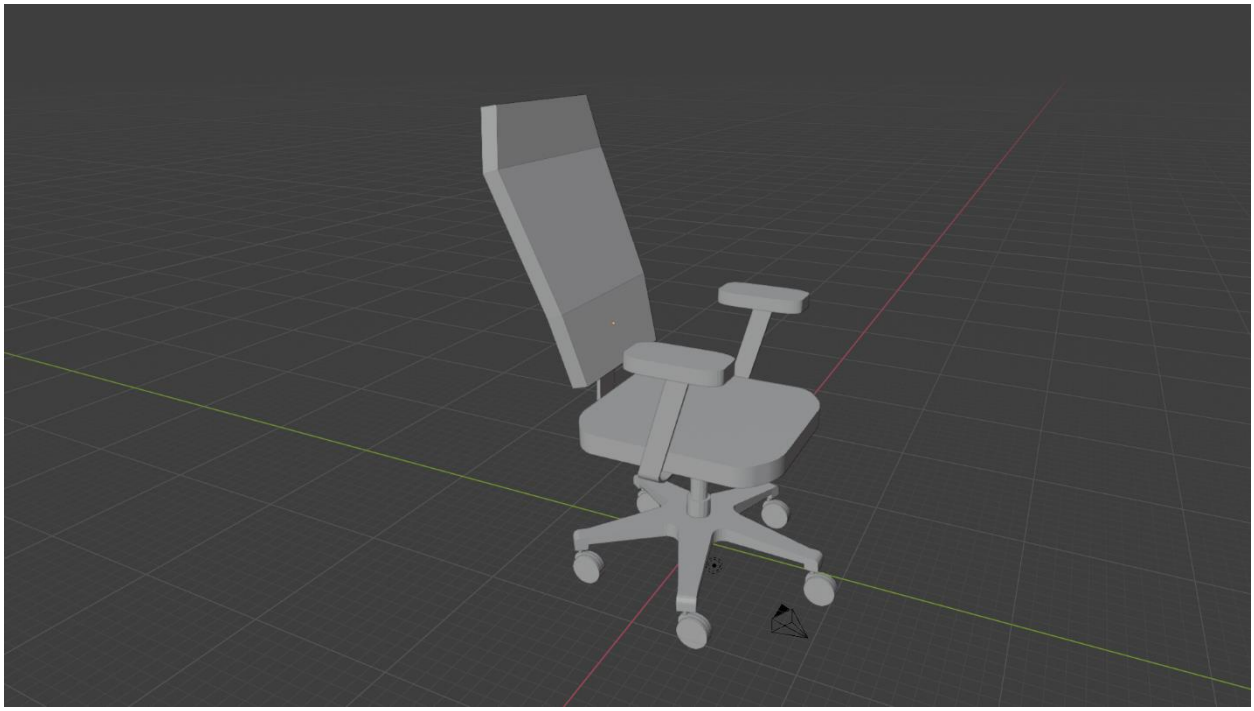
SMART CHAIR



University of Central Florida
Department of Electrical and Computer Engineering
Senior Design 1 2019
Dr. Lei Wei & Dr. Samuel Richie

Group 7
Mackenson Jean – Electrical Engineering
Annavay Kean – Computer Engineering
Bonarine Ramjas – Electrical Engineering
Thien Nguyen – Computer Engineering

1. Executive Summary

The project is a smart chair to encourage healthy habits in an office job. One of the many issues facing office workers is back pain caused by long periods of time sitting coupled with poor posture. The product would include a ‘pad’ that would be placed onto an office chair and would gather and interpret data on the user’s sitting trends. The smart chair would monitor how long the user sits and remind them to get up at a regular interval. The device would also monitor the user’s posture and provide feedback. The smart chair would pair with an application on the user’s phone to provide feedback and reminders. The feedback will come from research related to common spine issues and simple techniques to provide relief.

It is beneficial for all people and offices because the smart chair will be powered by a low voltage power load and rechargeable. The battery itself will power all sensors, LEDs and Bluetooth transceiver. Due to the back pain resulting from sitting too long in a chair, the smart chair will vibrate after a predetermined or user defined interval, and it will tell you how long you should get up and stretch or walk for. The data being used to advise and correct the user’s habits will primarily be collected using a grid of load or pressure-based sensors. By using these, the design could be considered more “plug and play” for existing office chairs.

2. Project overview

In order to better define our objectives, motivations, and responsibilities within the scope of this design, the following sections will discuss those very aspects. This process served to keep the project within its definition as we were going through the design process.

2.1. Background

The National Institute of Neurological Disorders and Stroke claims that back pain is the most common job-related disability. Around 80% of adults will experience back pain during their lifetime and 25% of adults have experienced back pain recently.

According to Spine-health's website, sitting in office chairs involves a 'static posture' that can cause or worsen back pain. The increase in back pain is caused by over-stretching the spinal ligaments due to posture used when sitting in an office chair (slouching). A prolonged slouch posture can cause damage to the spine. It is advisable to maintain proper posture when sitting in an office chair to reduce the cause of this pain. Ergonomic chairs can assist in improving back support but will not stop back pain unless the proper posture is used [1].

According to Spine-health, proper posture includes making sure your office chair is adjusted appropriately as well as your sitting posture.

How to determine if the office chair is adjusted correctly:

- User's elbows should be able to reach the keyboard and mouse while maintaining a 90-degree angle with the spine.
- Ensure that there is not too much pressure on the lower thigh by fitting two fingers between the seat and the user's thighs. If there is too much pressure on the thigh, correct with a footrest or raise the chair height.
- Check the chair back by ensuring a fist can fit between the end of the chair and the user's calf. If there is not sufficient room, adjust the back of the chair.
- The user's back should be pressed to the back of the chair such that the lower back support built-in to the chair pushes the lower back slightly outward. This will protect against back pain and will encourage better posture.
- User's eye level should be level with the center of the monitor. Adjust monitor as necessary.
- Arm rests should be high enough to slightly lift user's shoulders. This will reduce pressure on spine.

Regular standing breaks can aid in reducing back pain induced by office chairs. Spine-health recommends getting up from an office chair at least once every half hour for at least one to two minutes.

2.2. Roles & Responsibilities

Annavay:
Annavay was the lead for the mobile application's user interface design due to her prior experience in the area. She also worked on the application's backend to store user accounts and user data. She worked with Thien to ensure proper communication between the application and the hardware.

Thien:
Thien worked along with Annavay on the backend to take in information from user and allowing them to customize a set amount for a timer and notification. He also helped on the integration process between software and hardware to communicate over WiFi to synchronize with the application, allowing it to notify users and set guidelines.

Bonarine:
Bonarine took point on the sensor & peripheral (Vibrate, WiFi module) integration during the design and test phases. He supported Mackenson with PCB design, using it as an opportunity to learn eagle, and share responsibilities as stated below.

Mackenson:
Mackenson took charge to design the PCB and that we need for the project, and work with Bonarine to implement the power that was used for the sensor, WiFi and vibrate systems. As a group, we worked together to integrate the software and hardware while testing proper functionality.

2.3. Specifications & requirements

This section includes the self-defined requirements for the project. A summary of the project architecture is show below to aid in the discussion.

Figure 2.1: High level Project overview

2.3.1. Software

- The application should allow the user to log in.
- The application should save user data.
- The application should protect user privacy.
- The application should have default settings in accordance to scientific evidence available.

- Default time interval between walking reminders should be 30 minutes.
- The application should remind the user to get up after specified time interval.

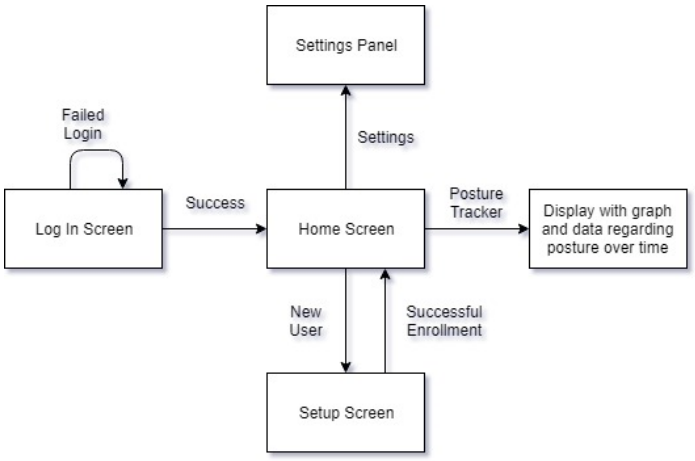


Figure 2.2: First Draft User Interface

2.2.1.1 Agile

In order to maximize efficiency during the development and ensure that the project requirements are met, the developers used Agile development.

Agile is a model for sustainable development cycles. Agile advocates for the use of ‘sprints’ which serve as chunks of time devoted to specific feature development. After the customer and the project manager negotiate requirements, the developers plan out a series of sprints. Each sprint will concentrate on developing one section of the application. No other requirements can be added to development once a sprint has begun. Each sprint should last no more than one month. During a sprint, there should be a fifteen-minute tag-up among developers where each member must respond to the following questions: What did I accomplish yesterday? What am I going to work on today? Only under very rare circumstances should a sprint be canceled.

After a sprint has ended, the development team should have an extended meeting (usually a few hours). During this meeting, the members should discuss how the last sprint went. Did something not get completed? Did something go particularly well? After discussing this, it is necessary to plan the next sprint. At this time, requirements can be added or updated [2].

2.3.2. Hardware

- The device should update the sensor input readings every 30 seconds.
- The device should pair with a phone app.
- 2-day standby battery life, minimum.
- PCB not to exceed 200 cm squared.
- PCB Housing should not exceed 1500 cm cubed.

- Sensor array should be able to load a minimum of 100lbs (45.4kg).
- Vibrate module should operate between 200-400 Hz or rpm equivalent.

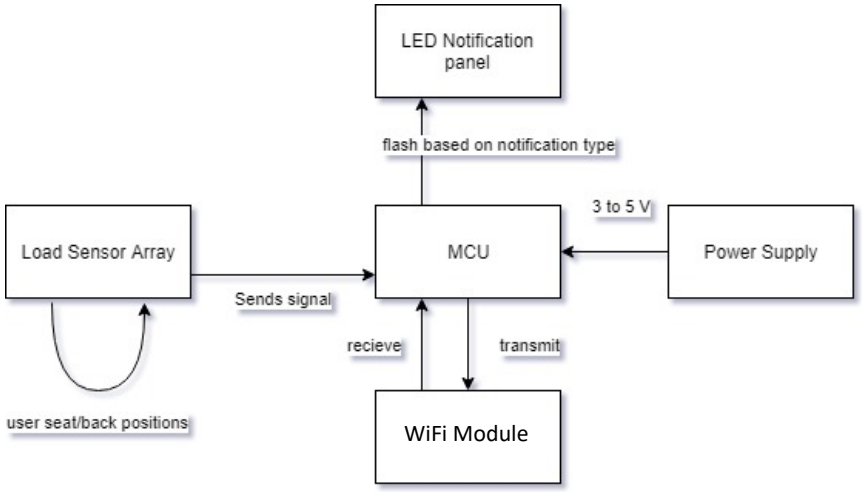


Figure 2.3: First Draft Hardware layout

2.3.3. Disclaimer

The intent of this project was to study common issues associated with poor sitting habits including slouching, leaning, crossing, legs, and prolonged periods and through various data gathering techniques, offer suggestive actions to the user that could serve to reduce or mitigate, the previously referenced issues as well as offer data trends for the . It was not intended to permanently heal or remedy more severe issues such as herniated discs, nerve damage/pain, poor circulation due to preexisting conditions, or issues that require ongoing treatment from a medical professional.

2.4. Milestones

Hardware

- Circuit Design & software simulation including calculations and filter/gain stages
- breadboard simulation with test parts
- Design PCB & simulate
- Order final parts
- Order PCBs
- Solder the parts on the PCB and reflow
- Test PCB

Software

- Storyboard the user interface
- Design Database

- Learn how to create a mobile application for android and learn how to use local storage.
- Create Database and populate with sample data.
- Create skeleton UI and get interaction with backend.
- Communication with hardware to send commands to vibrate modules and receive sensor data.
- Finished version of UI.
- Fully functional application backend.

Integration

- Send and receive data via Bluetooth
- User setting controls vibrate modules
- Read sensor data from PCB

Task	Primary	Secondary	Status
WiFi integration	Thien	Annavay	Completed
Database development	Annavay	Thien	Completed
microcontroller	Rudy	Mackenson	Mounted
power supply	Mackenson	Rudy	Completed
rechargeable battery	Mackenson	Rudy	Integrated
force sensor integration	Rudy	Mackenson	Completed
vibration module	Rudy		Integrated
user interface	Annavay	Thien	Completed
PCB Design	Mackenson	Rudy	3 iterations (final)

Table 2.1: Tasks & Responsibility

2.5. Constraints

This section served to illustrate the constraints imposed upon our design based upon decisions we made in subsequent sections relating to software and hardware.

House of Quality

This house of quality served as a guide to our technical and customer focused requirements. The result of which is coincides with a defined specification that ultimately served as a benefit or limitation to our design.

	Technical Requirements	Standby Battery Life	PCB size	Sensor Load	Vibration module frequency	Sensor reading	WiFi interface	Push notifications	Database for user data.	Simple application interface
Customer Requirements		∨	∨	∨	∧	∧	∧	∧	∨	∧
Receive feedback on current posture	∧	0	0	0	+	+	0	+	0	+
Chair must maintain mobility similar to unmodified chair	∨	-	-	0	0	0	0	0	0	0
Comfort of chair must be similar to unmodified chair	∨	-	-	-	-	-	0	0	0	0
The chair should pair with a mobile application.	∧	0	+	0	0	-	+	+	+	+
The application should provide meaningful feedback.	∧	+	0	0	+	+	+	+	+	+
The application should keep a history of user data.	∨	0	0	0	0	+	+	+	+	+
The application should be simple to use.	∧	-	0	0	-	-	+	+	+	+
	Specifications	≥2 days	≤230cm^2	100lbs	200-400Hz	every 30 sec		5 minutes (for demo)		≤10 clicks for a task

Table 2.2: HoQ

3. Research Related to project

This topic will converse the numerous and possible technologies with handy or approximate relation to the Smart Chair. In order to do more research about our project, this part will dump interested in existing projects or products that are previously done by a company or anyone that are extremely current to our anticipated development where we are focus in life which helath system for the professors and people who have being working for a long period seating in a chair.we came to this idea an addition to gain more detailed empathetic of the core technologies that are fundamental to this final project so that we could developped our knowledge to show the skills that we have and getting more educated in the option and collection more education by doing research and develop everything that we have learned in class so that we can use them in our world. By picking this project, we look at each part and research about them to see what they can do and their functions in the project, then choosing the best that can be used for the Smart Chair. Two years ago one group from University of Central Florida have a chance to come with the idea and work on it, but it not work well [3]. There are a lot of smart Chair on the market, but they do not do the same work that we want our Chair to do. Our project will be the first on the market to help people protect their health due to the function of the chair.

3.1. Spine/posture health

Below we will discuss some of the motivation and research behind the need for our device, and some of the issues we can potentially solve. In a subsequent section, we will discuss the suggestions we intend to offer the user to help mitigate some of the issues that arise from prolonged sitting and poor seated posture.

3.1.1. Skeletal

The vertebra column (the human spine) can be grouped into 5 regions: cervical, thoracic, lumbar, sacrum, and coccyx, however the last two begin the pelvis, tailbone, and peripheral structures of the body's lower region. The spine as a whole system is comprised of vertebrae which house the central nervous system, interconnected by sponge like material known as a disc. Our project focuses on the midthoracic down to coccyx region of the spine as this is where most issues arise from sitting with poor posture for long periods of time. It should be noted that with proper posture, long periods of sitting can cause fatigue in similar regions of the body. The lumbar region of the spine is by far the most common area affected by poor posture. Focusing on the lumbar spine, the region typically follows a *lordotic curve,* or more simply, a backwards "*C*" and deviations in this shape are where discomfort begins; note that from person to person this curvature naturally favors a flatter or more arched shape [5].

3.1.2. Muscular

In addition to the skeletal, there is also the muscular anatomy of the back. While much of the focus traditionally goes to the spine with regards to back health, the muscles play a far more important role and tend to be where many of us feel discomfort [6].

3.1.3. Circulation

An issue that is often overlooked with respect to sitting for extended periods of time is circulation. Sitting requires less effort than most activities which leads a person to usually being at resting heart rate; often associated with lower blood pressure [7]. This coupled with several muscle groups holding fixed positions, and the constriction of blood vessels in certain regions, specifically those supplying lower extremities, can lead to reduced circulation within the body. The results of poor circulation vary from discomfort, such as the tingling of a limb “falling asleep”, to more severe lifelong issues.

3.1.4. Center or Gravity

An integral part of this project was learning how to interpret sensor data from the chair in order to determine how the user is sitting. One idea was to use the grid-like sensor data to determine the user’s center of gravity. In a peer-reviewed article published by Very Well Health, they discussed a study where it was determined that lower-back pain correlated to a center of gravity further back than average. This study also concluded that individuals with lower-back pain would have strength and balance challenges when trying to improve posture [4].

3.1.5. Recommended Exercises

According to DoYouYoga.com, there are yoga poses to help alleviate back pain. Below we will discuss some of these poses in order to aid in the alleviation process [5].

Exercise Disclaimer

All of the exercises described here should be performed at the user’s own risk. These recommendations have not been sanctioned by a medical professional and should be approached with caution.

Downward Facing Dog

How to Do:

1. Begin on the floor on hands and knees.
2. Lift knees away from the floor.
3. Stretch legs such that hands and feet are supporting all weight. The body should be making an upside-down V shape.
4. Hold for one to three minutes. Then move into Child’s Pose [6].

Reclined Pigeon Pose

The reclined pigeon pose works to stretch the hips, which can help alleviate lower back pain.

How to Do:

1. Start by lying on the back with knees bent.
2. Move the right ankle over the left knee. Place both hands behind the left knee.
3. Slowly pull the left knee toward the chest.
4. Repeat on other side [7].

Knees to Chest with Slow Rock

How to Do:

1. Begin by lying on back.
2. Draw knees up toward chest. Hug knees with arms.
3. Tuck chin against chest. Try to place as much of spine on the ground as possible. Hold and release.

Do not perform this pose if pregnant and beyond first trimester [8]!

Supine Twist

It is important to note that this type of pose can cause herniated disks if not performed correctly. If performed correctly, it can help with back pain.
How to do:

1. Lay on your back.
2. Move your right knee over to the left side of your body. Hold.
3. Return to lying flat on your back.
4. Move your left knee over to the right side of your body. Hold [9].

Sphinx Pose

How to do:

1. Lay on your stomach.
2. Move your elbows under your shoulders, keeping your forearms parallel to the floor.
3. Press upward in a curve. Hold [10].

Thread the Needle

This pose is for all experience levels. The aim of the pose is to stretch the shoulders. This pose can help with neck, back, and shoulder pain. Do not attempt this pose if knees, shoulders, or neck is injured [11].

How to do:

1. Begin on hands and knees with wrists beneath shoulders and knees beneath hips.
2. Slide right arm under left arm. Right palm should be facing up.
3. Hold for 1 minute. Repeat on other side.

Double V Pose

How to do:

1. Begin on stomach supporting weight on forearms.
2. Adjust right forearm such that fingers are pointing left. Adjust left forearm such that fingers are pointing right.
3. Slowly move hands in the direction the fingers are pointing. Rest head on yoga block.
4. Hold, then switch sides by moving the rear forearm to the front. Repeat [11].

Child's Pose

How to do:

1. Begin on hands and knees.
2. Move big toes toward each other and move knees outward. Rest hips on heels.
3. Move hands straight out and lower chest. Let forehead rest on mat. Hold [11].

Cat and Cow Pose

This pose is a general-purpose stretch that can help with spine flexibility. It can help to correct spine alignment with persistent use. Pregnant users should not perform the cat pose, only the cow pose [12].

How to do:

1. Begin on hands and knees with wrists beneath shoulders and knees beneath hips.
2. Cow Pose: Drop stomach toward ground and lift chest and chin toward ceiling.
3. Cat Pose: Draw back up and round back up toward ceiling.
4. Repeat 5-20 times.

Arm Across Chest Pose

How to do:

1. Sit straight up on a chair or on the floor. The neck should be elongated and the shoulders should be relaxed.
2. Stretch out the right arm and cross it over your chest such that it is parallel with the ground.

3. Turn your head to look over your right shoulder. Hold, then switch sides [12].

Eagle Arms Pose

How to do:

1. This pose can be done sitting or standing. Elongate the neck and straighten the back.
2. Stretch arms out such that they are parallel to the ground and perpendicular to the chest.
3. Bend the right arm at the elbow such that the fingers on the right hand point skyward.
4. Wrap the left hand under and around the right arm such that the palms of the hands touch. Hold, then switch sides [12].

Pose	**Lower Back Pain**	**Upper Back Pain**	**Neck / Shoulder Pain**
Cat - Cow	✔	✔	✔
Thread the Needle		✔	✔
Supine Twist	✔	✔	
Sphinx Pose	✔		
Double V Pose		✔	✔
Child's Pose	✔		
Arm Across Chest Pose			✔
Eagle Arms Pose		✔	✔

Table 3.1: Summary of yoga poses for back pain**.**

3.2 Sensing User Presence

A key aspect of the design's functionality is knowing when the chair is occupied by its user. This relates to several hardware and software components. If the chair is not occupied, we would not want the WiFi module enabled and transmitting data, we would also like the sensors and power supply to disable or enter low power functionality (if applicable). In the event that this product was to enter a commercial market, many of these requirements would apply to standards relating to energy efficiency, which are

listed in the respective section of this paper. In order to sense the user's seating habits and posture we begin with the true or false, is the user present? This will be determined with a proximity sensor located in the backrest of the chair and the ten piezo sensors located throughout the base and back of the chair. There are a several methods to accomplish this. One involves using an interrupt associated with the state of the proximity sensor and some number of the piezo sensors. Once the combination is triggered the interrupt flag can be raised, activating the other sensors, begin to take readings and establish a connection with the user's cellphone via Bluetooth link. From there, another interrupt can be set to trigger after the sensors do not receive data for a predetermined amount of time which can reset the flag for the first interrupt and trigger a low power/off option for the device.

3.2.1 Sensing Posture

One of the more complicated parts of this project was deciding how to determine if the user is sitting with proper posture. Ultimately, the method we to do this involved a modified distance averaging equation which utilized sensors placed in a known grid pattern with measured distances from an origin. These are then weighted based on the amount of load placed on the sensor, which is then used to determine trends and provide feedback to the user.

3.3 Similar products

According to a report done by market research firm *MarketsandMarkets*, the IoT health devices market has growth projections of more than tripling by 2023, from approximately twenty billion dollars (USD) in 2018 [13]. This market includes various wearable technologies, measuring apparatus', and other connected devices designed to digitize the consumer health market. In the following sections we will discuss some products that overlap in marketspace while discuss some of their pitfalls which we hope to improve upon.

3.3.1 Upright GO

The upright GO is a body contact adhesive device that primarily marketed towards users with poor standing or walking posture. This focus does not take into account users who sit for prolonged periods of time. The device is attached to the users upper back, along the thoracic region of the spine, specifically, between the shoulder blades. This suggests the device focuses on slouching or forward leaning bias'. Priced at approximately eighty dollars plus tax and shipping, at the writing of this design brief. With an average rating across amazon and the OEM website of four out five stars, this device fills the need of its market. The direct contact with the user and the necessity of continually purchasing adhesive strips in order to wear the device may dissuade potential users of such device. Our device aims to provide the user with a plug and play solution, once the chair is calibrated to its user, the only maintenance required is charging the device [14].

3.3.2 Wearable device (non-smart)

Another tried and true method for correcting posture is the use of tension straps. These devices are capable of targeting upper, lower, and combination areas of the back. The principle is simple, a strap holds the problem areas in a fixed position, using tension to prevent the user from returning to a poor posture. The straps that target slouching typically do so by pulling back the shoulders, this is because a square shoulder position prevents rounding of the back. These devices range in cost from tens to thousands of dollars, based on several factors including target region(s) and how custom it is to the user (note that some of these devices come with medical recommendations from medical professionals). Something to consider with a device like this is that since the strap is holding the tension, it may take the user longer to develop the muscular strength required to hold the body in the optimal position, if at all. The device also covers a relatively large portion of the body, as such if worn under clothing it would require periodic washing or even replacing, and if worn over clothing may cause the user discomfort or even awkwardness [15]. Referring to custom medical devices, these typically come in the form of braces. These devices tend to be bulkier and the primary driver of cost is custom molding to the user. These devices are designed with the intent of having the user wear them long term as the amount of support they provide reduces the likelihood of muscular development required to maintain a healthy posture long term. These devices can serve a workforce demographic that sits long term, as we defined for our project, but the bulk and cost of these devices justifies our project's purpose and cost.

3.3.3 Ergonomic chairs

The initial inspiration for this project is derived from the structure of a traditional office chair. While most office chairs offer some level of ergonomic comfort, the primary driver of posture correcting chairs is cost [15].

	Upright Go	Strap/harness	Ergonomic chair
Cost	+	0	-
Overall Effectiveness(user response based)	+	+	+
Lower back	-	+	0
Upper back	+	+	0
Lower extremities	-	0	-

Table 3.2: Competition Benefit Table

3.4 Hardware

This section serves to summarize relevant technologies and considerations with respect to the various hardware and software to be used in our design. The topics covered are

not final decisions but necessary research and decision matrixes weighing requirements of our design.

Hardware specifications

For the hardware description strategy, the whole team sat together and constructed a plan. Our first plan was to protect humans that have being sitting for long periods in chairs which results in back pain. We came with the Smart Chair and the components that would make it smart like sensors, printed circuit board, software and various electronics components. The actual Smart Chair is able to provide user trends about seating positions. The load sensors are placed to the right and left side of the chair on the bottom and on the front and the back of the chair's bottom. The load sensors are activated when they sense pressure, and it will send data to the computer software and the app that will be available to collect information. The sensors should be the monitor of the project and control the load to the user. All the data that has collected will be stored in the internal memory of the system and will be display in phone screen or any computers that have access to the system to the user can read really well without mistakes. The sensors should be installed solidly in the chair because they might receive a lot of movement from the user specially the user's weight. It will be installed inside the chair where no one can have access with them because they have been powered with electricity. They are all flat enough to not disturb the users when they sit on it. The structure has an assortment of sensors operating in a microprocessor unit as a driver and a battery that will be connected and is rechargeable, inside to power the printing circuit board and an outside connector that is used to charge the battery. This battery is implemented inside the case that showed below. As we know, the chair will not be impacted, it can be moved everywhere like indoor and outdoor that why we use the rechargeable battery to power the components in case there is no electricity.

Hardware Issues

We have some latent hardware matters in the Smart Chair. One of those is how to get the instruments wires out of the case safely without being disconnected. Sensors are an electrical part that have power on them and are connected to the chair that is available for people to sit on. We chose this design to save people's life due to the back pain and how to sit normally in a chair for long time with stretching or doing some exercises. The hardware will be in contact with people because evertime someone seat, they have in impact on the hardward because they are hardware not software. The most problems that can cause some impact are the wires. We will be having about eight sensors in the chair, six in the regular basis and two in the back where people can cut the wires and being electrocuted. To prevent that, we have decided to add the sensors under the sponge and use the best quality of wires that will make the chair safe by piercing a holes in the in the right corner of the case and course the wires from the Printing circuit board to the sensors, the power supply and the rest part of awareness to make sure the project is safe for the users and send the data properly accurate from the components to the software usage.

3.4.2 Power Supply

The goal of this project was to develop a sophisticated chair that could prevent health problems yet be an easy to use office chair that would make professor's life and people who work in an office better. With health problems that can kill people, the idea of smart chair as a senior design project used technology to make people more aware. In the world of technology, power supply places an important role in the product. In addition to power supply, we decided to one of the best standards used in our project so that we can use the national and international standards so our product could market around the world. A main objective was to have safety in our power supply because using electrical components can cause severe damage.

We chose a power supply that would meet international standards. The power supply needed to have the proper conductor and insulator to protect the chair against electric shock and make sure it was safe to move around because the people using the chair may not sit in one place, so it is important to use a power supply that meets all the safety requirements because it also has a battery power supply implemented. As you know, batteries are the hazardous. It is important to use standards to regulate the chair because people are going to sit on it. Batteries can cause severe burns and can explode when full of charge and used under a large weight.

The power supply that we used in this project meets the American and European standards to prevent electric shock such as, fire, explosions, burns, and hot wires. We had to do all the calculations necessaries to pick the power supply cords and the adapter systems that were used to convert form AC to DC meet the requirements (standards) to prevent against all the shock. As we know, every project should meet the national and international standards, in following sections are some standards that use around the world for any projects that contain power supply specifically explosive stuff like batteries that can kill and cause some negative effects on humans being and materials.

Back in the day, precisely in the 19th century, electricity was a major problem in the world. Thanks to some scientists in the 20st century, now we can use various power technologies in our product at low cost. The smart chair used a power AC source of 120V and 60 Hz for North America which designated that the current alternates 60 times per second in the substation where it comes from. As our design used DC current, we needed a wall adapter that convert AC to DC to run power to our board. All the equipment in the smart chair have used 3.3 or 5-volt direct current. We needed to stepdown the voltage that come from the wall then using a rectifier and a voltage regulator to keep the components running at a constant voltage so that the components would not burn and explode due to unstable voltage rate.

First, the AC power in the wall was 120V. AC to DC converter took 120v AC and converted to 12v DC. We used a stepdown transformer from 12V to 5V because the most of our components are using 3.3V to 5V. These calculations run our systems that are using low power material. As you know the input voltage is not always perfect, we

needed a capacitor connected in parallel to the load, regulator and a diode to regulate the DC voltage from the transformation AC to DC so that the battery used in the system can charge automatically and keep the charge for an amount of time because the smart chair is connected to the wall. To do so the battery was charged and offers standby time of 2 days with low power options.

According to Wikipedia using, a rectifier in the project helped us to take the alternating current and transform it to Direct current. To reach the amount of current that we are using on the load. As we learned in electronics classes, we have two (2) types of rectifier: half-wave rectifier and full-wave rectifier. The half wave can be used in a single phase like our project, but it has a negative effect because it is ignored the negative sinusoidal wave to being zero that we had in the input as a step function we had one half of the input waveform that reached the output waveform which means it had a lower voltage.

Compare to the half-wave rectifier, the full wave rectifier is converted the full input waveform to one with a polarity constant. The advantages of using the full wave rectifier was, it is more effective and converts both polarities of the input to direct current and it produces the voltage higher than the half-wave, and it has a positive effect on the negative sinusoidal input where it creates a continuous power supply on the circuit. Due to the advantages of the full wave on the circuit, we used it on our project to stabilize the direct current that we have. We accomplished the change from the alternate current to direct current that will need the main important electrical supply to achieve the function of a full wave rectifier that we calculated and built the design by using the diodes, capacitor and resistor to connect to the sensors.

Using a full-wave diode connection can help us have a better project because it had 4 diodes connection to get the transformation that we needed. One of the diode functions helped to get the direct current in the right direction and have two pairs of diodes that conducted the direct current DC in one direction. Having four diodes in the rectifier is divided in two halfwave, one half wave for the positive direction and the other halfwave for the negative direction of the cycle that helped us to get the positive cycles to go the right direction that we needed to power the load in direct current DC. Through to the full wave, we had to add a capacitor that should place in parallel with the load to regulate the current going to the load which is the straight line that we needed to have in the direct current DC. The reason that we needed the capacitor was to charge and discharge the alternate current AC input when they are switching one to the other by a sinusoidal wave to correct from sine wave to line that is needed to get in direct current for the transition from one peak to the other peak. The capacitor stored charge as case of a rupture where it is provided backup voltage to the load and prevented the components without any loss of power.

Power usage

Moving the chair everywhere was the most important aspect in picking a power supply to achieve the longest distances from the wall that could reach the AC source, a

rechargeable battery-operated source is employed for the last long possible charge that can power the chair for the distance. A lithium-Ion 9 Volt battery and 1200 mAH was using in this project to power all the features materiel that are essential to monitor the chair and preserve it from discharge. It persisted functional at the very minimum of one-week life that can bring the chair everywhere with a full guarantee of charge because we were using the low power mode system on the chair. When the chair is not in movement the low power mode function is used to save energy and battery life. The battery and a power supply of 120V AC input power to connect to the wall to charge the battery. The power supply had a 9V DC output to charge the battery that needed the direct current as output.

As we are using a rechargeable battery, it allowed the chair's owner replaces the battery in case it has something wrong without the assistance of a professional that will charge them a lot of money, they could do maintenance on the battery with a free cost. However, the chair by itself has several dedicated technical services on the software that will allow the customers to see all the information related to the battery and the components inside the smart office chair. The software indicated the battery's life, and when they had to do a quit maintenance on the chair like the battery position and when to change the battery to make the world technology easier and applicable for all situation. The battery had some commands to the board and send signal when it is full of charge and discharge, then beeping, and the red LED was lighting to allow the user that it time to charge. As long as the battery is full the green LED will be light up. One of our purpose in this project was using the printing circuit board to distribute the power in each electrical component. A 9V battery was sending too much to power the PCB because it required a 5V to run and 3 to 4V to the rest of the component. We were using a step-down autotransformer with a combination of a rectifier to find the amount of voltages that allowed to power the PCB and the electrical component to get the accurate quantity of power at an unchanging rate. Selecting a 9V lithium battery was the perfect choice because it gave the amount of voltage that each component needs at low cost.

3.4.3 Load sensors

In order to obtain consistent and reliable data about the user's seating habits an array of sensors was utilized. The composition of this sensor array was determined based on weighing cost, efficiency, accuracy, and integration. Below, we will explore the different types of sensors that would have applications in our design as well as draw comparisons between each to conclude the ideal fit, based on the previously stated parameters. The beam style load cell, in its simplest iteration, is comprised of a deformation element, such a spring, a strain gauge, and some form of housing these elements. As the element deforms the strain gauge will also deform, thus resulting in a change in the strain gauge's resistivity. This results in a signal on the order of a few to tens of millivolts. The element paired with the strain gauge determines the range and longevity of the sensor, these are typically larger devices when compared to sensors such as piezoresistive force sensors. The strain gauge offers longevity of the sensor and is useful in applications where the load can be static for prolonged periods. The

load cell can be subcategorized as one of the following: pneumatic, hydraulic, or electric. For the purposes of this discussion we will be referring to electric options [16 - 17].

The device converts a compressive (or tensive) input signal to a variable voltage output. Taking a look at the load cell technology, the sensor is comprised of a point of contact that is attached to a sealed cylinder filled with a conductive matter, with some output wires, housed in a sealed container. This container is usually cylindrical. The material inside the housing will undergo a change in electrical property which is read as a voltage or current change. From this change, typically a wheatstone bridge circuit will sense a change in resistance, providing an output signal, which can be processed and interpreted. Sensor calibration can be achieved with calibration weights and the specific sensor's datasheet, detailing the corresponding output. These cells are very similar to the strain gauge where a change in electrical property is induced under load. The key distinction is that here, a voltage is induced, as is characteristic of piezo-resistivity. This works well in situations where the load continually varies. In applications where the load is static, i.e. it is applied and does not change over an undefined period, we find there won't be an accurate output.

3.4.4 Proximity sensors

Sensing the surrounding area for objects with respect to a reference point is the purpose of a proximity sensor. These will typically emit some range of electromagnetic radiation, and look for either changes in the returning field, or the return field itself. The type of proximity sensor required is based on the object(s) being targeted. In the following section we will discuss common proximity sensors and the basic theory behind each. From this we will conclude the ideal choice for our design. Sensing the distance between the user's upper back and their chair back requires a sensor that will be subject to being covered for long periods of time (when the user is actively in the chair and seated in an appropriate posture). Similar to ultrasonic sensors, microwave sensors detect proximity/motion of an object using waves. The key distinctions are in the spectrum of operation and the speed and accuracy improvements due to the use of higher frequencies, over ultrasonic sensors. They come with their own drawbacks which include higher susceptibility to interference, which is due to their spectrum of operation being the same as a large number of wireless communication devices and methods [18].

3.4.5 Amplification circuit

With the use of many of the aforementioned sensor technologies, their outputs register on scales as small as one-one hundredth of the input. One such sensor would be the compressive load cells that operate on a few volts and register electrical changes in the range of millivolts or tenths or ohms. In order to turn these outputs into useable data we must magnify these minute changes [27]. This is where the use of various gain circuits comes in. We may consider a simple op-amp circuit where we can calculate the necessary gain required to supply our MCU with a reliable, interpretable signal. Op-

amps offer low cost gains and simple calculations to achieve consistent results [19]. Ultimately we chose voltage dividers for integration in order to save time in the PCB design as well as space on said PCB.

3.4.6 Microcontrollers

At the core of most digital electronic devices, is a module that interfaces peripheral input, processes them and provides to the end user in a meaningful way. For many of these types of devices, this module is a microcontroller. By standard design, a microcontroller consists of some onboard memory, a central processing unit (CPU), and I/O pin(s) which can be general purpose. These devices are often considered self-contained systems, or specialized computers based on their applications. The versatility of an MCU, makes it an ideal choice for our design, in comparison to an FPGA or actual computer. With respect to programmability, there is a wide range of languages and development platforms. These will be discussed in another section. Microcontrollers can increase in complexity based on the number of GPIO's, operating supplies, the addition of other sub systems on the integrated-circuit (IC), such as analog-to digital converters (ADC's,), AC-DC converter IC's, and communication protocols.

3.4.6.1 Communication Protocols

As mentioned before, communication protocols are a crucial aspect of transmitting data, as a subsystem of the microcontroller. This determines what and how data can be transmitted. I2C, bus, UART and several other communication protocols offer unique capabilities over another as we will discuss below. We keep in mind a few of the considerations being made for our project which include the ability to receive data from multiple inputs while providing data to multiple outputs in near real time, the timing intervals required for these sequences. Also, we should consider the types of data in and out such as the ability to support ADC.

3.4.7 Haptic Motor

A crucial aspect of our posture correcting measures is notifying the user of poor seating positions, and interval-based reminders to stretch. These would offer a less distracting method of communicating with the user as compared to a flashing light or sound. This is countered by the fact that the vibration could startle the user, resulting in consequences such as breaks of concentration, discomfort, and in some cases, severe health issues. It should be noted that if done in a manner similar to the haptic drivers of cellphones, these consequences could be minimized. Below we will explore the common types of haptic sensors used in cellular devices.

3.4.7.1 LRA

LRA's rely on an AC voltage to drive a voice coil that is in contact with a spring-loaded mass. Vibration is achieved when the resonant frequency of the spring is produced by

the voice coil. This offers fewer moving parts which improves the longevity of the part however, in small applications where DC is the preferred power supply a transformer is required. This also allows for more precise and varied patterns of vibration, but intensity is limited by the resonant characteristics.

3.4.7.2 ERM

As the name suggests, the ERM relies on offset mass that rotates. This typically operates on a DC supply. They are comprised of a small motor, an eccentric mass, and a shaft connecting the two. It is akin to a washing machine that isn't properly balanced, the result is the machine shaking, or vibrating [30]. The feedback is varied by the input, providing for more range and intensity of the driver, however because we rely on moving parts, the precision and patterns that can be produced is limited compared to LRA's. these motors are far more common that LRA's, provide better market access and costing [20].

3.4.7.3 DC Coin Module

Ultimately, our design utilized the *Dowonsol* DC coin vibration modules for the chair based user notifications (as opposed to the phone notifications). This design choice is the result of multiple including cost and size compared to other market available options. Seeing as we already designed our power supply, we needed an option that could operate between 3.3-5v. this module operates between 2.8-5v. We purchased ten modules at a cost of ninety cents per module on *Amazon*.

3.4.8 Batteries

To achieve this project, we needed backup power for smart chair can be more dependable and helped people working in the office more active. As Smart can be indoor and outdoor, we decided to do it smart by using all equipment necessary to make it comfortable for people and used a 9V battery which is not heavy for the chair and can charge for one week without connected to the wall. This is enormously important to find a better way for our professor and people working in office to have a smart chair that tell them when to have some exercises and taking a walk. We used 9V lithium battery because it had adequate voltage to be able to use on our apparatuses proficiently, it is last longer because it was using a full charge for at least a week**,** it can be replaced easily in case the maintenance is needed, and it can be in the board without taking a risk of unprotected battery.

The implement of a 9V battery was good enough to power the chair. Since our system ran with 4 to 5V, we needed to decrease the power dissolute of the battery by using voltage divider circuits analysis to find how we are using the 9V battery in each of the component where we can choose the perfect stability between proficiency and size. When we pick this project, we already thought about how to make the office chair smart for people and can be used with backup power which is a battery. The battery was working in the main circuit connected to board and shared the positive end with the

load, and we connected negative to the ground side where it was protected by the battery safety and standards. The diode was connected to the end of the positive side to the battery that came in action when the electrical component is not getting power. As in the project we accompanied a rectifier to the in the input of the PCB to the key load, the full-wave diode was used as open circuit in contradiction of all the contrary side of the current that was in the project.

Having a Bluetooth to connect to cellphone and computer to display information about the chair, different number of sensors for the load, we used LED that will allow the customer when to stand up and doing exercises as we have more items that will consume power, we planned to have the power usage that was around 50 watts which is not going to be bad for the smart chair. All the components was using 5V from the battery which are involved energy that we need to control because we want the smart chair to use less energy as possible well-organized and precise to power the systems with all electrical components that we needed to make this project works. We used the American standards in our project where we were using the power from the industry that we have in the wall which is 120V and 60 HZ frequency that we used in United States of America. We used the wall power to recharge the chair when the battery is about to discharge, and we added a battery to deliver power to the chair when the adapter is not connected, and the chair is outdoor. The smart chair project was able to use outside the US specially in Europe where they use 220V and 50 Hertz, in that case we will need another adapter that can connect to the European system as the input and gave the same amount of voltages as the output to provide the chair. At that time, we did not need to redesign the PCB where we use the international safety in our project.

3.4.9 Hardware selection

Below, we will discuss the various products being considered for our design and ultimate decisions as well as justifications. This section will also be the location of design revisions and changes with respect to components as with any design, though it is not wanted, changes can occur.

3.4.9.1 Breadboard Testing

We chose a circuit board based on the project we have and how we wanted the board to look like. As the board is inside such that no one can see it, but it is safe to be considered the types that must fit the design where all the symbols and laws were respected. According to Wikipedia there were so many types of printed circuit board PCB such as the board can flex, double flex and single. Our project was not sponsored by any companies where we have to pay everything out of pocket, we decided to use the double layer not only for the cost also for the manufacture and it is easier to use than the others. Double layers are safer use to use for the electrical and electronics components such as Diodes, Microcontroller and Bluetooth Module.

Board Testing

We tested our material in breadboard that we used in the lab just to make sure the part that we have non defecting and get the result that we needed to have for our project. We had to connect some of the electrical and electronics components to the board by using a function generator, multimeter and other components to determine whether or not all of them or working properly as the way we want them. We used the breadboard as an example before we start soldering our components so that we could embraced the simulated the circuit.

Hardware design

As we know the PCB should be inside a box, we decided to design a 3D printing box that could fit for the printed circuit board PCB. We had a 3D wood printing that a dimension of 20 inches by 15 inches so that the PCB and all the other components can sit inside for safety because everything that has electricity flow on it should be protected to people. Smart Chair was able to move around and sell to world, we used SolidWorks to design the 3D wood board so that the dimension can be shape because plastic 3D printing cost too much. It was more protected with the plastic cover because in electricity we used plastic as an isolate that means even there is water in the area, it should be protected. As we are doing this project out of pocket, we did not want to make it too much expensive that's why we are using a 3D wood board to design the shape.

3.4.9.2 Microcontroller

The selection of microcontroller was the hardest choice to do in the project because it is the brains that will control all parts in our projects. The most important thing that the microcontroller did as it acts as the main driver for our sensors and will serve as the brainpowers of the project as it has dictated the actions based on the sensory understandings received from the chair. The microcontroller has input and output ports which will allow us to connect our various sensors and electrical components that our chair will possess. The Microcontroller control units was tasked with handling of subroutines and interrupt service routines to perform the necessary calculations to determine the proper feedback response.

Selection

With our list of needs for this project, we established a list of guidelines necessary to achieve the functionality we are after. One of these guidelines pertains to the amount of general-purpose input and output pins, GPIO, an MCU is able to offer. Our design was required many sensors to interface with the microcontroller control unit, so it was necessary that we selected an option that contains enough GPIO pins. Another guideline is the operational voltage. Most of the embedded sensors currently out in the market are designed for low power environments of less than 5V. We must ensure that the MCU is able to appropriately handle power which will not overload any of the sensors being used. We will talk more about the topic of powering the sensors in the

section detailing power consumption. Another important factor was memory. Specifically, RAM and program storage memory. We ensured that the MCU had enough storage capacity to load our compiled program onto the storage space. Also, we must ensure the supplied amount of RAM memory is sufficient to run our program without experiencing delays in runtime functionality or loss in data integrity. The last guideline we have for our MCU requirements is the selected CPU. We ensured that the CPU we select was able to execute the processes in a timely manner. Being able to program the clock in various modes was of high importance as this because it was allowed our project to run efficiently while utilizing the low power modes available. This saved power consumption while ensuring that we were able to run our program without any trouble.

Texas Instruments MSP430

A microcontroller that is frequently used in Computer Engineering classes at the University of Central Florida, UCF, is the Texas Instruments MSP430 microcontroller. This MCU has ultra-low-power consumption which is convenient for portable applications that run on battery power. This MCU features five low-power modes which ensures we are able to turn off all unnecessary components to save power while not compromising processing speeds. We decided not to go with this option because its board support package, BSP, comprises the Universal Asynchronous Receiver-Transmitter, UART. It is a hardware feature intended to help with the transmission of data while allowing for customization of data format. The reason why we did not like this feature is because it allows us to send a message only after the preceding message has finished which is responsible for latency issues while interfacing with the MCU. Another pitfall from this MCU is the fact that it has a 500 MHz clock speed which is quite fast and would sufficiently keep our program executing flow running favorably, however the faster clock speed could lead to issues with power consumption. For these reasons we decided to focus on other MCU options in order to ensure our power consumption is low while ensuring smooth operation of our program code [21].

Texas Instruments F280049PMSR

Another frequently used MCU in Computer Engineering classes at UCF is the Texas Instruments F280049PMSR. This MCU was used in our Embedded Systems laboratory course and performed well in most of the labs assigned. This system features real time control making it an excellent choice for closed-loop control applications. It has a Trigonometric Math Unit that contains predefined trigonometric operations and functions that are optimized for runtime and the Virterbi/Complex Math Unit. This allows for calculations using complex numbers. The board also features a wide array of self-diagnostic tools such as the missing clock detection circuit and a brownout reset circuit for detecting problems with voltage. This package is managed by an embedded real-time analysis sensor which corrects problems as they occur. A feature that is lacking in this selection is that it only has a single I2C line whiling having two for CAN, SPI, and SCI and 1 for LIN. This is a problem because we have more than 1 sensors that would require communication via the I2C line. This means that we would have significant problems with our project had we chosen to select this MCU [22].

TI MSP430FR5739SRHATEP

The MSP430 offers a breadth of versatility while bringing a familiar development environment and community. Having coursework derived experience with the msp430 it is a strong choice for our platform offering a 16-bit architecture, 24MHz clock frequency, 16 KB of memory, 1kB of RAM, and 16 ADC channels. We also have a number of interface types to choose from as follows: *IrDA, SPI, UART, & I2C* [36]. Seeing as this controller typically comes in a tape and reel setup, the package size is quite small, which helps with the minimizing of our PCB area and comes in at roughly 7.72 per unit (under 10 units). Being a part of the TI value line, this controller features low power capability offering and operating range between 2-3.6v. while not an ideal operating range, the sensors being considered would not conflict with this specification [23].

Microchip PIC24FJ1024GA606-I/PT

Looking at other MCU options, we decided to look into the PIC24FJ1024 family. This MCU is known for its hardware real-time clock and calendar system with time stamping that can be used for logging. This feature allows for the reporting of data collected over a period of time. As an example, if we wanted to notify the user of the number of hours they have spent sitting over the course of a month, we would be able to do so with this application. This is a highly beneficial feature as it would allow us to continue to build on our project by implementing new ideas while still allowing us to do the basic functions. Additionally, the sleep and low power modes are modifiable to selectively shut down peripherals to save power automatically when the system sees fit. Another great feature in this MCU is that it supports 3 I2C lines, 3 SPI lines, and 6 UART lines which would provide a total of 12 communication lines for us to connect our sensors to [37]. The chip has a 16-bit architecture which would not provoke any issues with the selected sensors we will be using in our project while ensuring that we have a sufficient amount of data bits to store and process our sensory data. A couple of cons for this MCU is that it 5 times and only allows for up to 5 external interrupt sources which restricts the number of features we can implement in our design. Also, the chip lacks native ethernet and WIFI connectivity support which means we must have a separate module to handle the connectivity for our project [24].

Microchip Technology SAMA5D2

The SAMA5D2 chip is another ultra-low-power ARM cortex A5 processor based MPU that we considered for our design project. This MCU runs at processing speeds of up to 500MHz and has ARM NEON SIMD engine which has high processing speeds and is very efficient. This MCU provides many security functions to protect the consumer's code from being copied by a third-party vendor. While we may have not considered this in the past, this feature is very beneficial if we were to produce our chair in real life as it would prevent a competitor chair maker from taking our technology without our permission. Moreover, the external data transfers are secure preventing hackers from

taking the data and using it for malicious means. In our case, it would prevent someone from taking the information gathered from the chair and using it to send malicious code to the MCU to cause it to fail. Some additional features of this MPU are low system cost and high integration with 4-layer PCB with less than 200uA retention mode with fast wakeup. A downside of this MCU is that it included 2 communication lines for I2C which would prove a problem considering we have several sensors that would need to communicate repeatedly with the MCU. Some benefits of the MCU is that it comes with support for LCDs, keyboards, and touchscreens built in. This would be great for add on featured in later additions of our chair such as tracking the amount of time spent typing to ensure the user is spending enough time stretching to prevent arm strain. In addition, to this it came with 7 different types of RAM controllers as well as 10/100Mpbs ethernet support built in. The bus width is 32-bit wide which again presents some issues of underutilization of the data buses again since our sensors are only 16-bit [25].

Atmel ATmega2560

The selection of MCU's from microchip technology is vast with their acquired *Atmel* portfolio of *ATMEGA*xxx's. Specifically, the ATMEGA2560U16 offers a 16MHz frequency (internally clocked with an XO), This MCU allows for up to 256 KB of programmable flash which would allow for our project to be of relatively large file size. It contains 2 8-bit timers and 4 16-bit timers for a total of 6 isolated timers. It also offers a total of 86 I/O's, and 16 ADC channels [39]. These ADC channels are crucial to our design as we have 8 sensors and may accommodate more should our measurements require a higher degree of accuracy, as defined in our engineering specifications. This MCU employs real-time clock. Another great feature is that it has six low power modes to ensure efficiency and power savings. The MCU allows for four external interrupts which is slightly less than the previously looked at MCU [39]. One of the benefits is that this MCU has a total of 86 general purpose I/O lines which means we will be able to connect several components onto the MCU without any issues. Another plus in this MCU is that it uses the highly favorable 16-bit computer architecture. The chip contains 4 UART, 5 SPI, and 1 I2C serial ports for digital communication [39]. This is by far one of the best options we have as it allows us to simultaneously communicate with the various components we have in our design. Additional requirements that are met with the selection of this microcontroller are the size constraints of our pcb as this is smaller than the TI equivalent. In terms of cost this controller does price higher than the ti at 12.04 per unit (under 25 units). This controller offers an operating voltage of 5 volts with a tolerance of ten percent (plus or minus), this is more ideal for the sensors being considered [26].

MCU	F280049PMSR	ATmega2560	PIC24FJ1024GA606-I/PT	MSP430FR2311IPW16R	ATSAMA5D21C-CUR
Manufacturer	Texas Instruments	Atmel	Microchip Technology	Texas Instruments	Microchip Technology
Program Memory (kB)	256	256	1024	3.75	160
Data Memory (kB)	100	8	32	8	128
Clock Frequency (MHz)	100	16	32	16	500
GPIO	26	86	53	11	128
Operating Voltage Range	1.2 - 1.3V	1.8V – 5.5V	2 - 3.6V	1.8V -3.6V	1.2V
Size L x W (cm)	33.6 x 33.6	1.6 x 1.6	.07 x .07	.05 x .04	1.4 x 1.4
Price (USD)	$10.81	$12.20	$4.41	$1.62	$6.71

Table 3.3: MCU Decision matrix

Final Decision

In the end, our team has decided to choose the Atmel ATmega2560. We decided to use this MCU because it had a vast number of power modes which was ensured that we get the most out of our battery life. It also contained a large number of I/O pins so that we can connect more components to our design without any issues, such as the need for several multiplexers, or other secondary components required for proper communication between the sensors and user. While we are limited in number of UART connections, we are pleased to have additional SPI and I2C connections which we can utilize to make additional communications with the rest of the components in our design.

We also liked that it had a large amount of program memory which means that we were able to store code that possess a larger number of complex subroutines and multitude of different states to cover a wide variety of situations to ensure that we give the user the most capabilities possible given the data collected. Because the selected MCU has 16-bit architecture there would be no issue writing backward compatible code since all the sensors that will be utilized are specifically for this type of system. The trade of from this design along with the other MCUs such as the Microchip PIC24FJ1024GA606-I/PT is that we would not be able to have a calendar feature which would keep track of usage for a period of time. This would have been a great feature to add as it would allow us to give the user a detailed report of their usage along with suggestions for how they can

improve their mobility given the amount of time they spend sitting down. We feel that while this feature would have been great to have, we would like to start our project with an MCU that has a sufficient number of I/O pins to connect all of our components and have sufficient amount of storage space to keep all of our code on the onboard memory. We realized that if we would have gone with the PIC MCU we would have had even more program memory at about 1024 KB which would have allowed us to store all sensory data internally without any problems. This was another reason that made us really consider going with the PIC MCU as the stats for this product seemed to be much better than the ATmega option. One feature that drove our worries away was that the ATmega MCU had significantly more documentation available with regard to programming and pin location. This means that programming all of our components with this MCU would be much more straightforward in comparison to the PIC MCU. We thought this was of high importance because while our team possess great computer engineering skills sufficient enough to code our program, we wanted to find an MCU that was straightforward to program on so that we could focus our time on developing more features rather than spending time troubleshooting errors due to hardware abnormalities

3.4.9.3 Piezoresistive Sensor

Based on research and specifications of our project as it relates to sensing the user, the sensor type that meets most of these constraints is the piezoresistive force sensor. To briefly recap, the piezo sensor applies a known voltage across a variable resistance voltage, as pressure (or force) is applied to the material, the resistance will change. This change in resistance is typically linear and can be accurate to within 5 percent. This accuracy typically requires a higher cost. While the sensor cannot accurately determine where within its physical housing the pressure is applied, in an array it can very well offer data on weight distribution relative to other sensors. An unintended benefit of this is helping to maintain a degree of user privacy with respect to weight. In the subsequent sections we will discuss some of the sensors being considered as well as include a final decision matrix to determine the best suited sensor for the chair. These considerations will include cost, surface area, load range, drive voltage, and response delay. It is important to note that based on the sensor selected, the driving circuit will be constrained to best suited for said sensor. the driving circuit can be a number of choices including inverting/non inverting op amp or voltage divider. The reason for this is to amplify the change in signal seeing as the resistance change value caused by a load is in the magnitude of tenths of a percent, relative to the initial resistance value [29].

FSR-406

From preliminary research and based on the development platform we begin our considerations with the *Interlink Electronics* force sensitive resistive -406. This is a development kit part designed for actuated load sensing; repetitive task (and range of force). This part offers multiple integration methods which include the following: voltage divider, adjustable buffers, multi-channel digital interface, variable threshold switch, & current to voltage converter. The list provided are methods of application for this device

which also amplify the changes in resistance in order to provide usable voltage outputs, which are converted to force estimations. While cost effective this component suffers from the most common issue of piezoresistive sensors, accuracy between sensors can vary as much as forty percent. This is due to several factors which include manufacturing processes, choice of resistive elements, drift (measurement change based on length of time load is static), and signal amplification and processing. The rectangular shape of this sensor would provide optimal coverage when placed in an array which would allow for more precise measurements of weight distribution [30].

Spark Fun (SEN-09376)

the most cost-effective choice on the list, and seemingly readily available at the time of writing this brief, the *Spark Fun -09376.* This component comes recommended for development projects based on the Arduino platform [31]. Spark fun also offers a development board for this component which allows for plug and play solutions without the need for an amplifier circuit. The cost of the development board would drive up the price as well as the size of our PCB but would allow for better tuning of the component as well as a controlled test environment.

TEK Scan (A502)

In this section we discuss the most costly option, that also provides quantitative superiority with respect to the sensors discussed previously. This sensor offers several benefits including large surface area and low response time. With respect to the component size being a plus, it should be noted that too large of a sensor may result in a lack of precision in our measurements. With this sensor we also have a variable drive voltage option that changes the load range of the sensor; discussed in the datasheet, we see that lower drive voltages offer much higher ranges, this can be tuned according to design requirements which will continue to evolve in the design and test phases. An positive for this component is that the datasheet also provides a recommended driving circuit with a low cost op amp choice. This would reduce development and test time for a driving circuit. This in turn will allow for more time to focus on final design implementation and tuning the sensor range for precise, usable array data. The op amp mentioned is the *Microchip MCP-6004* [32].

TEK Scan (A301)

Another TEK Scan component is the A301. This option offers a circular sensor area, the first for the options being considered. With a diameter of just under point four inches. This sensor offers a maximum load rating of 445N [33]. while smaller compared to the previously discussed sensors, there are a few advantages to its shape. If used in conjunction with a larger sensor, say of a square shape, we can distinguish between predetermined areas or “zones” of a region and more specific subsections within a region. For example, if we define four zones within the base (or seat) of the chair using four large square area sensors, the smaller sensors of the A301 size, could be placed around or in between the zone defining sensors. This would provide data on not only

which region carries the most weight (relative to other regions) but also where the weight within the region is distributed. How this data will be used and interpreted will further be discussed in the testing section of this paper.

	A502	**A301**	**SEN-09376**	**FSR-406**
Price per unit	$23.24	$12.18	$11.25	$12.99
Surface Area(in^2)	4	0.375(diam)	3.0625	2.25
Response time(µs)	≤ 5	≤ 5	≤ 5	≤ 5
Load range(N)	44kN max	444N max	*	100N max
Drive voltage(V)	Variable	Variable	*	*
Amplifier circuit	Multiple	Multiple	Volt. Div.	Multiple

Table 3.5: Force sensor table

Final Decision

While cost is an important driver in the decision-making process for choosing an optimal sensor, we chose to prioritize accuracy and area covered in our decision-making process. This is due to the fundamental nature of these components with respect to the vast number of subsystems that will rely on this device. Based on the varying sizes and availabilities of the discussed items, we have chosen a combination setup of the TEK Scan options, *A502 & A301.* This combination will allow for the establishing of zones within the chair, while maximizing the surface area covered. While the spark fun option would be the most cost effective, the leads are not designed to be soldered, which would be an issue for the permanent connection we hope to make on our PCB. As for the FSR while both cost effective and more solder friendly, we find that the surface area is small, which would require more sensors to provide similar feedback. While our microcontroller selection can support a large number of inputs, these sensors require an ADC to process the input signal before becoming a usable form of data. As such, we wish to not use all the ADC channels on our chip, though the support is there. A benefit of using the two different sensors from *TEK Scan* is that we can average the cost per sensor, which comes to nineteen dollars per device. Since they Are sold in four or eight packs, we will consider ordering one eight pack of each sensor in order to test and develop while also allowing ourselves room for error should receive a damaged component or have a catastrophic test result.

3.4.9.3.1 Relative sizing

Below is an illustration of our two sensor sizes relative to a quarter. Their sizes play a crucial role on how they will be distributed in what we refer to as an array. Ideally, during testing we intend to place the four large sensors in quadrant locations, while filing the spaces between them with the smaller sensors. We are also considering the inverse of this design where the smaller sensors will define the quadrants and the larger ones will be placed in between. This we be determined on how we map the sensor outputs in a

tabular format and define zone activity distribution which will correlate to various seating postures.

Figures 3.2 & 3.3: A501 &A302

3.4.9.3.2 Testing the sensor

In order to verify the functionality of the force sensors, we devised a method to power them and define a reasonable response based on the following parameters: is the return signal amplified, what is the supply voltage, how much force is being applied (this must be some form of standardized weight for calibration. Referencing manufacturer provided test data, we were be able to determine with a high level of certainty, whether the sensors are operating within specification. After verifying functionality to manufacture spec, we must confirm our design theory that the sensors can be amplified using the MCU. We will rely on circuit analysis and reference designs to aid in the integration of the crucial components.

Calibration

In testing the sensors *Tekscan* recommends the sensor's be calibrated before initial use. According to their implementation guide this is done by uniformly overloading the sensors several times; doing so allows sets a reference by which the sensor can be tuned. What is happening during this process is we are looking for a specific voltage output from the sensor based on a ten percent overload (110% loading). Whatever the deviation is from the desired output we can tune the sensor by supply voltage or, line resistor. This line resistor acts as a voltage divider between the supply voltage and the sensor.

Mounting

Due to the construction of the sensor, it is important that care is taken in mounting the devices. They are designed for point or distributed loading, not shear force measurements which means forces are exclusive to the z-plane we do not want lateral pulling or pushing on the sensor. however, as their name suggests the *flexiforce* sensor

is somewhat malleable which will prove useful for the contours of our chairs seat. Again, following the OEM's recommendation, double stick adhesive tape is used to mount the sensors in place. By using tape versus a permanent adhesive, we can ensure that the sensors don't experience uneven or biased load distribution.

Proper loading

Referenced in the previous section is the concept of load distribution on the sensor. we can load the sensors in a few ways. One involves simply mounting them to the base plastic of the chair and then applying the foam padding on top, allowing the foam to act as the buffer and distributor for us. Another method would be to "puck" the sensors; add a hard material onto the sensor's readable area and let the foam rest upon that. In testing we find that either method is suitable, however since user comfort is important to us, the use of pucks may affect comfort for sensitive users.

3.4.9.4 Development board

In the designing and testing of this device, we find it would be beneficial to have a board upon we can make rapid changes and real time test before moving to a final decision that end up printed on a board that we cannot modify. As such we will discuss the use of a readily available development board that uses the a closely related variant of the MCU we have chosen.

ATMega Development board

In order to simulate proof of concept such as receiving data from our sensor array, processing the data, sending via WiFi and controlling the user feedback, we needed a board that allows for plug and playability, at a relatively low cost and ease of use. Specifically, we need to supply our *TEKSCAN A series* sensor's with a 3.3v supply, receive a signal which will be amplified with voltage divider, suggested by the manufacturer. For most of our development and prototype embedded software, we will rely on the *Egeloo ATMega R3* development board. Pictured below this board offers usability of most of the microcontrollers pins. This includes the multiple transmit and receive channels for UART functionality, the 16 ADC channels which we intend to use for our analog force sensors. Through development, which will be discussed in the design section, we found that the MCU's ADC channels offer variable gain, up to twenty times. this would reduce space on the pcb (as we would not require two op-amp circuits), and the only real cost is a reduction in bit value from 10 to 7. The effects of this will also be discussed in the design section. This board offers a 5V DC input, which requires a wall outlet AC/DC power supply, or a USB-B port which is also used for programming. By using this board we can also reference a functional design while designing our PCB, this maximized the use of our time while we focused on other aspects of the project, as was required. We also used this design to make improvements or simply try a different approach as we intended to learn while developing a functioning product. This board also utilized a variant of the microcontroller we selected [34].

3.4.9.5 PCB

As we know, printed circuit board PCB was the main function that was sending connection everywhere in this system from the software to hardware in the smart chair. In our project, the PCB was inside a 3D printed box to protect the electrical and electronics components against water. It was under the chair and have wires connected from the PCB to sensors, also the PCB had a microcontroller and a wireless communication system which we used the WIFI module that was able to send signal to the electronic components and was using for the app. We have more details about the printed circuit board for Smart Chair to have more ideas how we were able to design the board and fully function to distribute the work for each component.

3.4.9.5.1 PCB Terminology

There are so many method we can use to generate the printing circuit board that will be fit for the design that we need. In a project, there is no precise design must be surveyed because there are so many companies that have their own rules of the design and the components, also there were many diverse categories of software you can use to design your project. There are companies that just been there to design and create the printing circuit board for people. In order to successfully make and design the PCB, it was very significant to comprehend the terminology that were required to design and how to make it possible for everyone to understand the foundation and to familiarize with the term PCB that everyone who are working is this domain should know and the advantages [36].

Solder Mask

According to sparkfun tutorials, solder mask is the main principal thin layer that has been applied to the copper that which are employed on the printed circuit board. It is composed of polymer that are been used on the board to joint into the solder pot and the electrical connections which can protect the circuit against short-circuit. Solder mask is used everywhere around the world because it is the main protection of the board to protect it against oxidation and prevent the solders from causing fire and damage the PCB that are sometimes very difficult to built. It is also work as an electrical insulation in the board, and it usually have a green color to insulate the copper traces from accidental contact with other components like metal, solders, wires and more electrical and electronics components. The solder mask layer helps designer to solder to the right or correct places and making more spaces to prevent solder jumpers around the board that can electrical shock and the whole circuit. It makes the design more efficient and safe to use.

Silkscreen

Silkcreen is the most important layer in the design of a printing circuit board because it

applies to write in the board the components name such as the numerical values, the letters an dthe symbols so that any designer can work in unfinish project. It permits the user to get-together the components easier and designates the humans to better understand the board. The numbers, letters, and symbols on a circuit board can be executed on the silkscreen with different color like white which is the most recommended color in the industry. When you look at all the circuit board, you can imagine all the indications are silkscreen white label to indictae the functions of each electrical and electronics components from the pin of each material to the LED if the users are using them. White silkscreen is the most common and popular in every aspect because it is visible to everyone that has being working in the printed circuit board. Silkscreen is required for all printed circuit board PCB that used screen of polyester across all the aluminium frames. The PCB manufacturers use silkscreen to provide the manufacture name mark and identify the traces that allow problems. The main standard color for the silkscreen is white but companies can also use different color like yellow, black, red can use. There are three methods for applyimg the silkscreen on the board.

- Manual Screen Printing which can be operated when line are superior than 7 mil and the registration acceptance is 5 mil which the cheapest of all three that has being using in th eprinted circuit board industry.
- LPI which stand for Liquid Photo Imaging delivers more precision and legibility than manual screeing and is active when line widths are larger than 4 mil
- Last DLP which is Direct legend printing is the most exact and the uppermost cost for consumables because it is the most understandable printing of the letters compagnies.

3.4.9.5.2 Printing

Printed circuit board is a mechanical component that had numerous electrical and electronics components that were soldered on the PCB. The process took a while because we had to wait at least two weeks for the printed board to be chipped from China. A vendor of printed circuit board used some materials like copper, silkscreen and solder mask to print their PCB which make us have to wait a little longer for it because we did not have that many companies in the United State of America printed PCB and they were so expensive. When we got the printed circuit board, we needed to solder all the necessary components such as Diodes, Capacitors, Resistors etc. all of these were using wires that can connect them together, it can printed wire that connected all the electrical and electronics components like Bluetooth Module, load Sensors and the Microcontroller. These wires were all secures because each of had its own work on the printed circuit board PCB. Below is the example of one of the materials using by the manufacture to show how our printed circuit board PCB looks like.

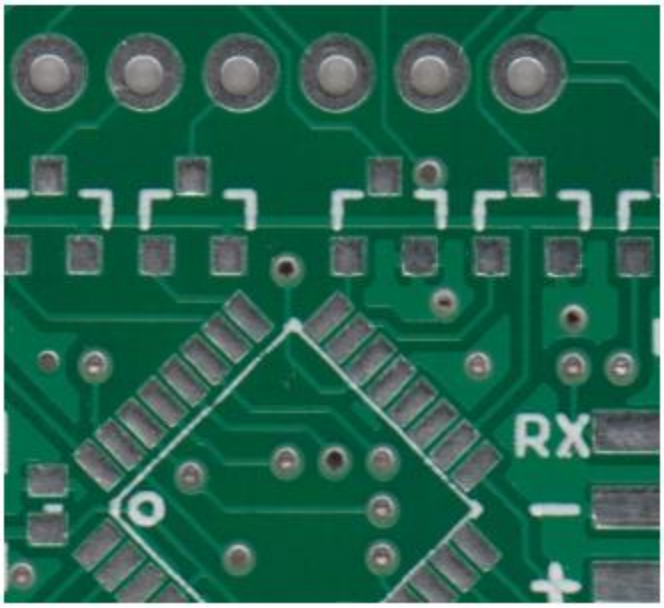


Figure 3.4: Example of Soldermask Board

3.4.9.5.3 Powering PCB

On every project that used printed circuit Board, we must think first about how it will work. The PCB must be provided by one of the best ways that almost all material has to deal with is power. Powering the PCB must be provided with the electricity which is the flowing of electron in a system. We had to contrivance a female pin that was linked on the printed circuit board, and it connected to the board so that the PCB can be powered, then we decided to have our source from the wall where it will provide us an amount of voltages that we need to power the microcontroller that will be using for the Microchip that is ATMEGA and the amount of power that we have should be enough to power all the electrical and electronics components that will be used in the printed circuit board PCB.

The Board had to power 8 sensors, Microcontroller, WiFi and more where we need to have enough voltage to power all of those things, to do that we add all the components power together, and we multiplied by a factor of 0.2 so that we can have the increase voltage in case we error in the system. This increase voltage will be 20 percent which is the percentage error and any types of blackout that going to he happened. As you know, the wall is 120V, we will be using a step down transformer that will be connected to a DC/DC converter where we had to add a rectifier on the system so that we can have our power rectify in the normal voltage that we needed for the microcontroller and the components in the projects.

The parts that has been used for the Smart Chair have a lot of changed necessities that wanted to be encountered for them to function correctly. A project had different parts that has been powered like Microcontroller, Bluetooth module, Chip, all the electrical and electronics components such as resistors, capacitors, rectifier and voltage regulator

that will be using all different types of voltage necessary. The Smart Chair cannot be working perfectly as the way that we want because one component can cause a lot of damages. All the components that we were using, we make sure that we look up the datasheet to see the function and the voltages rating that the components are needed to work perfectly. We were using eight sensors, all of them the I2C pins that we selected on the Microcontroller Microchip which were required from 3 Volts to 4 Volts that was associated to the pins that we choose from the Microcontroller where they shared the same voltages that we need and provided by the power supply. We used two pins on the Chip that we were using which is ATMEGA where connected our material like the sensors that we are planning to do first because they are simple to use and connect to the Microchip. Those pins are paired because we should connect them from male to female pins so that we could have the proper power that we needed from the chip Microcontroller which are from 3 Volts to 4 Volts.

As the Bluetooth Module are using 3.3 Volts power, we need to have the voltage regulator to have the maximum power that we needed from the wall and convert it to the amount power that we needed for the WIFI module. In this, the voltage regulator played a role of step down so that we could have the accurate power for the Bluetooth Module.

3.4.9.5.3.1 Regulating PCB voltage

In our project, we needed a voltage regulator in our printed circuit design because we needed different amounts of voltages in our design. In Smart Chair had a wall that is 5 Volt nominals and about 2.2 to 3 amps with a maximum output power of about 20 watts. As we know the definition of a voltage regulator, it is automatically used to decrease and increase the voltages to a constant voltage level that was going to system. It took the voltage from 9 Volt and bring it down to the amount of voltages that we were needed to use for the printed circuit board PSB 5 Volts, and the voltage regulator that we were using for our design has 3 Pins. Each of the pin was connected to a different part of the system which were the left pins was the output, the middle pin was the common sometimes used as ground and the third and last pins was the input where we used it as 9 Volts input, and each the pin can be deliver a current of 1.5 Amps. In our Case a voltage regulator was a perfect fit in our system, but we did not really need it because in our design, we want to make we have the normal voltage that we really needed to provide all the material such as Microcontroller, WIFI module and sensors so that our project can use the nominal voltages that we design for it.

However, we still used two voltages regulator in case something happened because we really care about safety. Our main power source was coming from a source that can have a problem at any moments, for our safety, we usede a wall source that will a DC/DC that can give a 5 Volts output and 2 Amps. This can give some problems to the Bluetooth because it is 3.3 Volts. For that we needed a voltage regulator for the Bluetooth to give 3.3 Volts to the Bluetooth and the Vcc power supply for the Microcontroller and connected to all Vcc pins that we needed to power all the pins that we were using for the Microcontroller Control Unit, and the second voltage regulator that we used in the project was using as the step down voltages that came from the battery.

The battery was used when the power wall is not connected, but the battery is 9 Volts that is not appropriate to connect directly to the printing circuit board PCB that will need 3.3 to 5 Volts. The second one used to step down the 9 Volts battery to the regular voltages that we needed for our project. When the Smart Chair was not connected, we needed enough power to send to the electrical and electronics components such as sensors, Microcontroller and WiFi if the main power is not connected or off [37 - 38].

Name	Description	Voltage	Current
Microcontroller	High performance with 75 pins and 10 IC pins with 16-bit timers memory	3V – 4V	18 mA I/O pins
Sensor	8 piezoresistive force sensors	-0.5V – 5V	*2.5*mA typ.
Bluetooth	High performance to Arduino with 13 pins w/ 16-bit timer memory	3V-5V	15mA

Table 3.6: Component voltage & current requirements

3.4.9.5.3.2 Switch

In a design that we are using two types of power for the equipment, we need to implement a switch that will allow us to adjust the power from one source to the other. The switch can be manual or automatic, its roles will be to accommodate the electrical energy from the power supply to two diverse sources that are going to use by the printed circuit board. The first power supply will be connected to the wall to use the alternative current AC current then the transformer will down the voltage and convert it to direct current DC voltage. The second power supply source that will provide the circuit is the battery where we are going to utilize the battery as the backup power, or we can use it the second source of power that will provide the smart chair automatically when there is no power. As you know, the switch can be manual or automatic, it is better to use both, but it is going to cost a little extra money for the components that we will be using.

My group has decided to use a manual switch to make it easier for the users. We really want to use automatic switch as long as the battery is full of charge, the board will not use the power that has provided by the wall. For the safety of the users, we will use the manual switch just in case of something happened in the board, the users can use the backup power by the switching the manual switch that will provided outside the box under the chair. This switch can be controlled the way of the current flow in a printed circuit board that will be able to turn ON and OFF whenever the Smart Chair is not in movement or no one uses it. Also, the manual switch will be able to use whenever an electric shock happened because it will help people to close the components that use in a latch of electric circuit. Finally, we are using manual switch on the Smart Chair

because it is very important in an electric circuit specially in a printed circuit board that will be closed inside a box [39].

3.4.9.5.4 Design software

A major aspect of our project was the design and layout of PCB. In order to design our PCB we first needed to have a schematic showing how our components connected and situated in relation to one another. This ensured our ability to maximize the space used on our PCB, while simultaneously minimizing the cost of our printing the board. Below we will discuss the development processes and environments for designing our schematic and subsequent PCB design.

Eagle

An *Autodesk* application, eagle offers the ability to design both a schematic and PCB in tandem. Off the bat, this reduces design and development time as we can do clearly see how the digital circuit will look while, weighing the concurrent effects of placing these components on a PCB. Component placement on the PCB affects several factors such as efficient use of space, complexity of traces, and cost of components. These in turn, affect several constraints of our design, if they themselves are not already such.

This software is an electronic design automation (EDA) program which enables us to sync schematic diagrams, component placements, PCB wire routings, and comprehensive library content into a build which allows us to construct a printable circuit board. Constructing a PCB on EAGLE is quite simple given that the system allows you to import device libraries with the file extension of. LBR into your design so that you can implement a circuit board design for practically all applications. Also, if a library is currently not available for a device, one is able to create a rendering of the electrical device within Eagle and store it as its own library. With this, no matter the complexity of your project, we are able to implement our design with ease.

Once the circuit schematic has been created, files are stored in the. SCH extension which makes it easy to transport the build project and will allow us to open the project in another workstation. This was very beneficial because it allowed us to share the work of designing the PCB because we could each share the .SCH file and make adjustment to the design, save our progress, and forward it to the next teammate to continue working on the project. Once we finished our schematic, the next step was to create a board layout. The board layout is the file that PCB fabricators use to print the PCB designs.

To create a board layout file which has a. BRD extension, we simply used our schematic file to generate a board layout file. Upon doing so, all our electronic devices are imported into the board layout file. From here, we could reposition our components on the board layout so that we avoided 90-degree wire connections because it could cause problems with the design. We also placed our components in such a way so that we can minimize the overall size of the PCB. With technologies improving exponentially, the goal has always been to minimize the area consumed by electrical devices. For this

reason, we aimed to make our PCB design as small and compact as possible. Furthermore, using eagle can make the job easier for us because it gave us the possibility to order all the electrical and electronics components that we were using in the project at a low cost. Eagle also offers an industry standard for components, as they were typically vetted, through various libraries and often provided by the manufacturer. This offers better prototyping as the dimensions of components come with a higher degree of certainty. Speaking more on prototyping, being that eagle is more widely used, when it comes to printing our board, and subsequent iterations, we can share the actual eagle file with the board manufacturer. This helps in the event of a small change being needed, the manufacturer may be able to support this rather than going back and forth on the design, or worse yet, receiving a printed board with nonfunctional components. In subsequent sections we will see the schematic and PCB design [40].

OrCAD

The first software that come to our mine for the printed circuit board PCB was OrCAD. It is free for student and some companies are using it as their main software to design a PCB. After Download the software, it seems really hard to use but it is very exciting and commanding. OrCAD has a lot of main parts that can help with a design through interception to completion where you will be able to get it done easily as long as you really know what you doing, Using OrCAD will help you with the library system because it is different from eagle library [41]. Eagle library requires you to write the name of the components exactly the same way as it looks in the library storage which is really hard to find. OrCAD library is simple and support about 32 copper layers which is used for the auto router that calculate so many tracks for the PCB that connect all the pins. our group uses Eagle because it is free for all students and we learned it in Junior Design class. We have a better idea of how to design any PCB from the schematic and the board layout. The biggest advantage that make us using eagle is the fact that each student has it download in their laptop and you can find it in almost all computers at University of Central Florida.

Easy EDA

Easy EDA is a web-based application, specifically used for schematic design. This can be used for quick prototyping [42]. While not as widely used as eagle, this application offers the use of user submitted components. While some may be unverified, this can offer a certain degree of clarity in design for components that cannot be easily found through eagle’s libraries. Ultimately for our project, having members familiar with eagle, we chose not to use this application.

Final Decision

Ultimately, we chose to use Eagle for development of our schematic and PCB design. It offered, more familiarity as well as more comprehensive library selections with respect to parts.

3.4.10 Schematic Overview

Based on our preliminary research and hardware selections we have come up with a schematic overview of the system. this illustrated how we intended to connect our various components including sensors, Bluetooth module, power supply, and several I/O connections. This is illustrated in this section followed by a description of what is presented.

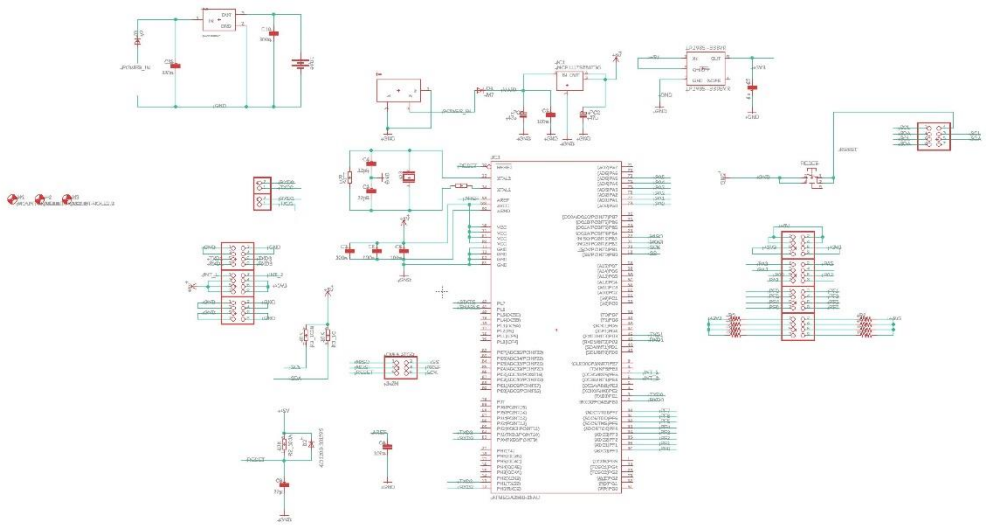


Figure 3.5: schematic overview

3.4.11 Labeling Sensors

Utilizing an array of sensors can become a daunting task when it comes to remembering which sensor corresponds to what pin on the microcontroller, or which sensor is being referenced in a line of code. For our application the designers have come up with a zoning and labeling system for the array. For conceptualizing and explanation purposes, let us establish a point of view. On an *xyz* plane where z is out of or into the page, and x is left and right, y up and down, we place our chair at the origin (center of the grid) and the front half of the seat lies in quadrants I and II (from cartesian coordinate quadrants), and the rear half lies in quadrants III and IV.

In each quadrant there is one *A502* sensor and one *A301* sensor. now that the view point and quantity has been established we can create a labeling scheme. The larger sensor is labeled for its zone, while the smaller is labeled S, this removes confusion as to which sensor we are speaking of in each zone as there are only two, one of each type.

As for labeling the four zones, we have chosen FR, FL, BR, BL which correspond to front right (quadrant I), front left (quadrant II), back left (quadrant III), back right (quadrant IV). Two examples of the labeling scheme would be as follows: FR1→ front

right region, sensor 1 (4x4 inch sensor), RL2→ rear left region sensor 2 (0.375”diam. sensor). the table below lists all the entire labeling scheme devised and will serve as a reference throughout or design and development process.

Sensor name	Region	Size	Quadrant
FR	Front right	4”x4”	I
SF	Right	0.375”(diam)	I
FL	Front left	4”x4”	II
SL	Left	0.375”(diam)	II
BL	Back left	4”x4”	III
SB	Rear	0.375”(diam)	III
BR	Back right	4”x4”	IV
SR	right	0.375”(diam)	IV

Table 3.7: Force Sensor labeling

3.5 Software

In order to meet the requirements of the project, the developers needed to create a mobile application that paired with the physical chair. A database was necessary to preserve user data over time. The application therefore had to be able to communicate with the database and with the PCB on the chair. Outlined in the following sections are the technologies considered for each category and their compatibility. Decisions involving which technologies used in the development of the application were derived from the data compiled below.

3.5.2 Security

The consideration given to data security in application development has increased over the years. The main cause of that heightened awareness being the long line of security breaches that can compromise information ranging from passwords to credit card information to social security numbers. Ensuring that an application is secure is a major concern in software engineering. According to Veracode, having strong security measures in place can improve a company’s reputation, thereby winning over more customers, and can prevent the legal and financial consequences of breaches [43].

Discussed below is a summary of the measures that Upwork recommends using during development to write a secure application [44].

- Regarding the portions of the application that will reside on the user’s mobile device:
    - Protect the application code and the API with encryption. If the source code is readable, someone with malicious intent can examine it for weaknesses.
    - Test the source code for vulnerabilities.
    - Design code to be patch-able. Make it easy to fix security vulnerabilities after a breach.

    - Find the balance between the best security and the best performance.
    - Just because an app store says that the app is secure, does not mean that it is 100% invulnerable.
- Security measures for the back end:
    - Consider using containerization to encrypt back end storage.
    - Have penetration tests performed to pinpoint network vulnerabilities.
        - This measure is probably beyond the scope of the project.
    - Encrypt the database and encrypt connections between the front-end and back-end using a VPN, SSL, or TLS.
- Use strong authentication methods like OAuth2, JSON web tokens, or OpenIDConnect.
- “The more data that’s stored locally on a device … the more vulnerable it is.”
- If the data must be stored on the user’s device, encrypt it.
- Avoid storing critical data on the user’s device (ie. credit card number).

SQL Injection

A topic that was not discussed in the Upwork article is SQL injection. W3 defines SQL injection as when a malicious user inserts SQL code into a text box that was not intended to receive SQL code. W3 also states that SQL injection is one if the most common ways that web-based applications are hacked [45].

SQL Injection Techniques

W3 discusses what techniques are used to hack an application via SQL injection. The article describes three methods to inject SQL into a text box [45].

1. 1 = 1

Injecting a constantly true statement like 1 = 1 can manipulate a select statement in such a way that rather than returning a specific row from the database, it could cause the database to return an entire table.

2. “” = “”

Inserting “”=”” into a username or password field could provide the hacker with a list of all of the usernames and passwords since “ “ = “ “ is always true.

3. Batched SQL statements

Entering an SQL command into a textbox that begins with a “;” can allow for the hacker to send batched commands. This is a way to send a second SQL command that is concatenated to the end of the first (and intended) SQL command. An example could be: ; DROP TABLE Students;

How to prevent SQL Injection

SQL injection can be prevented by using SQL parameters. SQL parameters are safer than inserting a variable into an SQL command because the SQL engine will check to see if the parameter data matches what the column is expecting. SQL parameters are characterized by the @ symbol [45].

XXS Attacks

XXS attacks (Cross-site scripting) are different from SQL injection in that these attacks do not damage the application, but the user. XXS attacks are possible when the application uses client-side scripting and when the application dynamically uses user data without first validating said data. If a XXS attack is successful, the hacker will gain control of the user's browser.
In order to prevent XXS attacks, it is important to validate input data and to encode output data [46].

Storing Passwords

Password storage can prove itself to be a major vulnerability in applications lest the developers take steps to protect user's information in even of a breach. Major websites have made the mistake of saving passwords as plain text. On March 21, 2019 Facebook released a notice that user passwords were stored in plain text which made them readable to Facebook employees [47]. Protecting user data is a high priority in application design. In order to protect users of the project's application, it was important to the developers to study methods to protect privacy in event of a security breach.
According to a GeeksforGeeks article, storing passwords in plain text is the least secure option [48]. There are several ways to improve the protections for passwords by way of making it harder to recover passwords if a database is accessed illegally.

Hashing Passwords

Hashing passwords is the more straightforward method of protecting passwords. The idea behind hashing passwords involves a hash function that maps to the user's passwords. The hashed version of the password is what gets saved on the database.
But hashed passwords are still vulnerable. Hackers can use a rainbow table to work around hashed passwords. A rainbow table is a table of all of the different hashes of a dictionary of words. The hacker can match the hashed passwords in the database to the rainbow table and derive the passwords from there [48].

Salting

We can improve upon the hashing method described above via salting. Salting is adding an extra string like '%67@#_a' to the end of a password, then sending the concatenated string through the hash function. This improves the security of the

passwords because the hacker would also have to obtain the salt in order to retrieve passwords.

Using dynamic salting is even better than static salting. Dynamic salting involves generating a new salt for each user. The dynamic salt is added with the static salt to the password before it is hashed. Even though the dynamic salt would have to be stored in the database, it is still another step toward more secure password storage. In order to retrieve a password, the hacker would have to get both the static and dynamic salt, then generate new rainbow tables. The salts will increase the time it takes for these rainbow charts to generate considerably [48].

Bcrypt and Scrypt

Even though using dynamic salting will go a long way to slow down a hacker, GeeksforGeeks still recommends using Bcrypt or Scrypt to protect passwords [48].

3.5.3 Framework

When choosing a framework for this project, the project members had a set of primary concerns that directed the research. Those primary concerns were:

- Framework must be low-cost, preferably free.
- Framework must have ample documentation and tutorials available.
- Framework system requirements must include both developers' computer specifications.
- Framework must support needs of the project.

Described below are the best options for the project, given the constraints described above. Due to the impact of a framework on a project, the developers determined that compilation of a list of framework options and the use of careful analysis and elimination of frameworks under consideration would provide the due attention necessary to ensure success of the project.

IOS-Exclusive Development

To develop for Apple devices, one must use Apple's IDE, Xcode. Apple is currently on Xcode version 10. The coding languages accepted in Xcode are Swift, C, C++, and Objective-C. Xcode supports three main version control softwares: GitHub, GitLab, and BitBucket. Xcode provides built-in simulations for code, so developer's will not require an Apple device to test the software's (isolated) functionality. However, since neither of the two developers have Apple devices, this will strain resources during the integration phase. According to development forums, developing for Apple devices is particularly difficult since there are a lot of regulations on what your application is allowed to do and documentation is hard to come by [49 - 50].

System Requirements

- **Windows**
  - **Xcode not available for Windows operating systems. Will require virtual machine.**
- **Mac**
  - **Intel i5 or i7**
  - **4GB of RAM (minimum)**
  - **128 GB of storage (minimum)**
- **Linux**
  - **Xcode not available for Linux operating systems. Will require virtual machine.**

Android-Exclusive Development

The development environment for the Android operating system is Android Studio. Extensive documentation available on Android's website [51]. Android Studio supports the following programing languages:

- Java (all of Java 7, only subset of Java 8) (preferred)
- Kotlin (preferred)
- C/C++ (libraries only)
- Python
- Corona

In the previous iteration of this project, *Posture Perfect*, the application was developed for Android in Android Studio [3].

System Requirements

- **Windows**
  - **Windows 7, 8, or 10**
  - **3 GB of RAM (minimum)**
  - **2 GB of disk space**
  - **1280 x 800 screen resolution**
- **Mac**
  - **Mac OS 10.10 to 10.13**
  - **3 GB of RAM (minimum)**
  - **2 GB of disk space**
  - **1280 x 800 screen resolution**
- **Linux**
  - **GNOME or KDE desktop**
  - **64-bit version capable of running 32-bit**
  - **GNU C library 2.19 or later**
  - **3GB of RAM (minimum)**
  - **2GB disk space**

- **1280 x 800 screen resolution**

Cross Platform Development Environments

Instead of writing an application that would be native for either Android or IOS, an app can also be written to be platform independent. Cross-platform frameworks are becoming increasingly popular in the software development sphere. This category of frameworks is known for its cost effectiveness, time saving, and ease of maintenance provided by a single codebase. Described below are the most promising options for cross-platform frameworks to be discussed in detail.

Flutter

Flutter is a new development tool that was released by Google to develop web frontend and mobile application. Being a Software Development Kit (SDK) from Google, this also comes with complete framework, widgets, and tools to allow developers to create applications that can be easy to build and deploy visually pleasing, quick, and smooth mobile apps that can be used for Android and iOS [52].

One of the many benefits of using Flutter as compared to another SDK, is that it can be cross-platform, being able to run on both Android and iOS, instead of separately. Having to run on one codebase rather than multiple SDK, makes it more convenient and saves time for developers. Having the utilities already built into the system provides very accessible tools such as widgets as customizable design for developers. These beneficial functionalities can be broken down as follows:

- Object-Oriented Language that is based on **Dart**, making it easy to learn
- Built-in Widgets, that can be drawn from its own high-performance rendering engine
- Widgets that are fast, visually pleasing, and customizable
- Ability to create own custom app design, with UI elements available
- Architecture based on React, simple and reactive programming

Pros:

The advantages of using Flutter, not only for developers but also users as well, since it speeds up the development process of the mobile application, while also reducing the cost of production [53]. Development teams would greatly benefit from building an app with aesthetically pleasing UI with smooth animation.

- **Code Writing**
  - Mobile Application Development will be faster and dynamic
  - Making changes are more seamless through the process known as **Hot Reload**

    - **Hot Reload** is the process of editing code by updating code into source file while the Dart Virtual Machine stays running [54]
    - Speed of Hot Reload is in the milliseconds to run and test for development and experimenting
    - Designing application made simpler and more comfortable for development process to improve for both designer and tester
    - Less time and building, unlike other native app development which requires rebuilding to retest and experiment the application which can take minutes

- **Multiplatform**
    - Codebase can run on Android and iOS on the same code development
    - Does not need to depend on platform, such as iOS requiring MAC to develop apps
    - Built in library and widgets to create designs for application

- **Testing**
    - Multi-Platform allows developers to save time from running on different system
    - One codebase for an application will work the same on different platform
    - Process Quality Assurance is faster to maintain the level of desired quality of product for close attention to detail on every stage in development

- **Speed**
    - Functions in smooth and fast working environment
    - Seamless transition of mobile application to hanging and cutting while scrolling
    - This section can be further explained in Flutter Technical View

- **Design**
    - Creating widget is made easier with library and customizing existing ones

- **User Interface**
    - Flutter can work with older platform while maintaining functionality of newer supporting systems

- **Minimum Viable Product**
    - MVP is the conceptual process when new product development learns of consumer needs and what requires satisfaction through feedback or collective status

Cons:

The disadvantages of Flutter are that since it is new in the developing environment, support is not very abundant for this Software Development Kit. While many libraries are yet to be implemented to Flutter, features are not fully available like any other SDK to develop applications.

- **Libraries and Support**
    - Not every functionality has been featured to Flutter as compared to native development
    - Developers would need to mainly use Google support in order to fully utilize beneficial libraries
    - Even without having full support of libraries like native development, Google has full libraries necessary to develop applications
- **Lack of Continuous Integration Support**
    - Flutter is not widely known compared to other continuous integration platform due to being primarily Google SDK
    - Custom scripts will be necessary to create application using Flutter because of libraries that might not exist at the time for development

Development of mobile application connects many technologies together, with Flutter, the advantages outweighs the disadvantages since it can well support business and development teams rather than having the risk of lacking support. Developing applications with that functions with high-performance while maintaining to be aesthetically pleasing proves to be beneficial while it also runs faster than another SDK to develop such as Android and iOS platform.

Applications have a user end and developer end perspective in the platform that it is used from the native code to communicate between widgets, location, audio, camera, Bluetooth, sensors etc. while keeping in mind the cross-platform nature to run on developing architecture. Mobile applications functions by connecting frontend code to API which communicates with the database, similarly web development runs on similar process. Having cross-platform application is complex since other framework utilizes different mobile and web technologies, with the first cross-platform that was capable to engage were based off JavaScript and WebViews that can now be found on numerous families of framework such as PhoneGap, Apache, Cordova, Ionic and many more.

Viewing applications run with HTML, the structure of the display of the code, which can be then shown on WebView by having it connect with JavaScript to operate and make viewing the application all possible. HTML is what creates the page of the application while JavaScript is how it functions in code. Web development framework like ReactJS have been a favorable library become a main tool for designing web platform.

Flutter provides viewing very much like React Native by taking different methods to perform while avoiding problems when compiling the program from the JavaScript bridge. This bridge acts as the communication between services in native code to connect the realm of JavaScript and Native.

Flutter is Google’s cross-platform mobile development framework. It is free and open sourced. The platform is C/C++ based. The language supported by Flutter is Dart. Flutter’s main selling point is that the developer only has to code up the project once and the program can then run on multiple device operating systems (Android, IOS,

Windows). Offers stateful hot reload which allows developers to make modifications to the application while it is running. This speed up the debugging process. Flutter has a plugin (FlutterBlue) to use Bluetooth to communicate with a microcontroller (works for both Android and IOS).

A testimonial from Abbey Road Studios claims that they were able to get their application developed and released in ten weeks with a team of three people [55].

System Requirements

- **Windows**
  - **Windows 7 SP1 or later**
    - **64-bit**
  - **400 MB of storage**
  - **Windows PowerShell 5.0 or later**
  - **Git for Windows 2.x**
    - **Also needs the 'Use Git from Windows Command Prompt'**
- **Mac**
  - **64-bit macOS**
  - **700 MB of storage**
  - **Access to: bash, curl, git 2.x, mkdir, rm, unzip, which**
- **Linux**
  - **64-bit Linux OS**
  - **600 MB of storage**
  - **Access to: bash, curl, git 2.x, mkdir, rm, unzip, which, xz-utils**
  - **The libGLU.so.1 library**

Hot Reload (Flutter)
This feature allows experimenting on building UI, fix bugs, updating from the source code to automatically make changes thanks to the **Dart Virtual Machine** to make the process possible. By using an application made by Flutter editor or from a terminal line, this can allow modification of Dart files to making projects so long as the IDE or editor in used is capable of supporting Flutter. This is especially helpful on rebuilding widgets in order to save time and reloading the source code, Hot Reload shows visible changes made from automatic execution of the application. Hot Reload has a feature that preserves the state of which the application enables the design to be viewed with the most recent changes, this is known as **Stateful Hot Reload**. This feature is useful to maintain the current state while changes are made to keep desired behavior, an example of such would be having user login. Stateful hot reload modify changes without having to reenter credentials to login after several navigations of changes. Unless the code affects the state to the point of not being fully functionally app, then the behavior would go from hot reload to hot restart.

Changes made may not be visible to the UI immediately, this is usually rooted within the **main()** method where the root of the **widget tree** must be made to affect the UI with Hot Reload. Otherwise **Hot Restart** is executed to run the code from the beginning with the

new version of changes made in the main() method to display the widgets after modification are build. There are cases where hot reload will not be able to be applied due to constraints, these are normally cases such as changing enumerated types to regular classes or vice versa, these **limitations** are not supported by hot reload. The process of this includes the compilation of last edited code that is executed by hot reload are libraries from changed code, main, and any other affected library [54].

Widgets (Flutter)

Creating designs for applications can make appealing user interface that can also function in efficient interactive experience that communicate well with data management. Widgets can make apps accessible, take inputs, design layouts, responsive routing, scrolling, visual behavior, and display and style text [56]. Instead of giving information on the user screen with listing long and strenuous text that can be daunting to read, since the dawn of smart phones and even beyond, the idea of using icons to hold the data and app in place, allows simpler and organized access on screen while also saving battery life in the process [57]. Designing widgets can be creative and exciting that brings emersion for the user, some key useful features to use widgets are as follow:

Basic Widgets

- **Row** – this feature helps organize the display window of an application in horizontal view
- **Column** – like row, this also organize the display window but in vertical view
- **Text** – stylizing the font or writing of the text can make more easily readable layout
- **Icon** – this is a useful tool to create graphical drawn glyph to represent functionality to interact with the app
- **Button** – an on-display feature that lets user click to a different page or action

Layout

- **Container** – a form to position and combine effects of the surrounding with padding and aligning of the design, this could mean the parent display or outside background of an app
- **Padding** – the child display after a container, this creates the inner design containing frontal view of different functionality or actions of the app
- **Center –** position the dimensions of the constraints of the widget itself from either container or padding
- **Align** – positioning the child widget to match the dimensions of the parent
- **Aspect Ratio** – much like center and align, this gives the dimension of the app that matches the size of the screen that the app host

Anatomy of a Flutter Application

In order to successfully write an application with Flutter, the developers will follow several tutorials available online as part of the learning process. Summarized here will be the information gathered from those observations. This discussion will be based on the default demonstration code that Flutter provides [58 - 61].

1. Generate a new Flutter project. This new project will come with pre-written demonstration code. This demonstration code should be deleted before writing an application. The main dart file should begin with import statements.

Important and Useful libraries:

Library	**Purpose**
material	Provides Flutter widgets that are using Material Design
firebase_auth	Provides Flutter support for Firebase authentication
firebase_database	Provides Flutter support for the Firebase RealTime database.
swipedetector	Detects up, left, right, and downward swipes (to be used for navigation)
charts_flutter	Provides Flutter support for detailed charts and graphs.
progress_indicators	Provides enhanced loading indicators

Table 3.8: Flutter Libraries Relevant to the Project

2. The first line after the import statements begins with: void main() => runApp(...);. The '...' is to be replaced with the class that is used to create the application. This line serves as the entry point for the application. The main method calls the runApp method. The runApp method displays the widget (or class of widgets) that it is passed.
3. Below the entry point of the application is a class that is composed of multiple widgets. In the default demonstration code, this class is called myApp. A class can extend StatelessWidgets and StatefulWidgets. Stateless widgets are widgets who's values cannot be changed (all values are final). Stateful widgets are widgets who's state can change during operation of the application.
4. Within the demo code's myApp class, is an override of the build() method. This method is used to determine how the widget(s) should be displayed. The build method is overridden for building widgets who depend on the current state of the listenable (current state of the animation).
5. The overridden build method will return a new MaterialApp object. This object will contain attributes like 'title' and 'theme'. This object also includes an variable, 'home'. The home variable in the demo code creates a new instance of the MyHomePage class.
6. The MyHomePage class exists outside of the MyApp class. It extends StatefulWidget. This class's purpose is to hold the values provided by the parent (MyApp) so that those values can be used by a State widget.
    1. This class also contains a call to the createState method. The createState method creates a mutable state for the widget. This method should be

overridden by subclasses for the purpose of creating a new instance of the subclass's associated State subclass. For the example code, the createState method creates an instance of the user interface for the widget.

7. After the MyHomePage class is the _MyHomePageState class. This class extends the State class. The State class holds the internal logic for a stateful widget. The purpose of this class is to implement the user interface. Typically, a 'state class' like this one is used to hold information that is subject to change during the life of the widget.
    1. Within the _MyHomePageState class is an integer counter variable and a incrementCounter class. The incrementCounter class's purpose is to call the setState() method. The setState() method notifies the State when the state of the counter changes.
    2. Within this class is also another override of the build method. This occurrence of the build method is going to return a new Scaffold object. A Scaffold object, according to the Flutter documentation, will provide a basic layout structure with material design. This Scaffold object accesses some of the values of the parent, like the title, but also defines some new structure. After passing the value for the title, the sample code defines a centered body.
        1. The 'Center' serves as a layout widget that takes one child and centers it within the parent. Within the body, is a child that is a 'Column' widget. The Column widget will take its children and arrange them vertically. This widget has properties that can be used to control its sizing and alignment. In the demo code, the main axis alignment is set to center. The main axis is the vertical axis since the children of the column are stacked vertically. To control the horizontal axis, the cross axis property should be used.
        2. The children of the Column widget will be a text label and a display for the value of the counter. The list of children is considered an array of widget.
        3. Outside of the Center body widget is where the button to increment the counter will be. This button is defined as a floating action button widget that uses an onPressed property to specify the method to call when the button is pressed (the incrementCounter method from the _MyHomePageState class), a tooltip (a property that describes the button), and a child. The child of the button is an icon that places a '+' symbol on the button.

PhoneGap

PhoneGap is a cross-platform mobile development framework developed by Adobe. It is free and open sourced. PhoneGap uses HTML, CSS, and JavaScript for development. PhoneGap maintains access to native technology. It produces hybrid applications via a cloud compiler. PhoneGap supports access to most of a device's native features (ie. Camera) [62].

System Requirements

- **Must have Windows or Mac OS**

While other frameworks were considered, these four appeared to be the most likely to fulfil development requirements for this project. Below is a summary of the data.

	Android Studio	IOS	Flutter	PhoneGap	Appcelerator	Sencha Touch
Pros	IDE is compatible with three main operating systems.	Easier to work with than Android Studio.	Free & open sourced	free & open sourced	extensible & open	Cross-platform (mobile and desktop)
	Documentation easy to locate.	Big Audience	Fast development with Hot Reload	Cross-platform compatible	Cross-platform compatible	
	uses Java	Exclusivity sells	Cross-platform compatible	uses HTML, CSS, and JavaScript	supports 5,000 device and OS APIs	
Cons	IDE is difficult to work with.	IDE is incompatible with the developers' computers and phones (Mac OS only).	Isn't supported by web browsers	not good for graphics intensive applications; inconsistent rendering	support team is slow to respond	commercial licensing feature is confusing
	Not cross-platform compatible	The developers do not have experience programming on apple machines.	Young language; limited libraries	sub-par documentation	some rewrite needed for *true* cross-platform compatibility	limited themes

Table 3.9: Cross-platform Development Environments Pros & Cons

After evaluating the benefits and drawbacks of writing the application for Apple or

Android devices, the developers decided that while it is easier to program in Xcode, Apple does not support the developers' computer systems. The developers could still work in Xcode but would require virtual machines and an IOS device for testing. Ultimately, it would be more cost and time effective to use a cross-platform framework for the project due to the types of devices owned by the project team and the time constraints.

The developers eliminated Sencha Touch as a possibility due to the restrictions of the free tier [63]. Since the other frameworks provided additional functionality at no cost. Appcelerator was also not selected for additional discussion due to the extra code rewriting needed for the application to work on multiple operating systems. The developers will be on a strict schedule and re-writing the application to support multiple operating systems is not feasible.

Ultimately, the developers had to decide between Flutter and PhoneGap. Flutter was chosen as the framework for the project due to its speedy development and stateful hot reload which will drastically speed up development. Since Flutter was chosen as the framework, the application was written in Dart.

3.5.4 Important Ideas of the Dart Language

Dart is a programming language that can be easy to learn, compile shared codebase, productive, and ahead of time general purpose tool for developers [64]. The programming language itself is similar to objected oriented language such as C# or Java, also similar to JavaScript (vastly different from Java) that can be used to develop and connect to frontend to backend, meaning user interface and database. Experienced programmers of objected oriented language can Dart as a language much easier to comprehend and learn, and even for less experienced programmers can go forward straight into Dart alone with simpler syntax of code [65]. Aside from the ease of use with Dart, it can also be loose and strong typing, meaning unpredictable producing outcomes of results for loose typing or strict to rules of syntax compilation for strong typing, but both can be applied and simple to move between other languages through Dart.

Sharing codebase across other platforms is difficult, doing so requires portion of the codebase to be ported to other platforms. Dart simplifies the process of sharing codebase due to the nature of being capable compiling natively for both Android and iOS by using the framework for mobile application development tool through Flutter. Since Google supports cross platform, with Dart and Flutter developed by Google, the transition between other platform has been supported all while having the language based on object oriented. Cross platform does not only apply for mobile application, but web and other forms of development. Using Dart programming language through Flutter as the Software Development Kit (SDK), compilation runs ahead of time (AOT) to allow the application working natively and just-in-time (JIT) which develops for local testing [66]. The process of running Dart is much faster as it operates live as code is being changed and no longer having to reload with saving for changes to occur, this gives developers a much faster time to create applications.

Listed below are some of the fundamental ideas that describe how the Dart language functions. Understanding the main ideas that a programming language is built on is important for the developers to successfully transition from a similar coding language (in this case, Java) to the new language.

- If it can be placed in a variable, it is an object.
- An object is an instance of a class.
- Even an integer is an object
- Dart can infer variable type.
- Dart offers support for generic types.
- Like Java, Dart supports top-level functions and variables, static functions and variables, and nested functions.
- Dart does not use the Java keywords public, private, and protected. Instead, if an identifier starts with an underscore, it is private.
- Identifiers must start with either a letter or an underscore.
- If a variable is specified as dynamic, it is not restricted to a specific type.
- Dart offers built-in support for: numbers, strings, filename, arrays (lists), sets, maps, runes, and symbols.
- Dart numbers are either an integer or a double.
- Strings can be created using either single or double quotes.

Side Notes on Dart

Dart is an interesting language with syntax that the developers have not previously encountered. In order to improve the developers' comprehension of the code, included below are some of the new and noteworthy syntax encountered during project research [66].

Symbol / Syntax	**What does it do?**
=> (arrow)	This symbol is shorthand for functions that have only one expression. This is used to replace { return expression; }.
$variableName or ${expression}	This is string interpolation
var	A method of declaring a variable without declaring its data type.
??	If null operator

Table 3.10: Dart Syntax

3.5.5 User Data

In order to manage user data, the project required user sessions. The project needed new users to create accounts and for returning users to sign in. Accounts allow a user to transfer devices without an interruption in service (ie. Transfer to a new phone). In order to provide user sessions, local storage for the session data was needed.

In order to save posture data, the developers considered the use of a combination of an embedded database and a server-side database. The embedded database would reside on the user's mobile device. The server-side database would reside separate from the user's mobile device. The server-side database will act as a centralized database to service all users of the application. The embedded database will exist as an instance on the user's mobile device and will hold only that user's data. If an embedded database is not used, the application is to be in frequent contact with the external database which is costly for both the user (in terms of fees on data) and the developers (in terms of fees on access to the central database). The embedded database will synchronize with the central database when convenient (when the user is on wifi).

3.5.5.1 Local Storage

In order to achieve the objectives of the project's application, the developers needed to use persistent data. Persistent data is data that has been preserved after the processes that created, modified, or otherwise used this data have terminated. Within the project, the developers anticipate needing persistent data to store settings such as notification preferences, maintain sessions, and incremental data on the user's posture pending synchronization with a database [67].

Saving data on mobile devices can be done with either, internal, external, shared preferences, and databases. The options to store app data gives users and developers more resources methods of storage. Protecting privacy is an important factor in storing data, to prevent sensitive information from being widely accessible from other apps. Allowing other apps to access private information requires the consent of the user.

The benefits of having data being internally stored on local device is that the data can be private, making information less widespread across other apps. This makes other apps unable to access said data, to protect privacy of the user, making less direct access. Meaning that the system can provide a directory in the file system to privately store the data and organize the apps necessity. Files can vary from photos, videos, text document, music, downloaded files and such [68].

While the file stored is kept in the internal storage, uninstalling the app can also remove that save file from the data. This is one of the downsides of storing files in internal storage. This method can be prevented by storing cache directory of the save data into the file system of the device. Applications have private cache directory that functions to prevent lose data in case of the user wishes to remove the app [69].

Even though cache directory is a viable solution to losing data upon removal of the application, internal storage space can be limit on the device. With limited storage space, Android can delete portions of the cache files to recover space. It can be an option to rely on the system to clean up files, but uncertainties can persist since it is not known to the user on which data can be removed. Maintaining cache by the user is the more secure option so that they can choose which save data to maintain or remove.

To have the Internal Storage function, file objects must use reference to store data by using **FileOutputStream** within the code in order to allow the content to access data only from that specific application. After the data is being stored, accessing this requires internal file directory by using **getFilesDir()** method for the application to retrieve the data from storage. Creating an access point for the directory requires another method, by typing **getDir()** in the code to return reference. To provide file reading preference, the code must include Scanner objects to complete file reading method by using **openFileInput(filename)**.

Read/Write Files

Reading and writing to files to save persistent user data is a very simple method. The path_provider plugin is a platform-independent method to access common app storage locations on a variety of mobile devices. This plugin provides access to both the cache and the document directory. Using this method to save temporary user data is easy and there is a comprehensive tutorial available via Flutter to facilitate incorporating the method into a Flutter application [70].

SQLCipher

SQLCipher appears to be very similar to SQLite. It is based on an SQLite database but adds encryption and allows migration from and Android database. Encryption can be implemented on an SQLite database via an extension, the Encryption Extension. Since the developers are also not migrating data from an Android database, SQLCipher seems like it is not the best fit for the project [71].

SQLite

While having a database to store massive data helps in the long run, there is the option of SQLite Database, which are more application specific online storage [85]. Since SQLite is fully available and functional for apps to a lesser degree as compared to a full SQL database, it is required to have knowledge of SQL in order to use SQLite. For mobile application development, using SQLite full features necessary with power and speed as an option for Android and iOS while SQLite also has features of Data Binding.

SQLite is a local database that provides better performance than reading and writing to files. It is free to use and is platform independent. SQLite use in a Flutter application is supported by the ‘sqflite’ plugin. According to SQLite, it has several advantages over static file read/write [72 - 75]:

- Better performance
    - 35% faster than file read/write
    - Load what you need, rather than loading and parsing entire file
- Portability
    - Compatible on all x32 and x64 operating systems (including both little and big endian).
    - Multiple processes can read/write to the same file simultaneously without interference.
- Reliability
    - Continuously updated to prevent data loss
    - Fewer bugs
- Accessibility
    - Data tends to last longer than the application itself.

SQLite is in the public domain, therefore free to use.

Shared Preferences

Another method of storing data is by allowing the user to enter key or value pair of primitive data types, such as Boolean, Float, Int, Long, or String, which are written in XML (eXtensible Markup Language), a language similar to HTML for online storage, with these files that exist throughout multiple sessions. Shared Preferences is an API (Application Programming Interface) that stores information of the user from an interface that access and modify the data of keys and values by hashing, strings of characters that are changed to a lesser and shorter length of data that holds the key and value representing the original stored data. In the case of mobile devices such as Android, the xml file that was transformed can be stored in private directly, while applications can have multiple Shared Preferences of file storage location.

Shared preferences is a method for storing local data in an Android application in <key, value> pairs. The plugin that would permit this kind of storage for a Flutter application is shared_preferences. This plugin would work for both IOS and Android, but not any other operating systems. Therefore, the developers do not anticipate this method of local storage to be the best solution for the project requirements.

The developers decided to use an SQLite database to manage local storage due to its performance advantages over static file storage and its continuous updates that protect against data loss [76].

In order to synchronize the embedded database with the central database, a web service will be required.

	Read/Write Files	SQLCipher	SQLite	Shared Preferences
Pros	Simple and common	Encryption	Cross-platform compatibility	Easy
	No plugins needed	SQL-based	High-performance and reliability	Lots of resources to use for guidance
Cons	Slow	Uncommon choice -> limited tutorials	Requires plugin to work with Flutter	Not fully platform-independent
	Unsecure		Requires plugin to add encryption	Requires plugin to work with flutter

Table 3.11: Local Storage Summary

Creating a location for Shared Preferences requires an object that the context of the method can retrieve that object representing the Shared Preferences [85]. This can be done do by typing **SharedPreferences = getPreferences()** object into the code. In the application of Shared Preferences, multiple files can exist and getting the object needs accessing the editor using the **edit()** method to add value and use **put()** method of any data types between Boolean, Float, Int, Long, or String, along with being able to remove the key or value by using the **remove()** method.

3.5.5.2 External Storage

Like internal storage, external storage can be used outside of the mobile application or internal storage of the device. This can be done by storing on a computer, or removable SD card and such. Every Android device is capable of sharing data by transferring to external storage and transferring files is also a viable option, such as USB connection to load/copy files from mobile device to a computer. Due to the nature of external storage, it does not guarantee accessibility after the stored data is transferred, since it makes a copy of the file in the system.

Save files in the system maintains on the device while it is active and stored, removal of the application deletes any old files. The files are no longer accessible after the application has been removed from the device. Unlike internal storage, the benefits of having external storage is having a backup of the data, making accessibility of the backup files can be restored once the device re-acquires the application. This is beneficial for the user in the case of having limited storage space in the device, unless external storage can be placed into the device such as SD card to extend space.

Extending storage such as SD card is very favorable to acquire more space, and the installation and removal of the SD card is simple and feasible method of external storage. Relieving the user of concern from storage, since the save data in the SD card can be easily plugged in onto another device.

Creating a filesystem for External Storage is similar to Internal in terms of the development environment, with the difference that External allows removability of the storage device [85]. Providing removability of the device gives External Storage the ability to share stored data to other devices and can be read by any and even all applications.

In order for files to create a directory to allow external devices to store data, **Environment API** must be used to make this possible. The choice of having files be exposed or secure can be given as an option for the user, while developers implement such files on external storage to be saved on private or public space. Using private space allows the user to be able to delete when the app is uninstalled by using the method **getExternalFilesDir()** within the code, while public on the other hand allows accessibility to all files which can be scanned by using the method **getExternalStoragePublicDirectory()** implemented within the code. This gives External Storage more flexibility than compared to Internal Storage, since directory requires parameter to be parsed by the method using **getDirectoryType()** to results a return. Directories are not always set by default, to create location in the storage, the device must be called by implementing the method **mkdirs()** before beginning to save files [77].

3.5.5.3 Database

Aside from storing saved data on devices and having to transfer files to another device physically by connecting a cable from mobile to a computer, or an SD card to another separate device, a more convenient method to store data would be having a database. Unlike the Internal and External Storage, the use of Database allows accessibility across multiple platform and devices at once. Instead of making copies of save files and transferring to another device, the database can automatically transfer the data whether it can be stored on one device or many, this can make the user experience simpler and less tedious [78].

The usage of database creates more user friendly and uncomplicated accessibility due to connection of online storage. Android is capable of supporting SQLite databases for storing data, while databases tend to be specific to the app to avoid data to cross over apps without the user's consent. What makes SQLite beneficial for the developers and users, is the power and speed that comes in full features within a database in correlation to the application.

After deciding on which framework to use for the project, the choices for a database management system were narrowed down to those listed and discussed below. The primary goals when selecting a DBMS were:

- The DBMS must be low cost or free.
- The DBMS must work for both IOS and Android.
    - The purpose of this concern is to prevent the developers from having to create and manage two databases (one for Android, one for IOS).

- The DBMS that will function as the embedded database must be lightweight.

3.5.5.4 SQL

A relational database is a database defined by its collection of related tables. Each table has a primary key, and each value within the table is a <key, value> pair. The language used by a relational database is SQL (Structured Query Language) [80 - 81].

3.5.4.4.1 Database Normalization

Database normalization is a technique used to eliminate redundancy in database design. Redundancy can cause a wide range of issues if not removed from the design. Redundancy can cause insertion, update, and deletion anomalies that invalidate data. Redundancy can also increase the size of the database unnecessarily. TutorialsPoint provided an in depth explanation on the 4 main forms of database normalization [80].

Requirements for First Normal Form (1NF)

- Determine which data values are needed. Define the type of data that will be stored for that value (ie. Integer or string). Then group relating values (columns) into a table.
    - Note that each column should only hold one value.
- There should be no repeated groups of data. If a table that should hold transaction information also holds the customer's address, name, and email the table needs to be split into a customer data table and a transaction records table.
- Each table must have a primary key.

Requirements for Second Normal Form (2NF)

- The tables should meet the requirements for 1NF.
- There must be no partial dependencies on the primary keys.
    - Example: Still using the transaction idea, if the transaction data table holds a customerID, customer address, orderID, and order details assume the customerID is a primary key for the customer table and the orderID is the primary key for the order table. The customer address is derived from the customerID. The customer address should be stored in the customer table so it is redundant to have that information in the customer/order table. Since the orderID is the primary key for the order table, it is also redundant and in violation of 2NF rules to have the order details located in the customer/order table.

Requirements for Third Normal Form (3NF)

- The tables should meet the requirements for 2NF.
- All transitive dependencies need to be removed.

- A transitive dependency is when columns b, c, and d contain data that can be derived from the data in column a, but columns a, b, and c are included in a table with other attributes. In order to remedy this, create a second table for b, c, and d with a as the primary key. Then the original table should only contain a and the other data.

Boyce-Codd Normal Form (BCNF)

Boyce-Codd Normal Form is a stricter version of 3NF. According to StudyTonight, a table is in BCNF if: For X -> A, X is a superkey.

3.5.4.4.2. MySQL

According to Oracle, MySQL is the world's most popular open-source database [81]. The project member responsible for the database management has previous experience with MySQL. MySQL can work with Flutter applications via drivers.

According to Datamation, MySQL has the following advantages [82]:

- Data Security
  - MySQL is known for its reliability and data security. It is the DMBS of choice for many popular websites and is frequently chosen for products that perform a high volume of transactions.
- Scalability
  - Datamation claims that MySQL's scalability is 'unmatched'.
- High Performance
  - MySQL provides superior performance with its high transactional speed and indexing.
- Up-time
  - MySQL offers '24/7' uptime.
- Transactional Support
  - MySQL provides strong row-level locking to prevent deadlocks.
    - For most projects this should be a primary concern, but there will probably not be a lot of row modification within the application for this project.
- Low Cost
- Open Sourced

For Establishing MySQL Connection

In order to establish a connection to a MySQL database via a Flutter application, the plugin, mysql1, will be used. The plugin serves as a driver for MySQL that is compatible with the Dart programming language [83].

How to Connect to a MySQL database via Flutter:

1. Enter the database connection details into a new ConnectionSettings instance.

2. Use the connect() method to establish a connection. This method returns a connection object that needs to be stored in a variable.
3. Use the query() method to send a query or update to the database. This method will return the SQL results and they need to be stored in a variable.
4. Use a for loop to parse the data in the object query() returned.

3.5.4.5 NoSQL

NoSQL is a non-relational database management system. Non-relational databases were designed to cope with modern development trends and to provide higher performance. NoSQL databases are also more scalable than their relational counterparts and have dynamic schemas. Dynamic schemas allow the developer to alter and redefine the schema as the project progresses. According to the MongoDB website, there are several types of NoSQL databases: document databases, graph stores, key-value stores, and wide-column stores.

Document databases involve pairing a ‘key’ with a ‘document’. The document can contain key-value pairs, key-array pairs, or another document.

Graph stores provide data storage in the form of a network. This type of database would work well for networking applications (like Facebook).

Key-value stores rely on the key-value pairs. Values can have variable types. This is the simplest NoSQL database.

Wide-column stores are unique because they store data by columns rather than rows. This type of database is best for handling queries when there is a lot of data.

The NoSQL database that is provided by Firebase is the recommended NoSQL companion to a flutter application.

Of these four NoSQL database types, it is most likely that the project would require a key-value store or a document database. We would not need the graph stores as the application data would not be networked together. The wide-column store would not be a good fit since our application will not be performing massive and frequent queries.
In general, it looks like a NoSQL database is more than we will require for the nature of this project. Since a relational database will probably fulfil all of the development needs, and since one of the developers has prior experience with MySQL, it is unlikely that we will be using this technology [84].

3.5.4.5.1 RealmDB

It is important to note that RealmDB is not free. The lowest tier of service should be sufficient for this project, but at the price of $30 per month.

According to RealmDB's website, the following are their selling points:

- Offline First
    - The RealmDB has an embedded and central database model that self-synchronizes when the mobile device has signal.
- Cross-Platform compatibility
- Can be used as a 'RESTless' middleware
- Realtime data synchronization
    - Supports reactive applications
- Edge computing
    - Permits developers to cache data anywhere.

Currently, there is no native support for RealmDB with the Flutter SDK. In 2016, a RealmDB developer claimed that it is unlikely to be supported due to the 'substantial' nature of the work that would be involved. Due to this roadblock, RealmDB is probably not a good DBMS to use with Flutter [85 - 86].

3.5.4.5.2 Couchbase Lite

Couchbase is a NoSQL database management system that provides support for SQL-based queries. The Couchbase website has a page on pricing, but no prices listed. A quote is available upon request. Couchbase Community Edition is free to use [87].

According to Couchbase, the perks of their mobile database include:

- High availability and disaster recovery
- Security: Authentication and Authorization
- Performance and scaling via adaptive indexes
- Embedded-central synchronization via web sockets
- On-device encryption (Enterprise edition only)

There is a plugin called 'Fluttercouch' that offers compatibility between Flutter applications and Couchbase mobile [88].

	SQL	**NoSQL**	
DBMS	**MySQL**	**RealmDB**	**Couchbase**
Pros	Familiar to developers	Offline-first synchronization	High performance and availability
	Free & open sourced	Realtime data synchronization	Embedded-central synchronization

Cons	Rigid Schema	$30/month for lowest-tier service	On-device encryption and other features restricted to enterprise edition
	Must be hosted, which will add to project cost	No support for Flutter SDK	Enterprise edition is costly

Table 3.12: Central Database Summary

Due to the compatibility of Flutter and Google's NoSQL database, Firebase, the developers decided to use a NoSQL database for the application. The developers used a NoSQL database as the central database and attempted but did not execute an embedded database. While an embedded database would have reduced database reads, it was determined that this method caused the application to be unable to consistently refresh data. Firebase will be discussed in section 3.5.6.

3.5.5 Database-Application Interfacing

Each combination of technologies has different requirements in order to communicate. Some combinations are already patched together via open-sourced plugins, while others require the developer to write an API (application programming interface). A smaller selection of technologies are designed to work together, requiring little to no effort from the developers. In the table below are the possible combinations of technologies and if and/or how they can be connected. While it is important to select the best technology for the intended purpose, if the database cannot communicate with the application it will cost the development team a lot of time. It is best to avoid a combination of technologies that are not meant to work together or appear to have a significant workload associated with getting communication established [89 - 100].

	MySQL	SQLite	SQLCipher	Read/ Write Files	Shared Preferences	RealmDB	Couchbase

Flutter	PHP	Sqflite package	Flutter_sqlcipher package	Path_provider plugin and the dart:io library	Shared_preferences (IOS and Android)	Not available - would require custom plugin	Couchbase-lite-flutter (under development)
Android	Content Provider /PHP	Must be custom written	android-database-sqlcipher	DeviceFileExplorer	SharedPReferences interface	Custom API needed	Self-contained within couchbase

Table 3.13: SDK & Storage compatibility

The API/plugin will provide synchronization between the embedded database and the central database. Since the developers decided to use the Flutter SDK and Firebase, Google's libraries will be used to initiate and maintain database connections.

3.5.5.1 Data Binding

The ability to use SQL allows fetching data from online and delivered to devices, making less effort for the user to retrieve data. Having databases using SQL, uses the technique known as **Data Binding**, which is the process of retrieving date by fragments from the location of the element in the UI and matching the element in the database. Data Binding provides seamless usage of library in the development stage for efficiency and ease of use for mobile application [101].

- **Support Library**
  - Data binding support its own library that is available to be use for Android 2.1 (API 7) platform to the latest

- **Data Binding Layout**
  - Configuration is required to perform this layout files
  - Files are auto-generated, otherwise without it then it would be needlessly generating random file layout
  - This technique allows layout files to use layout root tag
  - Providing layout root tag allows UI view that can be indicated in the build system
  - Layout root tag allows files to be processed for data binding
  - Without layout root tag cannot be processed for data binding

- **Data Binding Activity**
  - Capabilities of layout file that utilizes data binding loads layout in different way
  - Class that utilizes data binding auto-generates layout file creates default class name that are already set
  - Default names of layout file are capitalizing first letter of each words followed by underscore
  - Code runs faster with data binding because it traverses view hierarchy upon calling for specified view
  - Layout traverse in single runtime because of no more type casting returned views

- **Binding Objects**
  - Binding user object to layout file to assign proper fields directly from layout file
  - Doing so does not require activity to perform when User Object display properties that corresponds to specified fields, such as finding each specific User object
  - Activity to bind objects performs by assigning fields as whole rather than singular field object

- **Data Objects**
  - Can be Java or JavaBeans object that express the evaluated text attribute
  - Objects such as User can bind to automatically generate the Binding class
  - Variable of the layout data such as the user will be generated by creating and binding the object in the Activity (Data Binding Activity)

- **Binding Integers**
  - Having integers can be used in many ways for the user, such as age, weight, height, contact, files, text etc.
  - Data that contain text generally do not accept integers, but can be done by using methods into a string
  - Saving data into string can store the data of a text or integer so that it can be converted to desired output in the layout file

- **Imports**
  - Data Binding library allows importing classes into the layout file
  - Java allows importing by referencing layout file from a string that is created from a method

- **Data Binding Expressions**
  - Special characters can be complicated to convert into, but storing it into string allows expressions to be stored

    - Java allows expressions to be included which is readily available with useful operators

- **Updating Data Binding Objects**
    - Objects can be displayed by listing and mapping layouts to applications
    - Updating objects does not usually affect the UI, in order to have objects reflect its update into the UI, this requires the use of **Observable Fields**, **Observable Objects**, and **Event Handling**
    - **Observable Fields** – allows accessor methods without needing values directly accessed
    - **Observable Objects** – requires implementing objects to the interface known as **Observable**, or extending it by using **BaseObservable** class
    - **Event Handling** – planning, setting, accessing events by using Method references or Listener bindings to target methods

3.5.6 Firebase

Firebase is an SDK (software development kit) with numerous capabilities that can create multiplatform applications all the while giving the develop test lab and crash reporting to diagnose errors. Backend side of the application can be supported much faster due to the abilities of having Realtime database feedback while editing the software, keeping the file storage, and hosting solution without needing external sources under the same SDK. Authentication to log users in is a simple process with minimal resistance, and while implementing notifications, allowing cloud messaging, and index of the app can be a seamless development process under Firebase. Testing new configurations, does not need to take delay from the user, since implementing new structure can be done in Realtime with a feature unique to Firebase, the Analytics feature, this allows the application to be observed and insights on how components are working for developers and users. The Analytics feature is useful for giving statistics of how the application can be used, so that developers can use this information to either fix bugs or improve upon features. This SDK can be proving useful for developing Android, iOS, and Web applications [102].
In regards to the database, Firebase would help streamline integrating with the front-end. Firebase would make handling sign-up and sign-in functions easier as it manages user sessions and allows the users to sign in and sign up with their Facebook, Twitter, Google, or Github accounts. Firebase would also help with providing custom home screens based on user preferences and user tracking.

3.5.6.1 Firebase for User Sign In/Up and Custom Screens

Firebase offers two ways to use it for user accounts and sessions. There is a drop-in authentication feature, FirebaseUIAuth that performs all of the essential functions necessary including password resets. The other option is to use Firebase SDK Authentication. This option allows the developers to decide which features are needed in the application. Those optional features include:

- Email and password authentication
- Authentication via other accounts (Facebook, Twitter, Google, and Github)
- Authentication via SMS
- Custom authentication to work with pre-existing authentication systems.
- Guest/temporary authentication

Firebase also allows for the customization of screens without having to re-publish the application. In order to use this feature, a series of steps must be followed. First, Remote Config must be set up to hold the parameters for the customizable elements. Second, set up Analytics to define audiences for user targeting. Lastly, configure Remote Config to customize elements based on Analytics or user properties [103].

3.5.6.2 How does using Firebase impact the database?

Firebase also functions as a NoSQL database. Firebase can also handle offline data management, as well as synching when the user's device does have signal. But this ability was not implemented due to the latency issues that this caused. Data in a Firebase database is stored as a structured JSON tree [104 - 105].

The developers had to decide between using a MySQL database and manually handling sign-in and data management or using a Firebase back-end and having to work with unfamiliar technology. Since Firebase claims to save time for development and would prevent bugs that would be caused if the developers had to handle most of the back-end manually, the developers decided to plan on using Firebase. In order to transition to Firebase, the developers worked to transition the planned MySQL database to a Firebase database by way of tutorials [106].

3.5.6.3 Costly Mistakes with Firebase

An article published by a small Spanish development group, GreenLionSoft, highlighted some of the issues that can arise when using Firebase. This development group switched to Firebase and over the course of one month accumulated a bill from Google of over $1000. In the article, they described their misunderstandings and mistakes that lead to this huge bill. Summarized below are the recommendations from the development group to avoid such issues [107].

Tip #1: Use short labels.
Firebase charges based on the amount of data stored ($5/GB) and downloaded ($1/GB). Using short labels for data fields reduced the size of the data stored in the database significantly (reduced storage by 34%) and helped to decrease the storage bill.

Tip #2: Do not enable keepSynch
This was what GreenLionSoft claims was the main source of the hefty bill. The developers *thought* that keepSynch would only synchronize the user's device with the cloud storage if there had been a change in data. What actually happened, was

Firebase synchronized by downloading a fresh copy of the user's data from the cloud (replacing existing data) every time the application was started. When the use makes modifications, the application updated the data stored on the cloud because uploading data is free. The solution to this problem involved disabling keepSynch and replacing the functionality with a local cache that only existed for the life of the application. This solution did have a downside: If a user manipulates the application in two or more devices simultaneously, the data will not actually synch and update until the application is restarted on both devices.

3.5.6.4 Users in Firebase Projects

Every_application in a Firebase project shares the same user database. User objects represent user account. A user instance is independent from an Auth instance. A user object is made up of a fixed set of properties: a user ID, an email address, a name, and a photo URL. The user object properties can be updated by the user. Any additional user data that needs to be added must be stored separately.

User data is derived automatically from the initial sign in / sign up process. If a federated identity provider is used, the user data will be populated based on that provider's data. A user instance keeps track of every provider linked to the user.

When a user signs up / signs in, that user becomes the 'current user' for that Auth instance. If the page is reloaded, the user is not logged out because the Auth instance persists the user's state. When a user signs out, that user ceases to be the current user.

The current state of the use can be tracked using listeners. Listeners are notified anytime a relevant event occurs such as: user signs in, user signs out, user changes password, et cetera.

Auth tokens are used by Firebase and Flutter to identify users. Auth tokens come in three varieties: Firebase ID tokens, identity provider tokens, and Firebase custom tokens. Firebase ID tokens are created by Firebase on user sign in. These tokens are signed JSON Web Tokens that uniquely identify the user in a project. These tokens can be used to identify which user is signed in. Identity provider tokens are (usually) OAuth 2.0 tokens created when a federated identity provider is used for sign in or sign up. These tokens are used to verify successful sign in and are then converted into a format that Firebase can use. Firebase custom tokens are created when a user signs in or signs up using the application's custom authentication service. These tokens are used much like identity provider tokens and exist as JWTs that hold a private key [108].

3.5.6.5 How to Implement Firebase Authentication

There are two methods of implementing user authentication with Firebase: FirebaseUI Auth or Firebase Authentication SDK [109]. As previously discussed, FirebaseUI Auth is a drop-in service that handles all of the authentication process. The Firebase

Authentication SDK allows the developers to pick and choose what aspects of Firebase's authentication system to use. Below is a summary of Firebase's guide to enabling sign in with both tools.

Implementing FirebaseUI Auth:

1. Select sign in methods and provide any additional data they require. This can be done in the Firebase console.
2. Set up the user interface. The UI can be customized via the FirebaseUI settings.
3. Enable the sign-in flow. This includes importing the FirebaseUI library and picking the permitted sign in methods.

Implementing Firebase Authentication SDK:

1. Select which sign in methods to use and provide configuration data. This can be done in the Firebase console.
2. Implement sign in flows. Federated identity providers may have different sign in flow requirements.
3. Pass the sign in credentials to the Firebase Authentication SDK.

Based on the research gather into backend data management solutions, the developers decided to move forward with Firebase serving as the database as well as many backend functions like the authentication system. If the developers encounter a major issue with Firebase, the secondary plan is to use a MySQL database and will manually handle authentication.

3.5.6.6 Firebase Realtime Database

This feature of the Firebase SDK is a seamless process of using Realtime functionality of connecting to the database all while working on the improvements of the application, without the delay testing and running implementation process. Having Realtime implementation helps developers with faster improvements since it can be tested and run concurrently, making it no longer necessary to load and reload the experimentation of the development. This can also be beneficial while the chance that the application is offline, since Firebase allows storing and synching data in Realtime through web, mobile, or simultaneously throughout every platform at the same time. While being offline, the data is stored at a local cache on the device to serve and store changes, so whenever the user gets back online, data that was stored on the local device gets synchronized to the database.

- **Realtime**
    - This method works by data synchronization of every changes that occurs to any connected devices upon updates are made
    - This is different from HTTP (Hypertext Transfer Protocol) request that works by communicating between the client and server to response to changes

- **Offline**
    - The responsive of Realtime can still function by storing data to local devices while waiting to be available online, that way it begins synchronizing when ready
    - Changes from one platform will synchronize with the database and update changes across every platform the data serves

- **Accessibility**
    - For being a multiplatform system, Firebase can be accessed directly from either mobile device or web browser without needing a server for the application
    - Having security and validation for the data is still available due to Realtime synchronizing, the rules allow data to be read and written when it is executed

- **Multiple Database**
    - Having scale of how much data can be a problem, but Firebase scales by splitting the data by having it stored across multiple database instances
    - Accessing database can be better controlled and managed by Realtime for each databases instance when authentication is requested [110 - 111]

3.5.6.7 Firebase Communication to Microcontroller

Connecting the database to microcontroller using Firebase requires set of libraries that can be use in the Arduino IDE, which is a problematic process to setup and obtain connectivity. The setup process needs the database authentication key and the database API URL from the software side through either the web or mobile application, while on the hardware side, it is more than simply typing code into the Arduino IDE to implement connectivity. The code will then allow communication from the database and the Arduino with the API URL of the host, database secret that sets up the authentication in the settings, can complete the process to successfully connect to the database [112].

The biggest issue is establishing communication of microcontroller with software application when developing internet or wireless exchange of data between devices. Methods can be implemented such as Bluetooth or Near-Field Communication (NFC) to remotely accessing devices and database mainframe, methods such as HTTP (Hyper Text Transfer Protocol). While the communication process for device and software can be complex even for simple task, Google Firebase serves as the intermediary medium to allow connection. The use of Realtime database (explained in Flutter section of Realtime Database) provides simple development for application with Firebase SDK along with libraries for the microcontroller.

Developing applications using Firebase is much more simpler than traditional database creation like MySQL, Firebase Database has data stored while in the format of JSON which is an easier access for storing application data. Implementing new elements in Firebase is simply clicking the (+) sign on the top right from the screenshot below, which is available and seen from the Graphic User Interface (GUI) to allow minimal effort in changing elements and accessing the database that can serve as a huge time saving method for developers. While a new element is stored in the database, it becomes hash (giving a generated ID) so that the data transmit and receive to other devices while being recognized only during the exchanging communication of designated interface [113].

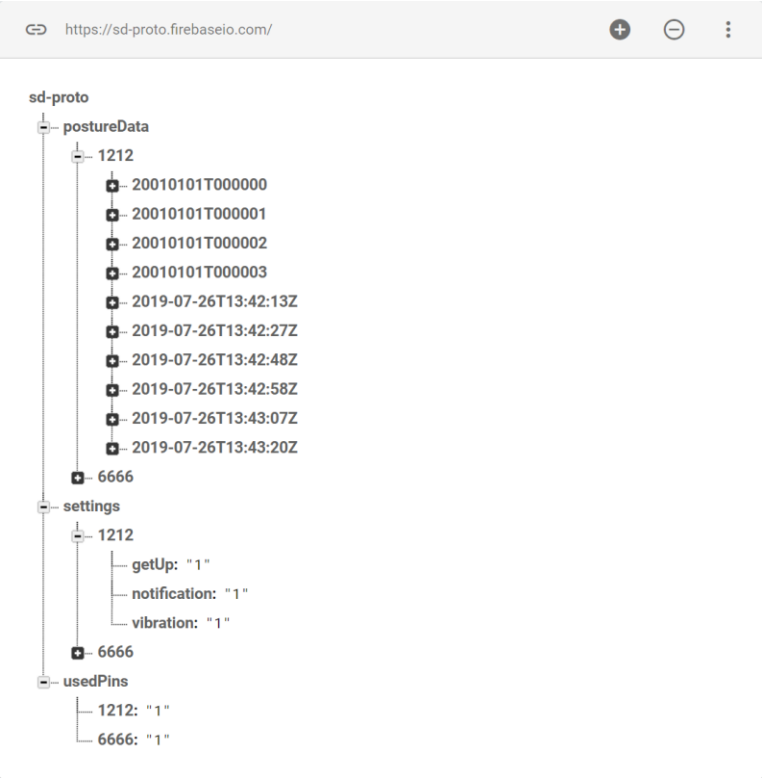


Figure 3.6: Implementing new elements

What makes this important in terms of hardware is the implementation of the stored data that can be transmitted wirelessly, with the use of Firebase library and wireless device which is the Bluetooth module. Using the Firebase Arduino Library is easy to implement, which requires obtaining an authentication key and API URL. Abstracting data with proper wiring of the circuitry, while also calls JSON into parsing data from the database can be done through C language along with sets of Firebase classes. **FirebaseArduino** serves as the main class to interact between the Microcontroller and Firebase that can be complex through usecases and control but can be simple to design and implement. Within the class, specific public functions that can interact with the database must be declared to set parameters. **FirebaseObject** is set to store values into Firebase to a more lesser and single value of structure that can construct the JSON, return values in Boolean, int, float, string JsonVariant, or any data types.

3.5.7 Bluetooth

Connection of software to hardware can be complex and convoluted, in the case of mobile devices, sending and receiving data has abundant methods and among many ways to connection, Bluetooth is a manageable option. Mobile devices support Bluetooth to allow wireless connection of exchange of data between the software and hardware thanks to the provides API and framework between mobile and Bluetooth.

Android Bluetooth

Bluetooth can support multiple platform; the Android platform can support the network stack to allow the communication between devices. The wireless exchange of data uses the framework to access and connect from Bluetooth with the Android API, to enable point-to-point features. Android has a built-in function to allow multiple Bluetooth devices connectivity all at once by using the API in the Android system application for scanning, query local adapter, establishing channels, and transferring data while all with the capabilities [115].

- Scan Bluetooth Device
- Query local Bluetooth Device
- Establish RFCOMM channels
- Connectivity to other device service discovery
- Transfer data of another device
- Manage multiple connection

Bluetooth pairing requires data to be transmitted between devices in order to enable the communication process. Security levels can be set to place precaution of connectivity between devices by pairing and bonding, doing so requires permission to request, accept, connect, and transfer data. Setting up applications to take Bluetooth connectivity requires permission of the location of devices to be scanned and can also be used to gather information for the process to transfer data. To set permission between connectivity application and device, declarations must be made for an access point to initiate the device to be enable for discovery by declaring **ACCESS_COARSE_LOCATION** or **ACCESS_FINE_LOCATION**.

Application allows profile for any device connected via Bluetooth, giving it an identity to distinguish from other devices wirelessly since Android 3.0 with the full support of the API. Profiles comes with interfaces such as Hands-Free, that supports mobile device to connect to headsets, vehicles, audio, and health device. Proxy is setup to connect objects of the profile data to establish transmission.

- **Headset**
    - Bluetooth API gives profiles for headsets in applications, which can be used in mobile devices or tablets

    - Profiles are given a proxy that is provided in Android to control devices

- **Audio**
    - Profiles can also be provided for audio devices that contains high quality audio
    - A2DP (Advanced Audio Distribution Profile) is used for devices that can switch connection between audio devices

- **Health**
    - Profiles for health devices such as heart-rate monitors, blood meters, thermometers, and scales etc.
    - Since Bluetooth can exchange data between connectivity, profiles can provide saved data

Health Device Profile (HDP) was introduced in Android 4.0 (API level 14) to provide applications and devices for constant communication in monitoring health. The API to use HDP can includes key classes to be described in interfaces. HDP has main concepts for health departments that provides beneficial services to patients.

Bluetooth Pairing

It is important to know which device can be compatible for communication to ensure that connectivity can be supported. Device that can be supported will send discovery point for scanning in local area for any enabled Bluetooth devices to request transmission and information exchange. The process of pairing Bluetooth with enabled devices is often called discovering, inquiring, or scanning. Device that are within range of discovery will respond by requesting and accepting information, to exchange name, class and unique address. After pairing is made between connection for the first time, request can be set automatically as the device will be remembered for the information already paired which is then saved. There are distinguishable differences between being paired and being connected.

- **Paired**
    - Being aware of existing devices that share an already established link for authentication and encrypted connection
- **Connected**
    - Sharing an RFCOMM channel that are capable of transmitting data between Bluetooth devices
    - Pairing will be established once initiation occurs to encrypt the connection with API

To pair devices in order to make it discoverable, the devices must be querying to return objects to bond and connect, that way it can be recognized, and data will bring up the profile for specific device. The querying process will then acquire the name and MAC

(Media Access Control) address to initiate connection begins as an asynchronous process to return Boolean value in the scanning inquiry.

Bluetooth Low Energy:

Running Bluetooth on devices can drain battery life faster than normal while being active, by keeping connectivity and communication between other devices. To save power for the user in order to make usage of mobile device for longer periods of time thanks to latest Android API connection. The new android 4.3 latest update for API (level 18) platform, provides support built into the device for Bluetooth Low Energy (BLE) for connectivity of the API. Since BLE has a stricter usage power consumption in comparison to classic Bluetooth, it operates on key terms and concepts from GATT, ATT, characteristics, descriptor, and service that can be more explored below [116]:

- **GATT (Generic Attribute Profile)** – Specification profile that links BLE to send and receive data in short pieces called “attributes” that every BLE uses.
- **ATT (Attribute Protocol)** – optimization of running BLE devices that was built on GATT that is identified as Universally Unique Identifier (UUID) from standardized 128-bit format.
- **Characteristic** – a way to contain single value that can either be analogous to a class
- **Descriptor** – defining attributes of characteristic value that could be readable description from within range of the characteristic value, or even units of measurement
- **Service** – collections made from characteristic that can list GATT profiles, such as Heart Rate Monitor and Heart Rate Measurement

Bluetooth vs BLE:

It is important to know the difference along with the advantages and disadvantages that can prove to be beneficial in projects for wireless communication. Bluetooth technology was created with the purpose of not relying on wired connection to exchange data, which is proven useful for short-range and long-range connection that can be implement in devices such as headsets, hands-free call in vehicle, transferring files wirelessly, and control devices. Built into the technology of Bluetooth is frequency of 2400-2483 MHz of band, that allows operations to send data by splitting into packets. While having the purpose of continuous streaming of data in applications, a new version of the technology is Bluetooth Low Energy that has more efficient power consumption all the while being capable of sending more data at close range, making this more marketable for consumer products [117].

Bluetooth comes in different generations from 2.0 to the newer 4.0 which are BLE for low power consumption to perform the same functionality and better. This is possible because BLE stays in sleep mode unless a connection is being initiated. The latency it takes for BLE to connect would be in the few milliseconds while Bluetooth 2.0 takes

about 100ms to establish connection. This makes BLE for more towards an industry and technological advancement purposes such as:

- Blood pressure monitor
- Fitbit devices
- Monitoring sensors
- Geographical Beacon
- Public transportation app

While both Bluetooth from 2.0 and BLE have different purposes, the 2.0 version is widely used for exchange large data transfer while it consuming battery much faster and costing more, on the other hand with BLE, that is mainly used for application based systems that does not necessarily need exchanging large data but running on lower power consumption and cost less.

Bluetooth Communication to Android:

Bluetooth modules and Mobile devices such as Android can communicate with the use of microcontrollers to integrate between software and hardware. Using microcontrollers to implement the communication between software and hardware are main factors in electronics for creative prototypes of experiments like the Bluetooth to mobile device. Bluetooth is a useful tool for short-range wireless communication to other electronic devices in developing hardware connectivity along with mobile application for software implementation using microcontroller. While there are multiple microcontrollers exist for electronic experiment, the prime device used will be the ATmega 2560 for the practice of the experimentation.

Implementing Bluetooth connectivity with mobile device is uncomplicated since writing the code does not use libraries but rather just simple and only needs the serial transmission and setup methods. Locating the serial transmission is simply done by connecting the right wire to the pins of the microcontroller assigning to that specific command in the code along with the baud rate for transmission. The serial transmission should be set at the proper baud rate of 9600 as the default, any higher baud rate can cause the transmission to be more susceptible to noise that would disrupt the connectivity. Once the Bluetooth connects to the microcontroller, the next process would be sending and receiving data to devices that have paired with the Bluetooth.

Establishing the communication with the Bluetooth to the mobile device requires both to be enabled for discoverability in order to have both be paired. On mobile devices such as Android, enabling the option to scan other devices should display multiple devices and detect the Bluetooth module. Pairing request is send to the module from Android, and a pop-up option window will display on screen with password input. This form of connection is based on models for client-server that waits for connection request or sends out the request to the server, meaning the Bluetooth device.

For Establishing Bluetooth Connection with Flutter

In order to establish a Bluetooth connection via a Flutter application, the plugin, Flutterblue was used. Flutterblue is a plugin that works for both Android and iOS devices. Flutterblue provides support for scanning for Bluetooth devices, connecting to devices, and interacting with those devices [118].

How to Connect to a Bluetooth device via Flutter:

1. Create an instance of flutterblue
2. Use the scan() method and a listener to obtain the scan result. This will look for Bluetooth enabled devices nearby. A scan can be terminated with the cancel() method.
3. Use the connect() method and a listener to connect and verify connection to a device. The connection can be terminated with the cancel() method.
4. Once the connection is established, use the readDescriptor() and writeDescriptor() methods to communicate with the paired device.

3.5.8 Password Recovery

A satisfactory password reset model makes it easy for a user to reset their password (less than 1 minute) and allows a user to reset their password while keeping their data safe.
If the Firebase authentication system is used, the developers will not have to handle password resets as Firebase will manage those functions. If a different authentication system is used, the developers may have to handle password resets [119].

According to Troyhunt.com, the two best and most secure way to facilitate password resets is to:

1. Have the server generate a temporary password and email that password to the user.
2. Email a unique URL that handles the password reset.

The article also highlights the importance of never using the method of 'reminding' the user of their password. In order to use this method, the passwords must either be stored in plain text or they are encrypted. This is not the best way to store passwords [120].

3.5.9 Push Notifications

Push notifications were used in the application to remind the user to stand and walk around every ninety minutes. There will be different approaches to performing push notifications depending on if a MySQL database or a Firebase database is used.

In order to perform push notifications with Firebase, the application will need to use Firebase Cloud Messaging (FCM). This service has a free tier up to a threshold that this project will not reach. There are two types of push notifications via FCM: data messages and notification messages. For this project, we will be using notification messages.

Implementing this strategy will require slightly different code for Android and IOS devices [121].

It does not look like there is support for performing notifications with a MySQL database.

4 Standards & Constraint

In design there is the need to set boundaries in order to set a scope while being able to define a successful result. These can be boundaries set by a customer, a budget, or various other factors; often times the object setting the boundary is itself one, such as budget. This where constraints are derived from. They can both help and hinder the design process, and in some cases the quality of work one achievable. As a benefit, they can help reduce things like feature creep and cost. When it comes to offering a product to consumers, regulations, laws, and requirements can become lengthy; standards offer a way to meet a number of these at once without the hassle of combing through them all. Standards can be applied to the design process, sub-systems within the design, moral practices with respect to end user's data and privacy and a number of other aspects as they relate to our design. Refer to this section to gain relevant insight into the constraints and standards of our design and how we have defined the success of our separate sub-systems.

4.1 Distribution - App Store Requirements

The easiest and safest way for a user to obtain the application is to download and install the application via an application store like Google Play Store or the Apple App Store. In order to have the application available on these devices, the application must be submitted to the app stores for approval. There are a series of privacy and safety requirements that applications hosted on these stores must meet. These requirements will serve as project constraints.

Note: According to ClearBridgeMobile.com, the Apple App Store has much stricter requirements than the Google Play Store [122].

4.1.1 Google Play Store

In order to upload an application to the Google Play Store, the developer must have a developer account. Obtaining a developer account involves a one-time $25 fee.

How to upload an application to Google Play, according to ClearBridgeMobile.com:

- Make sure application functions correctly. Fix any bugs.
- Ensure application does not exceed maximum file size. If it does, files can be modulated to an extent.
    - Android 2.3 and up: 100MB
    - Android 2.2 and lower: 50MB
- Enter application details, graphics, and screenshots.
- Upload files
- Perform alpha and beta tests
- Publish

Note: After 'publishing' via the developer account, there will be a small delay before the application is visible to others (a few hours) [123 - 124].

4.1.2 Apple App Store

The application must pass Apple's App Store Guidelines. Below is a summary of guidelines from Apple that are relevant to the project's functionality [125].

Safety Guidelines

- Apps should not contain offensive or inflammatory content (ie. discrimination, animal abuse, pornographic material, prank phone calls).
- Apps that contain user generated content must have a filtering and review plan.
- If application will be in the kids category, must not contain targeted ads or in-app purchases
- Must not promote/allow physical harm (ie. inaccurate medical information, remind users to check with a doctor)
- Provide a method of bug reporting.
- Protect user data.

Performance Guidelines

- The application should be complete (including metadata and placeholder text).
- Do not use the app store for beta testing.
- Make sure metadata is correct and up to date.
- Do not include undocumented features in the application.
- Ensure device compatibility.
- Use power efficiently.
- Declare additional equipment/peripherals needed to use the app.
- App must only use public APIs.
- Apps should be self-contained in their bundles.
- Application must not transmit a virus.
- App must be compatible with IPv6 addressing.
- Apps must not alter the functionality of the device.
- Do not submit apps with empty or sample banner ads.
- Must request user permission to record user activity.

Business Guidelines

- Applications should not require the user to perform additional tasks such as rating or reviewing the application.

Design Guidelines

- Application design must not be 'copycat'.
- Application must fulfil a useful purpose and not be 'creepy'.
- Application must not repurpose keyboard shortcuts.

Legal Guidelines

- Application must protect user data.
- Application must comply with privacy best practices, ‘applicable laws’, and the Apple Developer Program License Agreement
- App must include link to privacy policy in metadata.
- App must get user consent to collect data.
- Application should only access data necessary to perform core functionality of the application.
- Application cannot ‘use, transmit, or share’ personal data without user consent.
- Data cannot be used for multiple purposes without notification and consent.
- Application must not store health information in iCloud.
- Application must not include ‘borrowed’ content.
- Application must abide by the Developer Code of Conduct

Given that the application meets the requirements summarized above, the application should be permitted to distribute itself via the Apple App Store. According to ClearBridgeMobile.com, these are the steps that should be taken to submit an app for review to the Apple App Store [123].

How to submit application to Apple Store:

1. Create iTunes Connect app record
2. Build application.
3. Archive application.
4. Application must run and pass iTunes Connect validation tests.
5. Upload app to iTunes Connect
6. Submit app for review
7. Release application.

Due to the time constraint of the project, the developers declined to distribute the application via Google Play or the Apple App Store.

4.2 Chair electricity Standards

Electricity is very dangerous in the project that we have been in because it can kill people in a second. Safety in smart chair was our main purpose because we did not want to put people live in danger. The PCB was under the chair and rely on some sensors that were in the back of the chair, those sensors have wires connected to the board that has electricity flowing on it. As we know the smart chair will be cleaned by using water, disinfectant and many more stuffs that can put life in danger, we want to use the maximum safety necessary to protect the chair specially when they use it outdoor and it is raining. As we know water and electricity do not mix, we wanted safe wires and material in the design.

4.3 PCB Standards

Printing circuit board which is PCB was invented in 1936 by Paul Eisler and has been using by the Association Connecting Electronics industries that take responsibility to standard the assembly and production requirements of all the electronics equipment assemblies. We are using the PCB in the design because all the projects that have electronics components require a PCB that has a schematic diagram to help identify different path and the standard of the PCB. In North America, there is an association calls Institute for Printed Circuit IPC standard that has associated with every project. Some company are decided to use their own standards for each project, unfortunately, they cannot pass and sell the product without using one of the popular standards the association has provided and creates six printed circuit board manufacturers in 1957 by some of the company like [126]:

- Embedded Technologies
- Printed Boards
- Assembly Technologies
- Design Standards
- Printed Electronics
- Design Standards

To design a PCB, then we needed to use of the standards provide above because every project that has printed board on it should meet the requirement provided by the association connecting electronics industries. Our PCB design were set the general requirements because our project should be safe enough to protect people's lives no to destruct. As Smart can be on the market one day, we make sure that we use IPC-221B Printed Boards standards for our PCB design because it certifies the consistency of our product that will be on the market nationally and internationally. We were using these standards on Smart Chair to help us attain desired results, and it made our merchandise status and dependability as a strength on the health system market to save people who are being sitting and working for a long time without stretching their bodies. As the standards are divided in 3 classes of standards that can works on our project as the fabrication of the PCB and the design; we decided to use the most efficient and can protect all the electronic components in our PCB design which is class 2. The class 2 standards were set for building the PCB that use in building electronics because the PCB dependable is more advanced than the other classes. As you know, Smart Chair were set to perform unceasingly, this class has been defined by IPC A-610 which is for prolonged life without intermittent service. Class 2 was dedicated for all electronic service that can repair and use again by human such as, Smart Chair, Computer mother board, Cellphone board because this class has no void [127].

4.4 Flutter Development Standard

For the front-end development, the developers abided by the Flutter code standards to promote efficient code and ease of reading.

4.5 Disability Standards for App Development

For the project, the development team researched accessibility standards set by W3. These standards are in place in order to improve access to technology for users with disabilities. Described below are measures that can be taken to increase the inclusivity of an application [128].

- Provide alternative text for images.
- Provide a keyboard alternative to mouse/finger movements. If a user cannot use a mouse, they may be using a speech-to-text device that can mimic keyboard strokes.
- Provide transcripts for audio.

Due to the time constraints of the project and limitations on available libraries, the developers did not include accessibility settings within the application. But, Flutter has built in support for screen readers so that aspect of accessibility was still accomplished via the framework.

4.6 User Privacy Standards

User privacy is becoming increasingly important to application developers, application users, and governments alike. Governments are beginning to enact privacy standards to protect their citizens. The European Union enacted privacy standards that impact every application that has a single EU citizen's data stored in it. Abiding by these privacy standards is important for both the user's privacy and the company's (or application's) reputation and global availability. It will also reduce penalties for noncompliance [129].

4.7 Mobile Application Standards

Mobile application has become widespread in global market for software and technology for consumer usage and innovative design, and as part of core modeling to develop application now available on small devices, standards are set to create a universal understanding on how mobile devices function. Great potential can be achieved through mobile devices that allows users access to web, app, planner, media, and information that can support daily life, while developers can expand the capabilities and enhancing cross device design. Standards can be set in principles, interfaces, patterns, and guidelines to developing mobile applications [130].

4.8 Firebase Standards

To ensure efficient use of resources, Firebase has a set of recommended practices. They have been summarized below [131].

All variants of the same application should be placed on the same Firebase project. However, if the variants use different build bundles, then they should be separated. There is no limit on the number of applications that can be placed within the same project, but there is a limit on the number of OAuth 2.0 client IDs.

If two or more applications do not share the same data, they should be located in separate projects to prevent the complications and privacy concerns that accompany multi-tenancy.

4.8.1 Principles

The overall Principles can be numerous ways to create a platform that can document and function natively with individual operating system on each device. Mobile platform can greatly benefit consumer to allow a system that create an experience to unify others with an ecosystem to support. Other platforms can also support a unifying experience with its own ecosystem, with the day to day usage of devices enhancing the user to fully utilize such feature, the mobility of a small platform that goes straight into pockets makes convenience the key leverage to the capabilities. While mobile devices can be identified as a phone, applications can exist in cross platform to scale functionality among ecosystem created with similar experience. The overall Principles can be broken down as follow:

- **Platform**
    - An important factor is documenting the necessity such as the pattern and components that comes in the native operating systems in the application such as Android and iOS
    - Designing platform that is native to the device should remain consistent with the operating system and guidelines to focus on optimal quality
    - Native operating system to the platform can improve and evolve with new guidelines to enhance interface

- **Benefit**
    - Importance of consumer needs is a prioritize focus in developing mobile application that offers capabilities to support day to day needs
    - Designs should be able to bring together multiple users to unify an experience within an ecosystem across any device.

- **Device**
    - Device should have capabilities such as touch, voice, pressure, location tracking, accelerometer, notification, etc.
    - The design of the application should revolve around being able to be utilized on the device that can benefit not only the screen but beyond that

- **Scalability**
    - Mobile device can go beyond than just a phone that can talk and text, modern phones can search through the web, applications to plan and schedule, gaming platform, media for music and video, and many more
    - Capabilities on the phone can also be scaled on tablets with the same features and if not more, and grow beyond simple screen usage
    - Although challenges can be met by scaling interface between web and tablet designs of the patterns and guidelines
    - There can be similarity between mobile devices and full web applications on a computer that proves to be difficult to be used similarly

4.8.2 Interface Platform (Standard)

Another challenge met by developers is the compact and small size of **phones** which are typically less than 7 inches, that can only display more limited amount of information and screen usage as compared to the web on a computer. To compromise with the smaller size display of mobile devices, the fundamental design must include primarily the necessary information for users to search and view so that the device does not overload. Since mobile devices are the most convenient platform to access information for users, it can also be a useful device for business to quickly access and continue work that the user is not usually and readily available at an office.

Other than phones, another mobile device would be a **tablet**, which are greater than 7 inches of mobile display for the user interface that would allow more space for design and usability. While alignments do not have to be the same from tablets to phones, having larger interface display would greatly benefit by having more information for the fundamental design to include in the platform. While tablets have the designs that tend to have an experience similar to desktop to search the web, plan and schedule, workspace, and media, which is also available on phones, the main functionality of tablets can be considered as a hybrid device.

4.8.3 Patterns & Guidelines

Mobile applications have design patterns that follows to fit on small platforms to navigate through content while feeling quick and easy to use, simplicity should be the key factor in creating applications. Navigating content should have transition phase to display through applications to feel smooth and organize without feeling cluttered and disarrange.

4.8.4 Industry Implementation (Standard)

Establishing set of rules and guidelines for platforms to follow becomes a universal practice for industries to develop mobile applications while also evolving on new and inventive ideas for technology of the software. There are multiple set of standards the industry implements [132].

- **Visibility and Timing**
    - Notification is an important feature that allow users to view information which companies provide in applications to be relevant for the framework
    - Accessing data also collects and transmit important application features while prioritizing user accessibility with privacy
    - Sensitive information such as financial data will be held accountable on the company to provide and ensure secure utilization
- **Security and Data Retention**
    - Importance of security in transmitting data must be for legitimate purpose and should be access in the application only unless it is required to do so
    - Sensitive information should be given as option that lets the user to allow their data to be collected to avoid risk
    - Developers must implement this option in order to avoid danger of data being leaked to give users security in experience
    - Rules and regulations must be met for developers to publish the application that follows the app store and platform terms and services
    - Sensitive data must be stored on a time frame where it has the chance of being deleted after user no longer needs data stored in servers
    - Respecting the user privacy keeps their interest in the service of the application with procedures holding accountable of private information
    - De-Identification is another solution to ensure security of data by deleting previous data to reestablish identification by hashing and linked back to original source of user or device

- **Enabling Security Measures**
    - Applications are susceptible to security risks of accessing or transferring data from individuals to another that requires careful attention
    - Testing and solution must be taken for security measures for implementing retention policies for safeguarding data
- **Data Encryption**
    - Encrypting data allows authentications of the user personal data through transmission to provide protection
    - Proper utilization of server by avoiding SSL/TLS and other forms of communications of transmitting data
    - Sensitive information such as email, address, username, and password, must be encrypted to maintain authentication

- **De-Identification**
    - Like encryption, changing identification on multiple efforts makes it difficult for data to be at risk since it does not link to particular result
    - Numerous identification elements require scrambling encrypted data to have comparable link to that element
    - This method ensures higher security measures to terms of privacy to avoid possible risk while maintaining retention

- **User Authentication**
    - Logging in and out requires authentication of the user by inputting either email, address, username, and password for the session with validation
    - Implementing mobile client to ensure validation of the user to access their data from the server database where it is stored with sensitive encryption protocol
- **Accountability**
    - Responsibility of the company gives user the respect to hold the application development team accountable for managing the safety and privacy of their data
    - Protections of sensitive data should be fully integrated to the application to maintain the policy of company duties of terms and services
- **User Feedback**
    - Maintaining good relations with user can be provided by having feedback for opportunities to improve flaws of bugs of the application
    - Companies gain positive note from questions, contacts, queries, or complains of application interface and usage
    - Fixes and patched can follow from feedback to highlight functionality to bring more efficient solution

4.9 Power Supply Standards

According to POWER SUPPLY SAFETY STANDARDS, AGENCIES, AND MARKS which are responsible for the principle of electrical safety standards, the national and international safety ask to use the one of the standards that requires in your country and certify by the government to prevent fire, electric shock and injury should meet their requirements. Below are the standards that describes our power supply requirements for the safe use of the electric equipment, batteries.

Standards	Description	Details
IEC 60950-1	Safety of Information Technology Equipment.	Intended to prevent injury and damage such as electric shock, fire, dangerous temperature.
IEC 60065	Safety of video, Audio, and similar electronic apparatus.	Intended to protect against fire, electric shock and injury, electronic equipment and communication.
IEC 62368-1	Audio, Video, Information and communication technology equipment	Standards that currently govern companies marketing audio-visual products, computing and communications equipment in North America.
IEC 60601-1	Safety of Medical electrical equipment.	Covers the basic safety and essential performance applicable to medical electric equipment including surgical, monitoring, hospital devices and been revised so many times.
IEC-61010-1	Safety of measurement, Control and Laboratory equipment.	Requirements for measurement, control and laboratory equipment. It protects the electrical shock, fire and burns injury.
UL 1310	Safety, Standards requirements for 2 class power units.	Covers indoors, outdoors that use 2 power supplies and batteries. Uses for residential and industrial.
UL 600799	Standards that use for explosive atmospheres.	Covers electrical equipment that use for explosive atmosphere like construction, gas, combustible dust.

Table 4.1: Power Supply Major Safety Standards

4.10 Economic & Time constraints

Engineering is not only to solve mathematics and physics problems. It is to retain transforming upon a delinquent by present more solutions to the world that we have being living. In our technology, engineering helps us save life, money and make people lives more efficient by solving and inventing new products that can use in our world. Smart Chair could theoretically have wounding advantage expertise in technology including structure, health solution, sensors, mechanism and efficacy. In this part, we talked about time constraint and economic that all project need to be discussed because you needed to spend time on everything that has been doing to make it possible and spend money to buy the material, electrical and electronics components that we were needed to figure out and balance the best way possible to provide the best product to the public nationally and internationally.

First, economically we wew undergraduate students who funding our own project because we did not have sponsorship from any of the government and one of the providers. We had to add money together to buy the parts that we all need to make the project. However, we must work outside the classroom to have money to sponsor our own project that will help the whole world whether the Smart Chair was out of pocket budget or outcome the correct funds. With our project reasonable approximation at $600.^{00}$, it was noteworthy for our Smart Chair team to provide all the material and components necessary to build the project. Due to the back-health problem, Smart chair should be planned to design as close to picture-perfect as possible and slightly mistakes or inaccuracies could be possibly overwhelming to our budget to success and progress that we need to make to our project. It was important to pay attention to all the details and review every components from the bottom to the top once we complete our pamphlets and the research that can make the project a win to us because we want to use the amount that we can give to make it happen with the little time that we have left in front of us.

To complete all the assignments and projects, we needed to use the limited time that we have. Time of constraint mentions to the limits on the start and end of each task that we must do. As a group, we were specified a limited time window to function where we must sit and meet to focus on tasks and to meet the requirements that necessary to finalize the project. Working on this project gave us a better idea on how to do a better research and prove what we learned from the coursework. When we meet, we used the amount of time that we must discuss because this project was a passion of what we have learned from all classes and independent research. The most important step to maximize time was communicating and working while help us decreased inevitable confusion and improving efficiency on the work and timing. we prepared before every meeting so that we could find perfection and maximize efficiency where we can balance the economic part and the timing part where we all agreed to be on the same page and did not take time for granted, every hour we spent together, we make sure that we found a solution so that at the end of the deadline, we did not have to kill ourselves, and we tried to minimize the amount of pressure we had at the end of the project.

5. Initial Design

With preliminary research complete, and design decisions determined, we begin our proof of concept and prototyping. In the subsequent sections we will discuss the initial design of our PCB including the motivations for changes that may differ from our research. We will also verify the implementation of the Bluetooth module for communication between the user interface and microcontroller. Database development and the use of firebase will also be discussed. Initial design and prototyping is a crucial step in developing a fully functional product, it allows developers to visualize what works and what doesn't. when done with a strategic process, this results in discovering more efficient workflow processes, flaws in initial considerations which may require, revisiting or complete change

5.1 Software

The developers created the initial design of the application. This phase of project planning helped the developers identify any additional research or technologies that were needed to meet the project goals. In order to meet the project requirements, the developers deemed it prudent to use user stories in order to develop a more detailed list of functions the application must possess in order to better direct development.

User Stories

- The user should be able to install the application on their device.
- The user should be able to create an account.
- The user should be able to sign in.
- The user should be able to pair the application with a chair.
- The user should be able to see data from the chair displayed on the application.
- The user should be able to control how the application sends notifications.
- The application should be able to connect to the PCB.
- The application should be able to insert data into the database.
- The application should be able to read data from the database.

Software Requirements Derived From User Stories

- The application will meet system requirements of at least one type of mobile device.
- The application will allow account creation.
- The database will allow for the addition of new users.
- The application will allow for user sessions via sign in.
- The application will connect to a PCB via Wifi.
- The application will read posture data from the database and display the data.
- The application will save user preferences regarding notifications.
- The application will insert data into the database's posture data.

- The application will read data from the database's posture data.

5.1.1 Theming with Flutter

Choosing a proper theme is very important in application development. The theme of an application is like the cover of a book: designed to make a favorable first impression. Themes help to communicate the mission of the application.
Material Theming is now supporting Flutter. Material themes are easy to implement, predesigned themes to help create effective and clean user interfaces. The developers intend to incorporate Material design into the application [133].

Material Colors (subheading)

The Colors class is responsible for providing color selection within Flutter applications. The MaterialColors class is a subclass of the Colors class. The MaterialColors class provides up to fourteen variations on primary colors that resemble paint chips found at a hardware store. To access the color variations, call the main color from the Colors class, then add the shade variant. For example, to get a particular shade of pink, add this to the code: Colors.pink[400].
If the colors available to not meet the needs of the application, custom palettes can be created with the Material Theme Generator. It is recommended to pick two primary shades to start with.
Material Design also provides a small selection of built-in themes. One of these pre-made themes is the 'light' theme. These themes can be used by accessing ThemeData.
In order to ensure the coloring of the application remains accessible, Google's codelabs recommends maintaining a specific color contrast ratio of 3:1 for large text and 4.5:1 for smaller text.

Material Text (subheading)

Material Theming also allows the developers to control the text. ThemeData includes three text themes. Each theme includes small variants for special text like headings. Fonts must be imported to the pubspec.yaml file before they can be used.
Google codelabs recommends paying attention to the text size when selecting a font. For smaller fonts, it is best to chose a font that is clear, rather than stylish.

5.1.2 Application and Database

In order to ensure proper functionality and to reduce integration bugs, the developers saw it necessary to create preliminary plans for the application and the storage of user data. In the following sections, the developers outlined their initial plans for implementation of the database and the application including diagrams as well as outlines of the classes and libraries that will be needed to establish basic functionality.

5.1.2.1 User Interface

Below is the developer's initial plan for the user interface.

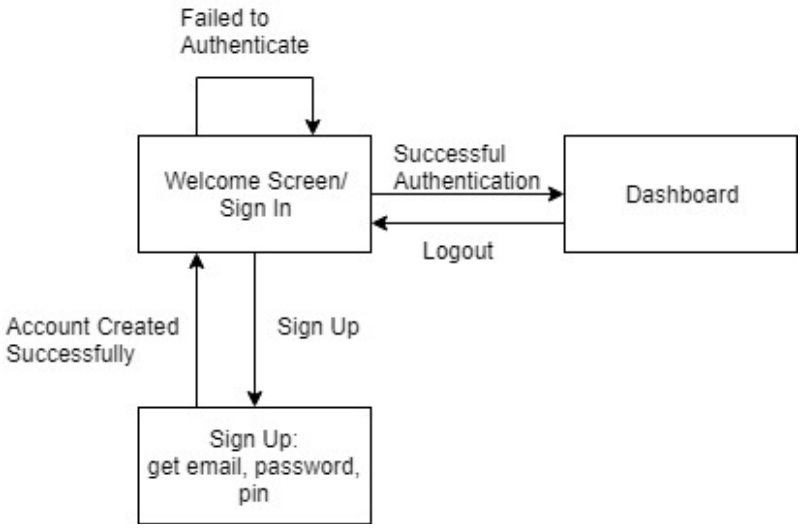


Figure 5.1: User Account Creation and Login

Note: Diagram made on draw.io

Navigation for a Multi-Page Flutter Application

A multi-page Flutter application is made up of a series of widgets that are constructed and displayed when the user performs a certain action. Navigating between those 'pages' is performed via routes. The different 'pages' should be defined as classes. For example, if there is a two page application that consists of a home page and a secondary page, two different classes will be required to create these pages: a class for the home page and a class for the secondary page.

To navigate from one 'page' to another, a Navigator is required. When moving from the home page to a secondary page, use Navigator.push() to add that route to the stack of routes. When moving back through the navigation, use Navigator.pop(). This method will pop the current route from the stack of routes. In the example provided by the Flutter documentation, these methods are called as the onPressed methods.

Sometimes data needs to be transferred between routes. This can be done by passing the extra data like one would pass parameters to a method [135].

Data Charts with Flutter

The application needed to provide the user with feedback on their posture over time. This will be done with a chart. To make charts in Flutter, the developers will be using the *charts_flutter* library.

In order to use the charts_flutter library, the library needed to be added as a dependency in the pupspec.yaml file. Once the dependency had been added, the construction of a chart/graph can began. For the purposes of this explanation, the Medium.com tutorial for this library will be used as reference [138 – 139].

How to make a chart:

1. Create a class for the chart. For the purposes of this document, let's call it MyChart.
2. Define data types for the data on both x and y columns.
3. Optional: use the Colors class to assign different colors to each column/row.
4. Modify the build method of the parent widget to generate this MyChart widget. This can be done by generating an array of charts where each data point is its own instance of a mini chart.
5. After creating the array of data points, use a Series widget to create a data series. The data series will contain a domain value (value for the x-axis), a measure (value for the y-axis), and a color assignment.
6. Finally, add the chart as a part of the Scaffold widget.

Settings Menu with Flutter

The developers used a list of settings with toggle-able controls to allow the user to control notifications. In order to save user preferences, the hardware pin was stored on the user's device. This was done via shared preferences [140 - 141].

The Flutter plugin, *shared_preferences*, provides support for accessing and managing shared preferences on both Android and iOS devices. This plugin will save the developers time by eliminating the need for code rewrite to meet the needs of both platforms.

How to implement shared preferences with Flutter:

1) Add the shared_preferences plugin as a dependency in the pupspec.yaml file. The library needs to be imported within the dart code file that will use it.
2) To access the shared preferences object, use the SharedPreferences.getInstance() method.
   a) To read an integer value from sharedpreferences use the SharedPreferences.getInt method. Variations of this method exist for accessing double, boolean, string, and stringList values.
   b) To write an integer value to shared preferences, use the SharedPreferences.setInt method. As with reading, variations of this method exist to accommodate additional data types.

5.1.2.1.1 General Design

In this section we will discuss the application-level development plan. Shown below is the initial user interface design created by the developers.

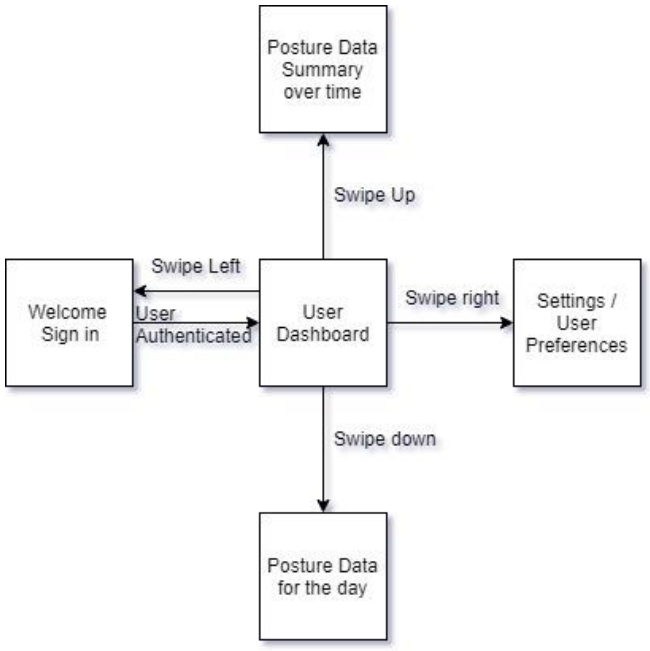


Figure 5.2: Initial User Interface

Adapting Initial Application Design to Flutter

After reviewing the anatomy of an example Flutter program and the implementation details of the main components, the initial design for the application needed to be considered within the abilities and limitations of Flutter. Refer to the image above for the original user interface diagram.

The posture data of the day screen was eliminated and it's data was moved to the dashboard. The dashboard was initially left blank because the project members were still deciding what data needed to be displayed to the user, so that was an easy change.

After shifting the daily posture data to the dashboard, the initial navigation plan becomes less intuitive. The four remaining screens were rearranged as shown in the diagram below.

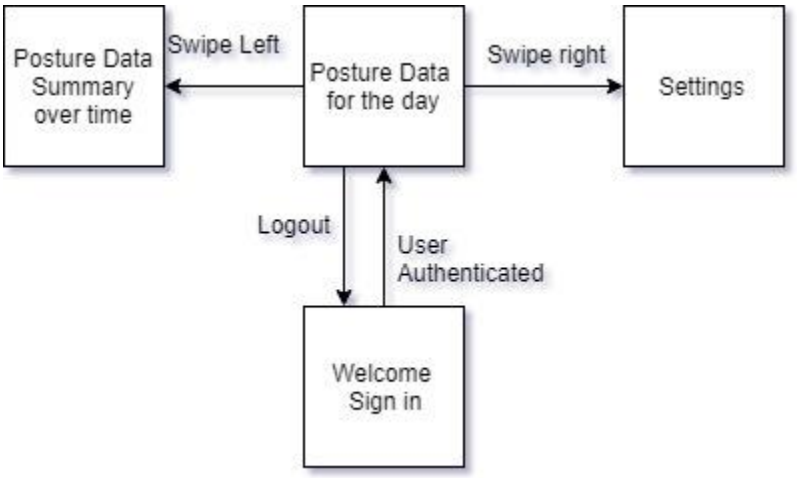


Figure 5.3. Updated User Interface Design

The developers anticipated that the user may need to make adjustments to the application during the application's lifetime. The setting menu is the portal through which the user will be permitted to adjust their preferences. The following are options the developers plan to include in the settings menu:

- Enable / Disable vibrations
- Enable / Disable notifications

The login / create account screen is handled by the Firebase Authentication system.

The posture data over time is displayed graphically. The data is displayed on a scatter plot such that the user's posture coordinates are shown on the Cartesian grid and correspond to the sectors of the physical chair.

5.1.2.1.2 Database Design with Firebase

According to an article from How To Firebase, the database functions best when the data is as normalized (shallow) as possible. Nesting data within the database will increase the cost and decrease the speeds of downloads. One important comment from the article highlighted the importance of finding a balance between normalized data and reducing extra downloads. The article used a transaction log as an example: If email addresses are frequently being pulled when transactions are pulled, it might be better to include a copy of the email address in the transaction log. This results in only the transaction log being pulled rather than both the users and the transactions. This method also increases the scalability of the data (39).

When retrieving data, it is better to use a child_added event than a value event. Value events return all nodes in an unsorted JSON object and will return all nodes every time a single value is changed. This is less efficient than a child_added event which fires for every existing child and fires again when a child is added. Also, since child_added fires for each child, it can work with orderBy modifiers, which means that the developers will not have to manually sort the data.

The article also recommends using queues to handle server processes. This highly scalable method involves adding any processes like changing usernames or transactions to a queue so that they are all handled in order and processes do not get lost.

The image below is a visual to represent the structuring of the NoSQL Firebase database. The securityAnswer field was omitted because Firebase will be handling password reset/recovery. The users data is restricted to what Firebase automatically gathers.

```
1  //Visualization of NoSQL database for Firebase
2
3  //Firebase has support for password resets, so
4  //security questions and answers were not included.
5
6  Users
7      >userID
8      >email
9      >password
10
11 Posture Data
12     >cogX
13     >cogY
14     >created_at
```

Figure 5.5. A Visualization of the NoSQL Firebase database.

5.2 Hardware

This section will serve to discuss the various testing and designing of hardware components that lead to the final design. This includes the PCB layout, testing the sensor array, development process for embedded code on the microcontroller, Integrating the WiFi module onto the PCB. We will also discuss design change decisions that arose, these in turn became revisions to previous sections of this design brief.

5.2.1 PCB

A requirement of our project is to design a PCB that will house all of our major hardware sub-systems including power supply, battery, Bluetooth module, and multiple DC/DC converters to power said systems. When approached as the design of a consumer marketed product, we treat all aspects as if they were to go into production and be sold to a consumer. As such we would not reasonably expect to market a device relying on the use of breadboards and development boards; a standalone purpose-built board is

the only suitable option. This is where the PCB comes in, and below we will discuss the design layout of ours.

5.2.1.1 Design of PCB

As previously mentioned in chapter 3, we decided to use eagle to develop our PCB. Below is an iteration showing all major components intended to be housed upon the board minus the nine-volt rechargeable battery and WIFI module, though we chose to leave a reasonable amount of space for these. We began by importing several component libraries into our design file and adding the ATMega MCU, to an off-center position of the board, keeping the needed space in mind. This was followed up with designing several sub-systems including the power supply, located at the very top left sector, denoted by three green octagons, leading into a diode to regulate the voltage. This goes into our five-volt DC converter, labeled *IC1*. Below this is our five to three-point three-volt converter, which will supply auxiliary devices including our sensors and Bluetooth module. The five volts will supply our microcontroller and vibration module. In the event that components require lower voltages, it will be more cost effective and less time consuming to implement voltage dividers (i.e series resistors) versus more DC/DC converters. To the right of our DC converters are several analog inputs, these can be used to signal components that do not provide feedback, such as the vibration module or LED had we chose to move forward with implementing a light-based notification system.

The far right of the board contains ADC inputs which would be used to power and receive signal from our force sensors. These pins connect to the MCU and utilize the built in ADC channels that also have variable analog signal gain. We also placed several ground pins in this location. To the far left of the board lie our transmit and receive channels for our UART protocols. These will be used for digital sensors, such as the proximity sensing device. There is also a set of digital I/O pins here. To the immediate left of the MCU is a set of Arduino plugin pins. Since much of our development is on an Arduino device, it would be useful to have a plugin from which we can configure our MCU based on the testing and designing we develop.

Finally, at the bottom of the board, just below the microcontroller lies a 16MHz crystal. This crystal served as our external clock source. The importance of the PCB design was that by placing components we can see how the traces were routed, and determine best locations for components to ensure neat, simple traces. Lines in red a top trace between components, while blue lines depict under traced. Below is the final iteration of our design, which took two previous versions to reach.

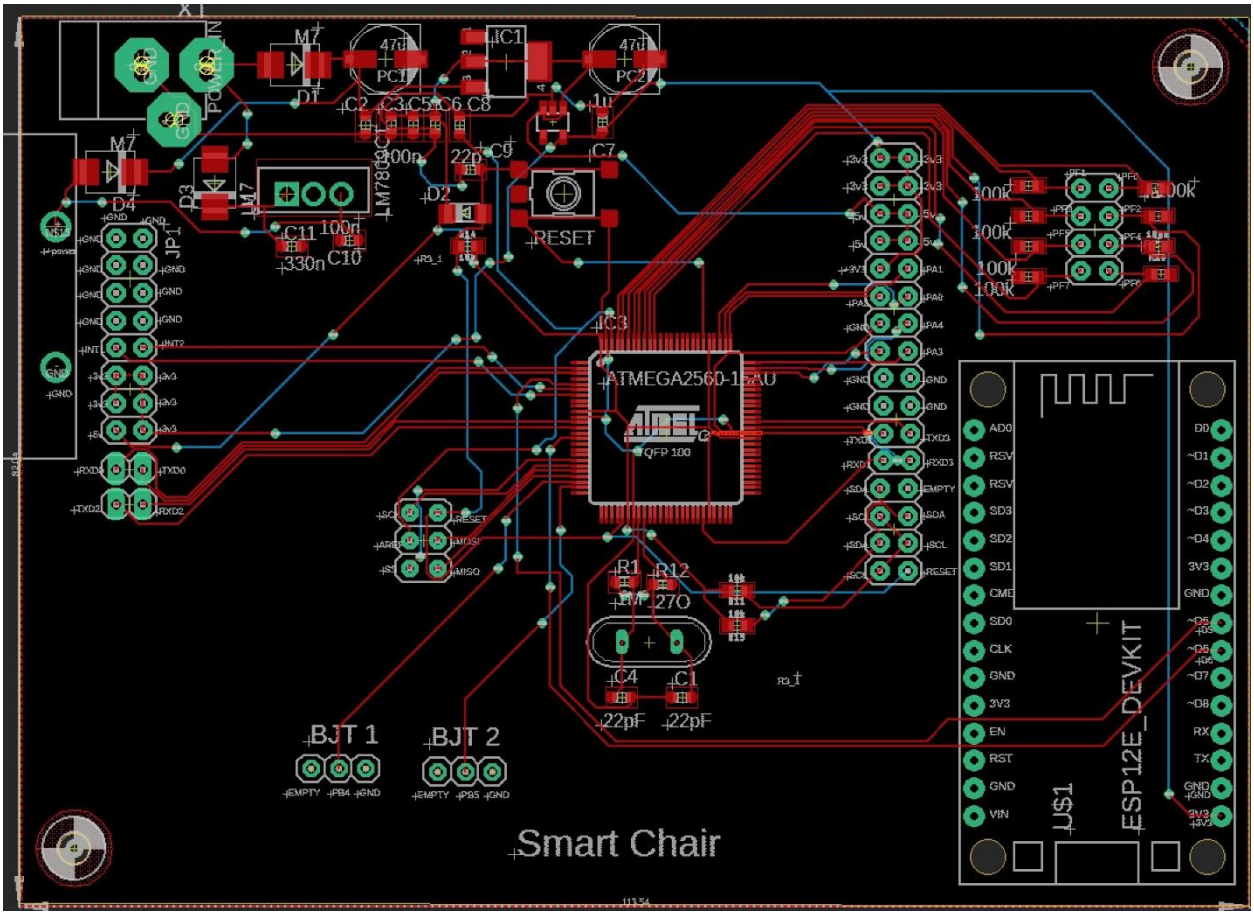


Figure 5.6: PCB design

5.2.2 Bluetooth Module

There are multiple kinds of Bluetooth modules that can be used for electronic experiments, but the main focus is primary between Bluetooth 2.0 ERD Modules and the Bluetooth 4 / BLE modules to compare which would contain more features, functionality, simplicity, and implementation. The number of pins can vary between the different version from 2.0 and 4.0, with the most common pins are EN, WAKEUP, STATE, BRK, KEY, or LED that can be available across any of these versions [154].

5.2.2.1 Bluetooth 2.0 EDR Modules

From this type of Bluetooth module, the most commonly used modules are the HC-05 and HC-06, with both being capable as slave device. The firmware built into these modules are different while they both have similar features while one has an advantage compared to the other, which would be the HC-05 having the capability to be master or slave that allows initiating connection to other devices, while the HC-06 has only slave functionality and only accepts connection of other devices. The key features are that both modules have the same hardware with the HC-05 having all 6 pins, while the HC-06 having only 4 pins, but the primary usage is the same [155].

5.2.2.2 HC-05

This Bluetooth module uses SMD module that is based on BC417 models and has a design to be used for transparent wireless serial connection that goes along with the setup of Bluetooth SPP (Serial Port Protocol). This module can cover range up to 9 meters, or 30 feet of signal that can function as master or slave. This module uses full duplex data transmission, meaning it has capabilities to transmit both direction of sending and receiving end of signal carrier at the same time. The pin configuration for the HC-05 includes Enable / Key, Vcc, Ground, TX – Transmitter, RX – Receiver, State, LED, Button. When running the experiment to send signal of the module, it has a default setting to be used when coding the microcontroller, with the follow [146]:

Default Setting

- **Name:** "HC-05"
- **Password:** 1234 / 0000
- **Communication:** Slave
- **Mode:** Data Mode
- **Baud Rate:** 9600, 8, N, 1
- **Command Mode Baud Rate:** 38400, 8, N, 1
- **Firmware:** LINVOR

Specifications

- **Voltage:** 4V – 6V
- **Current:** 30mA
- **Range:** <100m
- **Compatibility:** Serial Communication (USART) and TTL
- **Standard:** IEEE standardized protocol
- **Signal:** Frequency-Hopping Spread Spectrum (FHSS)
- **Operation:** Master, Slave, Master/Slave mode (simultaneously)
- **Usability:** PC and mobile with Bluetooth capability
- **Baud Rate:** 9600, 19200, 38400, 57600, 115200, 230400, 460800

Using the HC-05 for experiments can add full-duplex wireless functionality towards projects that can communicate across other microcontrollers, from mobile devices and PC to interface with applications. The USART and the baud rate at 9600 makes the process simpler to implement and configure the module through command mode. This module is widely used in electronic projects for wireless data transfer to and from PC and mobile but does not have functionality to transfer multimedia content. The benefits of using this module is having two operating modes, where one Data mode is capable of sending and receiving from other devices with Bluetooth functionality, the other data mode is AT Command mode to allows changes made onto the default setting using the pins. This module is simple thanks to the operations of Serial Port Protocol (SPP), while only needing to use up to 5V to power up from the pin with ground, this allows it to enter

to Command mode, unless the left was left idle for a while then the setting goes back to default which is Data mode.

5.2.2.3 HC-6

This is very much like the HC-05 Bluetooth module with the only difference being that the HC-06 is only a slave device. This makes this module more suitable for simpler projects for wireless data transmission that only requires the Bluetooth end for slave mode, with the same range as the HC-05 at 9 meters, or 30 feet. This module can be used for short range data communication that is setup from two different microcontroller or even any system with built-in Bluetooth. The transmission speed of this module can go up to 2.1 Mb/s while also being the most affordable Bluetooth module to use for projects and very flexible to implement. The signal of transmission uses frequency hopping spread spectrum (FHSS) like the HC-05, this type of signal avoids interferences across other devices in order to achieve full duplex transmission that was explained in HC-05 for directions of sending and receiving signal carrier. The pin configuration is identical to the HC-05, with the exception of Enable, LED, and button. The configurations include the Key, Vcc, Ground, TXD, RXD, and State [147].

Features

- **Bluetooth version** 2.0 protocol standard
- **Power level:** Class2(+6dBm)
- **Band:** 2.40 GHz, ISM Band
- **Receiver Sensitivity:** -85dBm
- **USB version** 1.1/2.0
- **Modulation Mode:** Gauss Frequency Shift Keying
- **Authentication and Encryption**
- **Voltage Operation:** +3.3V – +6V
- **Temperature Range:** -20ºC to +55ºC
- **Current:** 40mA

Advantages

- Optimal choice for short range wireless communication
- Wireless distance of less than 100 meters
- Simple to implement interface and communication
- Very affordable cost for module that functions all types of capabilities in present
- Very low consumption of power to maintain and capable of being powered by mobile battery to function
- Has capabilities to be interfaced with nearly all microcontrollers and microprocessors so long as there is a UART

Since the interface communication is done through the UART have the module to be functional, sending data to mobile or receiving data from the module can be establish through any device. Powering the module can be connected using the +5V pin from the standard regulated power supply with the UART establishing an interface, while the

connection to exchanging data from RXD to TXD and vice versa, but the TXD of the microcontroller to the RXD of the module is connected using voltage divider for the resistance. The voltage divider is used to convert the 5V down to 3.3V of the logic signal to make it more suitable.

For the programming portion of the Bluetooth module, the baud rate can be set to 9600 at default from the serial communication of the UART, and since the module is a slave device, a master device must be established from the microcontroller end to successfully complete the communication. Setting up a mobile phone for the master device in order to connect the module to have an interface communication, this is done by using a microcontroller to program the connection, thus allowing the Bluetooth module signal to transmit data. Once connection is established between both the mobile phone and the Bluetooth module, authentication with password will be asked from the master before the interface is complete, the default password is often 1234. The process can be further simplified by using libraries for the module to be called within the program, this communication can give commands to send or receive data through the UART serial communication.

5.2.2.4 Bluetooth 4 / BLE Modules

BLE (Bluetooth Low Energy) modules can be beneficial for saving power, and like the HC series, this also uses SMD module based on the Bluetooth SOC (System On Chip). The type of Bluetooth comes in 2 versions from the HM-10 series, the S version and C version that does not need pads from the bottom connector (USB connection). Both the C and S versions have 26 pads instead of the usual 34 from other modules, making this the cheater option to produce and manufacture. BLE itself is not considered an upgrade from the Bluetooth Classic modules, but instead it uses a different system and intention of usage.

5.2.2.5 HM-10

The basic component specifications of this module give manufacture cheaper and easier to produce while having the same operations for the C and S versions. Unlike the 2.0 modules, these BLE modules contain the standard UART serial connection that is more familiar to connect along with microcontroller. While being a Bluetooth 4.0, it cannot connect to 2.0 modules such as the HC-05 and HC-06, although being a BLE it is simpler to use and allows no actual control towards the BLE side of the module itself. The HM-10 modules can be controlled through the AT Commands that transmit data through serial UART connection. The pin configurations of the specification is listed below [148]:

Specifications

- **Voltage:** +2.5V to +3.3V
- **Current:** 50mA
- **Active State:** 9mA
- **Sleep Mode:** 50-200µA

- **RF Power:** -23dbm, -6dbm, 0dbm, 6dbm
- **Version:** 4.0 BLE
- **Baud Rate Pre-Firmware:** V700 at 9600
- **Baud Rate Firmware:** V700 up to 115200
- **Default Pin:** 000000
- **Default Name:** HMSoft
- **Based:** CC2540 or CC2541 chip

This module uses the standard UART serial communication which makes straight forward connection for microcontrollers with the same standard layer of UART. this connection from the serial UART controls the HM-10 through the AT Commands. It can also be used for mounting on breakout boards that allows the power more exposed from the connection to the breadboards process more seamless to activate the module from a 3.3V pin. Like the HC series, this module has similar pin configurations that includes, STATE, Vcc, GND, TXD, RXD, BRK. There is also a built in LED that blinks while waiting for connection, but once connection is made then the LED stays solid and can also change to pairing which leaves the LED to continue flashing to establish connection.

- **State Pin**
    - When there is no connection from this pin then the STATE is set to LOW
    - When there is connection from this pin then the STATE is set to HIGH

- **BRK Pin**
    - This is the break pin that can cancel connections of active transmission
    - This pin can bring down connection to LOW when breaks and HIGH to stop pin floating when the pin is pulled
    - No connection through the break can mean HIGH or LOW which has no effect

5.2.2.6 BLE Link Bee (Bluno Bee)

This module has capabilities to support master and slave switch with a long transmission range of up to 60 meters throughout any free space, which is one of the many benefits of Bluetooth 4 modules for latest design. The design for this module is more compact than the HC and HM series, that has voltage regulator integrated into the MCU that can support both 3.3V and 5V. This module can used as master and slave machine and even to each other to connect by point-to-point wireless transparent transmission. The Bluno Bee is useful for smaller and mobile devices that can be worn like a bracelet and can also achieve communication with real-time linking while having low power consumption. The important key features of configurations and specifications are listed as follow [149 - 151]:

Specifications

- **Bluetooth Chip:** TI CC2540

- **Frequency:** 2.4 GHz
- **Transfer Rate:** roughly 1Mbps
- **Modulation:** GFSK, Bluetooth Low Power, V 4.0
- **Power Consumption:** 10.6mA average, 8.7mA ready mode
- **Sensitivity:** -93dB
- **Input Voltage:** +3.3 DC
- **Temperature:** ~40 °C ~ +85 °C
- **Distance:** 15~20m enclose space / 30 open space
- **Size:** 32mm * 22mm / 1.26 in * 0.86 in

Features

- **AT Command**
- **Master-Slave machine**
- **Transparent Transmission**
- **Compatibility to Arduino**
- **Supports USB update to BLE chip program**
- **Supports Bluetooth HID**
- **Upgrade to BLE firmware**
- **Supports Android and iOS**

5.2.2.7 Implemented Bluetooth Module

Microcontroller to Bluetooth connection works as a master and slave device, where the microcontroller is the master, and Bluetooth is the slave for the device. This means that the master device gives commands to the slave device to communication with all functionality such as exchanging data. The Bluetooth has a set standard range for the wireless connectivity, it uses **short-wavelength UHF radio waves** that has a band range of **2.4 to 2.485 GHz**. The technology built-in Bluetooth can be embedded in any device, while the module communications with mobile devices from building personal area network (PAN), the range it can be reached from the band would be approximately 9 meters, or 30 feet. Each pin on the Bluetooth module serves a purpose, with the Enable/Key, Vcc, Ground, TX, RX, State as pins to connect with features for functionality.

- **Enable/Key**
  - The enable pin serves as a switch for the Data Mode and AT Command Mode
  - The switch for Data Mode is setting low power for the module
  - The switch for AT Command Mode is setting high power for the module
  - Normally at default, the module is set to Data Mode for low power

- **Vcc**
  - This is the pin that powers-up the module and serves as the supply voltage
  - Connection for supply voltage can go up to 5V

- **Ground**
  - The ground pin of the module to regulate the voltage to zero potential

- **TX**
  - This pin is the Transmit Serial Data on the Bluetooth module so that it connects to the RX on the other device to acts as the receiver
  - Serial data that gets transmitted is being received through the other end to give out data which goes to the pin to the receiver UART of the microcontroller

- **RX**
  - This pin is the Receiver Serial Data on the Bluetooth module so that it connects to the TX on the other device to acts as the transmitter
  - This acts to request serial data that is given to this pin, while the other end of the connection transmit from the microcontroller

- **State**
  - Serves as the connection to the LED on the Bluetooth module from the board itself
  - Functions to check if Bluetooth is functional and flash or blinks for feedback ensuring proper working condition

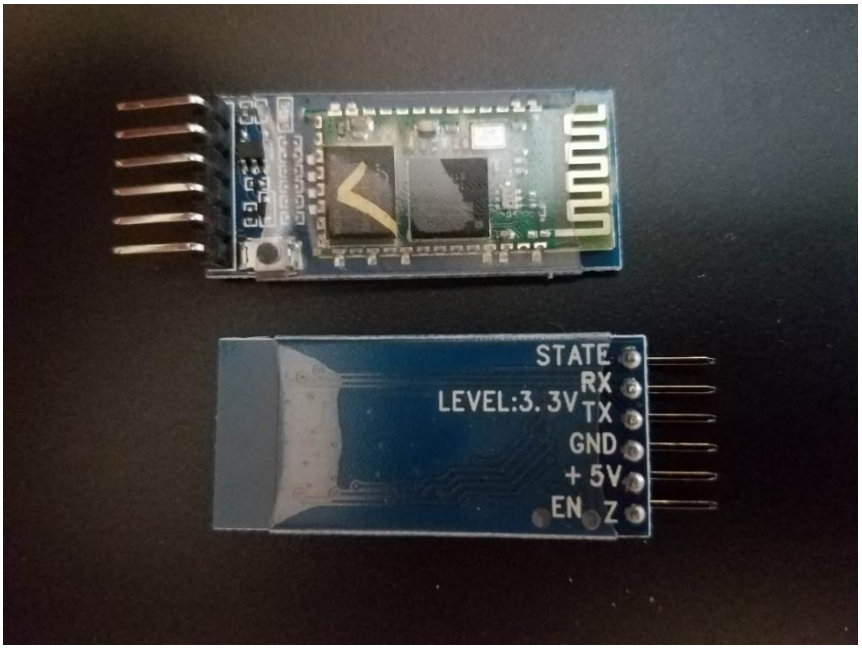


Figure 5.7: HC-05 module

Sending sensor data from Bluetooth to communicate is an uncomplicated process that requires few simple tools to conduct the experiment, which are microcontroller, Bluetooth module, USB cable, breadboard, and sensors. The microcontroller used for

the experiment will be the Arduino Uno, the Bluetooth module used will be HC-05, and all this will be connected to any dedicated computer to code the project. The experiment will require multiple procedures to follow right down from the wiring, Bluetooth connectivity, the coding process, transmitting data, and troubleshooting to have the electronic devices function, and the steps are as follow:

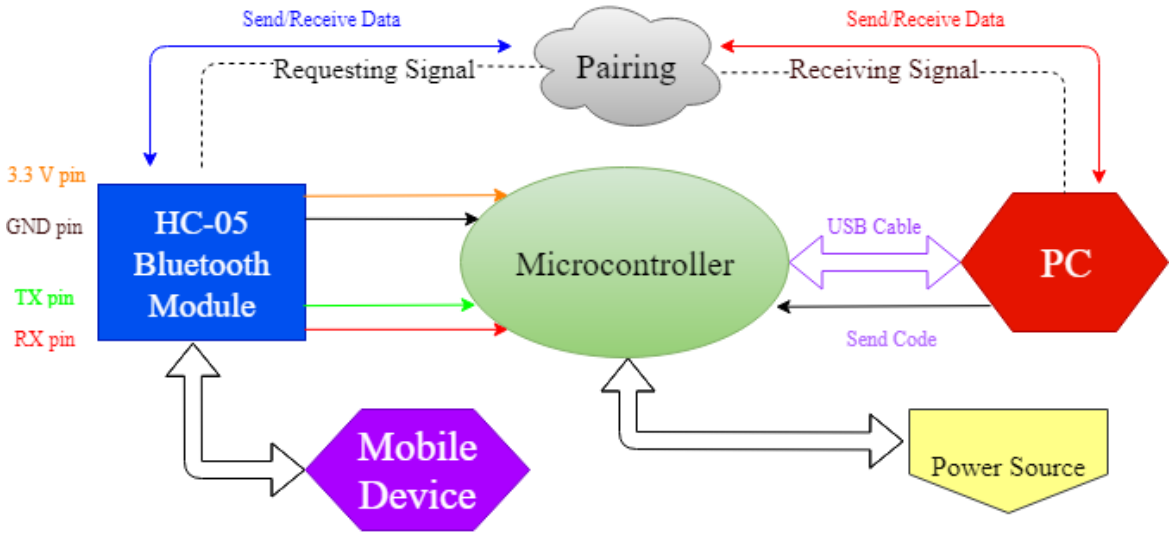


Figure 5.8: Datapaths

- **Circuit Setup**
  - The Bluetooth module (HC-05) will be connected by having the voltage lines through a voltage divider, that way the module does not get burned and fry the circuitry
  - The wire will be connected to a line, but because of the voltage divider, 5V may not be sustained in the line to pass through so it does not damage the module
  - On safer line to connect, the module can be connected to a 3.3V line, unless more power is required for other inputs
- **Bluetooth Connection**
  - The next step is pairing the Bluetooth module with any computer with built-in Bluetooth capabilities, otherwise an external dongle can be implemented
  - Connecting the Bluetooth can be done on the computer by going to the settings, locating the device, detecting the module, and pair with a pass code
  - Going to the main settings of the computer can be followed as such: Control Panel, Hardware and Sound, Add a Device
  - Detecting the HC-05 and typing a code of 1234 or 0000 to begin implementing to finish the pairing

- **Arduino Code**
    - From the sketch provided of the design to implement the experiment for proper functionality
    - Finding the correct ports of the board to connect the Bluetooth module can be found in the settings of the IDE
    - Once the correct port and board is selected, the TX and RX pins can be disconnected after uploading the code
    - Otherwise an error will occur when the COM port is busy if not disconnected after the code is uploaded
    - When uploading the code is successfully completed, reconnect the TX and the RX pins to the original port
- **Receiving Data**
    - After the code is uploaded to the board, in this step, the power source can be used to connect the board
    - After the code is uploaded and power source is used, USB cable can be disconnected from the board to the PC
    - This exhibits the step that no longer needs the PC to power the device and confirms that uploading the code is finished to use the power source on the board
    - Also showing that the sensor can function by remotely gathering the data and ready to transfer to the PC through Bluetooth
    - Notification setting can be set in order to see if Bluetooth and sensors are functional
    - Locating the COM port of the Bluetooth module can be set and display on screen on whether the data of the sensor is being send
    - An incoming and outgoing COM port can be set to signify sending and requesting data
- **Troubleshoot**
    - Data can be fully functional when the serial can be monitored on screen to show notification is set
    - In the case of data having delay and takes more time to successfully connect, this means that an error may occur in the process
    - Bluetooth communication problems occur due to signal interference, and can be avoided to ensure secure and firm signal
    - Rechecking the TX and RX pin connection to ensure functional communication
    - Changing to different voltage so that power can be properly distributed, connecting Vcc to 3.3 V and not 5 V
    - Loose connection can be an occurrence and stops the Bluetooth module from working, this can be seen on the module by checking if the LED blinks continuously to indicate if it is being powered
    - Disconnecting and reconnecting of the Bluetooth module to check by following the procedures previously listed and see if communication can function properly
    - Bluetooth module can be out of range from the PC
    - These steps followed can ensure simple Bluetooth experimentation

5.2.2.8 Bluetooth Breadboard Experiment

To test out the experiment to determine whether the Bluetooth module is fully functional, the module is then connected to the microcontroller to be programmed. The pins used on the Bluetooth module will be the Vcc, GND, TX, and RX, wired to the microcontroller to begin testing to ensure that it gives off signal to be detected to other devices. The Bluetooth module will begin blinking once it is connected and powered by the microcontroller, the module will continue to blink until it is properly setup to communicate with a mobile device. Connecting the Bluetooth module to the microcontroller requires proper connections of wiring, and an LED will be used to serve as an indicator, which can be used on a programmed to set other forms of notification by exchanging data. The proper wiring of the module and microcontroller can be set as follow:

Bluetooth Module to Microcontroller Analog Pins:

- **Vcc → 5V / 3.3V**
- **GND → GND**
- **TX → RX**
- **RX → TX**

The wire connection process of the Bluetooth module to the microcontroller functions in a master and slave device connection, where the master gives commands to external device to follow, in a transfer and receiving data process. Following the wiring connection, to power up the device it must be connected from the Vcc of the module to either 5V or 3.3V of the microcontroller pin, as well as ground to ground pin. The next wiring is to not be confused as to connect the wires of the same pin name, but rather of the opposite pin names, where TX pin is the transmitter serial data that gets connected to the RX pin which is the receiver rerial data. The TX and RX functions by having one end, the master uses TX to transmit data to the Bluetooth module, and begins by sending back data to respond which goes to the RX of the master where it receives data for communication. On the Bluetooth module side, it will blink continuously unless a connection is establish on any device, this confirms that the communication on the module is successfully established, and as the slave device, commands can be given to this from the master to exchange data to operate with other external connection either wired of wireless.

LED to Microcontroller Digital Pin:

- **Cathode (short pin / Negative) → GND**
- **Anode (long pin / Positive) → D2 (any digital pin is fine)**

This connection of an LED will serve as an indicator to recognize the exchange in data from master to slave device and can be modified in many ways to be set as notification module. The red LED on the breadboard will be used as an example experiment that

can be controlled from a mobile device, which is done through digital communication. Unlike the analog pins, which requires external parts to be controlled physically, the digital pins allows more intricate process to light up an LED by creating a mobile application that allows data to send from that app to the Bluetooth module, then the microcontroller, and then finally giving commands to the LED to light up.

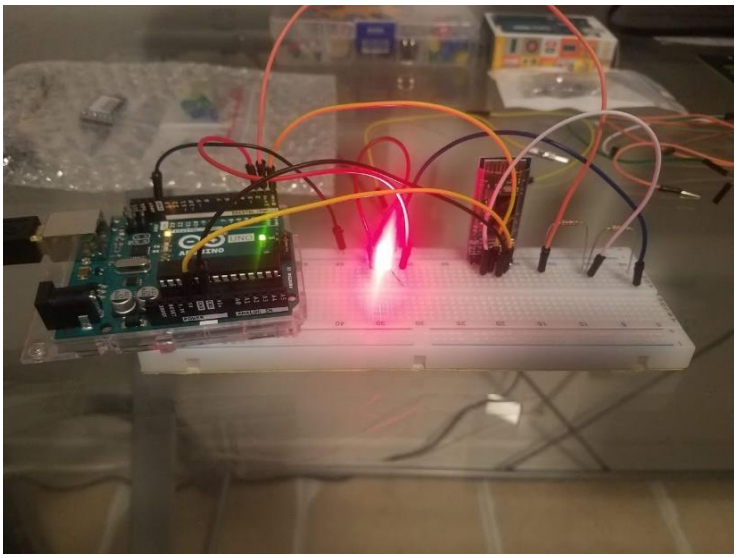


Figure 5.9: Bluetooth module test circuit

5.2.3 Wi-Fi Module

Wi-Fi connection is beneficial in wireless communication and has more capabilities than Bluetooth to work with due to having variety of access and more powerful features. What makes Wi-Fi stands out more than Bluetooth is that accessing data can be done remotely, rather than in proximity of a device as compared to Bluetooth. By having access to data online and information stored in a database, this gives Wi-Fi more flexibility in terms of wireless communication instead of limited by distance of a device.

5.2.3.1 NodeMCU ESP8266

This module is built as a Wi-Fi module, but it is a microcontroller on its own that is developed by Espressif Systems based in Shanghai. The ESP8266 module comes in different types ranging from ESP8266-01 and ESP8266-12 while all the modules from this line up has the same processor while only differentiating from each is the breakout board. Wi-Fi modules are important to connecting to the internet, although many equipment on its own are not able to connect to the internet, the capabilities to do so requires having a chip that enables that functionality [154].

Pin Configurations

- **16 GPIO pins:** PWN pins
    - Control peripherals such as sensors, LEDs, switches, etc.
- **ADC (Analog-to-Digital Converter):** channel access through A0

- **SPI:** 4 pins (SCK, MISO, MOSI, CS)
    - Communication through UART
- **I2C:** accessible support in internal pins
    - Master-Slave device
- **UART:** 2 main interfaces
    - **RXD0 and TXD0:** uploading code into the module
    - **RXD1 and TXD1:** can be used either module or microcontroller

Specifications

- **Operating Voltage:** 5V
- **Flash Memory:** 4Mb
- **Processor:** L106 32-bit
- **CPU Speed:** 80-160 MHz
- **RAM:** 32K + 80K
- **GPIO:** 16
- **ADC:** 1 and 10-bit

What makes this Wi-Fi module important in the ESP line-up as microcontroller of its own series, is that it can be built-in to Bluetooth module combo with full functionality. The benefits of having Wi-Fi and Bluetooth while having the purpose of high-performance with low power consumption all under one single chip module with full pin configurations. What makes the ESP8266 so important is that it has the same built-in features to the newer NodeMCU development board, the ESP32 with Wi-Fi and Bluetooth combo module.

Figure 5.10: NodeMCU

5.2.3.2 NodeMCU ESP32

Another type of NodeMCU development board is the ESP32, which has Bluetooth 4.0 and BLE that is also developed by based on Espressif Systems based in Shanghai

similar to the **ESP8266,** while the ESP32 module also works as both **Bluetooth and Wi-Fi combo module**. This would be beneficial for projects with high performance and ultra-low power consumption, the ESP32 is an integrated dual-core processor chipset with firmware updates along with an SDK that supports fast on-line programming that has available open source toolchains [164]. This module can be used for many purposes ranging from video streaming, Wi-Fi and Bluetooth enabling devices, home automation, mesh network app, hardware engineers, software engineers, that can provide solutions to develop with wireless implementation. It is designed to be the optimal module with a single chip capable of 2.4GHz from TSMC ultra low power 40nm technology to be compact with full functionality of both Bluetooth and Wi-Fi that is well optimized for power, RF, robustness, versatility, reliability, application, and profile featuring performance to match all category. The module includes multipurpose functionality that can be used from software to hardware interface with specifications as follow [154]:

Specifications

- **Processor:** Tensilica Xtensa 32-bit LX6 microprocessor
    - **Core:** 2 / 1 (depending on variation)
    - **Clock Frequency:** up to 240 MHz
    - **Performance:** up to 600 DMIPS

- **Wireless Connectivity:**
    - **Wi-Fi:** 802.11 @ 2.4 GHz up to 150 Mbit/s
    - **Bluetooth:** version 4.2 BR/EDR and Bluetooth Low Energy

- **Memory:**

    - **Internal:**
        - **ROM:** 448 KiB (boot and core functions)
        - **SRAM:** 520 KiB (data and instructions)
        - **RTC slow RAM:** 8 KiB (co-processor accessing)
        - **RTC fast RAM:** 8 KiB (data storage and main CPU)
        - **eFuse:** 1 Kibit (application and flash-encryption)
        - **Embedded Flash:** IO17, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1 (internal flash connection)

    - **External:** flash and SRAM up to 4x16 MiB
        - 8 Mib SRAM hardware encryption
        - Embedded flash no support for mapping with peripheral

- **Peripheral input/output:** capabilities of parts can be compatible with listed
    - Capacitive touch
    - Analog-to-Digital Converter

- Digital-to-Analog Converter
- Inter-Integrated Circuit
- Universal Asynchronous Receiver/Transmitter
- Controller Area Network 2.0
- Serial Peripheral Interface
- Integrated Inter-IC Sound
- Reduced Media-Independent Interface
- Pulse Width Modulation

- **Security:**
  - **IEEE 802.11 standard security:** WFA, WPA/WPA2, WAPI
  - **Secure boot**
  - **Flash encryption**
  - **1024-bit OTP, up to 768-bit**
  - **Cryptographic hardware acceleration**
    - AES, SHA-2, RSA, elliptic, curve cryptography (ECC), random number generator

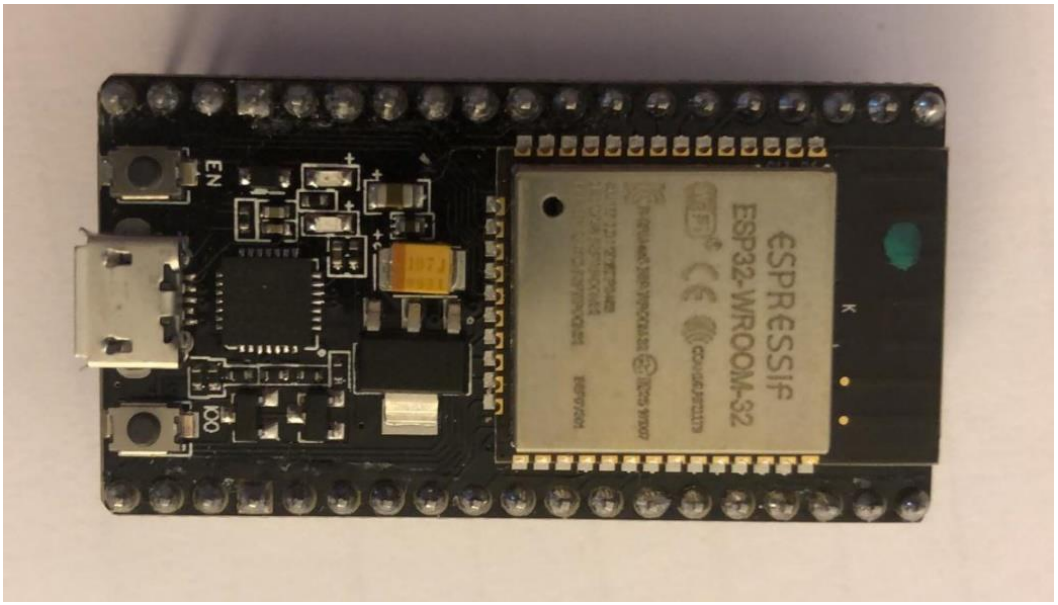


Figure 5.11: ESP32

The hardware built-in the ESP32 has the same design as the ESP8266 Wi-Fi module, while ESP32 has Bluetooth capabilities along with much more pins that can be used to solder on to boards. The developing boards of these modules have more quality electronic parts for precise soldering. While the ESP32 has different variations of technical manufacturing from **D0WD, D2WD, and S0WD**, it all has the same primary representation and ship design. One of the main important features of this module is the power consumption, different **power modes** can be operated from each resource.

Power mode	Active	Modem-sleep	Light-sleep	Deep-sleep	Hibernation
CPU	ON	ON	PAUSE	OFF	OFF
Wi-Fi / Bluetooth	ON	OFF	OFF	OFF	OFF
RTC Memory & Peripheral	ON	ON	ON	ON	OFF
ULP-coprocessor	ON	ON	ON	ON/OFF	OFF

Table 5.1: Power modes

5.2.3.2.1 ESP32 Wireless Capabilities

The main feature that will be used in our project is the wireless capabilities to connect to the mobile device via Bluetooth so that it can send data through wireless communication. Other wireless features, such as Wi-Fi also follows similar protocols and procedures for communication, while Wi-Fi allows capabilities of online and remote access. The features in this module has link controller and baseband that carry out protocols along with other low-level routines. Signal protocols such as modulation / demodulation, packets processing, bit streaming processing, frequency hopping, that is fully capable in Bluetooth version 4.0 and BLE. The main hardware features provide interfaces as follow, **UART of 4 Mbps, SDIO/SPI HCI, I2C interface hosting, PCM.I2S audio**. Connections to send data functions by having **device discovery, multi-connections, scan, asynchronous data, broadcast encryption, sniff mode, ping** are made sure that the device is discovered while transmitting and receiving signal.

5.2.4 Communication

In order to have hardware operate with software, the communication process is vital to integrate the utilizing of data transfer across the MCU and the Wi-Fi module, which then in terms send data to the database. In the project, the FSR would read the data from when a person sits on the chair where the sensors are placed or apply pressure, which activates the readings that would be read by the MCU, upon the activation and data going through the MCU, those data gets transmitted to the Wi-Fi module. This is done by implementing serial communication between the MCU and the Wi-Fi module, which is having wired connection of both devices, so the process of transferring data is established. Once data is received from the Wi-Fi module, and after a wireless connection is accomplished, the data that was received gets transmitted online to a database, where it can be stored so that it can be viewed and recorded.

5.2.4.1 Serial Communication

To establish the serial communication between the MCU and the Wi-Fi module, it is important to use the appropriate pins to connect both devices [157]. Assigning the pins on the development board and PCB itself aren't the only part necessary to connect both devices as the hardware implementation, software is also required to implement to perform the communication. Software implementation is established by using the RX and TX pins from both the MCU and Wi-Fi module, along with an important library that has to be included in the IDE, if not already there. The SoftwareSerial library functions to use software as digital pins to replicate the functionality from the hardware[158].

Aside from the serial communication, within the IDE itself, setting the baud rate is important to match the readings and reduce the amount of latency that is transmitting and receiving between both devices. The baud rate for both to operate at a desirable rate with low latency, would be at 115200, because the Wi-Fi module of choice, which is the ESP8266, must function at that rate due to the hardware limitation. With both devices using the RX and TX pins, it is important to note that the wiring connection should be opposite of each other when connecting between development boards where one pin operates as the transmitting pin, the other as the receiving pin [159]. This is done so by using the interface known as Universal Asynchronous Receiver-Transmitter, or UART for short.

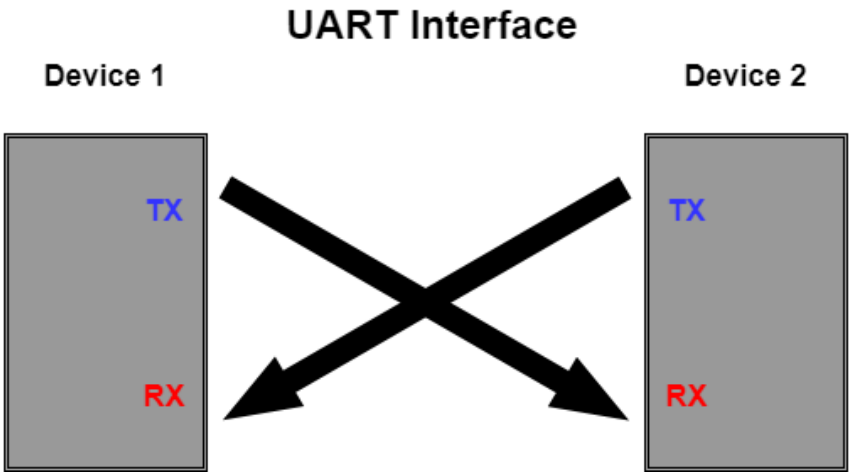


Figure 5.12: UART Interface Diagram

5.2.4.2 Software Serial Library

This library is important to include in the coding process, as the hardware comes with built-in to have support for the pins that is used, whether TX and RX pin, or digital pins which would suffice for the operation of the communication. The UART allows the MCU to receive and transmit data even while it continues to work other separate task of operations, so that transferring data does not get conflicted in the process, as long as the serial buffer has space of 64 bytes for the communication.

5.2.4.3 Wireless Communication

The next important method of communication is wireless connectivity via Wi-Fi that takes data from the MCU, to the Wi-Fi module, then finally to the database. Establishing connection of one point to the other end, requires the use of fullduplex connection of a client and server to act and send HTTP request from the client, then the server side would respond to the request[160]. This method of transmitting and receiving data, where the server side performs the action that takes the data to manipulate its inputs and outputs from an HTTP request on the client side. This is useful when sending real time data of string that is coming from the sensors, then received by the MCU, so the real time process remains prevalent while the Wi-Fi module acting as the client to send over to the server. Another method of client and server HTTP request is having the client extract useful and meaningful data, in which another method that was used in the project. The client HTTP request to extract data from the server was used to give commands to a vibration module, where a Boolean command is set activate it ON or OFF upon request from the server in real time. The HyperText Transfer Protocol (HTTP) is the protocol in which the world wide web to define messages, format and transmit actions from the browser to the user [161]. This method is the foundation of the internet and the primary source of exchanging data on the web with the client and server protocol. The client side that establish connection would be the ESP8266 Wi-Fi module to send request to the server, which would be the Firebase database by Google.

5.2.4.4 Firebase Database Connection

Firebase acts as the database for the project to store the data coming from the ESP8266 Wi-Fi module, which is cloud-hosted and access from all clients using Realtime synchronization [162]. Using Firebase is an integral part of the project because while data is secured, it can also be access by mobile devices as the client where the user can view the necessary data. The important of Realtime is vital in the project, because data coming from sensors are being read at a timed interval to continuously record data for posture measurements, which is activated when the user sits on the chair. Not only is it important for sending data from the ESP8266 to Firebase, but receiving commands also plays an important role, which Realtime benefits the process. Firebase sends a command to the ESP8266 to activate the vibration module to act as a physical notification device which can be set by the user.

5.2.4.5 Firebase Library

The library is required so that the ESP8266 can operate and send data to Firebase, while Wi-Fi SSID (Service Set Identifier) and password is necessary in the coding to connect to the internet, having Firebase credentials is also key in accessing the location on the cloud server. Firebase credentials require the host and authentication implemented into the coding, which is shown in Chapter 6: Code and Schematics section under 6.4 Serial and Wireless Communication Code.

5.2.4.6 Wi-Fi Implementation

The wireless communication of choice for the project is through Wi-Fi due to the powerful capabilities and numerous functionality that can be used. Not only does using Wi-Fi itself allow more operations, but the support of the communication that gives Wi-Fi more accessibility. Being able to access information remotely through online by storing data from a database, along with controlling devices to give commands remotely. This is a powerful feature to have by using Wi-Fi as the wireless communication, which not only requires hardware but also software to implement. The database that is used in this project is the Firebase database by Google, by allowing Wi-Fi connection to establish and send data to Firebase, the communication process is then implemented.

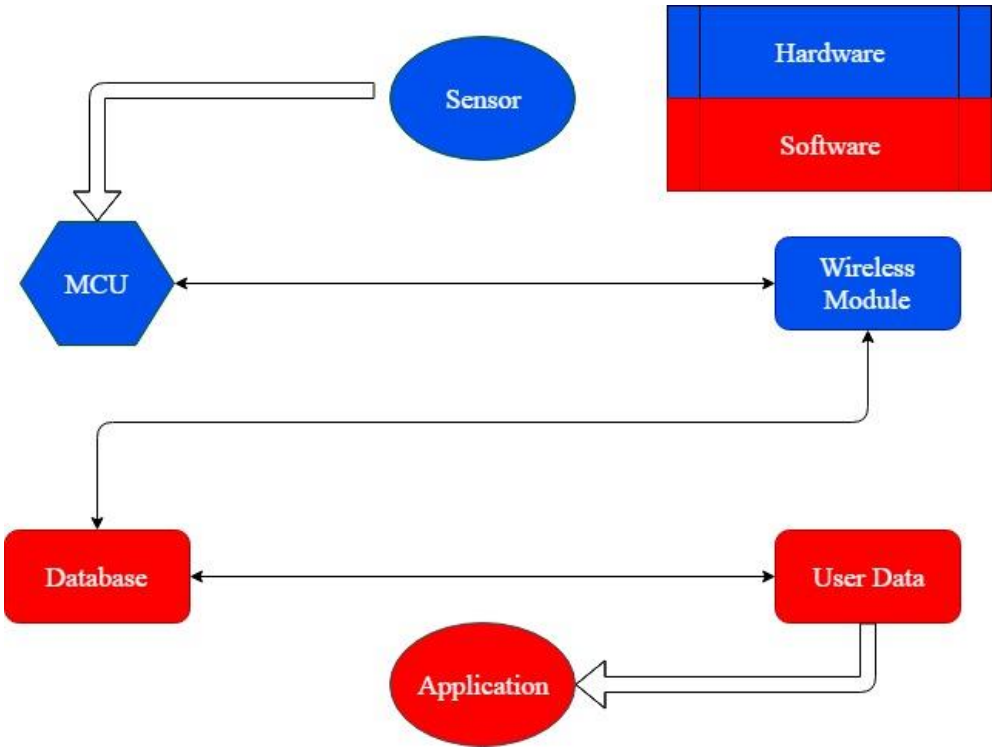


Figure 5.13: Communication Diagram

The diagram displays the process of how the hardware and software operate in tandem while the functionality activates. Once pressure is applied to the sensors, the MCU and Wi-Fi module would begin the communication process so that the data is transferred between both devices. On the hardware side, the entire development board of the NodeMCU ESP8266 Wi-Fi module gets placed on the PCB connecting to GPIO pins for communication, along with 3.3V and GND for powering on the module. The ESP8266 only needs 3.3V to power on instead of 5V, because using that much voltage could short circuit the device. Having the micro USB port is also useful for reading data, since that also functions to upload the code into the ESP8266, while displaying the serial monitor from the IDE on the computer, which allows viewing of the data that is being transmitted from the MCU.

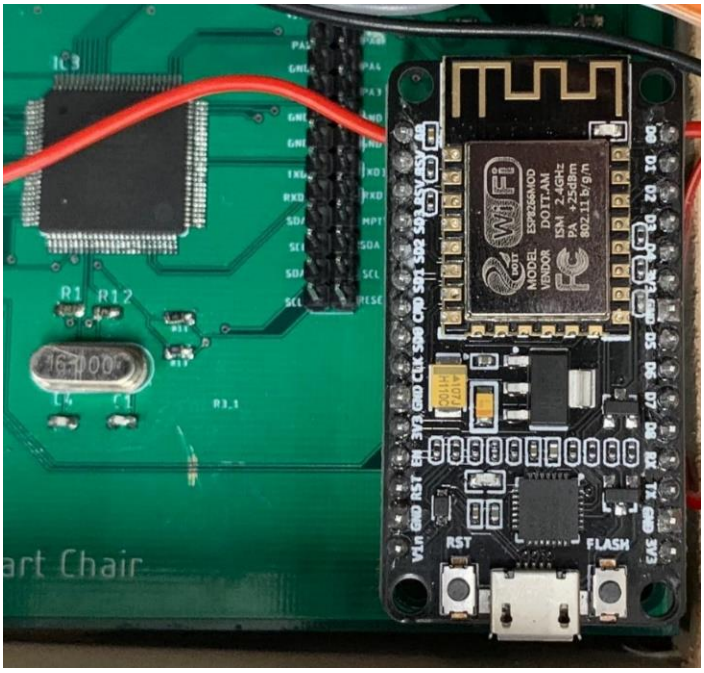


Figure 5.14: MCU Integration

5.3 Force sensor test experiment

In order to verify we received functional parts we setup a simple program on our development board that would spit out a value, up to 1023, based on the voltage coming to an ADC channel, on a 5V supply to our sensor. In doing this we found that the development board offered a pull up resistor option that allowed us to directly connect our sensor without a voltage divider. Upon initial testing using this methodology we found a variance of approximately eight percent for our sensors, given the same load. This is within manufacture specification, so when the sensor become permanently mounted, we will calibrate them to within one percent of each other. The image below depicts the final layout of the sensors on the chair in the setup which they were tested.

Figure 5.15: Sensor Layout

5.3.1 Conditioning Sensors

To further develop the implementation techniques for the force sensors, we conditioned them. This is suggested by the manufacturer as it helps with usability and reliability. Conditioning also allows us to acclimate the sensor to the typical load it will see over the span of its use. Conditioning, as previously described, involves overloading the sensor repeatedly, one hundred and ten percent to be precise while tuning the input voltage by observing the output voltage. This is followed by applying a typical load range expected for the sensor to operate on. A few key takeaways of this process based on using the voltage divider method for powering the device include ensuring the mounting surface is flat and clean. Not having proper mounting results in the sensor lifting, which results in poor sensor readings and calibration, as it calibrates to the rigidity of it's base. Another point to be mindful of is that in application the load is distributed more evenly across the sensor surface area, not like a point load where it is concentrated. There are several methods to remedy this, one is referred to as puck loading, where a puck shaped device covers most of the sensing area then a larger object is placed on top. This ensures any load in contact with the object above the puck can be fully sensed. Sensors that are conditioned offer far less discrepancy versus non-conditioned, where variations can occur in excess of fifteen percent.

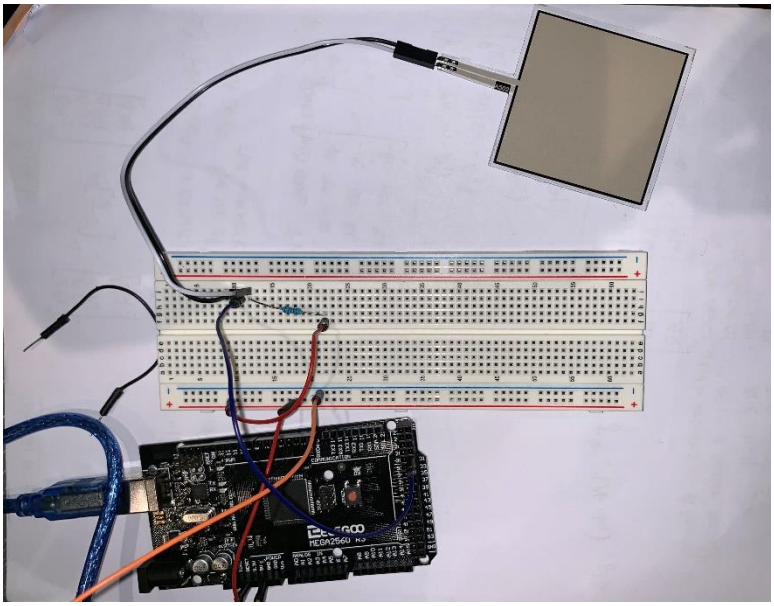


Figure 5.16: Sensor conditioning on breadboard

5.3.2 Embedded Sensor Code

In this section, we will discuss how the sensors are programmed and the logic used in our final design. We begin by initializing several variables to call later, including the labeling scheme for the sensors as they are placed on the chair. We then enable interrupts on the MCU and set the parameters for timers 1 and 3. Timer 1 is used to control the data read and write sequence from the sensors to the WiFi module. It also has the limits defined for triggering the sensors. Timer 3 allows the get up flag to be

passed to the database by checking the status of the sensors and using an iterator to check how long the sensor's are triggered for. In our final design the iterator for timer 3 is set to 30 seconds or roughly 3 data cycles. With the end to end delay of the system, we see this 30 second stretched to roughly a minute. This get up is triggered as a vibrate as well as notifications on the application, discussed in the corresponding section.

5.4 Proximity Sensor testing

Verifying functionality and compatibility of a component is key in determining if our design is on track for completion and if components will operate as expected. Sending and receiving a signal from the proximity sensor is fairly simple provided the manufacture test instructions. Implementing into our design requires an interrupt sequence, sensing interval, and a calibration operation. These three routines will allow the device to reliably sense where the user's back is in relation to a calibration region of the chair. The interrupt sequence will allow us to put the system in a lower power mode when no one is present while offering a wakeup routine once the user sits down.

Implementing with development board

To test we begin by connecting the sensor to a breadboard and using 3 lead connectors to connect this to the *Elegoo* development board. Using the *Arduino IDE*, we can use a few lines of code to ensure the sensor is receiving a supply signal and using a print function, see that the device is returning a signal. Ensuring calibration is simply done by using a ruler and set and measure predetermined distances. We can verify the sensing field by moving vertically and horizontally, maintaining the same distance from the sensor. calibration of the sensor is achieved by modifying the time interval used to generate the pulse. Ultimately, this feature of our project was removed due to time and usefulness based on the data being collected [155].

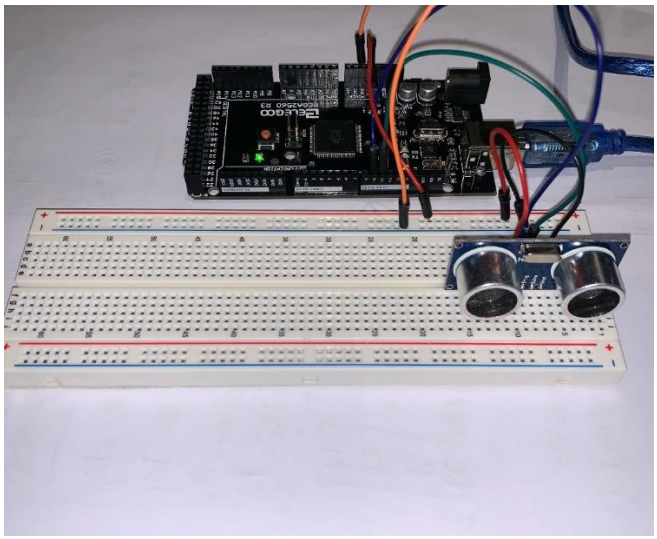


Figure 5.17: Breadboard testing of proximity sensor

5.4.1 Enclosure for PCB

The hardware design for Smart Chair is actual humble, but when you look at the inside of the case system design it gets more subjective multipart. The outside of the box has Smart Chair written on it and is designed using one of the box maker that is the most popular for all senior design project, called Makercase. The external part of the Smart Chair case we represented as the shell. The shell will be inches where the box width is 5.75 inches, box width 3.75 inches and the box depth will be 4.5 inches. The box will be made in 6 different sections that was collected to make the complete shell that we needed to use to store all the material. The first part will be the bottom where it has enough room to keep all the eletrical and electronics components that will be use for the hardware in this project. The base that we consider as the basement of the box same as the house when we built, it has the basement that strong enough to support all the floors that will be on top and the material that will protect it against a collapse. The bottom which is the the basement will be large and strong enough to hold the printed circuit board PCB where it will be provided the continuity of the power to the system.

It has a battery inside the box closed to the PCB that will provide power to the system in case we are out of electricity and the Samrt Chair is not connected to the wall that will provided the power to circuit board and the rest of the electrical components. The box has some have to be glued to protect the components against water because it will electricty as we know that water and electricity do not match. It has a hole the left corner that will be available for the wires that come from the wall power supply to connect to board and continuiying to the rest of the material that will use wires such sensors. As we know, when we have a closing box that has electrical and electronics components that function at the same time, we do not want a heating problem. We decided to add a ventilation system inside the box and in the right corner with some holes that are enough for the system to receive cool air and left out hot air. the box has a top to cover the box that will fit under the chair for the security if the box and all the components. It is secure enough to prevent the falling and causing by any chances the shock of the electrical components that can cause some severe injures due to the electrical components that we have inside specially the battery that is composed with chemical components.

The second most important layer in the design is the material thickness. The chair by itself is made with good material that protected humans and makes them comfortable in the chair. The material that we used to make the project success are good quality of sensor that detected how people sit in the chair and sending signal to the software parts then let them know if they are in good positions. The material Thickness will be one quarter which is 0.236 inches of the wood that will uy from Home Depot which is not expensive will be using the laser that is available in Texas Instrument TI lab to cut the wood. First, we were thinking about a 3D plastic printing, but due to the cost of the material an dthe project is out of pocket. We select to use a wood instead that will be provided the same work and safety with less money.
The box has some edge finger joints in all of the sections that are used to join the box together and using wood glue to assmble them. We intended to paint the box black to

better conceal it. The sensors wiring with good enough care and good wires quality protect people against getting electrocuted, and the wires will be long enough to connect from the printed circuit board that are inside the box to the area that need the sensors to be. The box is replaceable in case there is something wrong and open for a professional when they want to fix something inside the pcb, changeable the battery, connected and disconnected the sensors in case of them need to be replaced. Majority of the sensors that are sensing information about the status of people sitting and when they have been sitting in the chair that means sensors should be placed in the right position to have access to send information. The box itself is totally removable so that the user will be have access to the battery system in case the battery need to be changed. Due to the box dimension it fits under the Smart Chair and has the dimension to fit all necessary components.

Initial Design Architecture

For the Smart Chair architectural strategy, it was recognized by our group to use a 3-D printing using a software that is on all of the computers at University of Central Florida called Solidworks. The reasoning being was one of us knew how to use it, but he is not in charge for the hardware parts when it involved engineering projects. The team member that is an charge of the harware does not know how to use the solidworks, that's why we change our mind from Solidworks to Makercase wich is the easiest box maker for all the designer that has not really in designer box for all the user specifically engineering.

The structure itself will embrace all of four separated slices that will be collected together to engender the Smart Chair collective composition. Preliminary from the base and up to all the four corners, an open box, a protection to cover the box. Individually of theses apparatuses will be a 3-D printed and accumulated once all our devices have been combined to their corresponding designed locations. The base edifice will be 6''x 3''x 6''. This measure serves as the foundation for the box that will held the printing circuit board PCB and the mechanical components. the right and the left corner that will be using 3''x 1.5'' x 3'' to have a regular box to serve as protection and secure the components. we have a laser cutter plan sttings that will cut the horizontal margins 0.25 inches, vertical margins 0.25 inches, line stroke width for vector cutting 0.01 inches and line color 0.02 inches. The text engraves also have 3'' x 4'' for the position and 0.70 inches for the sizes that we are using to engrave the text characters and the position that will be better to add the text with dimension that you do not need glasses to see. Below is an example how the structure will be for the laser cutting plans before the structure of the Smart Chair box the will use to secure the components.

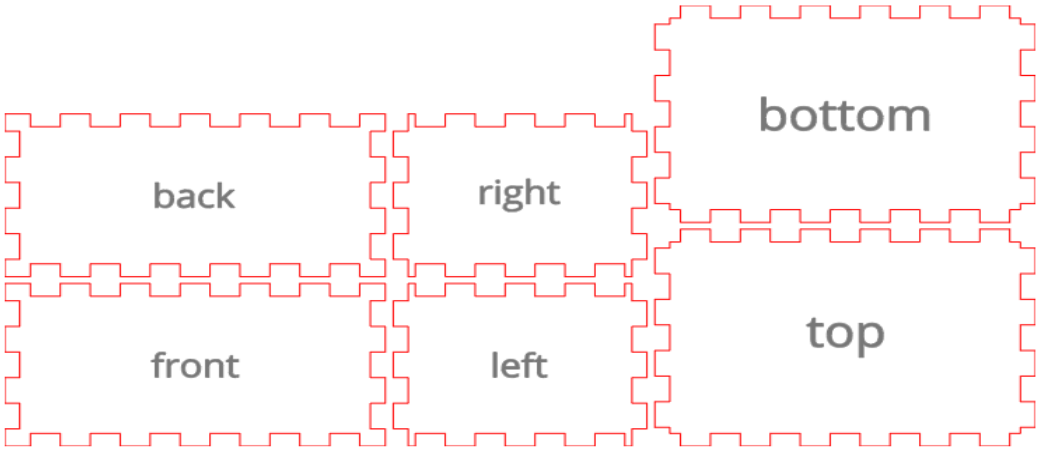


Figure 5.18: Laser cut diagram for housing

For the succeeding assemblage part, we put all of them together to have a box that have a rectangle form. The rectangle box has a hole in the middle of the left corner for that will be using for the wires that will connect to all the components. the holes will be 1 inche in diameter that is large enough for to use for the all the wires with a normal dimension but it can be challenging running the wires though the holes that will be connected from the sensors to the printing circuit board can change the design because it will be lengthened beneath hardware restrictions. Next addition to the box is the quiet ventillation fan that will be inside the box to prevent heating. We will use a laptop fan that has four pin connected to printing circuit board to cool down the board in case there is heating. The fan will be 5 Volts direct current that we have in the PCB and a current of 0.50 A. The model will be using in the design is AD6505HX-EEB [165] which is not going to use a lot of power, and it will fit inside the case to do the work that we need. The fan will comprehend and deliver a protective external security for the circuit that has electrical and electronics components. in the initial project, this appear like a prodigious clue because we do want the components either electrical and electronics to cool and use with their normal temperature so that they can give the amount of that we need to get and amount of time to protect them. Below is the assemblage of the structure that has shown the above figure [156].

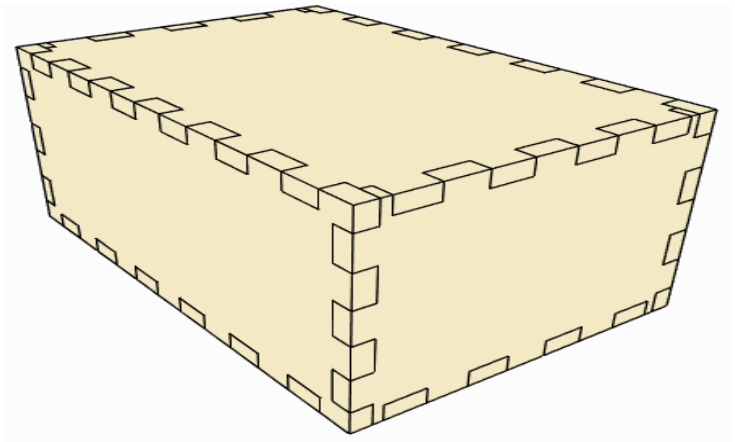


Figure 5.19: integrated housing

5.5 Milestones

In order to maintain a reasonable time table to achieve a successful design, we found it beneficial to come up with project milestones. Some of these have dates, while most have soft target deadlines just to keep the overview of the project in mind and have a metric to know where we stood.

Milestone Discussion

For the advancement of the project, we discussed the milestone every time when we meet to stay on track of the project. These momentous would be supplemented by a succession of dates in order make progress and get the project done on time. Senior design 2 was a little challenging because we did not have a full semester to complete the design. Instead of sixteen weeks, it was twelve weeks which required exceptional time management for the team.

The table below is a milestone overview of various aspects of the project which we felt were measurable and contributing factors to the success of the project. Some notable mentions include designing three pcb's, for which shipping constraints proved difficult to manage with overseas manufacturers. Accounting for the right quantity of small components (caps, resistors, etc.) early on in the design, when that many PCB revisions can occur is also crucial.

Milestone	Duration	Dates
Senior Design 1	16 Weeks	1/8 - 4/19
Project Idea Brainstorming	2.5 Weeks	1/8 - 1/23
Divide and Conquer Report	4 Weeks	1/23 - 2/23
60 Page Report	6 Weeks	2/23 - 3/29
100 Page Report	1.5 Weeks	3/29 - 4/12
PCB and Enclosure Designs	4 Weeks	3/29 - 4/22
120 Page Report	1.3 Weeks	4/12 - 4/22
Senior Design 2	12 Weeks	5/2 – 8/2
Construct Prototype	7 Weeks	5/1 – 6/19
Test Prototype	3 Weeks	7/1-722
CDR Presentation	1 Week	7/1-7/3
Final Report	2 Weeks	8/2
Final Presentation	1 Week	7/26

Table 5.2: Senior design milestones

Hardware

- Circuit Design & software simulation including calculations and filter/gain stages
- breadboard simulation with test parts
- Design PCB & simulate
- Order final parts
- Order PCBs
- Solder the parts on the PCB and reflow
- Test PCB

Software

- Storyboard the user interface
- Design Database
- Learn how to create a mobile application for android and learn how to use local storage.
- Create Database and populate with sample data.
- Create skeleton UI and get interaction with backend.
- Algorithm to analyze the data from hardware and to recommend posture adjustments.
- Communication with hardware to send commands to LEDs and receive sensor data.
- Finished version of UI.
- Fully functional application backend.

Integration

- Send and receive data via Bluetooth
- User setting controls LED's
- Read sensor data from PCB's

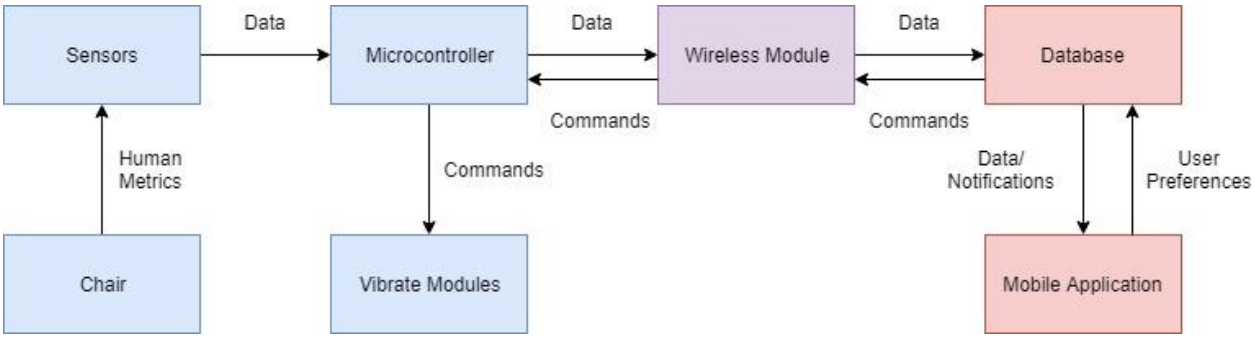


Figure 5.20: Project Management Visualization

6. Code and Schematics

This section of the document includes all of the code and schematics that were included in the final version of the project.

6.1. Embedded Hardware code

This section covers all the code required to operate the microcontroller as well as process the data coming from the sensors. Key sections of this code are labeled. They cover aspects including sending data to the esp module, receiving data from the esp module, toggling the vibrate modules based on condition, and defining the conditions for each position of leaning which read data. Some techniques employed in the execution of this code include interrupt routines, counters, iterator variables, parsing functions and variable type conversion operations.

```
#include <avr/interrupt.h>
#include <SoftwareSerial.h>
#include <String.h>
SoftwareSerial s(51,52);  //these are isp pins that work on the pcb int 1 & 2

//****************************

long count = millis();  //counter for 10 second data transmit interval
long getUp = millis();  //counter for 30 second get up notification
//define variables
//Large Sensors
volatile int FR ;   //A0
volatile int FL ;   //A1
volatile int BR ;   //A2
volatile int BL ;   //A3

//Small Sensors
volatile int SF ;   //A4
volatile int SB ;   //A5
volatile int SL ;   //A6
volatile int SR ;   //A7

volatile unsigned int vibeSet;  //check variable for vibrate on/off
volatile int userUp = 0;  //userUp flag

/* x & y points
measured values on chair if center is assumed to be zero--> will change once sensors permanently
mounted
Sensor Scheme
 1-FR,+x +y
 2-FL,-x +y
 3-BL,-x -y
 4-BR,+x -y
 5-SF,0 +y
 6-SL,-x 0
 7-SB,0 -y
 8-SR,+x 0
*/

//****************************

void setup() {

 s.begin(57600); //baud rate cannot go above 57600 b/c pwm will not work for vibrate module this
is for serial channels not serial comm
```

```
//initialize timer 1

noInterrupts();

TCCR1A = 0;
TCCR1B = 0;
TCNT1 = 0;

OCR1A = 62500;
TCCR1B |= (1<<WGM12); //CTC MODE
TCCR1B |= (1<<CS12);  //256 prescaler
TIMSK1 |= (1<<OCIE1A);  //enable timer compare interrupt

//enable timer 3
TCCR3A = 0;
TCCR3B = 0;
TCNT3 = 0;


OCR3A = 62500;
TCCR3B |= (1<<WGM32); //CTC MODE
TCCR3B |= (1<<CS32);  //256 prescaler
TIMSK3 |= (1<<OCIE3A);  //enable timer compare interrupt

interrupts(); //enable all interrupts

Serial.begin(9600);

  }

//timer compare interrupt service routine

ISR(TIMER1_COMPA_vect){

volatile signed int x1= 5;
volatile signed int x2= -5.5;
volatile signed int x3= -2;
volatile signed int x4= 1.5;
volatile signed int x6= -6.5;
volatile signed int x8= 6.5;

volatile signed int y1= 4.5;
volatile signed int y2= 4.5;
volatile signed int y3= -2.5;
volatile signed int y4= -2.5;
volatile signed int y5= 6;
volatile signed int y7= -6;

char toSendX[6];
char toSendY[6];
signed int temp;
signed int number;
signed int x;
signed int y;

signed int cogx = ((x1*(FR)+x2*(FL)+x3*(BL)+x4*(BR)+x6*(SL)+x8*(SR))/(6));
signed int cogy = ((y1*(FR)+y2*(FL)+y3*(BL)+y4*(BR)+y5*(SF)+y7*(SB))/(6));



//leaning forward
if(((FL)>=100)&&((FR)>=100)&&(count>=8854)){  //ADC check condition


//Serial.println(FR);
//Serial.println(FL);
//Serial.println(BR);
//Serial.println(BL);
//Serial.println(vibeSet);
Serial.print(count);
Serial.println(getUp);
```

```
   userUp = 0;
   x = cogx;
   y = cogy;


    if(x<0){
  toSendX[0] = '-';
  toSendX[4] = ';';
  toSendX[5] = 'x';
  x *= -1;
          }

     else{
    toSendX[0] = '0';
    toSendX[4] = ';';
    toSendX[5] = 'x';
         }
    for( int i=3 ; i>=1;i--){
      toSendX[i]=(char)((x%10) + 48);
      x = x/10;
                              }

s.flush();    //clear buffer
         Serial.print("X: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendX[i]);
      s.write(toSendX[i]);
                           }
   Serial.println();

    if(y<0){
      toSendY[0] = '-';
      toSendY[4] = ';';
      toSendY[5] = 'y';
      y *= -1;
           }

    else{
      toSendY[0] = '0';
      toSendY[4] = ';';
      toSendY[5] = 'y';
        }

    for( int i=3 ; i>=1;i--){
      toSendY[i]=(char)((y%10) + 48);
      y = y/10;
                              }

s.flush();    //clear buffer
          Serial.print("Y: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendY[i]);
          s.write(toSendY[i]);
                           }
    Serial.println();
        count = 0;
}


//  //leaning backward
 else if(((BR)>=100) && ((BL)>=100)&& (count>=8854)){


//Serial.println(FR);
//Serial.println(FL);
//Serial.println(BR);
//Serial.println(BL);
//Serial.println(vibeSet);
```

```
Serial.print(count);
Serial.println(getUp);

   userUp = 0;
   x = cogx;
   y = cogy;


    if(x<0){
  toSendX[0] = '-';
  toSendX[4] = ';';
  toSendX[5] = 'x';
  x *= -1;
          }

     else{
    toSendX[0] = '0';
    toSendX[4] = ';';
    toSendX[5] = 'x';
         }
    for( int i=3 ; i>=1;i--){
      toSendX[i]=(char)((x%10) + 48);
      x = x/10;
                             }

s.flush();    //clear buffer
         Serial.print("X: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendX[i]);
      s.write(toSendX[i]);
                          }
   Serial.println();

    if(y<0){
      toSendY[0] = '-';
      toSendY[4] = ';';
      toSendY[5] = 'y';
      y *= -1;
           }

    else{
      toSendY[0] = '0';
      toSendY[4] = ';';
      toSendY[5] = 'y';
        }

    for( int i=3 ; i>=1;i--){
      toSendY[i]=(char)((y%10) + 48);
      y = y/10;
                             }

s.flush();    //clear buffer
          Serial.print("Y: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendY[i]);
          s.write(toSendY[i]);
                         }
    Serial.println();
        count = 0;
 }

//leaning left

 else if(((FL)>=100) && ((BL)>=100)&& (count>=8854)){
  //Serial.println(FR);
  //Serial.println(FL);
  //Serial.println(BR);
  //Serial.println(BL);
  //Serial.println(vibeSet);
```

```
  Serial.println(count);
  Serial.println(getUp);
  userUp = 0;
  x = cogx;
  y = cogy;

    if(vibeSet == 1){
      analogWrite(10,140);
                    }

    else if(vibeSet == 0){
      analogWrite(10,0);
                          }

    if(x<0){
      toSendX[0] = '-';
      toSendX[4] = ';';
      toSendX[5] = 'x';
      x *= -1;
           }

    else{
      toSendX[0] = '0';
      toSendX[4] = ';';
      toSendX[5] = 'x';
        }

    for( int i=3 ; i>=1;i--){
      toSendX[i]=(char)((x%10) + 48);
      x = x/10;
                              }
s.flush();    //clear buffer

  Serial.print("X: ");

    for(int i=0;i<=5;i++){
        Serial.print(toSendX[i]);
        s.write(toSendX[i]);
                          }
    Serial.println();

    if(y<0){
      toSendY[0] = '-';
      toSendY[4] = ';';
      toSendY[5] = 'y';
      y *= -1;
            }

    else{
      toSendY[0] = '0';
      toSendY[4] = ';';
      toSendY[5] = 'y';
        }

    for( int i=3 ; i>=1;i--){
      toSendY[i]=(char)((y%10) + 48);
      y = y/10;
                              }
s.flush();    //clear buffer

  Serial.print("Y: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendY[i]);
      s.write(toSendY[i]);
                          }
    Serial.println();
    count = 0;
 }

//leaning right
```

```
else if(((BR)>=100) && ((FR)>=100) &&(count>=8854)){
  //Serial.println(FR);
  //Serial.println(FL);
  //Serial.println(BR);
  //Serial.println(BL);
  //Serial.println(vibeSet);
  Serial.println(count);
  Serial.println(getUp);
    userUp = 0;
    x = cogx;
    y = cogy;

     if(vibeSet == 1){
      analogWrite(11,140);
                      }

     else if(vibeSet == 0){
      analogWrite(11,0);
                           }

     if(x<0){
      toSendX[0] = '-';
      toSendX[4] = ';';
      toSendX[5] = 'x';
      x *= -1;
             }

    else{
      toSendX[0] = '0';
      toSendX[4] = ';';
      toSendX[5] = 'x';
        }

    for( int i=3 ; i>=1;i--){
      toSendX[i]=(char)((x%10) + 48);
      x = x/10;
                               }

s.flush();    //clear buffer

    Serial.print("X: ");

    for(int i=0;i<=5;i++){
      Serial.print(toSendX[i]);
      s.write(toSendX[i]);
                          }

   Serial.println();

    if(y<0){
      toSendY[0] = '-';
      toSendY[4] = ';';
      toSendY[5] = 'y';
      y *= -1;
            }

    else{
      toSendY[0] = '0';
      toSendY[4] = ';';
      toSendY[5] = 'y';
         }

    for( int i=3 ; i>=1;i--){
      toSendY[i]=(char)((y%10) + 48);
      y = y/10;
                               }

s.flush();    //clear buffer

     Serial.print("Y: ");
```

```
    for(int i=0;i<=5;i++){
      Serial.print(toSendY[i]);
      s.write(toSendY[i]);
                          }
    Serial.println();
    count = 0;
}

else if((((FR)<20)&&((FL)<20)&&((BR)<20)&&((BL)<20))){
  userUp = 1;
  s.write('*');
  userUp = 0;
  s.flush();
}

}


 ISR(TIMER3_COMPA_vect){

  if((((FR)>=20)|((FL)>=20)|((BR)>=20)|((BL)>=20))&& (getUp >= 30000)){
    Serial.println(getUp);
    s.write('@');
    Serial.print("sent @");
    getUp = 0;
    count = 0;
    Serial.print(vibeSet);

    if(vibeSet == 1){
      analogWrite(10,150);
      analogWrite(11,150);
      s.flush();
                    }

     else if(vibeSet == 0){
      analogWrite(10,0);
      analogWrite(11,0);
                          }

  }

 }



 void loop() {
  //large sensors
  FR = analogRead(A0);
  FL = analogRead(A1);
  BR = analogRead(A2);
  BL = analogRead(A3);

  //small sensors
  SF = analogRead(A4);
  SB = analogRead(A5);
  SL = analogRead(A6);
  SR = analogRead(A7);


/*USE TO SEE IF SENSORS ARE OUTPUTTING AS EXPECTED

Serial.println(FR);
Serial.println(FL);
Serial.println(BL);
Serial.println(BR);
Serial.println(SF);
Serial.println(SB);

*/

  if (s.available()>0){    //check if serial is available
```

```
    byte val = s.read();     //read value from esp, store in val
    Serial.println("RECIEVE");  //confirm code is running
    volatile int turnOnLED = val - 48;   //offset by 48 to recieve 1 or 0

  if (turnOnLED == 1){
    Serial.println("LED Turned ON");
    pinMode(5, OUTPUT);             // make external led ON
    pinMode(6,OUTPUT);
    analogWrite(10,0);    //left vibrate
    analogWrite(11,0);    //right vibrate
    vibeSet = 1;

    s.flush();
                      }
  else if(turnOnLED == 0){
    Serial.println("LED Turned OFF");
    pinMode(10,INPUT);              // disable vibrate
    pinMode(11,INPUT);               //disable vibrate
    vibeSet = 0;
    s.flush();
                            }

   else{
    Serial.println("Wrong Credential! Please send ON/OFF");
       }

}

  if((((FR)>=20)|((FL)>=20)|((BR)>=20)|((BL)>=20))){   //CHANGE back to 100 from 10
    count ++;
    getUp++;
                                                    }
  else{
    count = 0;
    getUp = 0;
  }
```

6.2. Mobile Application Code

Included here is all of the relevant code used to create the Flutter mobile application. Each screen has its own appropriately named file. In addition, there is also the *main.dart* file which serves as the entry point for the code.

6.2.1. main.dart

This file serves as the entry point for the code. This file holds global resources including firebase connection objects, user preferences, phone notification variables, and navigation controls. This file is also what controls the getUp timer. Since the homepage for the application is also the login screen, this file also services those needs.

```
import 'package:flutter/material.dart';
import 'code/dashboard_page.dart';
import 'code/settings_page.dart';
import 'code/signUp.dart';
//import 'code/database_test_page.dart';
import 'code/stretches_page.dart';
import 'code/longTermData.dart';
import 'dart:async';
import 'dart:typed_data';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:flutter/services.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';

void main(){
```

```
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  //global access to firebase connection information and user settings

  static FirebaseAuth firebaseAuth = FirebaseAuth.instance;
  static FirebaseUser user;
  static StreamSubscription deviceConnection = null;
  static List postureDataList = null;
  static String pin;
  static String prefsPin = 'pinNum';
  //global access to database connection
  static FirebaseDatabase database = new FirebaseDatabase();
  static DatabaseReference userDataReference;
  static DatabaseReference userSettingsReference;
  //static DatabaseTestPageState databaseData = new DatabaseTestPageState();
  static String pose;
  static String poseLabel;
  static bool vibration = true;
  static bool notification = true;
  static FlutterLocalNotificationsPlugin notificationsPlugin;


  @override
  Widget build(BuildContext context){
    return MaterialApp(
      title: 'Testing Sign In with Firebase',
      home: new WelcomeScreen(),
      routes: <String, WidgetBuilder> {
        '/WelcomeScreen': (BuildContext context) => new WelcomeScreen(),
        '/SignUp': (BuildContext context) => new SignUpScreen(),
        '/DashboardScreen': (BuildContext context) => new DashboardScreen(),
        '/SettingsScreen' : (BuildContext context) => new SettingsScreen(),
       // '/DatabaseTestPage' : (BuildContext context) => new DatabaseTestPage(),
        '/Stretches' : (BuildContext context) => new Stretches(),
        '/LongTermData': (BuildContext context) => new LongTermData(),
      }
      );
  }
}
class WelcomeScreen extends StatefulWidget{
  @override
  WelcomeScreenState createState() => WelcomeScreenState();
}
class WelcomeScreenState extends State<WelcomeScreen> {

  final TextEditingController emailCtrl = new TextEditingController();
  final TextEditingController passwordCtrl = new TextEditingController();
  bool validEmail = false;
  bool validPass = false;
  String errorMessage = '';

  initState() {
    // super.initState();

    var initializationSettingsAndroid = new AndroidInitializationSettings('@mipmap/ic_launcher');
    var initializationSettingsIOS = new IOSInitializationSettings();
    var initializationSettings = new InitializationSettings(initializationSettingsAndroid, initializationSettingsIOS);

    MyApp.notificationsPlugin = new FlutterLocalNotificationsPlugin();
    MyApp.notificationsPlugin.initialize(initializationSettings, onSelectNotification: onSelectNotification);

    //remind user to get up every x minutes
    const getUpCheck = const Duration(seconds: 30);
    var timer = new Timer.periodic(getUpCheck, (timer2) {
      //    print('timer ran out!');
      //check firebase to see if notif should be fired
      if(MyApp.user != null){
        //   print('checking if notification is needed');
        var notifQuery = MyApp.database.reference().child('settings').child(
            MyApp.pin);
        notifQuery.once().then((DataSnapshot snapshot) {
          if (snapshot.value != null) {
            if(snapshot.value['getUp'] == '1') {
              print('firing notification!');
              sendNotification("Take a break from sitting!");
              //reset flag in firebase
              var resetDB = MyApp.database.reference().child('settings')
                  .child(MyApp.pin)
                  .update(
                  <String, String>{
                    "getUp" : "0"
                  })
                  .then((result) {
                print("INFO: Database Write Completed");
              });
              initState();
            }
          } else {
            initState();
          }
        });
      }
```

```
    });

  }

  Future sendNotification(String feedback) async {
    if (MyApp.notification) {
      var vibrationPattern = Int64List(4);
      vibrationPattern[0] = 0;
      vibrationPattern[1] = 1000;
      vibrationPattern[2] = 5000;
      vibrationPattern[3] = 2000;
      var androidPlatformChannelSpecifics = new AndroidNotificationDetails(
          'notofication_channel_id', 'Channel Name',
          'Here we will put the description about the Channel ',
          vibrationPattern: vibrationPattern,
          importance: Importance.Max, priority: Priority.High);

      var iOSPlatformChannelSpecifics = new IOSNotificationDetails();

      var platformChannelSpecifics = new NotificationDetails(
          androidPlatformChannelSpecifics, iOSPlatformChannelSpecifics);

      await MyApp.notificationsPlugin.show(
          0, 'Smart Chair', feedback,
          platformChannelSpecifics, payload: 'Default_Sound');
    } else {
      print('Notifications turned off.');
    }
  }

  Future<String> getPinNumber() async {
    final SharedPreferences prefs =  await SharedPreferences.getInstance();
    print('fetching pin number...');
    return prefs.getString(MyApp.prefsPin)?? 'invalid';
  }


  Future onSelectNotification(String payload) async {
    showDialog(
        context: context,
        builder: (_) => new AlertDialog(
          title: const Text('Here is your payload'),
          content: new Text('Payload: $payload'),
        )
    );
  }

   signIn(BuildContext context, String email, String password) async {
    //remove white space
    String emailTrimmed = email.trim();
    String passwordTrimmed = password.trim();

    //check that form is valid
    if(email != null && password != null) {
      try {
        //attempt to sign in
        MyApp.user = await MyApp.firebaseAuth.signInWithEmailAndPassword(
            email: emailTrimmed, password: passwordTrimmed);
      } on PlatformException catch (e){
        //if sign in fails, display error message
          print('INFO: ${e}');
          setState(() {
            errorMessage = e.message;
          });
      }
      //if login is successful, execute if statement
      if(MyApp.user != null) {
        errorMessage = "";
        emailCtrl.clear();
        passwordCtrl.clear();
        print('INFO: ${MyApp.user.email} signed in.');

        //get pin from sharedPreferences
        var recievePin = await getPinNumber();
        print('recieved pin: $recievePin');

        if(recievePin != 'invalid') {
          MyApp.pin = recievePin;
          //establish new database session
          //save data offline until connection is reestablished - disabled due to problem caused regarding updating chart without
wifi
          MyApp.database.setPersistenceEnabled(false);
          MyApp.database.setPersistenceCacheSizeBytes(10000000);
          //get reference to user's document within database
          MyApp.userDataReference =
              MyApp.database.reference().child('postureData').child(recievePin);
          //pull posture data
          MyApp.userDataReference.once().then((DataSnapshot snapshot) {
            List list = [];
            for (var value in snapshot.value.values) {
              //was previously experiencing errors on parsing from json. Fixed by converting data to string then to int.
              var cogX = value['cogX'].toString();
              var xInt;
              var yInt;
```

```
            if(cogX != 'null') {
              xInt = int.parse(cogX);
            }
            var cogY = value['cogY'].toString();
            if(cogY != 'null') {
              yInt = int.parse(cogY);
            }
            var created_at = value['created_at'].toString();
            //add parsed data to list as a Posture object
            if(xInt != null && yInt != null && created_at != null) {
              print("Adding (" + xInt.toString() + ", " + yInt.toString() +
                  ", " + created_at.toString() + ")\n");
              list.add(new Posture(xInt, yInt, created_at));
            }
          }
          //update global posture data list with fresh data.
          MyApp.postureDataList = list;
        });
        //navigate to homepage and dismiss keyboard
        FocusScope.of(context).requestFocus(new FocusNode());
        print('going to dashboard...');
        Navigator.pushReplacementNamed(context,'/DashboardScreen');
      }
    }
  }
  else{
    errorMessage = 'Invalid login. Try again.';
    print('Must input email and password to sign in');
  }
}

goToSignUp (BuildContext context) {
  Navigator.pushReplacementNamed(context, '/SignUp');
}

@override
Widget build(BuildContext context) {
  return new Scaffold(
      appBar: AppBar(
        title: const Text('Welcome to Smart Chair!'),
      ),
      body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        crossAxisAlignment: CrossAxisAlignment.center,
          children: <Widget> [
            Container(
              width: 300,
              child: TextField(
                controller: emailCtrl,
                autofocus: false,
                decoration: new InputDecoration(
                  labelText: 'Email',
                  errorText: validEmail ? 'Required Field' : null,
                ),
              )
            ),
            Container(
              width: 300,
              child: TextField(
                controller: passwordCtrl,
                autofocus: false,
                obscureText: true,
                decoration: new InputDecoration(
                  labelText: 'Password',
                  errorText: validPass ? 'Required Field' : null,
                ),
              )
            ),
            Text(errorMessage,
                style: TextStyle(color: Colors.red)),
            ButtonTheme(
              minWidth: 300.0,
              child:  RaisedButton(
                  child: const Text('Sign In'),
                  color: Colors.blue,
                  textColor: Colors.white,
                  splashColor: Colors.purple,
                  onPressed: () {
                    setState((){
                      emailCtrl.text.isEmpty ? validEmail = true : validEmail = false;
                      passwordCtrl.text.isEmpty ? validPass = true : validPass = false;
                    });
                    signIn(context, emailCtrl.text, passwordCtrl.text);
                  }
              ),
            ),
            FlatButton(
              child: const Text('Create an account'),
              splashColor: Colors.blue,
              onPressed: () => goToSignUp(context),
            ),
          ]
      ))
```

```
    );
  }
}

class Posture {
  int cogX;
  int cogY;
  String created_at;
  Posture(this.cogX, this.cogY, this.created_at);
}
```

6.2.2. signUp.dart

The purpose of this file is to provide the user interface and methods to perform account creation. If an account is successfully created, the user is returned to the homepage (main.dart) to sign in. If account creation is unsuccessful, the user does not change pages and is served an error message.

```
import 'package:flutter/material.dart';
import 'dart:async';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:sd_proto/main.dart';
import 'package:intl/intl.dart';
import 'package:flutter/services.dart';
import 'package:firebase_database/firebase_database.dart';

class SignUpScreen extends StatefulWidget {
  @override
  SignUpScreenState createState() => SignUpScreenState();
}
class SignUpScreenState extends State<SignUpScreen> {
  final TextEditingController emailCtrl = new TextEditingController();
  final TextEditingController passwordCtrl = new TextEditingController();
  final TextEditingController pinCtrl = new TextEditingController();
  String errorMssg = '';
  String pinErrorMssg = '';
  bool pinIsValid = false;
  String verifiedPin = '';

/*  checkPinUpdate() {
    if(pinIsValid) {
      //determine if user has deleted pin after verifying
      if (pinCtrl.text != verifiedPin) {
        //reset verify button
        pinIsValid = false;
       // verifiedPin = '';
        //remove reservation from firebase
        MyApp.database.reference().child('usedPins').update(<String, String>{
          "" + verifiedPin: "0",
        }).then((result) {
          print(
              "INFO: Database Write Completed (pin ${verifiedPin} has been released)");
        });
        verifiedPin = '';
        setState(() {});
      }
    }
  }*/

  Future<bool> setPinNumber() async {
    final SharedPreferences prefs = await SharedPreferences.getInstance();
    print('setting pin number');
    return prefs.setString(MyApp.prefsPin, MyApp.pin);
  }
  setupUserPreferences(){
    //initialize vibrate and notification preferences to ON
    if(MyApp.user != null) {
      var dbRef = MyApp.database.reference().child('settings').child(MyApp.pin);
      dbRef.set(<String, String>{
        "notification": "1",
        "vibration": "1",
        "getUp" : "0",
      });
      //update global vars
      MyApp.notification = true;
      MyApp.vibration = true;
    }
  }
  writeInitPointsToDatabase() {
    //writes inital 4 values to firebase database to control shape of chart (so that(0,0) is center)
    //order of coords: FL (Quadrant 2), FR (Q1), RL (Q3), RR (Q4)
      List initCogX = ['-200', '200', '-200', '200'];
      List initCogY = ['200', '200', '-200', '-200'];

      if(MyApp.user != null){
        MyApp.userDataReference = MyApp.database.reference().child('postureData').child(MyApp.pin);
```

```
      int counter = 0;
      for(int i=0;i<4;i++) {
        DateTime now = DateTime.now();
        String formattedDate = DateFormat('yyyyMMddTkkmms').format(now);
        //01012001 00010
        //YYYYMMDDTHHMMSS
        String created = '20010101T00000' + counter.toString();
        counter++;
          MyApp.userDataReference.child(created).set(<String, String>{
            "cogX": "" + initCogX[i],
            "cogY": "" + initCogY[i],
            "created_at": "" + created,
          }).then((result) {
            print(
                "INFO: Database Write Completed (${initCogX[i]}, ${initCogY[i]})");
          });
 // });
      }
    }
    //dismiss keyboard
}
verifyPin(String pinCode) {
  String pin = pinCode.trim();
  //double check pin is not already in use before creating account
  var dbQuery = MyApp.database.reference().child('usedPins').child(pin);
  dbQuery.once().then((DataSnapshot snapshot) {
    if (snapshot.value != null || snapshot.value == '1') {
      print('Pin already in use');
      pinErrorMssg = 'This pin is unavailable.';
      pinIsValid = false;
      setState(() {});
    } else {
      MyApp.database.reference().child('usedPins').update(<String, String>{
        "" + pin : "1",
      }).then((result) {
        print(
            "INFO: Database Write Completed (pin $pin has been reserved)");
      });
      print('Pin is now reserved.');
      pinErrorMssg = '';
      pinIsValid = true;
      verifiedPin = pin;
      setState(() {});
    }
  });
}

Future<String> signUp(BuildContext context, String email, String password, String pin) async {
  String emailTrimmed = email.trim();
  String passwordTrimmed = password.trim();
  //create user account
  if(pinIsValid){
  try {
    MyApp.user = await MyApp.firebaseAuth.createUserWithEmailAndPassword(
        email: emailTrimmed, password: passwordTrimmed);
    errorMssg = '';
    setState(() {});
    //get pin from form to setup database
    MyApp.pin = pin;
    //save pin number as shared preferences
    setPinNumber();
    print('INFO: account created for ${MyApp.user.uid}');
    //send initial data points to firebase (the 4 points to control chart shape) and preferences
    writeInitPointsToDatabase();
    setupUserPreferences();

    //go to dashboard
    if (MyApp.user.uid != null) {
      Navigator.pushReplacementNamed(context, '/WelcomeScreen');
    }
    return MyApp.user.uid;
    }
    on PlatformException catch (e) {
      print(e);
      errorMssg = e.message;
      setState(() {});
    }

  }
  else {
    print('Must verify pin before account creation');
    pinErrorMssg = "Must verify pin before creating account!";
    setState(() {});
  }

}

clearForm(BuildContext context) {
  //check for null b/c if form is blank and cancel button is clicked causes error
  if(emailCtrl.text != null){
    emailCtrl.clear();
  }
  if(passwordCtrl.text != null) {
    passwordCtrl.clear();
```

```
    }
    if(pinCtrl.text != null) {
      pinCtrl.clear();
    }
    Navigator.pushReplacementNamed(context, '/WelcomeScreen');
  }

  doNothing() {
    return;
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: AppBar(
          title: const Text('Sign Up'),
        ),
        body: Column(
            children: <Widget> [
              Container(
                  child: Text('Please fill out the form below.'),
              ),
              Container(
                  child: TextField(
                    controller: emailCtrl,
                    autofocus: false,
                    decoration: new InputDecoration(
                        labelText: 'Email'
                    ),
                  )
              ),
              Container(
                  child: TextField(
                    controller: passwordCtrl,
                    autofocus: false,
                    obscureText: true,
                    decoration: new InputDecoration(
                        labelText: 'Password'
                    ),
                  )
              ),
              Row(
              children: <Widget> [
                Container (
                    width: 100.0,
                    child: TextField(
                      controller: pinCtrl,
                      enabled: !pinIsValid ? true : false,
                   //   onChanged: checkPinUpdate(),
                      autofocus: false,
                      decoration: new InputDecoration(
                          labelText: 'Pin Code'
                      ),
                    )
                ),
              RaisedButton(
                child: pinIsValid? Text('Pin Verified!') : Text('Verify Pin'),
                color: pinIsValid ? Colors.green : null,
                onPressed: () => !pinIsValid? verifyPin(pinCtrl.text) : doNothing(),
              ),
              ]),
              Container(
                child: Text(errorMssg,
                       style: TextStyle(color: Colors.red)),
              ),
              Container(
                child: Text(pinErrorMssg,
                       style: TextStyle(color: Colors.red)),
              ),
              Row(
                children: <Widget>[

                  RaisedButton(
                    child: const Text('Submit'),
                    onPressed: () => signUp(context, emailCtrl.text, passwordCtrl.text, pinCtrl.text),
                  ),
                  //add column for padding
                  RaisedButton(
                    child: const Text('Cancel'),
                    onPressed: () => clearForm(context),
                  ),
                ],
              )
            ]
        )
    );
  }
}
```

6.2.3. dashboard_page.dart

The dashboard page serves as the landing page after the user is successfully authenticated. From here the user can view their daily posture data on a scatter plot and access yoga pose recommendations. Navigation on this section of the application is primarily handled via swipe. If the user swipes to the right, the long term data page loads. If the user swipes to the left, the settings page loads.

```
import 'package:flutter/material.dart';
import 'package:sd_proto/main.dart';
import 'package:charts_flutter/flutter.dart' as charts;
import 'package:firebase_database/firebase_database.dart';
import 'package:intl/intl.dart';
import 'dart:math';
import 'package:swipedetector/swipedetector.dart';
class DashboardScreen extends StatelessWidget {

  List<GraphData> data = null;
  List<charts.Series<GraphData, int>> series;

  signOut(BuildContext context) {
    if(MyApp.firebaseAuth != null) {
      MyApp.firebaseAuth.signOut();
    }
    if(MyApp.user != null) {
      MyApp.user = null;
    }
    Navigator.of(context).pushReplacementNamed('/WelcomeScreen');
    if(MyApp.deviceConnection != null) {
      MyApp.deviceConnection.cancel();
    }

  }


  Widget buildChart(){
    data = [];
    print('in build chart');
    if(MyApp.postureDataList == null) {
      print('pdl is null in build chart');
      return Text('No Data Available');
    }
    else {
      print('in build chart: length ' + MyApp.postureDataList.length.toString());
      for(int i=0;i<MyApp.postureDataList.length;i++) {
        //convert posture data object to graph data object
        var freshData = new GraphData(MyApp.postureDataList[i].cogX, MyApp.postureDataList[i].cogY,
MyApp.postureDataList[i].created_at);
        //get datetime data for filtering data
        DateTime now = DateTime.now();
        if(freshData != null && data != null) {
          //check if data is either from the last 24 hrs or is the 4 points to hold the chart axes in place
        //  String dataT = freshData.created_at.substring(0, 7) + 'T' + freshData.created_at.substring(9);
        //  print(dataT);
          DateTime dataTime = DateTime.parse(freshData.created_at);
          if(((now.year == dataTime.year) && (now.month == dataTime.month) && (now.day <= dataTime.day))
          || (dataTime.year == 2001)) {
            data.add(freshData);
            print('added freshData to data list' + freshData.cogX.toString());
          }
        }
        else {
          return Text('Invalid Data. Cannot display.');
        }
      }
      series = fetchGraphData();
      return charts.ScatterPlotChart(
          series,
          animate: true,
      );
    }
  }
  readFromDatabase(){
    print('hit');
    List list = [];
    var userReference = MyApp.database.reference().child('postureData').child(MyApp.pin);
    userReference.once().then((DataSnapshot snapshot) {
      print('DATA: ${snapshot.value}');
      for(var value in snapshot.value.values) {
        //was previously experiencing errors on parsing from json. Fixed by converting data to string then to int.
        var cogX = value['cogX'].toString();
        var xInt = null;
        var yInt = null;
        if(cogX != 'null') {
           xInt = int.parse(cogX);
        }
        var cogY = value['cogY'].toString();
        if(cogY != 'null') {
```

```
          yInt = int.parse(cogY);
        }
        var created_at = value['created_at'].toString();
        if(xInt != null && yInt != null && created_at != null) {
          print("Adding (" + xInt.toString() + ", " + yInt.toString() + ", " +
              created_at.toString() + ")\n");
          //add parsed data to list as a Posture object
          list.add(new GraphData(xInt, yInt, created_at));
        }
  //        list.add(new GraphData(cogX, cogY, created_at));
      }
      print('exited for loop');
      //update global posture data list with fresh data.
      MyApp.postureDataList = list;
      //display list on db testing page for testing purposes...disable later
      //   showData();
      print('end2');
      series = fetchGraphData();
      buildChart();
    });

  }

  void reload(BuildContext context) {
    MyApp.postureDataList == null;
  //  data = null;
  //  series = null;
    //reload chart with fresh data
   // if(MyApp.postureDataList == null) {
      print('pdL is null');
      //pull fresh data from database
     //  MyApp.databaseData.readFromDatabase();
      readFromDatabase();
        print('hit2');
        print(MyApp.postureDataList);
        series = fetchGraphData();
        buildChart();
        print(data);
  //  } else {
  //    print('pdl is not null');
  //    buildChart();
  //  }
    Navigator.of(context).pushReplacementNamed('/DashboardScreen');
  }


  goToSettings (BuildContext context) {
    print('hit');
    var userReference = MyApp.database.reference().child('settings').child(MyApp.pin);
    //pull posture data
    userReference.once().then((DataSnapshot snapshot) {
      if(snapshot.value['vibration'] == '1'){
        print('vibration feature turned on');
        MyApp.vibration = true;

      } else {
        print('vibration feature turned off');
        MyApp.vibration = false;
      }
      if(snapshot.value['notification'] == '1') {
        print('notification feature turned on');
        MyApp.notification = true;
      } else {
        print('notification feature turned off');
        MyApp.notification = false;
      }
      Navigator.pushReplacementNamed(context, '/SettingsScreen');
      });
  }

  displayStretchPage(BuildContext context) {
    List imageURLs = ['cat-cow.jpg', 'thread-the-needle.jpg', 'supine-twist.jpg', 'arm-across-chest.jpg', 'childs-pose.jpg',
'eagle-arms.jpg', 'sphinx-pose.jpg'];
    List labels = ['Cat-Cow Pose', 'Thread The Needle' , 'Supine Twist', 'Arm Across Chest Pose', 'Child\'s Pose', 'Eagle Arms
Pose', 'Sphinx Pose'];
    //generate random number to select index of image
    var random = new Random();
    int scaledRandom = random.nextInt((imageURLs.length) - 0);

    //return image to display
    if (scaledRandom >= 0 && scaledRandom < imageURLs.length) {
      MyApp.pose = 'assets/' + imageURLs[scaledRandom];
      MyApp.poseLabel = labels[scaledRandom];
      Navigator.pushNamed(context, '/Stretches');
    }
  }
  @override
  Widget build(BuildContext context) {
     series = fetchGraphData();
    return new Scaffold(
        appBar: AppBar(
          title: Text('Welcome ${MyApp.user.email}'),
            actions: <Widget>[
              new IconButton(
```

```
              icon: new Icon(Icons.threesixty),
              onPressed: () => reload(context),
            ),
          new IconButton(
            icon: new Icon(Icons.close),
            onPressed: () => signOut(context),
          )
  ],
        ),
        body: SwipeDetector(
              child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              Expanded(
                child: Stack(
                  alignment: const Alignment(1, 1),
                  children: [
                  //  Image.asset("assets/seat.jpg"),
                    buildChart(),
                  ]
        )),
              RaisedButton(
                child: Text('I am experiencing back pain!'),
                onPressed: () => displayStretchPage(context),
              ),
            ]
        ),
        onSwipeLeft: () => goToSettings(context),
        onSwipeRight: () => Navigator.pushReplacementNamed(context, '/LongTermData'),
        onSwipeDown: () => reload(context),
        swipeConfiguration: SwipeConfiguration(
         horizontalSwipeMaxHeightThreshold: 50.0,
         horizontalSwipeMinDisplacement: 10.0,
         horizontalSwipeMinVelocity: 1.0,
         verticalSwipeMaxWidthThreshold: 100.0,
         verticalSwipeMinDisplacement: 50.0,
          verticalSwipeMinVelocity: 100.0
        ),
        )
    );
  }

   List<charts.Series<GraphData, int>> fetchGraphData() {
    print('fetching graph data');
    return [
      new charts.Series<GraphData, int> (
        id: 'Posture',
     //   colorFn:(_, __) => charts.MaterialPalette.blue.shadeDefault,
        colorFn: (GraphData point, _) {
          DateTime dataTime = DateTime.parse(point.created_at);
          if(dataTime.year == 2001) {
            print( dataTime.toString() + "i should be invisible!");
            return charts.MaterialPalette.white;
          } else {
            print("I am a valid data point");
            return charts.MaterialPalette.blue.shadeDefault;
          }
        },
        domainFn: (GraphData dataPoint, _) => dataPoint.cogX,
        measureFn: (GraphData dataPoint, _) => dataPoint.cogY,
        data: data,
      )
    ];
  }
}

class GraphData {
   final int cogX;
   final int cogY;
   final String created_at;

   GraphData(this.cogX, this.cogY, this.created_at);

}
```

6.2.4. stretches_page.dart

This file is rather small. Most of the heavy lifting for this page is handled by the dashboard's code. Here, we just have the build function for the screen that will show the yoga pose recommended by the application.

```
import 'package:flutter/material.dart';
import 'package:sd_proto/main.dart';

class Stretches extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: AppBar(
          title: Text(MyApp.poseLabel),
        ),
        body: Center(
              child: Container(
                child: Image.asset(MyApp.pose),
              ),
        )
    );
  }
}
```

6.2.5. longTermData.dart

This file functions very similarly to the dashboard page. Here the user can view long term data (so data from the current day and older). The main differences are the lack of yoga pose support (as that is handled by the dashboard page) and the different conditions for displaying data on the chart.

```
import 'package:flutter/material.dart';
import 'package:sd_proto/main.dart';
import 'package:charts_flutter/flutter.dart' as charts;
import 'package:firebase_database/firebase_database.dart';
import 'package:intl/intl.dart';
import 'dart:math';
import 'package:swipedetector/swipedetector.dart';

class LongTermData extends StatelessWidget {
  //List<GraphData> data = MyApp.postureDataList;

  List<GraphData> data = null;
  List<charts.Series<GraphData, int>> series;

  Widget buildChart(){
    data = [];
    print('in build chart');
    if(MyApp.postureDataList == null) {
      print('pdl is null in build chart');
      return Text('No Data Available');
    }
    else {
      print('in build chart: length ' + MyApp.postureDataList.length.toString());
      for(int i=0;i<MyApp.postureDataList.length;i++) {
        //convert posture data object to graph data object
        var freshData = new GraphData(MyApp.postureDataList[i].cogX, MyApp.postureDataList[i].cogY,
MyApp.postureDataList[i].created_at);
        if(freshData != null && data != null) {
          data.add(freshData);
          print('added freshData to data list');
        }
        else {
          return Text('Invalid Data. Cannot display.');
        }
      }
      series = fetchGraphData();
      return charts.ScatterPlotChart(
        series,
        animate: true,
      );
    }
  }
  readFromDatabase(){
    List list = [];
    var userReference = MyApp.database.reference().child('postureData').child(MyApp.pin);
    userReference.once().then((DataSnapshot snapshot) {
      print('DATA: ${snapshot.value}');
```

```
    for(var value in snapshot.value.values) {
      //was previously experiencing errors on parsing from json. Fixed by converting data to string then to int.
      var cogX = value['cogX'].toString();
      var xInt = int.parse(cogX);
      var cogY = value['cogY'].toString();
      var yInt = int.parse(cogY);
      var created_at = value['created_at'].toString();
      if(xInt != null && yInt != null && created_at != null) {
        //add parsed data to list as a Posture object
        list.add(new GraphData(xInt, yInt, created_at));
      }
    }
    //update global posture data list with fresh data.
    MyApp.postureDataList = list;
    //display list on db testing page for testing purposes...disable later
    //   showData();
    series = fetchGraphData();
    buildChart();
  });

}

void reload(BuildContext context) {
  //reload chart with fresh data
//  if(MyApp.postureDataList == null) {
    print('pdL is null');
    //pull fresh data from database
   // MyApp.databaseData.readFromDatabase();
    readFromDatabase();
    print(MyApp.postureDataList);
    series = fetchGraphData();
    buildChart();
    print(data);
//  } else {
  //   print('pdl is not null');
  //  buildChart();
//    }
  Navigator.of(context).pushReplacementNamed('/LongTermData');
}

@override
Widget build(BuildContext context) {
  series = fetchGraphData();
  return new Scaffold(
      appBar: AppBar(
        title: Text('Long Term Data'),
        actions: <Widget>[
          new IconButton(
            icon: new Icon(Icons.threesixty),
            onPressed: () => reload(context),
          ),
        ],
      ),
      body: SwipeDetector(
        child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              Expanded(
                child: buildChart(),
              ),
            ]
        ),
        onSwipeLeft: () => Navigator.pushReplacementNamed(context, '/DashboardScreen'),
        swipeConfiguration: SwipeConfiguration(
            horizontalSwipeMaxHeightThreshold: 50.0,
            horizontalSwipeMinDisplacement: 10.0,
            horizontalSwipeMinVelocity: 1.0,
            verticalSwipeMaxWidthThreshold: 100.0,
            verticalSwipeMinDisplacement: 50.0,
            verticalSwipeMinVelocity: 100.0
        ),
      )
  );
}

List<charts.Series<GraphData, int>> fetchGraphData() {
  print('fetching graph data');
  return [
    new charts.Series<GraphData, int> (
      id: 'Posture',
     // colorFn:(_, __) => charts.MaterialPalette.blue.shadeDefault,
      colorFn: (GraphData point,  ) {
        DateTime dataTime = DateTime.parse(point.created_at);
        if(dataTime.year == 2001) {
          print("i should be invisible!");
          return charts.MaterialPalette.white;
        } else {
          print("I am a valid data point");
          return charts.MaterialPalette.blue.shadeDefault;
        }
      },
      domainFn: (GraphData dataPoint, _) => dataPoint.cogX,
      measureFn: (GraphData dataPoint, _) => dataPoint.cogY,
```

```
        data: data,
      )
    ];
  }
}

class GraphData {
  final int cogX;
  final int cogY;
  final String created_at;

  GraphData(this.cogX, this.cogY, this.created_at);

}
```

6.2.6. settings_page.dart

The settings page allows the user to toggle the vibrate and notification preferences. If the preferences are changed, the application updates those flags in Firebase.

```
import 'package:flutter/material.dart';
import 'package:sd_proto/main.dart';
import 'dart:typed_data';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:swipedetector/swipedetector.dart';
class SettingsScreen extends StatefulWidget{
  @override
  SettingsScreenState createState() => SettingsScreenState();
}
class SettingsScreenState extends State<SettingsScreen> {
  //get preferences from firebase
  bool vibrationToggle = MyApp.vibration;
  bool notificationToggle = MyApp.notification;

  updateVibration() {
    if (vibrationToggle) {
      setState(() {
        vibrationToggle = false;
        MyApp.vibration = false;
      });
    } else {
      setState(() {
        vibrationToggle = true;
        MyApp.vibration = true;
      });
    }
    //update flag in firebase
    if (MyApp.user != null) {
      if (MyApp.pin != null) {
        var db = MyApp.database.reference().child('settings')
            .child(MyApp.pin)
            .update(
            <String, String>{
              "vibration": "" + (vibrationToggle ? '1' : '0'),
            })
            .then((result) {
          print("INFO: Database Write Completed");
        });
      }
    }
  }

  updateNotifications() {
    if (notificationToggle) {
      setState(() {
        notificationToggle = false;
        MyApp.notification = false;
      });
    } else {
      setState(() {
        notificationToggle = true;
        MyApp.notification = true;
      });
    }
    if (MyApp.user != null) {
      if (MyApp.pin != null) {
        var db = MyApp.database.reference().child('settings')
            .child(MyApp.pin)
            .update(
            <String, String>{
              "notification": "" + (notificationToggle ? '1' : '0'),
            })
            .then((result) {
          print("INFO: Database Write Completed");
        });
      }
```

```
    }
  }

  Future sendNotification() async {
    if (MyApp.notification && notificationToggle) {
      var vibrationPattern = Int64List(4);
      vibrationPattern[0] = 0;
      vibrationPattern[1] = 1000;
      vibrationPattern[2] = 5000;
      vibrationPattern[3] = 2000;
      var androidPlatformChannelSpecifics = new AndroidNotificationDetails(
          'notofication_channel_id', 'Channel Name',
          'Here we will put the description about the Channel ',
          vibrationPattern: vibrationPattern,
          importance: Importance.Max, priority: Priority.High);

      var iOSPlatformChannelSpecifics = new IOSNotificationDetails();

      var platformChannelSpecifics = new NotificationDetails(
          androidPlatformChannelSpecifics, iOSPlatformChannelSpecifics);

      await MyApp.notificationsPlugin.show(
          0, 'New Post', 'How to Show Notification in flutter',
          platformChannelSpecifics, payload: 'Default_Sound');
    } else {
      print('Notifications turned off.');
    }
  }


  Widget listViewConstructor(BuildContext context) {
    return ListView(
      children: <Widget>[
        ListTile(
            title: Text('Toggle Chair Vibration'),
            trailing: Switch(
                value: vibrationToggle, onChanged: (value) => updateVibration())
        ),
        ListTile(
          title: Text('Toggle Phone Notifications'),
          trailing: Switch(value: notificationToggle,
              onChanged: (value) => updateNotifications()),
        )
/*        ListTile(
          title: Text('Sign Out'),
         onTap: () => signOut(context),
        )*/
      ],
    );
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
        appBar: AppBar(
          title: const Text('Settings'),
        ),
        body: SwipeDetector(
          child: listViewConstructor(context),
          onSwipeRight: () =>
              Navigator.pushReplacementNamed(context, '/DashboardScreen'),
          swipeConfiguration: SwipeConfiguration(
              horizontalSwipeMaxHeightThreshold: 50.0,
              horizontalSwipeMinDisplacement: 10.0,
              horizontalSwipeMinVelocity: 1.0,
              verticalSwipeMaxWidthThreshold: 100.0,
              verticalSwipeMinDisplacement: 50.0,
              verticalSwipeMinVelocity: 100.0
          ),
        )
    );
  }
}
```

6.2.7. pubspec.yaml

This file details all of the dependencies used for creation of the mobile application.

```
name: sd_proto
description: A prototype for senior design project.

version: 1.0.0+1

environment:
  sdk: ">=2.1.0 <3.0.0"

dependencies:
```

```
  flutter:
    sdk: flutter
  firebase_core: ^0.2.5
  firebase_auth: ^0.6.2+1
  google_sign_in:
  firebase_database:
  flutter_blue: 0.4.2
  provider:
  intl:
  charts_flutter:
  flutter_local_notifications: ^0.4.4+2
  shared_preferences: ^0.4.2
  swipedetector:
  flutter_cache_manager: ^0.3.2
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.2

  flutter_test:
    sdk: flutter

flutter:
  assets:
    - assets/

  uses-material-design: true
```

6.3. Firebase JSON Data

Included here is a sample of what the data in the Firebase Realtime Database looks like when populated. This sample data was taken from the Senior Design Final Presentation.


```
{
  "postureData" : {
    "1212" : {
      "20010101T000000" : {
        "cogX" : "-200",
        "cogY" : "200",
        "created_at" : "20010101T000000"
      },
      "20010101T000001" : {
        "cogX" : "200",
        "cogY" : "200",
        "created_at" : "20010101T000001"
      },
      "20010101T000002" : {
        "cogX" : "-200",
        "cogY" : "-200",
        "created_at" : "20010101T000002"
      },
      "20010101T000003" : {
        "cogX" : "200",
        "cogY" : "-200",
        "created_at" : "20010101T000003"
      },
      "2019-07-26T13:42:13Z" : {
        "cogX" : -151,
        "cogY" : -66,
        "created_at" : "2019-07-26T13:42:13Z"
      },
      "2019-07-26T13:42:27Z" : {
        "cogX" : -159,
        "cogY" : -70,
        "created_at" : "2019-07-26T13:42:27Z"
      },
      "2019-07-26T13:42:48Z" : {
        "cogX" : -163,
        "cogY" : -69,
        "created_at" : "2019-07-26T13:42:48Z"
      },
      "2019-07-26T13:42:58Z" : {
        "cogX" : -165,
        "cogY" : -67,
        "created_at" : "2019-07-26T13:42:58Z"
      },
      "2019-07-26T13:43:07Z" : {
        "cogX" : -52,
        "cogY" : -59,
        "created_at" : "2019-07-26T13:43:07Z"
      },
      "2019-07-26T13:43:20Z" : {
        "cogX" : 136,
```

```
      "cogY" : -58,
      "created_at" : "2019-07-26T13:43:20Z"
    }
  },
  "6666" : {
    "20010101T000000" : {
      "cogX" : "-200",
      "cogY" : "200",
      "created_at" : "20010101T000000"
    },
    "20010101T000001" : {
      "cogX" : "200",
      "cogY" : "200",
      "created_at" : "20010101T000001"
    },
    "20010101T000002" : {
      "cogX" : "-200",
      "cogY" : "-200",
      "created_at" : "20010101T000002"
    },
    "20010101T000003" : {
      "cogX" : "200",
      "cogY" : "-200",
      "created_at" : "20010101T000003"
    }
  }
},
"settings" : {
  "1212" : {
    "getUp" : "1",
    "notification" : "1",
    "vibration" : "1"
  },
  "6666" : {
    "firePhoneNotif" : "0",
    "getUp" : "0",
    "notification" : "1",
    "vibration" : "1"
  }
},
"usedPins" : {
  "1212" : "1",
  "6666" : "1"
}
}
```


The purpose of the four data points that were dated for the year 2001 was for controlling the dimensions of the chart on the application. Due to the limitations of the library used, extra data points were needed to preserve the orientation of the graph such that the four quadrants of the graph corresponded to the sectors of the physical chair.

6.4. Serial and Wireless Communication Code

The code in this section functions as the communication between hardware and software, by implementing serial and wireless connection. As explained in chapter 5.2.4 on implementing the communication, the code contains the necessary components for the project with the software serial and firebase library. The software serial allows wired connection between devices and assigning the pins for transmitting and receiving using the TX or RX pins, or any other digital pins. The firebase library allows wireless connection to the database, with the proper assigned credentials from host and auth, while connecting to the internet using the SSID and password. Another important library implemented in the code is the timestamp as the WiFiUdp.h, which is important to set a time on when data is received to firebase.

```
// Sending Data Between Arduino and NodeMCU
// NodeMCU Code

// ************************ Headers and Libraries ************************

#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <NTPClient.h>
```

```
#include <WiFiUdp.h>

// ********************** Connection to the Network **********************

#define FIREBASE_HOST   "sd-proto.firebaseio.com"
#define FIREBASE_AUTH   "JHX8oWlHBy0OV2NSvrnziX9hJ297ypRT5OAxuitY"
#define WIFI_SSID       "The Bomb Galaxy"
#define WIFI_PASSWORD   "phuong3648"

#include <SoftwareSerial.h>
SoftwareSerial s(D6, D5); //Rx, Tx

// ************************** Center of Gravity **************************

signed int cogX;
signed int cogY;
signed int value = 0;
boolean negative = false;

// ******************** Vibration Activation Commands ********************

String fireStatus = "";
//int VIBRATOR = D3;

// ************************** Timestamp for FSR **************************

// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

// Variables to save date and time
String formattedDate;
String dayStamp;
String timeStamp;

// ****************** Setting conditions for connection ******************

void setup()
{
  s.begin(57600);
  // Uno can only work at baud rate 57600
  // Baud Rate at 115200 has less latency
  // 115200 does not activate vibrator on Uno
  Serial.begin(9600);
  delay(1000);
  pinMode(LED_BUILTIN, OUTPUT);
//  pinMode(VIBRATOR, OUTPUT);

  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting to ");
  Serial.print(WIFI_SSID);

  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("Connected to ");
  Serial.println(WIFI_SSID);
  Serial.print("IP Address is : ");
  Serial.println(WiFi.localIP());

  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
//  Firebase.setString("settings/1212/vibration", "1");

  // Initialize a NTPClient to get time
  timeClient.begin();
  // Set offset time in seconds to adjust for your timezone, for example:
  // GMT +1 = 3600
```

```
  // GMT +8 = 28800
  // GMT -1 = -3600
  // GMT 0 = 0
  timeClient.setTimeOffset(-14400);
  // Offset the timestamp by due to incorrect time reading
  // -14400 is by seconds, meaning 5 hours in seconds
}

// ***************** Analog Readings coming from Arduino *****************

void loop()
{
  // ***************** FSR Reading Data *****************
  if (s.available() > 0)
  {
    //read value from input buffer
    byte recieved = s.read();
//    Serial.print("Check value: ");
//    Serial.println(recieved);

    if(recieved == '@')
    {
      //send get up flag to firebase
      Firebase.setString("settings/1212/getUp", "1");
      Serial.println("Get Up Notification");
    }

//    else
//    {
//      Firebase.setString("settings/1212/getUp", "0");
//    }

    //process value according to flags
    if(recieved == 45)
    {
      //read a negative sign
      negative = true;
    }

    else if(recieved >=48 && recieved <= 57)
    {
      //read a numerical value
      value *= 10;
      value += recieved - 48;
    }

    else if(recieved == 59)
    {
      if(negative)
      {
        value *= -1;
        negative = false; //reset flag
      }
    }

    else if(recieved == 120 || recieved == 121)
    {
      //read an x or y
      if(recieved == 120)
      {
        cogX = value;
        value = 0;
      }
      if(recieved == 121)
      {
        cogY = value;
        value = 0;
      }
    }

    else if(recieved == '*')
```

```
    {
      //bad data sent, dismiss current value
      value = 0;
      cogX = 0;
      cogY = 0;
      Serial.println("User got up");
      s.flush();
    }

    // Gather the data from Sensor to print values
    if((cogX != 0 && cogY != 0) && (cogX < 500 && cogX > -500 && cogY < 500 && cogY > -500))
    {
      //print out coords
      Serial.println();
      Serial.print("(");
      Serial.print(cogX);
      Serial.print(", ");
      Serial.print(cogY);
      Serial.println(")");

      // Timestamp of the sensor data
      while(!timeClient.update())
      {
        timeClient.forceUpdate();
      }
        // 2018-05-28T16:00:13Z
        formattedDate = timeClient.getFormattedDate();
//       Serial.println(formattedDate);

      signed int x = cogX + 10;
      cogX = 0;
      signed int y = cogY + 10;
      cogY = 0;
      // Send cogX cogY to firebase
      Firebase.setInt("postureData/1212/" + formattedDate + "/cogX", x);
      Firebase.setInt("postureData/1212/" + formattedDate + "/cogY", y);
      Firebase.setString("postureData/1212/" + formattedDate + "/created_at", formattedDate);

    }

  }
  // The end of the s.available


  // ****************** Vibration Module Activation ******************
  // ******************* Outside the FSR Statement *******************
  fireStatus = Firebase.getString("settings/1212/vibration");

  if (fireStatus == "1")
  {
    // compare the input of led status received from firebase
    Serial.print("VIBRATOR Turned ON\t");
    Serial.println("Value: " + fireStatus);
    digitalWrite(LED_BUILTIN, LOW);           // make bultin led ON
    s.write("1");
  }

  else if (fireStatus == "0")
  {
    // compare the input of led status received from firebase
    Serial.print("VIBRATOR Turned OFF\t");
    Serial.println("Value: " + fireStatus);
    digitalWrite(LED_BUILTIN, HIGH);          // make bultin led OFF
    s.write("0");
  }
  else
  {
    Serial.println("Wrong Credential! Please send ON/OFF");
  }
  delay(500);
```

```
  // Conclusion of the code, print statement into serial communication
  // Once sensor code starts sending data, vibration statement starts
  // Proper baud rate is important to match the readings from Arduino
}
```

7. Appendix

This section serves to organize our works cited and permission requests. Below the works sited can be found sequentially numbered according to when they appear in the text. The permission requests section will house all communications and formal requests to use copyrighted, patented, and other types of protected documents in our report.

7.1 Works Cited

[1] "Anatomy and Function," *University of Maryland Medical Center*. [Online]. Available: https://www.umms.org/ummc/health-services/orthopedics/services/spine/patient-guides/anatomy-function. [Accessed: 20-Apr-2019].

[2] "What is Agile Software Development?," *Agile Alliance*, 05-Mar-2019. [Online]. Available: https://www.agilealliance.org/agile101/. [Accessed: 20-Apr-2019].

[3] *Home - Posture Perfect*. [Online]. Available: http://www.eecs.ucf.edu/seniordesign/sp2016fa2016/g21/. [Accessed: 21-Apr-2019].

[4] A. Asher and Cpt, "The Human Center of Gravity Defined," *Verywell Health*. [Online]. Available: https://www.verywellhealth.com/human-center-of-gravity-296568. [Accessed: 20-Apr-2019].

[5] "5 Yoga Poses to Ease Lower Back Pain," *DOYOUYOGA.COM*, 01-Oct-2015. [Online]. Available: https://www.doyouyoga.com/5-yoga-poses-to-ease-lower-back-pain-30237/. [Accessed: 20-Apr-2019].

[6] "Downward Facing Dog Pose-Adho Mukha Svanasana-Ekhart Yoga | Ekhart Yoga," *Ekhart Yoga online*. [Online]. Available: https://www.ekhartyoga.com/resources/yoga-poses/downward-facing-dog-pose. [Accessed: 20-Apr-2019].

[7] N. Rizopoulos, "The King of Hip Openers: Pigeon Pose," *Yoga Journal*, 16-Jul-2008. [Online]. Available: https://www.yogajournal.com/practice/pigeon-pose. [Accessed: 20-Apr-2019].

[8] Yoga Magazine, "KNEES TO CHEST POSE," *Yoga Magazine*, 26-Sep-2013. [Online]. Available: http://www.yogamagazine.com/knees-to-chest-pose/. [Accessed: 20-Apr-2019].

[9] A. Asher and Cpt, "Spinal Twist: Using Yoga for Back Pain," *Verywell Health*. [Online]. Available: https://www.verywellhealth.com/yoga-for-back-pain-supine-spinal-twist-297346. [Accessed: 20-Apr-2019].

[10] Y. J. Editors, "Sphinx Pose," *Yoga Journal*, 28-Aug-2007. [Online]. Available: https://www.yogajournal.com/poses/sphinx-pose. [Accessed: 20-Apr-2019].

[11] K. CollinsKelly and Nasm, "These 9 Relaxing Poses Relieve Pain in Your Back and Shoulders," *Paleo Blog*, 20-Feb-2019. [Online]. Available: https://blog.paleohacks.com/yoga-release-shoulder-back-pain/. [Accessed: 20-Apr-2019].

[12] S. Stevenson, “Yoga for Neck Pain | Neck and Shoulder Stretches,” *Openfit*, 08-Oct-2018. [Online]. Available: https://www.openfit.com/9-yoga-poses-to-help-relieve-neck-and-shoulder-pain. [Accessed: 20-Apr-2019].
[13] "Wearable Medical Devices Market worth 14.41 Billion USD by 2022", *Marketsandmarkets.com*, 2019. [Online]. Available: https://www.marketsandmarkets.com/PressReleases/wearable-medical-device.asp. [Accessed: 21- Apr- 2019]
[14] U. GO, "UPRIGHT GO", *Upright Tech*, 2019. [Online]. Available: https://store.uprightpose.com/products/upright-go-single?utm_expid=.jdD1ChnvTiKurlE4EhFuqw.0&utm_referrer=https%3A%2F%2Fwww.google.com%2F. [Accessed: 21- Apr- 2019]
[15] [12]"Amazon.com: Back, Neck & Shoulder Supports: Health & Household: Back Braces, Lumbar Supports, Shoulder Supports & More", *Amazon.com*, 2019. [Online]. Available: https://www.amazon.com/Back-Neck-Shoulder-Supports/b?ie=UTF8&node=8619205011. [Accessed: 21- Apr- 2019]
[16] “Bonded Strain Gauges,” *Instrumentation and Control Engineering*. [Online]. Available: http://instrumentationandcontrollers.blogspot.com/2010/11/bonded-strain-gauges.html. [Accessed: 20-Apr-2019].
[17] “Strain Gauges,” *Strain Gauges | Omega Engineering US*. [Online]. Available: https://www.omega.com/pptst/SGN_SPANRESISTORS.html?pn=SGN-4/20-E. [Accessed: 20-Apr-2019].
[18] “What's the difference between the motion sensors,” *Welcome to Brandon Lighting*, 11-Jul-2018. [Online]. Available: https://brandon-lighting.com/whats-the-difference-between-pir-ultrasonic-microwave-occupancy-sensors/. [Accessed: 20-Apr-2019].
[19] “ADS1231 (ACTIVE),” *ADS1231 24-Bit, 80SPS, 1-Ch Delta-Sigma ADC for Resistive Bridge Sensors & Weigh Scales | TI.com*. [Online]. Available: http://www.ti.com/product/ADS1231?keyMatch=ADS123&tisearch=Search-EN-Products. [Accessed: 20-Apr-2019].
[20] “Eccentric Rotating Mass Vibration Motors – ERMs,” *recision Microdrives*. [Online]. Available: https://www.precisionmicrodrives.com/vibration-motors/eccentric-rotating-mass-vibration-motors-erms/. [Accessed: 20-Apr-2019].
[21] Anon, (2019). *TI MSP430*. [online] Available at: http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/reference-designs.html#search?famid=342 [Accessed 21 Apr. 2019].
[22] "TI F280049PMSR", *TI*, 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/tms320f280049.pdf?HQS=TI-null-null-mousermode-df-pf-null-wwe&DCM=yes&ref_url=https%3A%2F%2Fwww.mouser.com%2. [Accessed: 21- Apr- 2019]
[23] “M430FR5739SRHATEP Texas Instruments | Mouser,” *Mouser Electronics*. [Online]. Available: https://www.mouser.com/ProductDetail/Texas-Instruments/M430FR5739SRHATEP?qs=/ha2pyFaduiwabCONVYtbcfa1yjVOouvgI5G9TxSrmE9Oo5y+xPPqAJH6LUGyT2c. [Accessed: 20-Apr-2019].

[24] "PIC24FJ1024GA610", *Mouser.com*, 2019. [Online]. Available: https://www.mouser.com/datasheet/2/268/PIC24FJ1024GA610-GB610-Family-Data-Sheet-DS3001007-1314526.pdf. [Accessed: 21- Apr- 2019]
[25] "SAMA5D2 Series | Microchip Technology", *Microchip.com*, 2019. [Online]. Available: https://www.microchip.com/design-centers/32-bit-mpus/microprocessors/sama5/sama5d2-series. [Accessed: 21- Apr- 2019]
[26] "ATMEGA2560-16AU by Microchip Technology | Microcontroller," *Arrow.com*. [Online]. Available: https://www.arrow.com/en/products/atmega2560-16au/microchip-technology. [Accessed: 20-Apr-2019].
[27] "Elegoo HC-SR04 Ultrasonic Distance Sensor Kit - Micro Center", *Micro Center*, 2019. [Online]. Available: https://www.microcenter.com/product/503220/elegoo-hc-sr04-ultrasonic-distance-sensor-kit. [Accessed: 21- Apr- 2019]
[28] "Ultrasonic Range Finder", *VEX Robotics*, 2019. [Online]. Available: https://www.vexrobotics.com/276-2155.html. [Accessed: 21- Apr- 2019]
[29] C. Pang, G.-Y. Lee, T.-il Kim, S. M. Kim, H. N. Kim, S.-H. Ahn, and K.-Y. Suh, "A flexible and highly sensitive strain-gauge sensor using reversible interlocking of nanofibres," *Semantics Scholar*, 29-Jul-2012. [Online]. Available: https://pdfs.semanticscholar.org/962c/c9e66904d86bdc22b518785038db1c06601f.pdf. [Accessed: 20-Apr-2019].
[30] "FSR 406 Sensing Resistor - Interlink Electronics | DigiKey", *Digikey.com*, 2011. [Online]. Available: https://www.digikey.com/en/product-highlight/i/interlink/fsr-406-square-force-sensing-resistor?utm_adgroup=Resistors&slid=&gclid=EAIaIQobChMI3Y69zcDh4QIVzJyzCh3TsgNJEAAYASAAEgKjwfD_BwE. [Accessed: 21- Apr- 2019]
[31] "SEN-09376 SparkFun | Mouser," *Mouser Electronics*. [Online]. Available: https://www.mouser.com/ProductDetail/SparkFun/SEN-09376?qs=/ha2pyFadug6mU392eSvum78YON5aunGg7TU89blaYI=. [Accessed: 20-Apr-2019].
[32] "Square Force Sensing Resistor | FlexiForce A502 Sensor," *Tekscan*. [Online]. Available: https://www.tekscan.com/products-solutions/force-sensors/a502. [Accessed: 20-Apr-2019].
[33] "FlexiForce A301 Datasheet," *Tekscan*, 05-Nov-2018. [Online]. Available: https://www.tekscan.com/resources/product/flexiforce-a301-datasheet. [Accessed: 20-Apr-2019].
[34] "[TUT] [C] Newbie's Guide to the AVR ADC," *AVR Freaks*, 23-Jun-2015. [Online]. Available: https://www.avrfreaks.net/forum/tut-c-newbies-guide-avr-adc?name=PNphpBB2&file=viewtopic&t=56429. [Accessed: 20-Apr-2019].
[35] *PCB Basics*. [Online]. Available: https://learn.sparkfun.com/tutorials/pcb-basics/all. [Accessed: 20-Apr-2019].
[36] O. Team, "Understanding PCB Manufacturing: Silk-Screening," *Understanding PCB Manufacturing: Silk-Screening*. [Online]. Available: http://www.omnicircuitboards.com/blog/bid/312861/understanding-pcb-manufacturing-silk-screening. [Accessed: 20-Apr-2019].

[37] SparkFun Electric, “uA7800 Series (Rev. J),” *SparkFun*, May-2003. [Online]. Available: https://www.sparkfun.com/datasheets/Components/LM7805.pdf. [Accessed: 20-Apr-2019].
[38] “admin,” *Pinout, Diagrams, Equivalent & Datasheet*. [Online]. Available: https://components101.com/7805-voltage-regulator-ic-pinout-datasheet. [Accessed: 20-Apr-2019].
[39] *Switch Basics*. [Online]. Available: https://learn.sparkfun.com/tutorials/switch-basics/all. [Accessed: 20-Apr-2019].
[40] "EAGLE | PCB Design Software | Autodesk", *Autodesk.com*, 2019. [Online]. Available: https://www.autodesk.com/products/eagle/overview?&mkwid=sCk26gIm5%7cpcrid%7c294276762468%7cpkw%7ceagle%20for%20pcb%20design%7cpmt%7ce%7cpdv%7cc%7cslid%7c0RUUq7nm%7cpgrid%7c37821440399%7cptaid%7ckwd-364337666048%7c&intent=EAGLE+Brand&utm_medium=cpc&utm_source=google&utm_campaign=&utm_term=eagle%20for%20pcb%20design&utm_content=sCk26gIm5%7cpcrid%7c294276762468%7cpkw%7ceagle%20for%20pcb%20design%7cpmt%7ce%7cpdv%7cc%7cslid%7c0RUUq7nm%7cpgrid%7c37821440399%7cptaid%7ckwd-364337666048%7c&addisttype=g&s_kwcid=AL!8131199977!3!294276762468!e!!g!!eagle%20for%20pcb%20design&gclid=EAIaIQobChMIwKTlornh4QIVyYuzCh3AmwNIEAAYASABEgKI_PD_BwE&gclsrc=aw.ds. [Accessed: 21- Apr- 2019]
[41] "Home Page", *Orcad.com*, 2019. [Online]. Available: https://www.orcad.com/. [Accessed: 21- Apr- 2019]
[42] “An Easier and PowerfulOnline PCB Design Tool,” *EasyEDA*. [Online]. Available: https://easyeda.com/. [Accessed: 21-Apr-2019].
[43] “The Importance of Application Security: A Few of the Benefits and Risks,” *Veracode*, 23-Jan-2017. [Online]. Available: https://www.veracode.com/blog/intro-appsec/importance-application-security-few-benefits-and-risks. [Accessed: 20-Apr-2019].
[44] C. Wodehouse, “8 Tips for Better Mobile Application Security,” *Upwork*, 05-Jul-2018. [Online]. Available: https://www.upwork.com/hiring/mobile/mobile-application-security/. [Accessed: 20-Apr-2019].
[45] *SQL Injection*. [Online]. Available: https://www.w3schools.com/sql/sql_injection.asp. [Accessed: 20-Apr-2019].
[46] “Cross-Site Scripting (XSS) Tutorial: Learn About XSS Vulnerabilities, Injections and How to Prevent Attacks,” *Veracode*, 18-Apr-2019. [Online]. Available: https://www.veracode.com/security/xss. [Accessed: 20-Apr-2019].
[47] “Keeping Passwords Secure,” *Facebook Newsroom*. [Online]. Available: https://newsroom.fb.com/news/2019/03/keeping-passwords-secure/. [Accessed: 20-Apr-2019].
[48] “How to store a password in database?,” *GeeksforGeeks*, 09-Feb-2018. [Online]. Available: https://www.geeksforgeeks.org/store-password-database/. [Accessed: 20-Apr-2019].

[49] “Is A MacBook Pro Good Enough For iOS Development? – LearnAppMaking,” *LearnAppMaking*, 20-Nov-2018. [Online]. Available: https://learnappmaking.com/ios-development-macbook-pro-good-enough/. [Accessed: 20-Apr-2019].
[50] *Apple Developer Documentation*. [Online]. Available: https://developer.apple.com/documentation/. [Accessed: 20-Apr-2019].
[51] “Documentation,” *Android Developers*. [Online]. Available: https://developer.android.com/docs. [Accessed: 20-Apr-2019].
[52] “Flutter in Mobile App Development – Pros & Cons for App Owners,” *iOS & Android Mobile App Development Company - Droids On Roids - Poland*, 28-Mar-2019. [Online]. Available: https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-cons-for-app-owners. [Accessed: 20-Apr-2019].
[53] M. Bellinaso and M. Bellinaso, “Flutter: the good, the bad and the ugly,” *Medium*, 23-Nov-2018. [Online]. Available: https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9. [Accessed: 20-Apr-2019].
[54] “Hot reload,” *Flutter by Google*. [Online]. Available: https://flutter.dev/docs/development/tools/hot-reload. [Accessed: 20-Apr-2019].
[55] “Abbey Road Studios (Flutter Developer Story),” *YouTube*, 05-Dec-2018. [Online]. Available: https://youtu.be/_ACWeGGBP4E. [Accessed: 20-Apr-2019].
[56] “Widget catalog,” *Flutter by Google*. [Online]. Available: https://flutter.dev/docs/development/ui/widgets. [Accessed: 20-Apr-2019].
[57] E. Ravenscraft and E. Ravenscraft, “How Widgets Can Actually Make Your Phone More Productive,” *Lifehacker*, 03-Sep-2014. [Online]. Available: https://lifehacker.com/how-widgets-can-actually-make-your-phone-more-productiv-1333180508. [Accessed: 20-Apr-2019].
[58] L. B. T. App, “Complete Guide to Flutter: How to Build a Real World App,” *YouTube*, 30-Apr-2018. [Online]. Available: https://www.youtube.com/watch?v=S59b-XFsyY8. [Accessed: 21-Apr-2019].
[59] 1005245086190346, “What the F**tter!? Understanding Flutter as an Android Developer,” *ProAndroidDev*, 02-Mar-2018. [Online]. Available: https://proandroiddev.com/what-the-f-tter-understanding-flutter-as-an-android-java-developer-2158086a2bd9. [Accessed: 21-Apr-2019].
[60] “Dart API docs,” *Flutter*. [Online]. Available: https://docs.flutter.io/index.html. [Accessed: 21-Apr-2019].
[61] 523457184653447, “Flutter: Creating Multi Page Application with Navigation,” *ProAndroidDev*, 21-May-2018. [Online]. Available: https://proandroiddev.com/flutter-creating-multi-page-application-with-navigation-ef9f4a72d181. [Accessed: 21-Apr-2019].
[62] “Adobe PhoneGap,” *PhoneGap Documentation | PhoneGap Docs*. [Online]. Available: http://docs.phonegap.com/. [Accessed: 20-Apr-2019].
[63] “Sencha Touch,” *Sencha.com*, 07-Mar-2018. [Online]. Available: https://www.sencha.com/products/touch/. [Accessed: 20-Apr-2019].
[64] M. Swehli and M. Swehli, “Why Dart is the Language to Learn of 2018,” *Medium*, 12-Mar-2018. [Online]. Available:

https://medium.com/@mswehli/why-dart-is-the-language-to-learn-of-2018-e5fa12adb6c1. [Accessed: 20-Apr-2019].
[65] W. Leler, “Why Flutter Uses Dart,” *Hacker Noon*, 26-Feb-2018. [Online]. Available: https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf. [Accessed: 20-Apr-2019].
[66] “A Tour of the Dart Language,” *Dart*. [Online]. Available: https://www.dartlang.org/guides/language/language-tour. [Accessed: 21-Apr-2019].
[67] “What is persistence and why does it matter?,” *DataStax*, 19-Jan-2018. [Online]. Available: https://www.datastax.com/dev/blog/what-persistence-and-why-does-it-matter. [Accessed: 20-Apr-2019].
[68] “Data and file storage overview | Android Developers,” *Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/data/data-storage. [Accessed: 20-Apr-2019].
[69] “How to store data locally in an Android app,” *Android Authority*, 20-Nov-2017. [Online]. Available: https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/. [Accessed: 20-Apr-2019].
[70] “Reading and Writing Files,” *Flutter by Google*. [Online]. Available: https://flutter.dev/docs/cookbook/persistence/reading-writing-files. [Accessed: 20-Apr-2019].
[71] Drydart, “drydart/flutter_sqlcipher,” *GitHub*, 22-Mar-2019. [Online]. Available: https://github.com/drydart/flutter_sqlcipher. [Accessed: 20-Apr-2019].
[72] “How to store data locally in an Android app,” *Android Authority*, 20-Nov-2017. [Online]. Available: https://www.androidauthority.com/how-to-store-data-locally-in-android-app-717190/. [Accessed: 20-Apr-2019].
[73] *Appropriate Uses For SQLite*. [Online]. Available: https://www.sqlite.org/whentouse.html. [Accessed: 20-Apr-2019].
[74] “SQLite As An Application File Format,” *Benefits of SQLite As A File Format*. [Online]. Available: https://www.sqlite.org/aff_short.html. [Accessed: 20-Apr-2019].
[75] Tekartik, “tekartik/sqflite,” *GitHub*, 11-Apr-2019. [Online]. Available: https://github.com/tekartik/sqflite. [Accessed: 20-Apr-2019].
[76] Tutorialspoint.com, “Android Shared Preferences,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/android/android_shared_preferences.htm. [Accessed: 20-Apr-2019].
[77] “Data and file storage overview | Android Developers,” *Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/data/data-storage. [Accessed: 20-Apr-2019].
[78] “Data and file storage overview | Android Developers,” *Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/data/data-storage. [Accessed: 20-Apr-2019].

[79] “What is a Relational Database Management System?,” *Codecademy*. [Online]. Available: https://www.codecademy.com/articles/what-is-rdbms-sql. [Accessed: 20-Apr-2019].
[80] Tutorialspoint.com, “SQL Tutorial,” *www.tutorialspoint.com*. [Online]. Available: https://www.tutorialspoint.com/sql/. [Accessed: 20-Apr-2019].
[81] “Oracle MySQL,” *Oracle*. [Online]. Available: https://www.oracle.com/mysql/. [Accessed: 20-Apr-2019].
[82] “8 Major Advantages of Using MySQL,” *datamation*. [Online]. Available: https://www.datamation.com/storage/8-major-advantages-of-using-mysql.html. [Accessed: 20-Apr-2019].
[83] “mysql1 | Dart Package,” *Dart Packages*, 28-Nov-2018. [Online]. Available: https://pub.dartlang.org/packages/mysql1. [Accessed: 20-Apr-2019].
[84] “NoSQL Databases Explained,” *MongoDB*. [Online]. Available: https://www.mongodb.com/nosql-explained. [Accessed: 20-Apr-2019].
[85] “Realm Platform,” *Realm*. [Online]. Available: https://realm.io/products/realm-platform. [Accessed: 20-Apr-2019].
[86] Realm, “Dart / Flutter Support · Issue #55 · realm/realm-object-server,” *GitHub*. [Online]. Available: https://github.com/realm/realm-object-server/issues/55. [Accessed: 20-Apr-2019].
[87] “COUCHBASE NAMED A LEADER,” *Couchbase*. [Online]. Available: https://www.couchbase.com/. [Accessed: 20-Apr-2019].
[88] Oltrenuovefrontiere, “oltrenuovefrontiere/fluttercouch,” *GitHub*, 10-Nov-2018. [Online]. Available: https://github.com/oltrenuovefrontiere/fluttercouch. [Accessed: 20-Apr-2019].
[89] N. Apps, “Flutter and MySQL,” *Medium*, 15-Jun-2018. [Online]. Available: https://medium.com/@thedome6/flutter-and-mysql-1d0dc9dfe4af. [Accessed: 20-Apr-2019].
[90] R. Rahiche and R. Rahiche, “Using SQLite in Flutter,” *Medium*, 12-Dec-2018. [Online]. Available: https://medium.com/flutter-community/using-sqlite-in-flutter-187c1a82e8b. [Accessed: 20-Apr-2019].
[91] “flutter_sqlcipher | Flutter Package,” *Dart Packages*, 13-Mar-2019. [Online]. Available: https://pub.dartlang.org/packages/flutter_sqlcipher. [Accessed: 20-Apr-2019].
[92] “Reading and Writing Files,” *Flutter by Google*. [Online]. Available: https://flutter.dev/docs/cookbook/persistence/reading-writing-files. [Accessed: 20-Apr-2019].
[93] Realm, “Dart / Flutter Support · Issue #55 · realm/realm-object-server,” *GitHub*. [Online]. Available: https://github.com/realm/realm-object-server/issues/55. [Accessed: 20-Apr-2019].
[94] hemeda3, “hemeda3/couchbase-lite-flutter,” *GitHub*. [Online]. Available: https://github.com/hemeda3/couchbase-lite-flutter. [Accessed: 20-Apr-2019].
[95] W. Leler, “What's Revolutionary about Flutter,” *Hacker Noon*, 25-Aug-2017. [Online]. Available: https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514. [Accessed: 21-Apr-2019].

[96] “Android SQLite Database Tutorial,” *AndroidHive*, 13-May-2018. [Online]. Available: http://www.androidhive.info/2011/11/android-sqlite-database-tutorial/. [Accessed: 20-Apr-2019].
[97] Sqlcipher, “sqlcipher/android-database-sqlcipher,” *GitHub*. [Online]. Available: https://github.com/sqlcipher/android-database-sqlcipher. [Accessed: 20-Apr-2019].
[98] “View on-device files with Device File Explorer | Android Developers,” *Android Developers*. [Online]. Available: https://developer.android.com/studio/debug/device-file-explorer. [Accessed: 20-Apr-2019].
[99] “Storing and Accessing SharedPreferencesEdit PagePage History,” *Storing and Accessing SharedPreferences | CodePath Android Cliffnotes*. [Online]. Available: https://guides.codepath.com/android/Storing-and-Accessing-SharedPreferences. [Accessed: 20-Apr-2019].
[100] “What Are APIs and How Do They Work?,” *ProgrammableWeb*. [Online]. Available: https://www.programmableweb.com/api-university/what-are-apis-and-how-do-they-work. [Accessed: 20-Apr-2019].
[101] “Data Binding in Android,” *Android Authority*, 16-Aug-2016. [Online]. Available: https://www.androidauthority.com/data-binding-in-android-709747/. [Accessed: 20-Apr-2019].
[102] “Firebase,” *Google*. [Online]. Available: https://firebase.google.com/use-cases/. [Accessed: 20-Apr-2019].
[103] “Firebase Authentication | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/auth/. [Accessed: 20-Apr-2019].
[104] “Firebase Realtime Database | Store and sync data in real time | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/products/realtime-database/. [Accessed: 20-Apr-2019].
[105] “Structure Your Database | Firebase Realtime Database | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/database/web/structure-data. [Accessed: 20-Apr-2019].
[106] Firebase, “Converting SQL structures to Firebase structures - The Firebase Database For SQL Developers #2,” *YouTube*, 22-Nov-2016. [Online]. Available: https://www.youtube.com/watch?v=ran_Ylug7AE. [Accessed: 20-Apr-2019].
[107] P. A. Martínez and P. A. Martínez, “Lessons learnt (the hard way) using Firebase RealTime Database,” *Pablo A. Martínez Andrés*, 24-Feb-2018. [Online]. Available: https://pamartinezandres.com/lessons-learnt-the-hard-way-using-firebase-realtime-database-c609b52b9afb. [Accessed: 20-Apr-2019].
[108] “Users in Firebase Projects | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/auth/users. [Accessed: 20-Apr-2019].
[109] “Firebase Authentication | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/auth/. [Accessed: 20-Apr-2019].
[110] P. A. Martínez and P. A. Martínez, “Lessons learnt (the hard way) using Firebase RealTime Database,” *Pablo A. Martínez Andrés*, 24-Feb-2018. [Online].

Available: https://pamartinezandres.com/lessons-learnt-the-hard-way-using-firebase-realtime-database-c609b52b9afb. [Accessed: 21-Apr-2019].
[111] "Choosing a Firebase Database For Your App: Realtime Database vs. Cloud Firestore," *Savvy Apps*, 04-Apr-2019. [Online]. Available: https://savvyapps.com/blog/firebase-realtime-database-vs-cloud-firestore-for-your-app. [Accessed: 21-Apr-2019].
[112] R. Rashmin and R. Rashmin, "Arduino to Android , Real Time Communication For IoT with Firebase," *Medium*, 22-Jul-2018. [Online]. Available: https://medium.com/coinmonks/arduino-to-android-real-time-communication-for-iot-with-firebase-60df579f962. [Accessed: 20-Apr-2019].
[113] DigitalOcean, "An Introduction to JSON," *DigitalOcean*, 18-Jul-2017. [Online]. Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-json. [Accessed: 20-Apr-2019].
[114] "Class Documentation¶," *firebase*. [Online]. Available: https://firebase-arduino.readthedocs.io/en/latest/. [Accessed: 20-Apr-2019].
[115] "Bluetooth overview | Android Developers," *Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/Bluetooth. [Accessed: 20-Apr-2019].
[116] "Bluetooth low energy overview | Android Developers," *Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/connectivity/Bluetooth-le.html. [Accessed: 20-Apr-2019].
[117] B. Ray, "Bluetooth Vs. Bluetooth Low Energy: What's The Difference?," *Link Labs*. [Online]. Available: https://www.link-labs.com/blog/Bluetooth-vs-Bluetooth-low-energy. [Accessed: 20-Apr-2019].
[118] "flutter_blue | Flutter Package," *Dart Packages*, 04-Feb-2019. [Online]. Available: https://pub.dartlang.org/packages/flutter_blue. [Accessed: 20-Apr-2019]
[119] "Password Reset Is Critical For A Good Customer Experience," *Auth0*. [Online]. Available: https://auth0.com/learn/password-reset/. [Accessed: 20-Apr-2019].
[120] Troy Hunt, "Everything you ever wanted to know about building a secure password reset feature," *Troy Hunt*, 26-Feb-2019. [Online]. Available: https://www.troyhunt.com/everything-you-ever-wanted-to-know/. [Accessed: 20-Apr-2019].
[121] P. Birdsall and P. Birdsall, "Enabling Firebase Cloud Messaging Push Notifications with Flutter.," *Medium*, 04-Sep-2018. [Online]. Available: https://medium.com/flutterpub/enabling-firebase-cloud-messaging-push-notifications-with-flutter-39b08f2ed723. [Accessed: 20-Apr-2019].
[122] "How to get your Website or App ready for GDPR and other Privacy Standards," *UVISION*, 18-May-2018. [Online]. Available: https://uvision.co/technology/how-to-get-your-website-or-app-ready-for-gdpr-and-other-privacy-standards/. [Accessed: 20-Apr-2019].

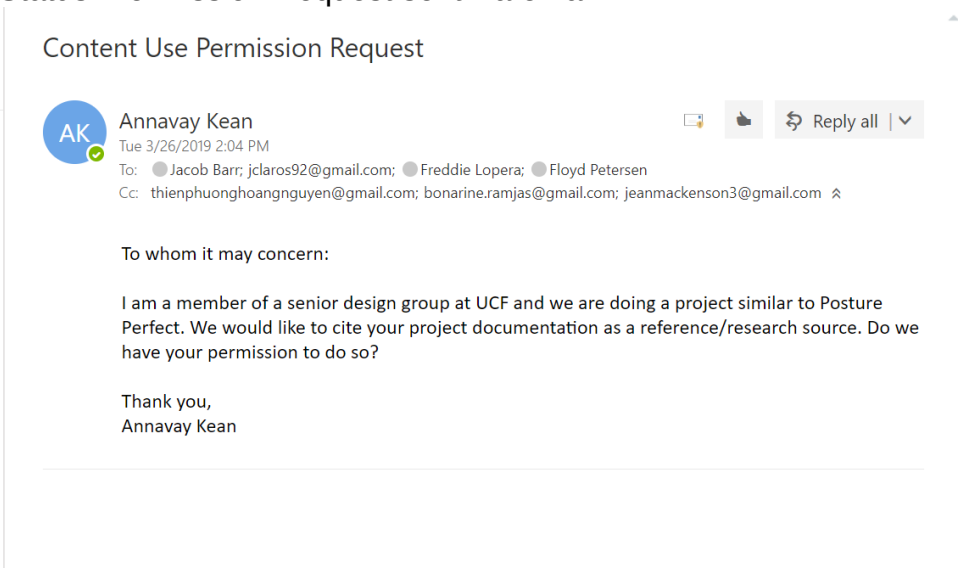
[123] “How To Submit An App To The Google Play Store,” *Clearbridge Mobile*, 13-Mar-2019. [Online]. Available: https://clearbridgemobile.com/how-to-submit-an-app-to-the-google-play-store/. [Accessed: 20-Apr-2019].
[124] “Upload an app - Play Console Help,” *Google*. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/113469?hl=en. [Accessed: 20-Apr-2019].
[125] Apple Inc, “App Store Review Guidelines,” *App Store Review Guidelines - Apple Developer*. [Online]. Available: https://developer.apple.com/app-store/review/guidelines/. [Accessed: 20-Apr-2019].
[126] N. Davis, “The History and Basics of IPC Standards: The Official Standards for PCBs,” *All About Circuits*, 20-Oct-2017. [Online]. Available: https://www.allaboutcircuits.com/news/ipc-standards-the-official-standards-for-pcbs/. [Accessed: 20-Apr-2019].
[127] “Class 2 vs. Class 3,” *Circuitnet*. [Online]. Available: http://www.circuitnet.com/experts/86649.html. [Accessed: 20-Apr-2019].
[128] *W3C*. [Online]. Available: https://www.w3.org/standards/webdesign/accessibility. [Accessed: 20-Apr-2019].
[129] “International Privacy Standards,” *Electronic Frontier Foundation*. [Online]. Available: https://www.eff.org/issues/international-privacy-standards. [Accessed: 20-Apr-2019].
[130] Y. So, “Designing for Mobile Apps: Overall Principles, Common Patterns, and Interface Guidelines,” *Medium*, 12-May-2017. [Online]. Available: https://medium.com/blueprint-by-intuit/native-mobile-app-design-overall-principles-and-common-patterns-26edee8ced10. [Accessed: 20-Apr-2019].
[131] “Understand Firebase projects | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/projects/learn-more#best-practices. [Accessed: 20-Apr-2019].
[132] *Industry Standards for a Mobile App Development*. [Online]. Available: https://devtechnosys.com/know-about-industry-standard-for-app-development. [Accessed: 20-Apr-2019].
[133] *Martyn Currey*. [Online]. Available: http://www.martyncurrey.com/category/Bluetooth/. [Accessed: 20-Apr-2019].
[134] W. Arshad and W. Arshad, “Flutter | Progress indicators plugin | Loaders on the way!,” *Medium*, 19-Aug-2018. [Online]. Available: https://medium.com/@wal_33d/flutter-progress-indicators-plugin-loaders-on-the-way-6aa983490253. [Accessed: 21-Apr-2019].
[135] “progress_indicators | Flutter Package,” *Dart Packages*, 19-Aug-2018. [Online]. Available: https://pub.dartlang.org/packages/progress_indicators. [Accessed: 21-Apr-2019].
[136] B. Vanbilsen, “Flutter SDK Tutorial - Basic Navigation and Routes! (App Development),” *YouTube*, 22-May-2017. [Online]. Available: https://www.youtube.com/watch?v=RLyw-_MLLTo. [Accessed: 21-Apr-2019].
[137] “Navigation & routing,” *Flutter by Google*. [Online]. Available: https://flutter.dev/docs/development/ui/navigation. [Accessed: 21-Apr-2019].

[138] M. Thomsen and M. Thomsen, “Beautiful, animated charts for Flutter,” *Medium*, 23-Mar-2018. [Online]. Available: https://medium.com/flutter-io/beautiful-animated-charts-for-flutter-164940780b8c. [Accessed: 21-Apr-2019].
[139] “charts_flutter | Flutter Package,” *Dart Packages*, 15-Feb-2019. [Online]. Available: https://pub.dartlang.org/packages/charts_flutter. [Accessed: 21-Apr-2019].
[140] Suragch, “Saving and reading data in Flutter with SharedPreferences,” *Medium*, 04-Jan-2019. [Online]. Available: https://medium.com/@studymongolian/saving-and-reading-data-in-flutter-with-sharedpreferences-bb4238d3105. [Accessed: 21-Apr-2019].
[141] “flutter_blue | Flutter Package,” *Dart Packages*, 04-Feb-2019. [Online]. Available: https://pub.dartlang.org/packages/flutter_blue. [Accessed: 21-Apr-2019].
[142] S. Biswas and S. Biswas, “Flutter: Adding Bluetooth Functionality,” *Medium*, 05-Mar-2019. [Online]. Available: https://medium.com/flutter-community/flutter-adding-Bluetooth-functionality-1b9715ccc698. [Accessed: 21-Apr-2019].
[143] “A Developers Guide To Bluetooth,” *Bluetooth Technology Website*. [Online]. Available: https://blog.Bluetooth.com/a-developers-guide-to-Bluetooth. [Accessed: 21-Apr-2019].
[144] *Martyn Currey*. [Online]. Available: http://www.martyncurrey.com/category/Bluetooth/. [Accessed: 20-Apr-2019].
[145] *Martyn Currey*. [Online]. Available: http://www.martyncurrey.com/hc-05-and-hc-06-zs-040-Bluetooth-modules-first-look/. [Accessed: 20-Apr-2019].
[146] “admin,” *HC-05 Bluetooth Module Pinout, Specifications, Default Settings, Replacements & Datasheet*. [Online]. Available: https://components101.com/wireless/hc-05-Bluetooth-module. [Accessed: 20-Apr-2019].
[147] “admin,” *HC 06 Bluetooth module pinout, features & datasheet*. [Online]. Available: https://components101.com/wireless/hc-06-Bluetooth-module-pinout-datasheet. [Accessed: 20-Apr-2019].
[148] *Martyn Currey*. [Online]. Available: http://www.martyncurrey.com/hm-10-Bluetooth-4ble-modules/#more-4463. [Accessed: 20-Apr-2019].
[149] “The Guide to Bluetooth Modules for Arduino,” *Into Robotics*. [Online]. Available: https://www.intorobotics.com/pick-right-Bluetooth-module-diy-arduino-project/. [Accessed: 20-Apr-2019].
[150] “Bluno Bee - Turn Arduino to a Bluetooth 4.0 (BLE) Ready Board,” *DFRobot*. [Online]. Available: https://www.dfrobot.com/product-1073.html. [Accessed: 20-Apr-2019].
[151] “BLE-Link__SKU_TEL0073_,” *DFRobot*. [Online]. Available: https://wiki.dfrobot.com/BLE-Link__SKU_TEL0073_. [Accessed: 20-Apr-2019].
[152] *Getting Started with NodeMCU ESP-12 using Arduino IDE: Blinking an LED*. [Online]. Available: https://circuitdigest.com/microcontroller-projects/getting-started-with-nodemcu-esp12. [Accessed: 20-Apr-2019].

[153] SeeedStudio, “ESP-32S Wifi Bluetooth Combo Module,” *Seeed Studio*, 2019. [Online]. Available: https://www.seeedstudio.com/ESP-32S-Wifi-Bluetooth-Combo-Module-p-2706.html. [Accessed: 20-Apr-2019].
[154] “ESP32 – Cheapest IoT WiFi and Bluetooth ready module,” *ESP32 - Cheapest IoT WiFi and Bluetooth ready module*. [Online]. Available: https://www.geekstips.com/esp32-review-idf-programming-tutorial/. [Accessed: 20-Apr-2019].
[155] “ADDA AD6505HX-EEB 5V 0.5A 4Wire For HP Pavilion DV6 DV6-6000 DV6-6029 DV6-6050 DV6-6090 DV6-6100 DV7 DV7-6000 650797-001 CPU Fan,” *Amazon*. [Online]. Available: https://www.amazon.com/AD6505HX-EEB-Pavilion-DV6-6000-DV6-6029-650797-001/dp/B06Y62PF25. [Accessed: 21-Apr-2019].
[156] “MakerCase,” *MakerCase*. [Online]. Available: https://www.makercase.com/. [Accessed: 21-Apr-2019].
[157] Iotguider. (2019). Serial communication between NodeMCU and Arduino. [online] Available at: https://iotguider.in/esp8266-nodemcu/serial-communication-between-nodemcu-and-arduino/ [Accessed 1 Aug. 2019].
[158] Arduino.cc. (2019). Arduino - SoftwareSerial. [online] Available at: https://www.arduino.cc/en/Reference/SoftwareSerial [Accessed 1 Aug. 2019].
[159] MYBTECHPROJECTS. (2019). Serial Communication between NodeMCU and Arduino. [online] Available at: https://mybtechprojects.tech/serial-communication-between-nodemcu-and-arduino/ [Accessed 1 Aug. 2019].
[160] Microcontroller Projects. (2019). Communication between two ESP8266 module. [online] Available at: https://www.microcontroller-project.com/esp8266-inter-communication-using-arduino-ide.html [Accessed 1 Aug. 2019].
[161] MDN Web Docs. (2019). An overview of HTTP. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview [Accessed 1 Aug. 2019].
[162] Firebase. (2019). Firebase Realtime Database | Firebase Realtime Database | Firebase. [online] Available at: https://firebase.google.com/docs/database/ [Accessed 1 Aug. 2019].

7.2 Permissions Requested / Acquired

Posture Perfect
Status: Permission Request sent via email.



Content Use Permission Request

AK

Annavay Kean
Tue 3/26/2019 2:04 PM
To: Jacob Barr; jclaros92@gmail.com; Freddie Lopera; Floyd Petersen
Cc: thienphuonghoangnguyen@gmail.com; bonarine.ramjas@gmail.com; jeanmackenson3@gmail.com

Reply all

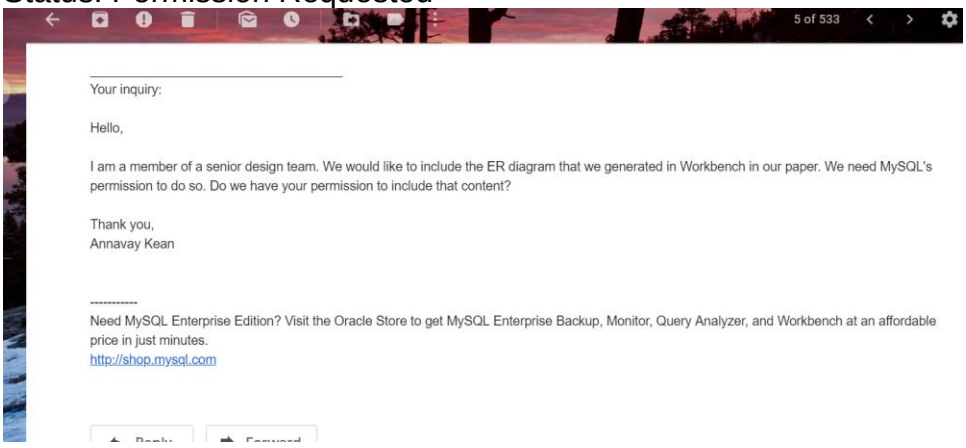
To whom it may concern:

I am a member of a senior design group at UCF and we are doing a project similar to Posture Perfect. We would like to cite your project documentation as a reference/research source. Do we have your permission to do so?

Thank you,
Annavay Kean

Draw.io diagrams permission granted:
https://desk.draw.io/support/solutions/articles/16000042494-what-are-the-usage-terms-for-diagrams-produced-by-draw-io-

MYSQL Workbench ER Diagram:
Status: Permission Requested



5 of 533

Your inquiry:

Hello,

I am a member of a senior design team. We would like to include the ER diagram that we generated in Workbench in our paper. We need MySQL's permission to do so. Do we have your permission to include that content?

Thank you,
Annavay Kean

Need MySQL Enterprise Edition? Visit the Oracle Store to get MySQL Enterprise Backup, Monitor, Query Analyzer, and Workbench at an affordable price in just minutes.
http://shop.mysql.com

MakerCase Box Diagram:
Status: Permission Requested