Senior Design II

Final Project Document

Study Spot Finder

Group 12

Andrew Kwong C.p.E.

Darwin Motato E.E.

Maria Herbas Gutierrez C.p.E.

Perla Del Castillo C.p.E.

Table of Contents

Table List

List of Figures

1. Executive Summary

In today's world, time has become an invaluable commodity. This is especially true for university students, who always seem to be in a rush to get from one class to the next, cramming multiple assignments in one night or even pulling all nighters in the library. The problem with time is that it only goes one way. Once it is used up, you can't get it back. That is why all we can do as humans is to value every moment we have and minimize any loss of time in our daily lives. Many applications nowadays have time calculated down to the minute, so you can know exactly when your next meeting is how long it will take you to get to your destination, or even when your food will arrive at your doorstep.

However, one big issue that hasn't properly been addressed yet is an issue that could save university students, some of the busiest people of all, immense amounts of lost time. Everyday students lose valuable time looking for a place where they can sit down and study for their classes. They will waste precious hours over the course of their entire education searching multiple floors and countless buildings for that perfect desk, the one with a comfy chair, a nearby outlet, etc. Instead they turn away, disappointed to find that their spot has already been taken. This process repeats again and again for many students - that wasted time could be utilized for additional studying time for that student's upcoming exam, or for preparation time before studying (e.g. getting food, water, materials)

Our team is proposing a reservation system for study spaces that will be easily scalable and accessible to all students. It streamlines the process of finding a proper space and instead allows the user to focus their time more on what matters: their education. Using a combination of a hardware device and a mobile app, students can locate their preferred spot with their desired amenities, reserve it, and arrive to find their spot reserved and ready for their use. Once they confirm their spot, they can rest easy and begin to do what they came to do.

This project was inspired by countless first and secondhand experiences. On far too many occasions we found ourselves searching floor after floor for spaces to accommodate our needs, adding pointless frustration and subtracting valuable time from our studying experience. We want to eliminate this useless, ancient process of trial and error and bring it to the 21st century, as so many things have been. More importantly, we want to improve the experience of all students through the use of technology and demonstrate that any problem can be solved with ingenuity and creativity.

2. Project Description

2.1. Motivation

As college students, we sometimes find ourselves struggling to find a study space. At the University of Central Florida, we might have it a bit harder due to the large number of students on campus and how far apart study spaces are from each other. This ends up wasting valuable study time, gives us frustration, and even can ruin a perfect day for studying.

Our solution is an interconnected network of hardware devices attached to study tables that allow users to find out which spots are available. These devices can be attached to different study tables in different buildings around campus. This device has an app that students can download to reserve a spot as well as view all available spots. The device has lights that represent different states such as: “available” in green, “reserved” in yellow, and taken in red, in order to easily communicate its status to students. The user can easily find available spots on the app, tentatively reserve it, and lastly confirm by arriving to the spot and inputting the correct keycode. We believe this solution makes finding a study spot a breeze and we hope that students spend more time studying than trying to find where to study.

2.2. Goals and Objectives

The following were our main goals for this device, which guided our decision making and design process for the rest of the project:

- The application is easy to use and will work in real-time.
- The device is small and not an inconvenience when attached to the table.
- The device is low cost and energy efficient.
- The device has buttons for the user to confirm a reservation
- The device participates in a wireless confirmation process when prompted by the web app
- The mobile app shows available spots for the user in the mobile app
- The mobile app shows relevant information to the user such as the amount of outlets and capacity at each spot
- The mobile app provides the user with the remaining time available to claim table once reserved
- There is a time out feature to renew the availability if the spot is not confirmed or if the spot reservation is canceled
- The mobile app gathers university information to limit use to students and collects PIDs to limit the reservation to only one user per student
- Admins can input the characteristics of study spots such as outlets, or computer inside of each study spot
- Admins have a different dashboard and sign-in information when using the app to configure the device

Desirables
The following goals are ideally what we wanted in our device but are not necessary to the main functionality of the device. These goals have less priority and are addressed only if there is enough time:

- The device uses a camera for user supervision of the area when the user is not present at the spot
- The device uses a microphone to determine the noise level of the environment.
- The device uses Ultrasonic sensors to detect human presence to verify user left
- The device uses PIR sensors to detect human presence to verify user has left
- The device uses an Accelerometer to detect human presence to verify user left

Optionals
The following goals are optional to this project, and were pursued only as stretch goals once all fundamental and desirable goals have been achieved:

- The app contains an internal map that the user can use to find the location of the spot
- The device has direct charging capabilities, including a port for easy recharge of the device.

2.3. Requirements

Power
If implemented, the device requires to monitor a single study spot and stays active during the entire time that the building is open. This means that it always needs to consume power. In order to keep operating costs low, it consumes a small amount of electricity. A logical starting point for a base requirement was equal to or less than the power consumption of a CFL bulb (~15 Watts), since they are on all of the time as well. This allows the device to be marketable and more desirable to consumers who are concerned with operating costs.

Also, since we were dealing with Wi-Fi technology, microcontrollers and other small electronics, we needed a DC power source of around 5V, a common voltage for logic devices. To supply this power, we used a battery, as well as a power management system in order to maximize the use of a single charge. Other smaller logic devices required a 3.3V input as well. To accommodate for different voltage requirements, voltage dividers were put in place to allow the correct voltages to be distributed. To further reduce the power consumption of the device, It operates around a watchdog timer in order to enter and exit sleep mode. This allows it to collect data at specified intervals, but quickly enters a low power mode when it is dormant.

Connectivity
Since the location is within a building, Bluetooth connectivity only needs to be confined to that building, which equates it to a Class 2 device (10 Meters). The actual physical

location of the study spot is stored in the app once it is initially configured to a desk so Students can find their way to it from outside the building. In order to speak to other devices, a mesh network must be established. As for connection to the database and, ultimately, to the user, the device needed a Wi-Fi module to connect to the internet at the desired building, to send and receive data from the database informing it about status changes and reservation confirmations.

Size

This device is convenient and simple, as it will appear on all the study spaces. To achieve this, it is as small as possible while still achieving its function. The size for this device was supposed to be within the volume of 14cm x 14cm x 16cm, yet the actual measurements are 10x10x19.5. This allows the device to occupy a minimal amount of space while containing proper electronics and components.

Ease of Use

This device is intuitive for students as this should be accessible to everyone. In order to achieve this, the device has one central, visible button to claim / release the study space. This button is large and also very reliable / durable since it will be pressed multiple times in a day. The rest of the interactions takes place in the application, which was designed to be user-friendly. To notify users of the study space's availability, there are 2 LEDs (Or a color adjustable LED strip) to indicate status. The simplicity of this design allows all students to easily adapt to using this device.

As for administrators, this device has to be configured when it is first installed in order to input important categorizing information. To make this device as User friendly as possible, there is a separate way to configure the device before it is set for use, which is simple, but secure as to prevent reconfiguration by students.

2.4. Project Specifications

The project specifications were derived from the Requirements section. All our requirements were listed into verifiable, unambiguous specs that satisfies the needs of the project. These project specifications are used to test and verify that our device meets all of our goals, thereby accomplishing its main function.

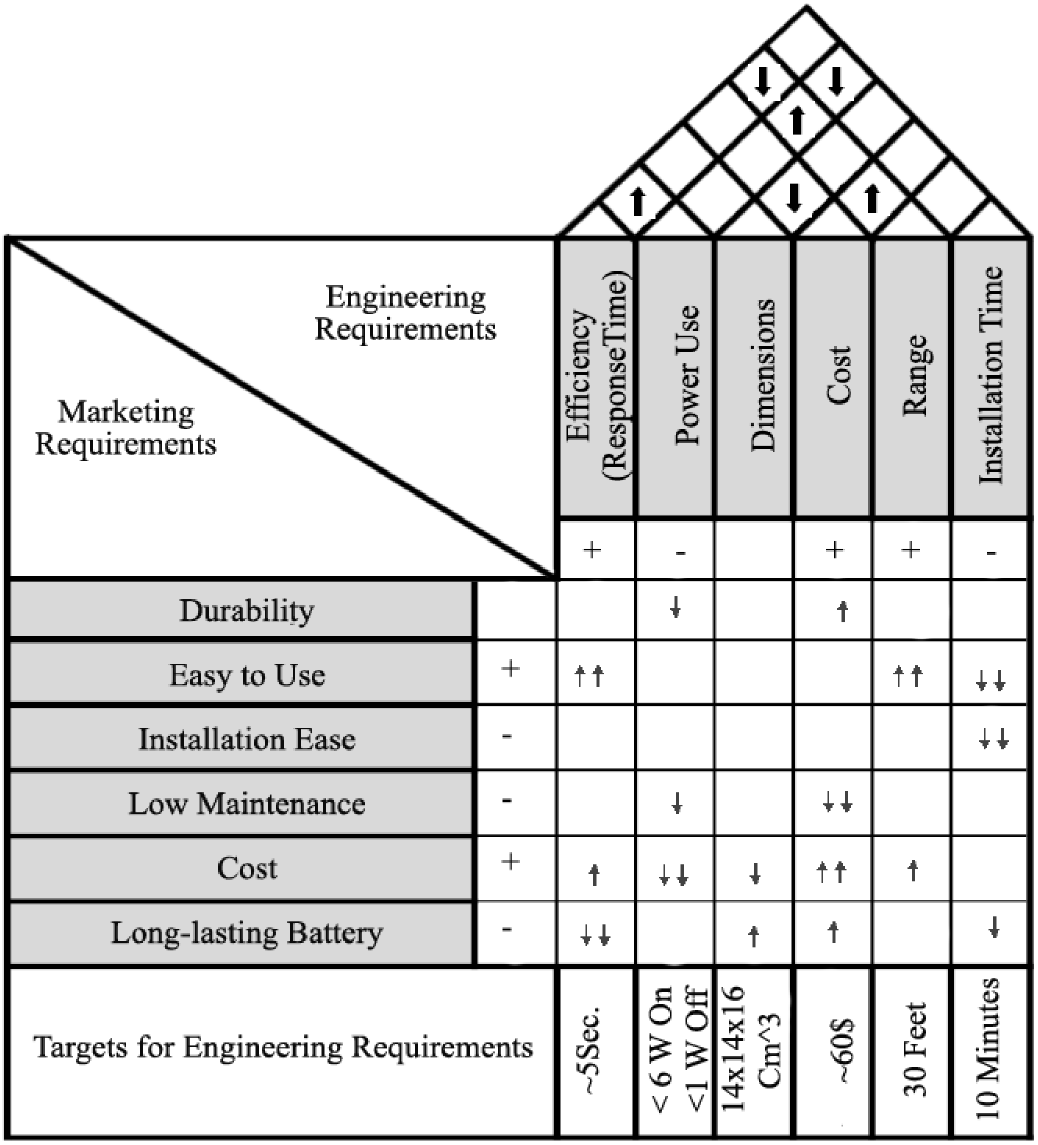
1	**Physical Device**
1.1	The power supply provides 9V with a tolerance of 15%
1.2	The voltage supplied to the LED circuit is 2.1V with a tolerance of 15%
1.3	The current supplied to the LED sub-circuit is 9mA with a tolerance of 10%
1.4	The device consumes <100mA when Wi-Fi is not required

1.5	The device begins transmitting data to Wi-Fi when required within 6 seconds of waking up from Low Power mode
1.6	The device changes its signal LED color to reflect its status within 4 seconds of its event trigger
1.7	The device weighs under 3 lbs.
1.8	The volume of the device does not exceed 14cm x 14cm x 16cm
1.9	The device sustains a reasonable amount of physical abuse from daily use
1.10	The device checks for status updates every 5 seconds
2	**Reservation System**
2.1	The mobile app reflects any changes to available spots within 5 seconds of an event trigger
2.2	The mobile app updates the database with a reservation within 3 seconds of the user sending a reservation request
2.3	The device enters reservation mode within 7 seconds of the user pressing the reserve button on the app
2.4	The device updates the database to confirm the reservation within 3 seconds of the user pressing the confirm button on the device
2.5	The device returns to low power mode within 4 seconds of the user cancelling the reservation
2.6	The app asks the user if they wish to keep the reservation after 5 minutes of submitting a reservation if confirmation has not taken place
2.7	The app triggers a status reset after 10 minutes of inactivity following a reservation request.
2.8	The device triggers a status reset after 10 minutes of inactivity following a reservation request.
2.9	The device remains awake during the entire process of the reservation
3	**Spot Configuration**
3.1	Any changes to device settings by admin reflects in database
3.2	Users are prohibited from making changes to spot configurations

Table 1: Project Specifications

2.5. House of Quality

Figure 1 lays out the customer's needs and how these needs can be met based on capabilities that the team provided. It considered the customer’s needs and translated these needs into a well-developed written plan by prioritizing customer’s needs based on which requirements were the most important. Moreover, it specified which ones were traceable requirements, provided structure and allocated resources. Table 2 shows the legend for the House of Quality Diagram for a better understanding of requirements and their importance.



Marketing Requirements \ Engineering Requirements		Efficiency (ResponseTime)	Power Use	Dimensions	Cost	Range	Installation Time
		+	-		+	+	-
Durability			↓		↑		
Easy to Use	+	↑↑				↑↑	↓↓
Installation Ease	-						↓↓
Low Maintenance	-		↓		↓↓		
Cost	+	↑	↓↓	↓	↑↑	↑	
Long-lasting Battery	-	↓↓		↑	↑		↓
Targets for Engineering Requirements		~5Sec.	< 6 W On <1 W Off	14x14x16 Cm^3	~60$	30 Feet	10 Minutes

Figure 1: House of Quality Diagram

Legend	
↑↑	Strong Positive Correlation
↑	Positive Correlation
↓	Negative Correlation
↓↓	Strong Negative Correlation
+	Positive Polarity
-	Negative Polarity
	Blank spaces mean no correlation

Table 2: Legend for House of Quality Diagram

2.6. Project Block Diagram

Figure 2 contains the overall block diagram of our device and how it interacts with the network and its android application to provide the user with an efficient way of reserving a study spot. Each block contains a system that was accomplished by the mentioned students. It also contains a division between the hardware and software components.

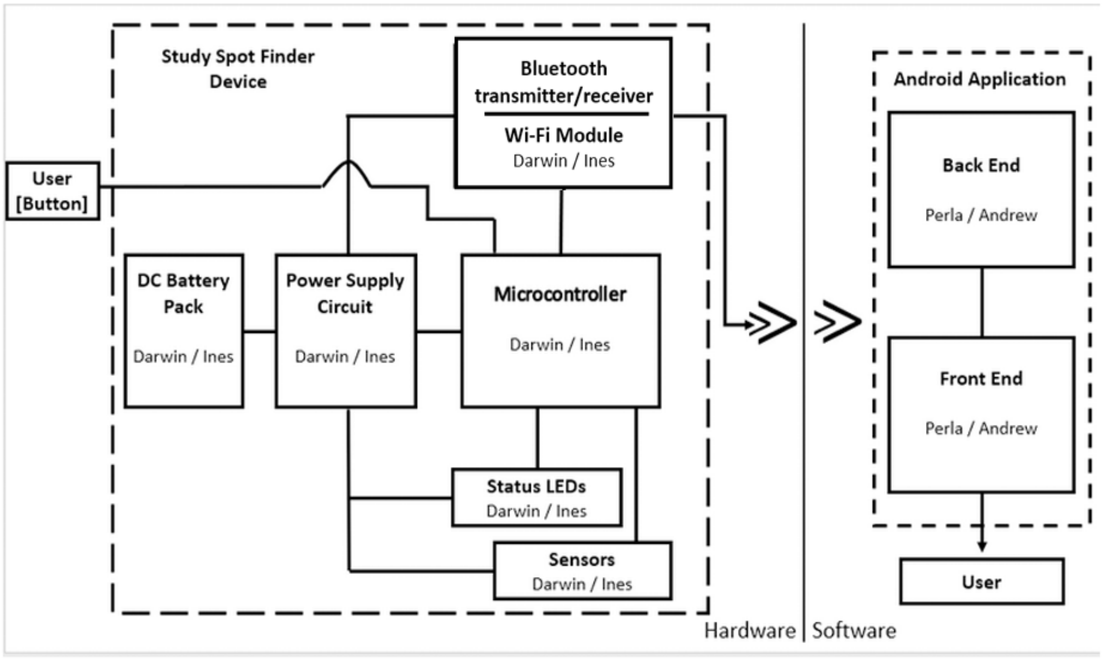


Figure 2: Block Diagram

In terms of hardware, the device is controlled by an ARM-based microcontroller that deals with the logic, sensor inputs, the Wi-Fi module, confirmation button/keypad, and the status LEDs. This device is powered from a rechargeable lithium-ion battery pack, which feeds into a power management circuit, designed by one of our group members. This circuit ensures the correct power is distributed to each device (~5V for Microcontrollers, 3.3V for Wi-Fi module), and reduces the loss of power (Proper grounding).

On the software side, the data from the Wi-Fi module on the Finder is sent in the form of a JSON String, where the Android application, which is written in either React Native or Android Studio, processes and presents the data to the user in an intuitive and friendly way. The data from the Finder is sent to an Amazon EC2 instance for extraction, transformation, and loading.

3. Constraints and Standards

In this section, we will discuss all related standards and constraints that apply or have an impact to our project. A brief overview will be given about the constraints and standards. Moreover, we will discuss how to address any constraints that are prone to affect us.

3.1. Constraints

Design Constraints

This product has considered both hardware and software constraints, so there were no surprises in the future when the product was in the building process. It is important to identify and address these constraints when partaking in any design decisions, in order to ensure that the final design was completed as close to our ideal image as possible. Multiple constraints are listed below which guided our design process:

Time Constraints

For this project, we were limited to 2 semesters. Therefore, we were limited in time to perform research, testing, development, and prototypes. Some sacrifices had to be made in all these areas in order to produce a finished product in the time allotted. Ideally, there would be more time needed to work on research and the best ways to minimize power consumption and maximize accuracy. Since this was not the case, the most time and effort were put into making a functioning product that meets as much of the requirements as possible, with less emphasis on marketability and consumer needs. In order to abide by these constraints, we established priorities in terms of what we wanted to do in order to allow us to accomplish the most important tasks first and secure a working product before working on less necessary features.

Dimension Constraints

The device used in this project can be placed on desks of all sizes, where students will be utilizing most of the available space. This means that the device is large enough to be noticeable and contain all the necessary components, but small enough not to create a distraction or inconvenience to the students that are studying.

Economic Constraints
The cost of this device was first and foremost limited to the budget of this team, but more importantly, it was limited by the fact that this product was made to be sold in mass quantities to work together and provide a service. Therefore, they are relatively cheap so that multiple devices can be used at once. This will allow for the greatest marketability and consumer desirability. To determine a reasonable constraint, the team observed similar products in the market and found an average selling point.

Manufacturability Constraints
This device required a housing to contain all the components. Considering the state of today’s “Maker” mindset and technology, it was relatively easy to manufacture complex housing and structural components using 3D printers and other common technologies, however, due to other factors, the team had to build a handmade housing. Some examples of research in these options were explored further along this document. These manufactured parts were by complexity, cost, and durability. Besides manufacturing certain parts ourselves, we were constrained by parts. This included PCBs, small logic devices, etc. We planned and designed knowing these manufacturing constraints in mind.

Power Constraints
In order to minimize the amount of maintenance required, we needed to keep the power draw of this device to be as low as possible to allow a single charge of battery to last for a long time. Having the battery run out too quickly requires a device to constantly need charge which is a problem that can compound quickly when you factor in the idea that this device will be sold in multiple units. It would be very tedious and unattractive to consumers if a constant recharge is required for this device. Sleep modes and other forms of energy conservation practices keep the power consumption low and allow the device to run for many hours or possibly days without running out of power.

3.2. Standards

Standards are important for industries building new products because it assists with the procedures and requirements for the design of the product. Standards ensure that the product is efficient and reliable through a set of rules, procedures, and requirements. The following standards were used to build the Study Spot Finder product in a safe and proper manner.

Battery Standards
This product was decided to be portable and a rechargeable battery was required for the system. In order to achieve this, ANSI C18.2M was used. ANSI is the American National Standards Institute which is a private non-profit organization in charge of standards for products in the United States. Batteries in the United States follow these standards. These standards mostly apply to portable rechargeable batteries using the following mixtures: Nickel-cadmium, Nickel-metal hydride, Lithium-ion.

These standards give batteries a high reliability where batteries are safe as long as it has not been manipulated by disassembling, crushing, opening or mutilating it. According to ANSI a person can use the battery safely without any battery failure. To prevent previous actions made to the battery in the product, we placed the battery in a case so students cannot access the battery.

PCB standards

IPC, also known as the Association Connecting Electronics Industries, is a trading company in charge of standardizing the production requirement of some electronic equipment. Also, these standards are created to provide electronic manufacturing industries a set of rules and procedures to follow. Some standards includes:

- Printed board
- Printed electronics
- Design standards
- Assembly
- Embedded Technologies
- Electronic enclosures

IPC-221B are standards used and followed for printed board designs used in commercial PCBs. These standards are general requirements for component required and PCB designs. Some standards needed were the following:

- IPC-T-50 Terms and Definitions
- IPC-2615 Printed Board Dimensions and Tolerances
- IPC-D-325 Documentation Requirements for Printed Boards
- IPC-ET-652 Guidelines for Electrical Testing of Unpopulated Printed Boards

These standards helped to build this project in order to achieve the desired outcome. The product has reputation, reliability, and profitability. Moreover, this universal standard approves global usage.

Soldering standards

The most common standards used in the industry regarding soldering connections and connection of electronic assemblies are the IPC J-STD-001, Requirements for Soldered Electrical and Electronic Assemblies and the IPC-A-610, Acceptability of Electronic Assemblies. These standards cover all forms of connections across all kinds of terminals.

Wireless standards

Since this device uses a Wi-Fi module, we needed to comply with all of the standards of wireless Wi-Fi communication (IEEE 802.11). In order to accommodate for our device, our budget, computing power, and simplicity, the most compatible version of this standard to move forward with was 802.11g. This set of standards encompasses most Wi-Fi devices, with backwards compatibility to the original 802.11 standards. These standards encompass the 2.4 GHz frequency range as well, which is most likely what we supported in our Wi-Fi modules.

Testing Standards

There is a useful standard provided by IEEE (IEEE 1012-2016 - IEEE Standard for System, Software, and Hardware Verification and Validation) that covers the entire cycle of the product that we are producing, which covers the verification and validation processes. This allowed us to make sure that all our requirements were accounted for in an organized manner.

Reliability standards

One of the last steps for a life cycle of a product is reliability. A product should be durable and reliable to be sold in the market else legal procedures will follow against the producer. Reliability standards are protection standards against legal terms in order to demonstrate the product’s full functionality for the duration estimated in the life span. These standards ensure that the product will have a meaningful use to the customer. If the product performs most of their features but it does not charge or the system goes down regularly, it can be considered as a defect product and can create issues in the future.

If a defect product gets in the market, we will need to provide replacement service or service to repair any damage component. The software section will be updated constantly to provide the latest configuration and fix debugs. The hardware section will need to inspect once every life span of the product in order to replace parts or update the design. Also, the product’s parts can be reused for other purposes. A reliable product will attract more customers which will increase the revenue. This revenue can then be used to upgrade and produce more products.

Programming standards

Many different programming languages were used during this project. One of the main languages was the C language used in programming the main microcontroller. The C language follows C11 standards (the formal name being ISO/IEC 9899:201). As for the software side, the data that will be exchanged between the microcontroller and database for processing is in a JSON format (JavaScript Object Notation) under the STD90 standard, which defines JSON, as a “lightweight, text-based, language-independent data interchange format”. JSON also lies under the standards ISO/IEC 21778:2017 and ECMA-404.

4. Project Research

The key idea behind this product is to provide students a space to study without wasting time looking for one. Today’s world is continuously making improvements and new products are being introduced on the market at a fast pace. There are many functional products with features that align to our ideal product. Companies such as Embrava and even certain universities themselves have already built products which target more efficient work environments through hardware or software applications.

Our product has a combination of both software and hardware, and research from existing products in both fields can greatly benefit our design process. Some of the most influential examples can be seen below. We also went over how the product was going to differ from our ideal device and what we could take away from products that we didn't see in the current market for what we were aiming for.

4.1. Existing Products & Technology

Embrava

Embrava is an intelligent desk signage for agile work environments. This device was created to target open-plan offices and agile layouts since these offices are prompt to interruptions, and difficulty to find a space or colleague. Moreover, it focuses on employee availability, workspace availability, workplace accessibility and workplace analytics. This company has two different products which has the same goals, but different features based on what the workplace needs. Figure 3 is the Blynclight Series and Figure 4 is the Desk Signage product.

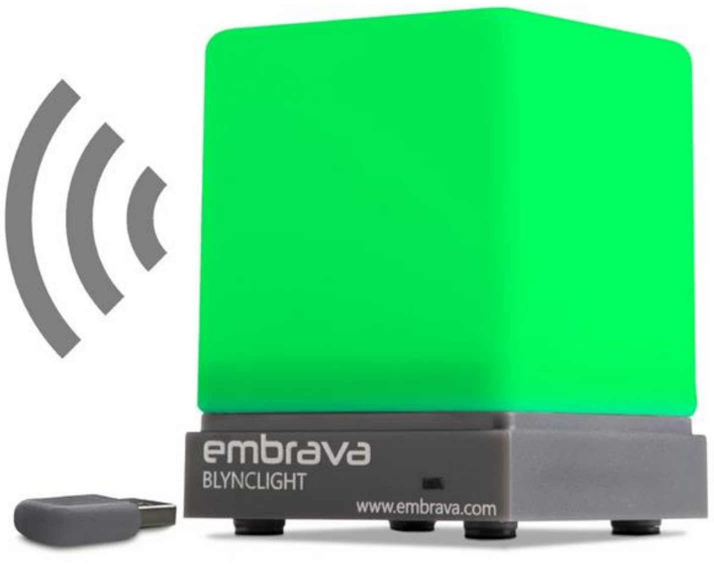


Figure 3: Embrava Blynclight Wireless

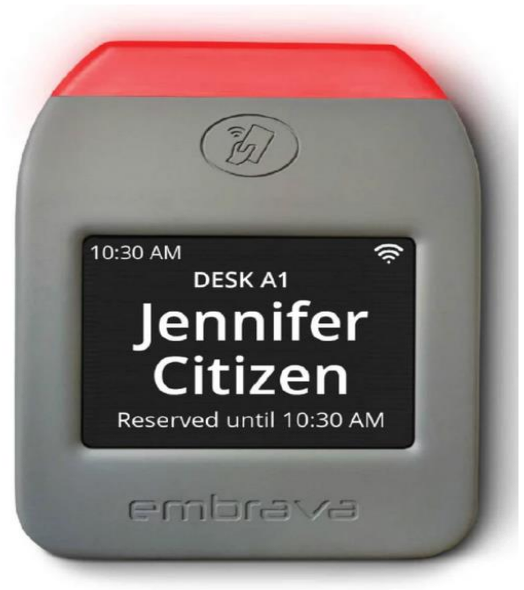


Figure 4: Embrava Desk Sign

These two devices have different device design however the functionality is the same. Both devices oversee telling other employees whether it is busy or not depending if it is on a conference or on a skype business conference. On these two devices, a reservation system is used which tells the availability of the user. Moreover, it uses a location system to locate the user. However, this design differs from our ideal Study Spot design because it requires the employee to dock their computer to the Spot station, which allows it to recognize the employee and mark the location as reserved. We allow user profiles to be made within the app in order to identify the user and implement a keypad confirmation in order to verify that the correct user has confirmed the sport.

University of Minnesota Study Space Finder App

The University of Minnesota has a web application where you can find individual study spots and study rooms for collaboration. This web application displays study spots in every building including an image of the spot itself.

Also, it contains useful information about the spot to help students pick the right area based on their needs. Some of the spot information includes monitor availability, printer access, whiteboards, outlets, noise level of the room, and type of seats (table/chair or lounge seating). Figure 5 shows an example of the mobile version. Here, a map is provided as well as detailed information about the selected study space. There is also a picture of the selected study space.

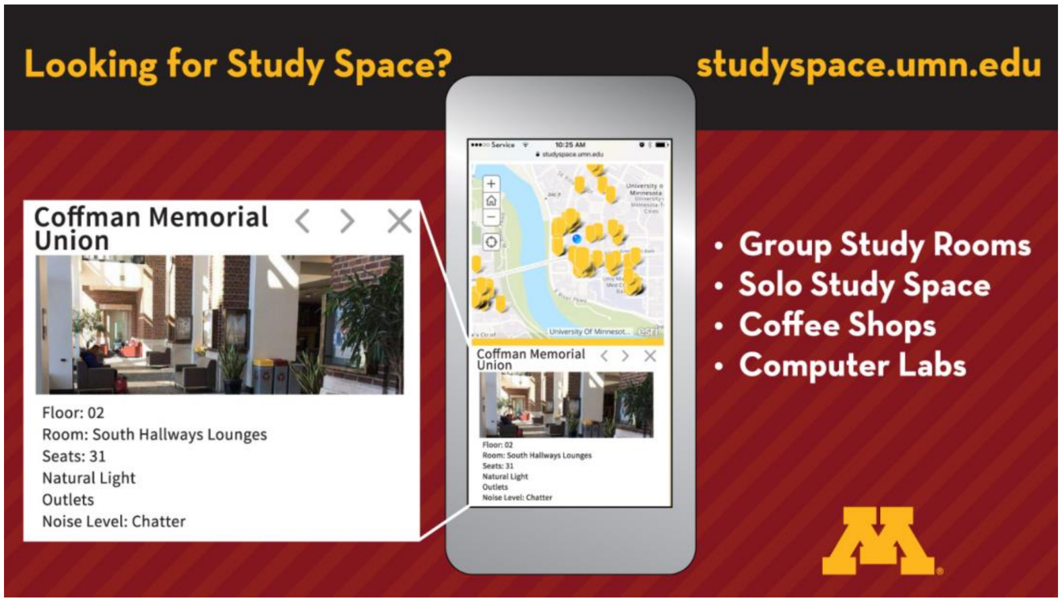


Figure 5: Study Space Finder Mobile App at University of Minnesota

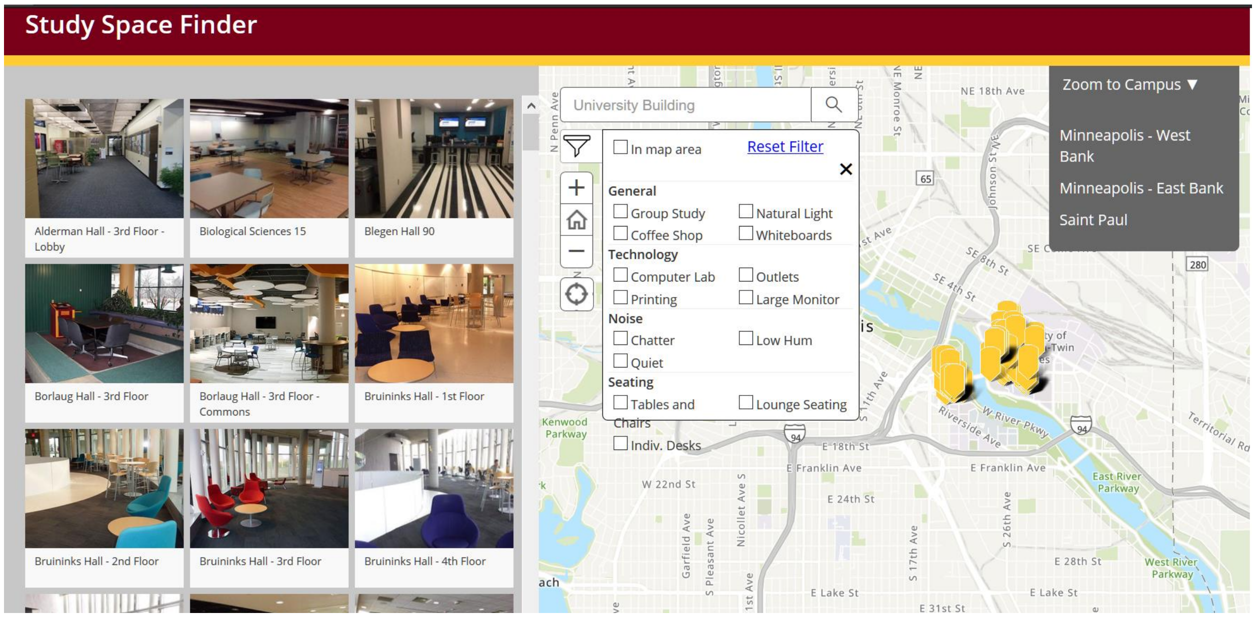


Figure 6: Study Space Finder Web App at University of Minnesota

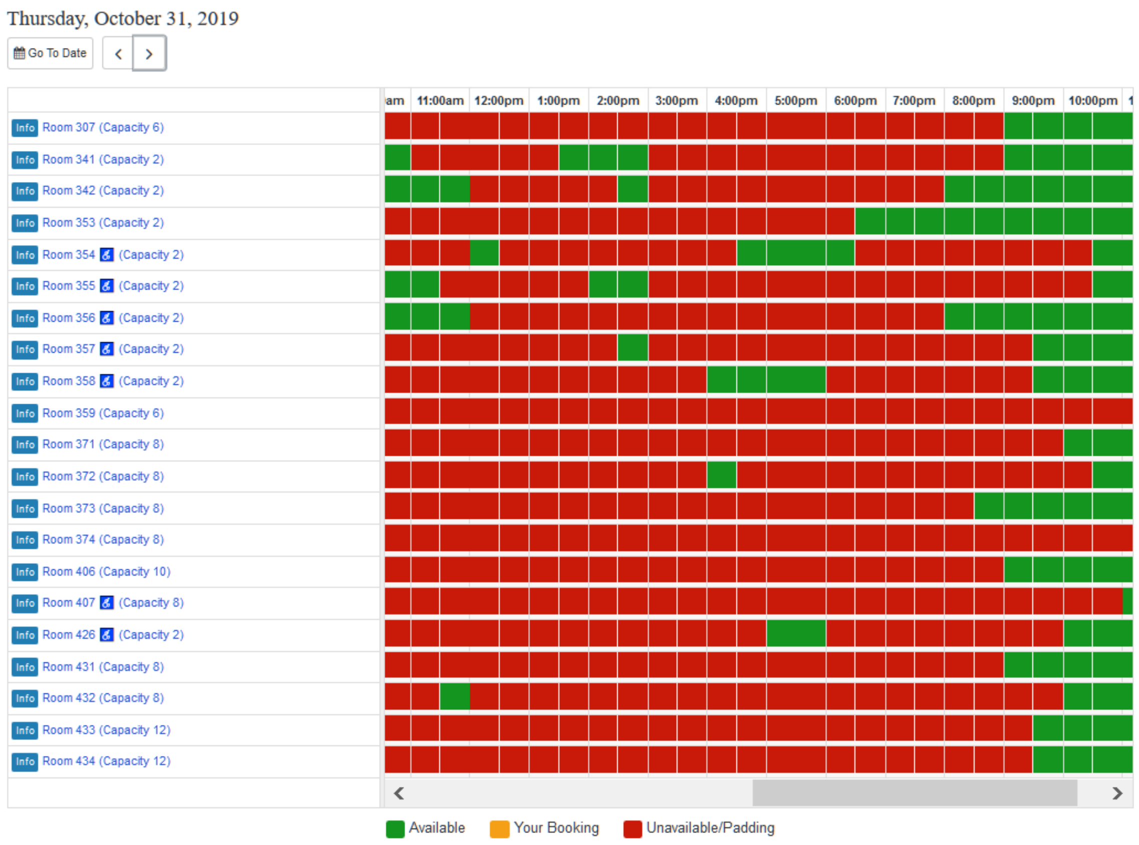
In Figure 6, we see the web version of the Study Space finder app. In this figure we can also notice the ability to filter the visibility of available spots based on checkboxes of what users desire in their study space. These factors were taken into consideration of the design for our Study Spot Finder. The big difference between our products is that the University of Minnesota's Study Space Finder system only finds study spaces, but there is no option to reserve them. This is a crucial step for our product.

UCF Online Room Reservation System

Since the University of Minnesota's Study Spot Finder can only find study spots and not reserve them, we turned to our own university's reservation system for something similar:

Study Rooms. On the website for the John C. Hitt library, there is a page for reserving study rooms in the library which closely resembles something we wanted to create for reserving study spaces in the same building.

The reservation page shown in Figure 7 shows all of the available rooms as well as their status and some details about the room. Once selected, the user can input their credentials and reserve the room. This system helped us in providing ideas for layout and the reservation process and interface, but for our product we focused mostly on instant reservations rather than making them hours or even days in advance. This was an unrealistic expectation for our application, as this is only for study spots such as desks.

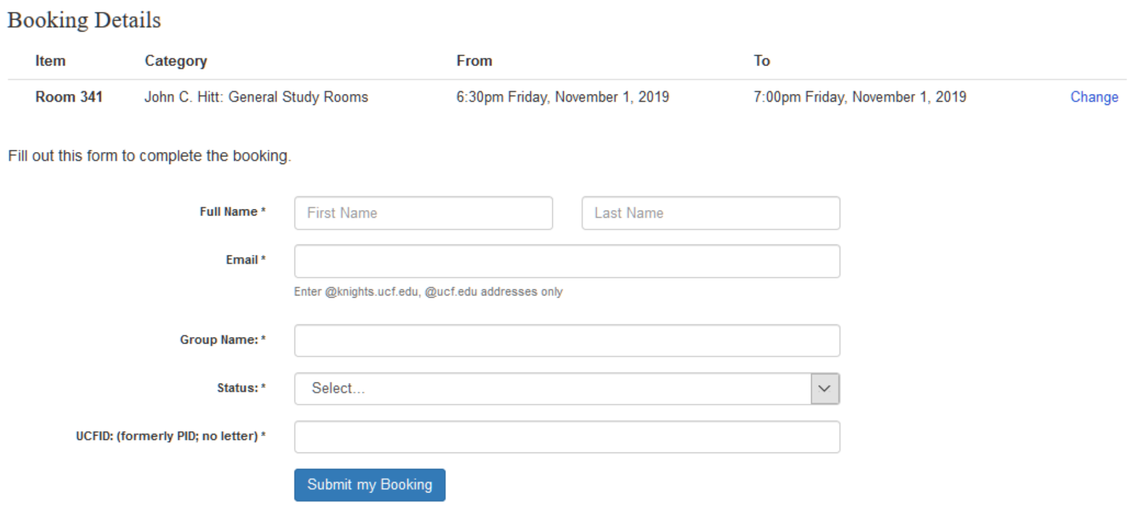

Thursday, October 31, 2019

Info Room 307 (Capacity 6)
Info Room 341 (Capacity 2)
Info Room 342 (Capacity 2)
Info Room 353 (Capacity 2)
Info Room 354 (Capacity 2)
Info Room 355 (Capacity 2)
Info Room 356 (Capacity 2)
Info Room 357 (Capacity 2)
Info Room 358 (Capacity 2)
Info Room 359 (Capacity 6)
Info Room 371 (Capacity 8)
Info Room 372 (Capacity 8)
Info Room 373 (Capacity 8)
Info Room 374 (Capacity 8)
Info Room 406 (Capacity 10)
Info Room 407 (Capacity 8)
Info Room 426 (Capacity 2)
Info Room 431 (Capacity 8)
Info Room 432 (Capacity 8)
Info Room 433 (Capacity 12)
Info Room 434 (Capacity 12)

🟩 Available 🟧 Your Booking 🟥 Unavailable/Padding

Figure 7: UCF Study Room Reservation System

Figure 8 demonstrates the process for obtaining information for the booking. One thing that is immediately visible is the bland user interface. However, they require the same information that we planned on using for study space reservation.


Booking Details

Item	Category	From	To	
Room 341	John C. Hitt: General Study Rooms	6:30pm Friday, November 1, 2019	7:00pm Friday, November 1, 2019	Change

Fill out this form to complete the booking.

Full Name *	First Name / Last Name
Email *	Enter @knights.ucf.edu, @ucf.edu addresses only
Group Name: *	
Status: *	Select...
UCFID: (formerly PID; no letter) *	

Figure 8: UCF Study Room Booking Process

4.2. Considerations for Future Technology

As we looked at existing products and technology we saw small advancements in technology to facilitate students to have a better experience at school and tools to succeed in classes. This included a busy light device from embrava and study space finder web app reservation system. Even though the idea design was not based on these previous technologies, they were great ideas found to begin the technology the team planned to build. Our product incorporated some features of these ideas as shown throughout the paper but the technology was improved and built according to our desired goals.

Our expectations for the project was that students find a study space without wasting time and in an organized manner. Sometimes, finding a study space is similar to finding a parking spot. We were hoping that this product reduces the amount of time looking for study space and uses them to study instead. In the requirements specification section, the product was divided into three sections, this means that improvements can still be made after the product was built based on the essentials.

For future improvements, the product is desired to have sensors that will improve its functionality and be more user friendly. Sensors will allow users to have more accurate information about the state of the space and also a microphone could be installed to provide information such as the level of noise of the space.
As of now, integrating these two systems accomplished our goal to solve the problem and including new components that they are lacking brings this device to another level.

4.3. Hardware Design Research

This section includes the research behind this product in order to compare the different hardware components available, which allowed the group to make the right choice and picked the best component based on the needs and overall functionality of the product.

This section also includes comparison tables and decision matrices to accomplish these tasks.

4.3.1. Buttons and Button Placement

In this section, we will be discussing the different desirable options for buttons needed for the reservation confirmation and to turn on and off. Advantages and disadvantages will be discussed as well as the best and final choice of buttons.

Reservation Confirmation Button

When a student reserves a study space, there needs to be a way to confirm that the student has made it to the study space to "confirm" the reservation. This process is unique to the student making the reservation, and it is easy and intuitive for the student. In order to make the process as easy as possible, inspiration was taken from many game shows and theme parks to imagine what a user-friendly button would look like. Ultimately we decided on a large push button on the top of the device. This button is very large in order to easily communicate its function as the "main" button to press.

We decided on a push button since the device could unknowingly be moved or pushed over if a switch or lever were to be used. Since the push button provides a downward force, it does not force the device into any unwanted direction. A push button was also ideal for triggering a single button interrupt, which was ideal for the process that we wanted to integrate it into.

The next decision was deciding whether to fabricate our own button or purchase a premade button. The advantages of making our own button was that we had more creative control over the design, but it would end up being more expensive (Buying separate parts) and would take longer to design (Housing for spring, mechanism to push the button). On the contrary, a premade push button did not have all of the ideal features that we wanted (LED inside push button), but it was less expensive and ready to implement. Ultimately, we moved forward with a premade button in the interest of time and money.

Our final choice for the reservation button was the Jiu Man JM-100mm Arcade LED push button, specified in Table 3. This button is large enough to attract attention as well as sleek. It also had support for LEDs, for aesthetic purposes.

Manufacturer	Site	Cost	Size	Weight	Material
Jiu Man	Amazon	$9.99	100mm	265 g	PVC and ABS Plastic

Table 3: Final Button Choice and Specifications

On/Off Switch vs On/Off button

Since the device should not always be “On” (i.e. when it is being transported or put into storage for any reason), it needed a way to trigger the power down of the device. In the research for this kind of button we had to take into consideration where this button would be located, and its susceptibility to being turned off by users. Table 4 demonstrates the advantages and drawbacks of using different forms of switches. This switch is located inside of the device so that it cannot be accessed by students.

On/Off type	Pros	Cons
Button	Takes up less space, simple action (Push), Easier to hide/ Prevent tampering	Can only be in 2 states (On/Off)
Switch	More intuitive for On/Off operation, Can be in 2 states	Takes up more space, Complex Action, Harder to hide/ Prevent tampering

Table 4: Comparison of Button vs Switch

Ultimately we decided to go with a push button which will have two states. The part we chose was the Judco model# J-188A-1, specified in Table 5. The button will be located inside the outer casing of the device, making it accessible only to administrators.

Manufacturer	Model	Site	Cost	Rating	Material
Judco	J-188A-1	All Electronics	$1.85	6 Amps @ 125Vac	Plastic

Table 5: On/Off switch Final Choice

4.3.2. LED Notification

LED’s are essential hardware components for many designs as they provide a visual representation of events, lighting for pathways, advertisements or means to display a state of a system. For this product purposes, LED provides an aesthetic appeal as well as it displays information about the state of the system. For example, red lighting is displayed when the study space is busy, yellow lighting is displayed when reserved and green lighting when space is available. LEDs in this case were added as an essential feature.

There are many different types of LED’s in which they differ on shapes, sizes, colors and efficiency. There are three types of categories of LEDs which include miniature, high power, and application-specific. The appropriate LED was selected based on the functionality and characteristics needed, which are to signal students from far and communicate to them the status of the desk. It also uses as little power as possible to extend battery life.

Application-Specific

This kind has several LEDs categories that fall into it RGB, Bi-color and Tri-color, Flash, Alphanumeric and lighting LEDs.

The two types considered for this product were Tri-color or RGB LEDs. Since we only needed 3 colors for our design, Tri-color LED lights would have been very handy, as they can be adapted to any color and therefore take the place of 3 separate LEDs. However, they would have been harder to implement logically and can have weaker brightness than regular LEDs. Single LED lights specialize in one color and can have a stronger glow, although their single function proves more time and power consuming. However, it was the simplest and strongest approach. Some other LED structures were also explored during the research for LED notifications.

LED Strip

LED strip lights is a new RGB LED technology with versatile characteristics that gives them an advantage against LEDs discussed previously. These characteristics include:

- Many individual LED emitters mounted on a narrow and flexible circuit board
- Low-voltage DC power required
- Variety of fixed and variable colors and brightness
- Can be adjust to length needed and includes a double-sided adhesive.
- Less connections and wired
- Variety of customization

Some types of LED strips depend on applications and constraints of the product desired. DC flex strips run on 12 Vdc, these LEDs can be easily installed due to the 3M adhesive it has as well as it is very convenient that is coated with a silicone cover which prevents it from water damages. There are also LEDs strips that are AC LEDs that require an outlet to run, it provides the same characteristics such as 3M adhesive and waterproof casing. However, these AC LED strips were not suitable for our product since our desired product runs on DC which includes batteries instead of outlets. LED rope lights are omni-directional LEDs, the usual form of packaging is in the form of rope light packaging.

Ring Lights LED

One option for the product was to use a similar LED that Alexa uses known as ring lights LEDs. These LEDs are fixed LEDs non-flexible in a round ring packaging. They are known for stability and usually run on low voltage. It is more frequently used for imaging purposes such as an add-on for phones cameras or as illumination for circle mirrors. Due to its brightness, most uses are on the imaging section since it helps to demonstrate any defects and display a brighter image. However, Alexa uses a similar LED but a thinner version on their devices.

LED Choice

After researching among the different options of LEDs and evaluating each option carefully, we created Table 7 to demonstrate the results of our findings. Ultimately we decided on the RGB LED located specified in Table 6.

Manufacturer	Model	Site	Cost	Voltage	Max Current	Luminosity
Jameco Value Pro	RGB LED	Jameco Electronics	$0.39	2-3.2V	~30 ma	5000-9000 mcd

Table 6: LED Final Choice & Specifications

We decided it was the best choice as all of the other options required a higher voltage. All of our logic devices operate at or around 3.3V and 5V, so having an LED falling under that range is more practical for the design of the circuitry for this device.

LED String Type	Operating Voltage (V)	Light Color	Additional Features
Flexible LED circuit	12V	RGB	Flexible, wide PCB
Full Color Flexible LED circuit	12V	RGB	DMX 512 compatible, flexible
Driverless RGB LED Strip Light	12V	RGB	Driverless, cuttable segments
Ring Lights WS2812B	5 V	Vary	Stability and efficient
CornholeRing Lights	4.5 V	Vary	N/A
Lumex	2.1V	R, Y, G	Simple, cheap, accessible

Table 7: Comparison of the different types of LEDs

4.3.3. Sensors

Sensors are not an essential requirement but fall into a desirable requirements section where if there was a slight possibility of including them based on time, the product would have included sensors which would have added another feature to the device. This feature was supposed to determine if the space is busy or available which was going to help the reservation system to be more accurate in case students forgot to cancel a reservation or missed a reservation.

Three main sensors have been discussed during the requirement specification process where the team concluded that ultrasonic, PIR and accelerometer sensors are best suitable for the desired performance of the device.

4.3.3.1. Ultrasonic sensor

Ultrasonic sensors are ideal for this design because it will determine whether there is a student in the study space or not. This sensor determines if there is any object by measuring distance using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. If no reflected wave is received, it means that no object has been found.

This device would be implemented to verify that a student is not present in the spot. This verification is needed if a student reserves a spot and never comes to claim it or forgets to claim it using the confirmation button. We want to be as thorough as possible before resetting a spot to not give students a worse experience. The following sensor options were explored in Table 8.

Sensor	Range	Cost	Resolution	Support
HC-SR04	2cm-400cm	3.95	4cm	Poor
LV-EZ	0cm-645cm	24.95	2.5cm	Good
PING	2.5cm-304cm	29.99	4cm	Great

Table 8: Comparison among Ultrasound sensors

For this design, HC-SR04 will be enough because the device is assumed to be placed on the table close to the student, therefore a wider range is not needed. Moreover, due to the fact that the product will potentially have three more sensors that will get data if the student is in the study space, having a poor support would not cause problems. Also, the cost is very accessible and will not overpass the economic constraint.

4.3.3.2. PIR sensor

PIR sensor, also known as Passive Infrared, allows you to sense motion and it is used to detect if there is a human moving within the sensor range. These sensors use low levels of radiation and the radiation is divided into two halves where in one half sees more or less IR radiation than the other so the output would swing high or low.

The PIR sensor would be beneficial to our project when we want to verify that there isn’t anyone present near the table and the reservation needs to be reset (perhaps due to a user leaving without pressing the button to finish their reservation). Table 9 displays the most commonly used PIR sensors.

Sensor	Range	Cost	Power Supply
Adafruit	Up to 7m	10	5-12V
Parallax 555-28027	Up to 9m	15.64	5V
Seeed studio Mini PIR	2-5m	4.60	5V
HC-SR505	3-7m	3.58	4-20V

Table 9: Comparison among PIR sensors

For this design, the Seeed Studio mini PIR would satisfy the desired requirements, as our distance needs are not very demanding (edges of a table from the center) and it has a low operating voltage. The cost is also the lowest making it an affordable and useful option.

4.3.3.3. Accelerometer sensor

Accelerator sensors can be used to measure the acceleration exerted upon the sensor. Two data given as output for this sensor are:

- Static Force applied on sensor due to gravity -> orientation detection
- Force exerted upon sensor-> movement detection

For this design we would want to use, in the future, the second data that will output the sensor. This will help us to detect movement in the table. As stated in the ultrasonic sensor paragraph, this will allow us to verify that a student is not present in the study spot before resetting it to ‘Available’. This will be used in conjunction with the other sensors. Some of the sensors that were considered for this design are shown in Table 10 which is shown below:

Sensor	Sensitivity	Cost	Power voltage	Frequency
MMA8451 2019 ADAFRUIT	4096 counts/g	7.93	1.95V to 3.6V	400kHz
EVAL-ADXL 372z	+-200g	26.1	1.6V-3.5V	400kHz
Adafruit-ADXL377	6.5mV/g	24.95	3.3V-5V	400kHz
Bosch-0273.141.148-1NV	4096LSB/g	1.5	1.2V-3.6V	10MHz

Table 10: Comparison among Accelerometer sensors

4.3.4. Casing Material Options

There were a variety of options for casing materials especially with new technology such as 3D printing and Laser Cutting. The casing material for this product did not have any limitations in comparison to other products where heat and water should be considered. Yet, these tools were not facilitated for this device, due to restrictions.

There were four main characteristics that this design should have considered when picking the casing material cost, weight, endurance and accessibility. Since this device targets the education market, the cost of the device should be low in order to do a mass production for the demand. Furthermore, as mentioned in the specification requirements, the weight of the device is light. Also, it should be considered that it will function among students, so the device has a good endurance to any falls or damages caused by students. Finally, this material is easily accessible for us to manufacture with and time effectively to increase our chances of success. Table 11a&b shows the different material options researched.

Materials	Pros	Cons	Cost
Plastic	Accessible Can have complex design Easy 3D printing Not breakable if dropped	Easy to melt under high temperatures	Low
Ceramic	Durable Stain resistant Heat resistant Maintainable	Expensive Set-up Heavy Breakable with pressure	Low

Table 11a: Comparison of different casing materials

Materials	Pros	Cons	Cost
Vinyl	Durable Maintainable Good for high traffic areas	Lifespan 10 years Installed over smooth underlayment	Low
Wood	Accessible Good aesthetic	Prone to water damage if no protection	Low
Material/Stainless Steel	Durable Maintainable	Heavy Short circuit concerns	High

Table 11b: Comparison of different casing materials

We decided to go with plastic as it is easily accessible, malleable, lightweight, and sturdy, which is perfect for our project. We ended up doing a handmade housing by using plastic PVC for the case.

Adhesive

In order to connect the pieces of our device together, we needed to use an adhesive. This adhesive would not have impacted the overall look and feel of the device, but should have held properly and provided a sturdy product that would not have fallen apart or feel flimsy. Table 12 demonstrates the options that we researched:

Adhesive	Advantages	Disadvantages
JB Plastic Weld	-Holds extremely well -Adapted specifically to bond plastic -Quick setting time	-Irreversible bond -Reaction can cause the plastic to melt and become disfigured
Plastic Glue Gun	-Extremely quick setting time -Reversible bond	-Can be very messy if not handled properly -Poor adhesion, can be broken -Takes up a lot of volume
Super Glue	-Holds extremely well -Can bond all materials -Small volume	-Takes time to set -Some glues set with a white color

Table 12: Adhesives comparison

Table 13 is a decision matrix that allows us to choose the best option for an adhesive.

	JB Plastic Weld	Plastic Glue Gun	Super Glue
Bond Strength	3	1	3
Appearance	3	1	3
Setting Time	2	3	1
Total	8	5	7

Table 13: Decision Matrix Adhesive

4.3.5. Battery and Battery Casing

The Battery must be able to supply enough power to the logic devices and LED components to allow them to function correctly, while also lasting for as long as possible and taking up minimal space. In order to access these tradeoffs, a table was formulated to compare different batteries that could potentially power our device. Our desire is for the device to have a rechargeable battery that can be easily replaced. Also, a stretch goal for this project would be to allow for a direct connection to recharge the battery from outside the casing. The first thing we explored in the research for this device was the battery material.

Lithium Ion vs. Lithium Polymer vs. Nickel-Metal Hydride

For the material of the battery, we looked at the three most common materials for rechargeable batteries. In terms of popularity, Lithium is a much more popular and widely-used material, as it can hold more charge than Nickel-Metal Hydride batteries. As far as Lithium-Ion versus Lithium-Polymer, the final decision was made using Table 14 & 15 to compare the advantages and disadvantages of each and how it will affect our project.

Type	Advantages	Disadvantages
Lithium Ion	-Higher energy density than Polymer, can power device for much longer without need for recharge. -Slightly less expensive than Polymer -Hold a charge better when not in use -Can handle fluctuations	-Slightly larger, bulkier designs not suitable for device. -Lithium Ion batteries more susceptible for aging-not good for reliability of device

Table 14: Rechargeable battery material comparison

Type	Advantages	Disadvantages
Lithium Ion Polymer	-Sleeker design allows for easier integration into the device, lighter design. -Better lifespan than Lithium Ion batteries.	-Lower energy density than Lithium ion batteries -More expensive to manufacture -Prone to loss of charge when not in use -More susceptible to damage due to fluctuations

Table 15: Rechargeable battery material comparison

Since our project requires both size constraints as well as durability and ease of use constraints, priorities have to be taken in order to select the correct battery. Since long battery life is more important to this project than size or weight, the Lithium Ion battery was selected as the ideal battery type. The longer lasting battery and ability to hold and provide a steady source of power is very beneficial to our power sensitive components such as the ESP8266. The downsides such as weight and size were to be accommodated in our housing design.

The final battery choice reflected in Table 16 was the Lithium Ion battery Pack provided by AdaFruit industries. This battery back was small, relatively light, and could power our device for a significant amount of time before requiring a charge.

Manufacturer	Site	Voltage	mAh	Weight
Adafruit	Adafruit	3.7	4400	95g

Table 16a: Final Choice Battery

Since we are using an ATmega328P chip, it requires a 5V, the original battery picked was not used. Instead, we decided to use a 9V rechargeable battery from Battery Junction. Shown in Table 16b.

Manufacturer	Site	Voltage	mAh	Weight
Battery Junction	Battery Junction	9V	1200	95g

Table 16b: Final Choice Battery (Actual)

4.3.6. 3D Printing & Software

Since our casing design is very complex and has to contain screw taps, fixtures for custom PCB boards, and aligns with ordered parts, the design for the case should have been made using a 3D modeling software which will then be printed using a 3-D Printer.

4.3.6.1 Software

Fusion360
This program is a cloud-based 3D CAD Program which allows teams to work on projects together simultaneously. This would be extremely beneficial to our group as it would allow us to work on it from anywhere. It is also a big plus that Fusion 360 contains a history of all changes, which can aid with the design process and prototyping. It has been described as having an easy learning curve and geared toward mechanical design, which aligns with our objective. It also comes with a 3-year free educational license for students, which really helps with costs.

Design Spark
Design Spark is a Free 3D Modeling software that is geared more towards hobbyists and includes a large built-in library. It is also straightforward and easy to navigate, which can help us prepare a design quickly. However, being free and beginner friendly makes it somewhat of a simpler software and could potentially be tricky to utilize once we design the more complex pieces or PCB enclosures.

Inventor
Inventor is a 3D CAD software developed by Autodesk, and is geared toward professional level mechanical design. This is an advantage for us because we will need multiple structures that will require carefully measured threads. Inventor is also free for Students, making it a strong candidate.

SolidWorks
Solidworks is a valuable and popular choice for modeling 3D parts. The visuals in Solidworks outrank similar CAD programs, which can be beneficial in visualizing our design and making design choices. Also, this program comes in UCF engineering computers which saves our group time.

Decision
Since none of our group members have much experience with 3D Modeling software, we decided to go for a software that was powerful enough to achieve our design, yet intuitive and simple enough to allow us to learn how to use it quickly. With all of this in mind, we selected to use Fusion360 for this project.

4.3.6.2 3D Printer & Material

UCF has their own in-house 3D Printers to print materials. There are also other resources such as maker workspaces or libraries across Orlando.Since there will be many prototypes as well as trial and error, it would be wise to use a cheap yet sturdy plastic. Our research narrowed the options down to 2 widely used plastics:

PLA
PLA is an easy material to print, and as common as ABS. It is also biodegradable, which would help with the marketability of our device However, it is slightly more difficult to manipulate which could mean it is less resistant to shocks. It also shrinks slightly after printing.

ABS
ABS is the most common of the plastics, with a slight flexibility and resistance to shocks, which would be very beneficial to our design as it runs the risk of falling from desk heights. It is also the cheapest plastic material, but shrinks in contact with air. This requires special treatment when 3D printing.

Final Decision
For this device we are moving forward with ABS plastic in order to keep the device as durable and low cost as possible.
Ultimately, we were unable to complete the Case Design using 3D techniques due to UCF closures related to COVID-19. We did, however stick to PVC and the same adhesives, in an attempt to stay as close to the original design as possible using traditional techniques.

4.3.7. Shape

For the shape of the enclosure, we decided on a few simple shapes, since our time was limited and we couldn't spend it on making a complex shape case. Table 17 outlines our possible case shape options as well as pros and cons for each:

Shape	Pros	Cons
Cylinder	-Structurally strongest shape -Easily viewable LED strip -Takes up less space	-If tipped over, can roll off of desk and become damaged
Cube	-Hardest to accidentally tip over -Structurally Sound -Larger space for components	-Shortest, hardest to see from a far distance -Takes up more space proportional to its height
Rectangular Prism	-Height will allow for easy viewing from far distances	-Easiest to tip over -Rectangular design not as attractive -Less space to house components

Table 17: Comparison chart for Enclosure Shape

4.3.8. Tamper Proofing

Since this product will be unsupervised for most of its life, it requires safety precautions against any actions that are not meant to be for normal users. These non-user actions include using the on/off button and using the keypad without permission

Since the on off button will be a push button with two states, it allows for the button to be concealed within the device, with access limited to those who remove the outer enclosure and expose the inner workings of the device. However, the outer enclosure will be attached with security screws and so will not be easily accessible except for those with the right key.

This will prevent everyday users from accidentally turning the device off and limit the number of visible buttons to allow for easier operation. The Keypad receives input, but it can be ignored as part of the software that is flashed into the microcontroller, preventing unwanted interference from keypad presses.

Ultimately we were not able to implement these Tamper Proofing ideas. We could, however, implement the On/Off switch withing the enclosure as well as a seal tight cap that is located under the device to limit access to the electronics.

4.3.9. Microcontroller

For all the input processing and the logic behind the reservation system on the hardware side, we researched ARM Microcontrollers. These microcontrollers also controlled switches that provides power to LEDs. Our research narrowed our choices down to the following boards:

NUCLEO-L011K4
This is a development board for the ARM based STM32 Microcontroller, which is a popular alternative to Arduino microcontrollers and considered a very good starting point for beginners of ARM programming. Since it is such a popular alternative to arduino, there is a lot of community support and resources available for programming the STM32 Microcontroller.

STM32F103C8T6 ARM STM32 Development Board
This board uses the same popular STM32 Microcontroller. The only differences between this board and the NUCLEO-L011K4 is that this board has less complexity, further reducing the cost but at the same time increasing the difficulty of programming and of preventing electrical damage to the microcontroller.

MSP-EXP430G2ET
The MSP-EXP430G2ET Microcontroller one of the easier development boards to work with, using the M430G2553 Microcontroller. This board is not only heavily documented by Texas Instruments and an active community of Makers, but it comes with a very comprehensive debugging software (Code Composer Studio) which can make the initial testing process extremely easy.

Raspberry Pi Zero W
The Raspberry Pi Zero W is an extremely powerful yet small development board with built in Wi-Fi capabilities and a large community of support and example projects. Since this device was designed with Wi-Fi and portability in mind, It meets the needs of our project.

Arduino UNO Wifi
The Arduino UNO Wi-Fi board is one of the most popular and common boards for beginners. That being said, it pays for simplicity and ease of use with low processing power and capabilities. There was a model introduced which contains an integrated Wi-Fi module, and with an extremely large community and open source software behind it, there is a lot of knowledge available for working with this microcontroller.

Microcontroller Comparisons
Although there are many boards available to take advantage of today's smallest and most potent microcontrollers, we were able to narrow our selections down to a few. Table 18 highlights the advantages and disadvantages of each controller to weigh their strongpoints and weak points. Table 19 is a technical comparison of each microcontroller based on a variety of essential fields.

Microcontrollers	Advantages	Disadvantages
Arduino UNO Wifi	-Wifi Capabilities integrated - Comes with easy to use Energia software -14 I/O Pins, 5 PWM	-Higher Cost 45$ -Lower storage space (RAM and Flash) -Lowest clock Speed
Raspberry Pi Zero W	-Wifi Capabilities integrated -Expandable Flash Memory - Small Dimensions	-Prone to crashes due to faults/failures -Hardest to program
MSP430G2ET	- Cost efficient - Code Composer Studio for advanced debugging	- Medium difficulty to program -Not as compatible with products that are not associated with Texas Instruments
STM32F103C8T6	-High performance STM32 microcontroller -Easy access Debug Ports	- Minimal Components - Little support, not as much documentation
NUCLEO-L011K4	-High performance STM32 microcontroller -Micro-USB Connection	- Little support, not as much documentation - Long lead time

Table 18: Microcontroller Advantage Comparisons

Technical comparisons	Arduino UNO Wi-Fi	Raspberry Pi Zero W	MSP-EXP430G2ET	STM32F103C8T6	NUCLEO-L011K4
Operating Voltage	5V	5V	1.8V to 3.6V	2.0 - 3.6V	1.65 - 3.6V
DC Current	20mA	16mA	15mA	25mA	16 mA
Digital I/O pins	14	40	47	38	38
CPU clock speed	16 MHz	1 GHz	25 MHz	72 MHz	32 MHz
Flash Mem	48 kB	Expandable	128 kB	128 kB	16 kB
Wi-Fi module	Yes	Yes	No	No	No
RAM	6.14 kB	512 MB	8 kB	20 kB	2 kB
Weight	25g	9g	80g	8.5g	7.6g
Cost	$44.90	$10.00	$10.37	$1.69	$10.32

Table 19: Microcontroller Technical Comparisons

Since our project is not particularly hardware heavy, our microcontroller won't require a large amount of RAM or Flash memory. However, whatever processes we do require should be done as quick as possible as our queries concerning the microcontroller are time sensitive. Another result of our project not being hardware heavy is the fact that there will not be as many peripheral devices attached to the microcontroller, so GPIO numbers are not as important. However, they need to be enough and specialized enough to support multiple PWM signals and buttons.

These factors were accumulated into a decision matrix in Table 20 in order to help choose the ideal microcontroller for this project. Taking into account all of these ideas as well as the information from Table 18 and 19, The highest-ranking microcontroller turned out to be the MSPEXP430G2ET. Its ease of use as well as its cheap cost and wide community of support allows it to be an easy starting point for this project. The small size and large quantity of pins also gives it greater flexibility and usefulness over other microcontrollers.

Decision matrix	Arduino	Raspberry Pi	MSPEXP430-G2ET	STM32-F103C8T6	NUCLEO-L011K4
Cost	1	3	3	4	3
Ease of Use	4	3	4	2	2
Power Use	1	2	3	4	4
Wifi Enabled	2	2	1	0	0
Size	1	2	3	4	4
Pins	1	3	4	2	2
Total	8	15	19	16	15

Table 20a: Microcontroller Decision Matrix

However, the team has decided to move on with the ATmega328P from the Arduino UNO board manufactured by Adafruit due to simplicity and compatibility with the Wi-Fi module chosen. Moreover, it has just the basic requirements the devices needed, so no waste of memory, pins, cost and DC current. The Atmega328P was obtained without bootloader software in order save money.

Manufacturer	Model	Cost	Voltage	Pins
Microchip Technology	ATmega328P	$2.08	1.8V to 5.5V	28

Table 20b: Microcontroller used in actual device

Keypad

This device requires an authentication process, and since the reservation button is used specifically to claim a reservation, there potentially needed to be a second form of input to verify that the use of the button is not simply an error or an unwanted user. This can best be done by using a simple keypad. Today's market has allowed for multiplexed keypads to be available for an extremely low price, and with the capabilities of microcontrollers to easily process these inputs, it is a feasible task to establish these keypads.

One option was to use a smaller version that consists of only 4 numbers. Although slightly less secure (4^n possible combinations), it is much smaller and also less expensive. The other option was searching for alternatives for this authentication process. The pros and cons of using a keypad have been laid out in Table 21:

Type	Pros	Cons
With Keypad	-Easier to implement authentication -Less reliance on software	-Bulky keypad potentially unattractive -Need workarounds for when keypad is pressed when not needed
Without Keypad	-More Sleek/ Attractive appearance -Less reliance on hardware	-Need to devise a more complex authentication system based entirely on software

Table 21: Comparison chart for Keypad

The final product was decided to be a 4 button keypad specified in Table 22.

Manufacturer	Model	Cost	Weight	Cable Length
Adafruit	1332	$2.95	2.83g	3.53”

Table 22: Final Choice for Keypad

4.3.10. Wi-Fi module

Since the device needed to communicate with the database, it required a Wi-Fi module aside from the Microcontroller to add network connectivity. The Wi-Fi module needed to have a low power mode to consume as little energy as possible when not in use, as transmitting data via Wi-Fi tends to be very power consuming. Our research had narrowed our results down to 2 different modules.

CC3100 TI

The CC3100 by Texas Instruments is a Wi-Fi module that it tailored to provide an easy IoT experience, supported by many resources from TI themselves, as well as a development board model designed to fit over the MSP-EXP430G2ET. This would have allowed for easy debugging and less risk of damaging physical components from improper wiring. One downside, however, was that the development board is only suited for an MSP Launchpad platform, and any sort of separate with this board would be extremely difficult as the circuitry for this device was very complex.

ESP8266

The ESP8266 is a popular Wi-Fi module that is considered a SOC (System on Chip) device. The microcontroller embedded in this Wi-Fi module is strong enough to handle some basic processing and can even support input from other GPIO pins. Although it is a popular board, there is no proper central support base like the CC3100, making it slightly more difficult to program and interact with the board. The technical specifications of each board are compared in Table 23 and a decision matrix was made as Table 24.

	CC3100	ESP8266
Cost	$23.38	$4
Operating Frequency	2.4 GHz	2.4 GHz
Input Voltage	3.3V - 3.6V	2.5V - 3.6V
Average Current Consumption	100 mA	80 mA

Table 23: Comparison chart for Wi-Fi Modules

	CC3100	ESP8266
Cost	1	3
Average Current Consumption	1	2
Ease of Use	3	1
Total	5	6

Table 24: Decision Matrix for Wi-Fi Modules

The final decision was to use the ESP8266. Overall, it is easier to implement, cheaper, and best suited to our needs. Since we did not require complex tasks and prioritize power efficiency over processing power, the ESP8266 will be a better fit than the CC3100. Table 25 shows the specifics of our final choice along with the price.

Manufacturer	Model	Price	Site	Weight
Espressif	ESP8266	$6.95	Sparkfun	1.9g

Table 25: Final Choice Wi-Fi Module

TI Code Composer Studio

Code Composer Studio is the main and most popular IDE for the Texas Instruments MCUs. This code composer is a powerful tool to write code on their platforms due to its effectiveness. It implements C code but marks it up. It is very efficient when it comes to determining errors caused by native support of a device.

It is very popular due to its ability to debug and test code used in other C IDEs. Furthermore, it is efficient to program every part and peripheral attached to the microcontroller which for this project will be required. Table 26 highlights the advantages and disadvantages of Code Composer Studio.

Advantages	Disadvantages
Texas Instruments creates CCS	Errors due to low time production
Natively support all MSP 430 series	Not a high quality of error reporting which slow down development.
Familiar with C code used in TI’s examples	Susceptible to crashes

Table 26: Advantage and Disadvantage for Code Composer Studio IDE

Arduino IDE

For this project, we decided to use Arduino IDE instead of the Code Composer IDE for the microcontroller. This IDE is by far the most popular for this product for many reasons. There are several amounts of documentation and example code for this IDE which makes it simpler and more effective to work with. Moreover, it is powerful, user friendly and easy to understand. In this project, rapid prototyping was a factor since there was no need to worry about re-flashing hardware. Table 27 outlines the advantages and disadvantages of using Arduino IDE.

Advantages	Disadvantages
Powerful IDE which supports Wifi Module	Lacks code prediction
Simple and aesthetic	Extremely simple which can be defined as a text editor with a compiler
Several resources	Lacks of error highlighting
Effective IDE	Natively supports Arduino microcontrollers

Table 27: Arduino IDE Advantage table

4.3.11. Serial Communication Technologies

SPI - Serial Peripheral Interface

Developed by Motorola, the SPI is a synchronous, full duplex master-slave-based interface. The synchronous part of this is that it uses separate data lines and an oscillating clock that keeps the data in sync. Since the clock is oscillating, meaning it has a rising or falling edge, it will start looking at the data line after it looks at the edge in order to read the next bit. The SPI bus has four signals - 1) master in slave out (MISO), 2) master out slave in (MOSI), 3) serial clock (SCK), and 4) active-low chip select (/CS), as described and pictured below:

Master in slave out- a unidirectional signal that sends data from the slave to the master device.

Master out slave in- also similar to the master in slave out, in terms of unidirectional signal, but the data is from the master to the slave.

Serial clock - the clock that synchronizes the signals, created by the master device.

Active low chip select - this tells the slave that it should wake up/power on in order to send/receive data, if there are multiple slaves that is.

When it comes to reading data at the edge of the clock, there are a couple of ways that the SPI accomplishes this, as the master can select the clock phase and polarity, which is represented as bits - the CPHA and CPOL bit, respectively. The CPHA bit is where the falling or rising edge of the clock is used to shift or sample the data, whereas the CPOL bit represents the transition of the chip selection for high to low end of the transmission, and vice versa. This leads to 4 possible states for sampling data:

SPI Mode 0 (Logic low clock polarity)
CPOL = 0, CPHA = 0: Data is sampled on the rising edge, shifted on the falling edge.

SPI Mode 1(Logic low clock polarity)
CPOL = 0, CPHA = 1: Data is sampled on the falling edge, shifted on the rising edge.

SPI Mode 2 (Logic high clock polarity)
CPOL = 1, CPHA = 0: Data is sampled on falling edge, shifted on the rising edge.

SPI Mode 3 (Logic high clock polarity)
CPOL = 1 CPHA = 1: Data is sampled on the rising edge, shifted on falling edge.

A popular reason that the SPI protocol is popular is that it is a simple protocol, such as a shift register.

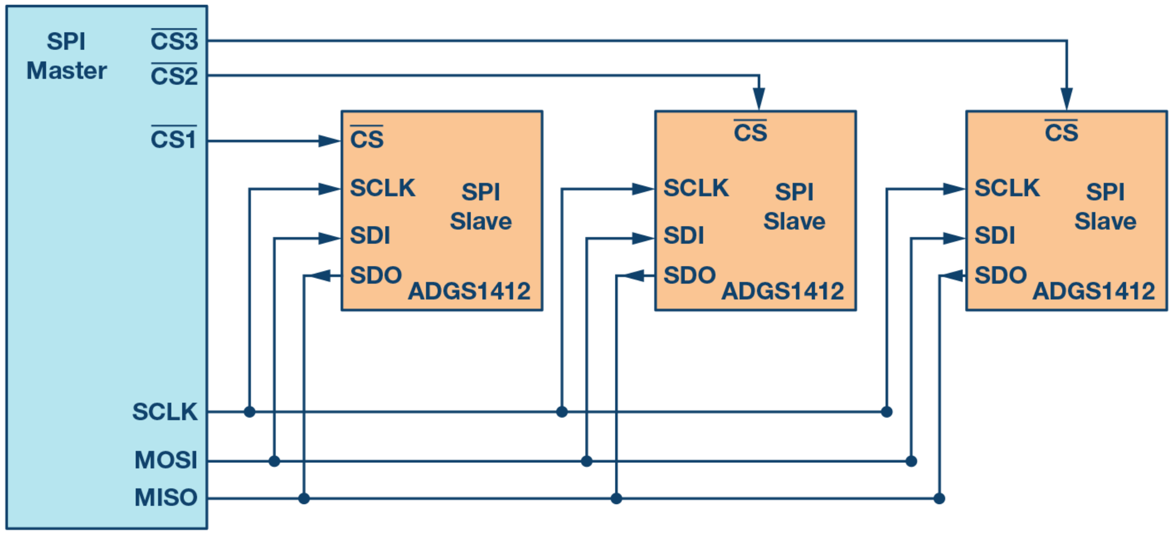


Figure 9: Multi-slave SPI configuration

As shown in the Figure 9 above, we can have multiple slaves in the SPI protocol, which required multiple slave select lines. This is one of the ways we connected multiple slaves to an SPI bus, the other way is to connect them using daisy chain method, in which a single slave select line is used, where if the data was sent, it results in all the slaves being activated simultaneously. Unfortunately, this required enough data to reach all of the slaves, as it overflows from slave to slave. This is situation specific, where it is mostly used for output only situations, such as using LEDs.

I2C - Inter-Integrated Circuit

I2C Communication was developed by NXP Semiconductors (formerly Philips Semiconductor) in 1982. Its main usage is for low speed communication between different integrated circuits. Like SPI, you can connect multiple slaves to a single master, which can be useful in certain cases such as having multiple microcontrollers logging data to a single memory card. I2C only uses two wires to transmit data between master and slave: Serial Data (SDA) and Serial Clock (SCL). Serial Data is a data line between the master and slave, while Serial Clock is a line that transmits the clock signal. I2C also happens to be synchronized as well, where the bit output is controlled by a clock signal, which in turn is controlled by the master, as shown in Figure 10 below:

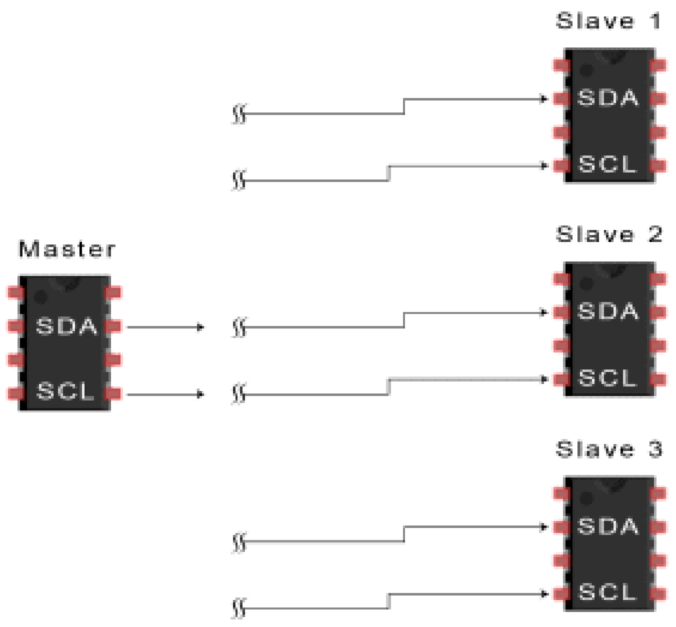


Figure 10: Illustrative diagram of I2C Connection

In terms of data transmission, data is transferred in messages, which are broken into frames of data - each frame contains the address of the slave, and the message that is being transmitted (Data Frame 1 and 2), as shown below in Figure 11.

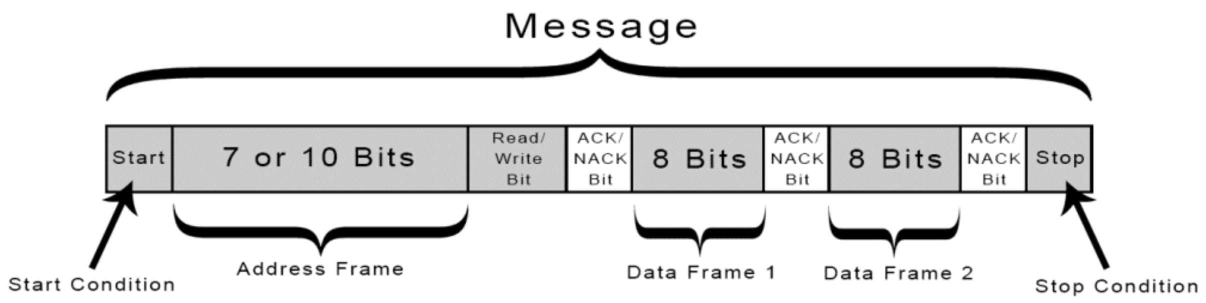


Figure 11: Message breakdown found within I2C

Breaking down each part found within the message, we can identify a few important segments:

Start Condition: the Serial Data line shifts from a high voltage to a low voltage before the Serial Clock line shifts from high to low, which signals the start of the message transmission.

Address Frame: It is a 7 or 10 bit binary address segment that identifies the slave when the master wants to communicate to it.

Read/Write Bit: A single bit that determines if the master is sending data to the slave or needs to read from it.

ACK/NACK Bit: An acknowledge/no-acknowledge bit that shows if the data frame was successfully received or not.

Stop Condition: the Serial Data line shifts for low voltage to high voltage after the Serial Clock line shifts from low to high, which ends the message transmission.

UART - Universal Asynchronous Receiver/Transmitter

Created by Gordon Bell, the UART is a computer hardware for asynchronous serial communication where the transmission and data speeds are adjustable. Its main purpose is to transmit and receive serial data - using only two wires in the process as well, similar to I2C. When it comes to communication, two UARTs directly communicate with each other, as shown in Figure 12.

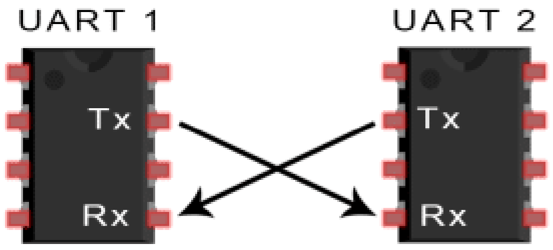


Figure 12: Sample UART diagram

As seen in Figure 12, there are only two pins - Tx and Rx, which stand for transmission and receiving. Unlike SPI and I2C communication protocols, the lack of a clock pin also means that it transmits data asynchronously, and instead uses start and stop bits to the data packet, so the other UART device knows when to start reading the start of the message. This can be specified by the baud rate, which determines the speed of the data transfer, expressed in bits per second (bps). Both UART devices must have the same baud rate, and is only limited to 2 UART devices, as they cannot have additional slave devices.

In terms of communication, UART communication is sent in the form of packets. Each packet as shown below in Figure 13, has the following:

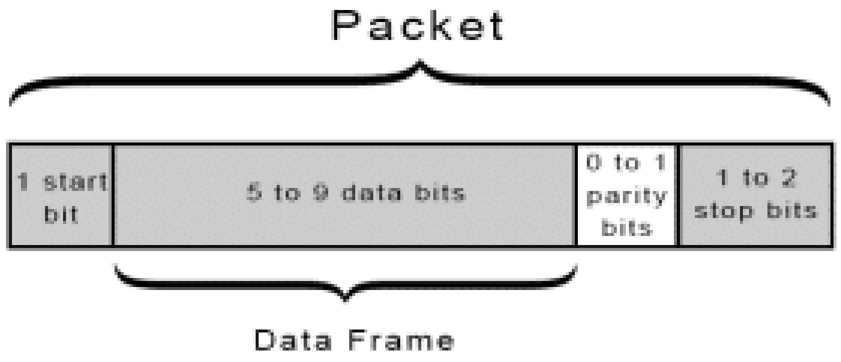


Figure 13: UART data transmission packet

Start Bit: Initially, the UART data transmission is at a high voltage level when no data is transmitted. In order to start transmitting data, the UART has the transmission line from high to low voltage, and starts reading the bits in the following data frame.

Data Frame: The data frame is where the actual data is being transmitted. It can be between 5-8 bits if a parity bit used, and if not, then the entire 9 bits can be used. In the majority of cases, the data is sent with the least significant bit first.

Parity Bit: It can be used to determine if the transmission has errors or not. If the parity bit is a 1, which is an odd parity then it means that the bits in the data frame should be equivalent to an odd number, suggesting it is free of errors. However if the parity bit is 0, and there are an odd number of bits in the data frame, it suggests that the bits in the data frame have changed in the UART.

Stop Bit: In order to signal the end of the data packet, the UART drives the transmission line from a low to high voltage for up to two bits.

Now that we know what is being sent out in the UART transmission packet, here's the following steps of how the transmission works:

1. The UART transmits data in parallel from the data bus.
2. The transmitting UART adds the start bit, parity bit, and stop bit to the data frame.
3. The entire packet is sent from the transmitting UART to the receiving UART, serially. The receiving end of the UART samples the data line at the pre-configured baud rate.
4. The receiving UART converts the serial data into parallel data and transfers it to the data bus on the receiving end.

Conclusion for Serial Communication
Since we are using not that much supporting peripheral devices in terms of slave devices, we are using UART communication as our main protocol.

4.3.12. PCB Design Consideration

When it comes to designing a PCB, there were many considerations to include, such as size and shape, electrical noise, thermal management, and placement of components - determined the final outcome of the board production and quality:

Size and Shape

The board shape and size was largely determined by the construction of the device. However if there was extremely limited space, we may have to use some flexible boards, which would have driven up the cost of the PCB. There were also other needs for needing multi-layer boards, which also could have driven up costs as well.

Electrical Noise

Electrical noise usually refers to shifts in current/voltage that is random has undesired effects which can lead to unwanted output in the circuit design. We wanted to reduce these unwanted effects by covering the board area with power and ground planes - a general good practice is to keep a face for the power plane and one for the ground plane. When it comes to traces, we had to keep them as small and thin as possible. We also had to keep the digital and analog circuitry apart from each other on the PCB.
The use of capacitors, specifically high-quality tantalum-polarized capacitors, was also valuable to filter noise from power and regulator supplies. And of course, we never used 90-degree turns on traces - the general practice was 45 degree trace around corners.

Thermal Management

When current runs over a trace or a component, heat is usually dissipated. The resulting heat is dependent on some factors, such as circuit design, power, and device characteristics. In order to reduce thermal resistance, we started with using the formula:

$$\theta = t/(A \bullet K)$$

Where,
t = is material thickness
K = thermal conductivity factor
A = cross sectional area

There were a couple of ways we could have reduced thermal resistance by adding thermal vias for vertical heat conduction, using thinner PCBs to reduce thermal paths, and using thick tracks/copper foil for horizontal heat conduction.
In terms of component placement, we can consider placing components that involve plenty of power usage in areas where heat can be best be removed. Some general principles is to place power heavy components around the middle of the PCB area, in order to make sure there is enough space for air circulation. Small sensitive components, such as transistors and integrated circuits, can best be placed in low-temperature areas. There are some physical thermal management solutions for PCBs: the use of thermal vias and heat sinks:

Thermal Vias

In terms of structure, a penetrating hole is opened into the board and if the board is double sided, and single layer board there is a copper foil that connects the bottom and top surfaces, which results in a lower thermal resistance

The recommended thermal via size is about 1.2 mm, and should be placed below the heat dissipating plate on the bottom surface of the component. It is also important to note that the vias be as close to the IC as possible.

Heat Sink

We had to know the difference of what material that we were using in the heat sink: copper or aluminum. Yes aluminum is a very popular choice for heat sinks, but there are some material and cost considerations for both. For copper material, they actually have more conductivity than aluminum (231 BTU vs 136 BTU for aluminum). Unfortunately in terms of weight, copper is more dense than aluminum (8940 kg/m^3 vs 2712 kg/m^3 for aluminum), which plays part of a reason why aluminum is a very popular heatsink on the market. The same goes for cost as well: aluminum is significantly cheaper than copper ($2463.00 per metric ton vs $6939 per metric ton for aluminum).

This information is summarized in Table 28:

	Aluminum	Copper
Weight	2712 kg/m^3	8940 kg/m^3
Heat Conductivity	136 BTU	231 BTU
Cost	2463.00 per metric ton	6939.00 per metric ton

Table 28: Summary of Materials Comparison

Given the comparison between the two materials, we can determine that we are using aluminum as the heatsink material.

Placement and Routing

The size of the PCB board is limited depending on our application, which means we had to consider how we place our components. There are some general practices that should be followed: keep surface mounted components should be on the same size of the board, and any through hole components should be placed on the top of the board, in order to reduce the number of assembly steps. In terms of orientation, similar components in the same direction to reduce soldering errors:

We also have to consider the use of through hole components and surface mount components as well - ideally we would wanted to use the majority of surface mount components for a professional presentation, and to keep the overall cost low. However, surface mount components are harder to assemble since, the team did not use an assembly company, we decided to use through hole components and only the voltage regulators as surface mounted.

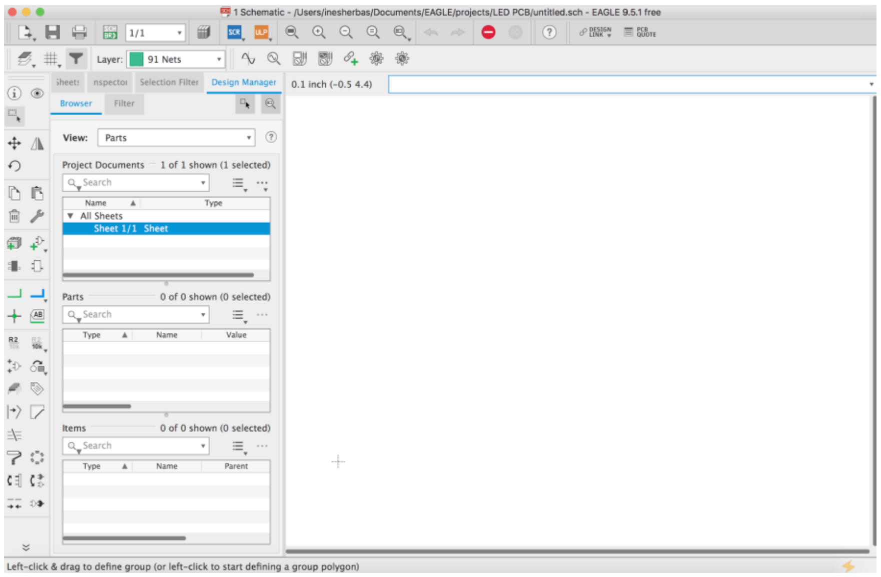
We also had to consider routing best practices as well - we had to keep PCB traces short as short and direct as possible between components - if we have the component placement forces a horizontal route trace on one side, then we had to route a vertical trace on the opposite side. For net widths, we needed to provide a 0.010’’ for low current digital and analog signals. For traces that carry more than 0.3 amps should be thicker. For calculating trace widths, there were also multiple various trace width calculator online that make it simple for the user to determine.

4.3.12.1 PCB Software Tools

In this section, we are going to go over some tools used for PCB creation and design. Some of the most popular software tools that we will cover are Eagle, EasyEDA, and Altium.

Eagle

It is an electronic design automation tool with scripting abilities complete with printed circuit board layout and schematic capture, and computer aided manufacturing abilities (e.g. visualize your PCB in the final build stages). It has some import support from other PCB tools, such as Altium. Eagle will become our preferred platform of choice, as we are familiar with some aspects of the software from Junior Design, and the fact that we have a free student license for it gives us more access to advanced features. Figure 14 shows the interface for the Eagle Software.



1 Schematic - /Users/inesherbas/Documents/EAGLE/projects/LED PCB/untitled.sch - EAGLE 9.5.1 free

Layer: 91 Nets

0.1 inch (-0.5 4.4)

View: Parts

Project Documents 1 of 1 shown (1 selected)

Name	Type
All Sheets	
Sheet 1/1	Sheet

Parts 0 of 0 shown (0 selected)

Type	Name	Value

Items 0 of 0 shown (0 selected)

Type	Name	Parent

Left-click & drag to define group (or left-click to start defining a group polygon)

Figure 14: Eagle PCB tool

EasyEDA

Easy EDA is also an electronic design automation tool with the ability to create public and private schematics, simulations, and printed circuit boards - instead of having the user install it on their computer, it is all online, meaning its free to use and accessible from anywhere! It is extremely user friendly and good starter platform to use if you don’t have enough knowledge about PCB design. In terms of library support, it has more than half a million libraries with footprints and symbols. It has also multi program import support such as EAGLE, LTspice, and Altium. Figure 15 shows the interface.

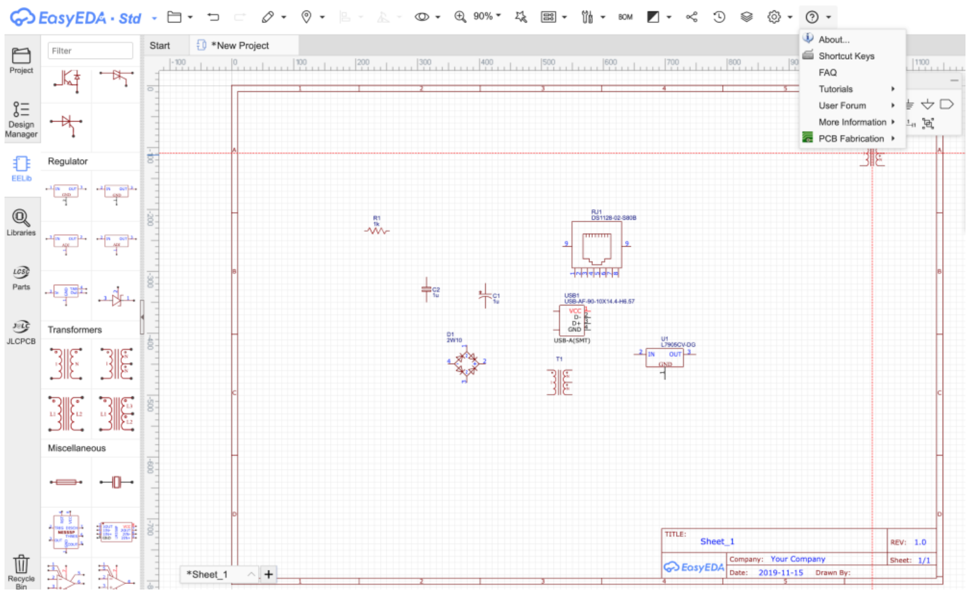


Figure 15: EasyEda User Interface

Altium

Many used by industry professionals, Altium does have a higher learning curve but is extremely user friendly, using the Windows standard behavior, has a flexible release management tool, a superior 3D visualization tool, and a fairly decent amount of import/export support, such as OrCAD, Eagle, DXDesigner It does have a very high license cost (~$7000) but for students there is a free evaluation period, and afterward it is only $100/year. Figure 16 shows the Altium User Interface.

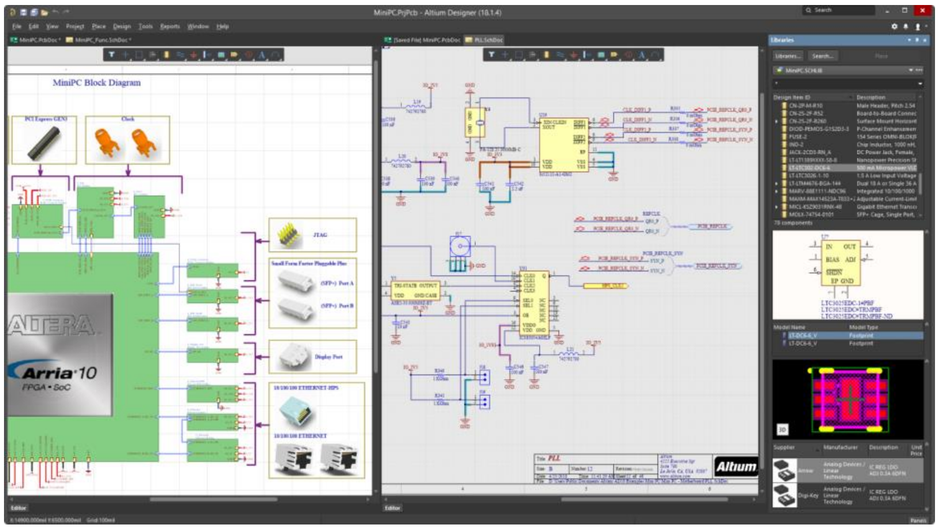


Figure 16: Altium User Interface

4.3.12.2 PCB Fabrication

Once our PCB design prototype has been designed, we need to look for a company to fabricate it. Some of the main points we used to make our decision was lead time and cost. As this project is time sensitive, we will need to fabricate our PCB with a company that can deliver a PCB as fast as possible, especially considering the fact that our first PCB design will certainly be suboptimal due to our lack of experience. We will also factor cost into this project since we want this to ideally be purchased as multiple devices to set up a unified system. This section will cover the pros and cons between three known PCB manufacturing and assembly companies known throughout Senior Design (one in China, two in Orlando) - JLCPCB, Quality Manufacturing Services, and FermiTron.

JLCPCB

According to JLCPCB's site, their company is China's largest PCB prototype enterprise which specializes in small batch PCB production and quick PCB prototype turnaround. In terms of fabrication services and production time, for a 1-2 layer board (which will be what we will most likely be using) it has a 1-2 day production time. Since it is shipping from China, we will anticipate a 15-20 day arrival time, which is not good for any last minute PCB emergencies. For pricing, it is only $2.00 for 5 1 to 2 layer boards, up to 100x100 mm in size, making it extremely cheap- but the shipping cost will add a bit more, depending on the amount of boards that we order plus any additional customizations. They also offer surface mount (SMT) assembly at their facility for a $7.00 core engineering fee, using their own components however.

QMS (Quality Manufacturing Services)

Based in Lake Mary, FL Quality Manufacturing Services is a full service Electronics Manufacturing provider, as an employee-owned company. Some services that they offer are PCB Assembly (throughhole/surface mount), rapid prototyping through their Fast Track program, and testing capabilities such as flying probe, in-circuit, functional, and device programming. Unfortunately, we do not have a price estimate, as we have to request a quote through a representative/website.

FemiTron
Born from the UCF Business Incubation Program — FermiTron's services include microcontroller design, FPGA design, RF design, PCB Layout, prototype testing, and mechanical design, just to name a few. Just like QMS, there is a quote that you have to get as well in terms of pricing, as their process is extremely involved - an inquiry phase, a discovery phase, a proposal phase, and a project kickoff phase.

Based on the overview of the PCB manufacturing companies, we will utilize JLCPCB for our initial PCB design, and if necessary we will use QMS as an emergency backup if we need printed PCBs on the ready.

4.4. Software Design Research

In this section, we will be going in detail and understanding different technologies available for us to use in our project. We will be discussing front-end and back-end possible technologies and comparing their abilities, functions, ease of use, and costs to choose the best options for our device.

4.4.1. Front-End

For front-end development there are plenty of programming languages to choose from. We looked at three different ones that work for mobile applications (iOS and Android) - JavaScript, Swift, and Java.

When it comes to mobile app development, there are factors to consider between native options and cross-platform solutions. If Swift and Java were being used, then we can only develop for a specific Operating System - iOS or Android respectively. On the other hand, if we used a cross-platform solution, then we can use JavaScript as our programming language and use React Native as our framework. This option allowed us to develop for both iOs and Android at the same time. In the next sections, we will be discussing these factors in details to understand which option was the best for our project:

4.4.1.1. Cross-Platform vs Native Apps

Performance
Native apps have a higher performance than cross-platform apps. That is because native apps have been specifically made for a specific platform, and this means they are following the quality guidelines and HMI guidelines.

User Experience
Native apps follow certain protocols depending on the operating system. They follow Apple's HMI guidelines and Google's. On the other hand, cross-platforms cannot do that

and they need to have a unified UX for the application, otherwise, it would be uncomfortable for the user if this one follows the guidelines from one or another.

Functionality

Native apps have access to all native APIs depending on their operating system. These are compatible with either Google or Apple products as well. These apps can even be used without an internet connection. On the other hand, there is limited functionality for cross-platform apps. This is because cross-platform apps do not have access to native APIs. Sometimes they might have limited access to OS features and might always need an internet connection to be used.

Support

The benefit of building a native app is that mobile app could be featured in either one of the stores. Apple and Google will always encourage developers to use their platforms. For cross-platform apps, there are always communities on the internet if there is ever a problem and since there is only one codebase, it is easier to make changes to the app.

Updates and Maintenance

For native apps, there has to be great management between different operating systems. If there is a change being made to one of the apps, then the change must be reflected on the other one as well. On the other hand, cross-platform apps are easier to update since there is only one codebase.

Development

In regard to native apps, it takes more time to develop these applications. You cannot reuse code from one platform to the other, and essentially you would need to start from scratch on the other operating system. On the contrary, cross-platform is a lot quicker to develop since code can re-used and there is no need to start from scratch.

Decision Matrix

Table 29 is a decision matrix between Native Development and Cross-Platform Development.

1 - Lowest; 2 - Good; 3 - Best

Decision matrix	Native App	Cross-Platform
Performance	3	1
User Experience	2	1
Functionality	2	1
Support	2	3
Cost	1	3
Updates/Maintenance	1	3
Development	1	3
Team Knowledge	1	2
Total	13	17

Table 29: Decision Matrix Application

4.4.1.2. Front-End Programming Languages

Now that we have covered native apps vs cross-platform apps, it is important to go in detail with which programming language we would choose. In the next sections, we will be going over the three languages and frameworks we mentioned before: Java, Swift and JavaScript with React Native.

Java

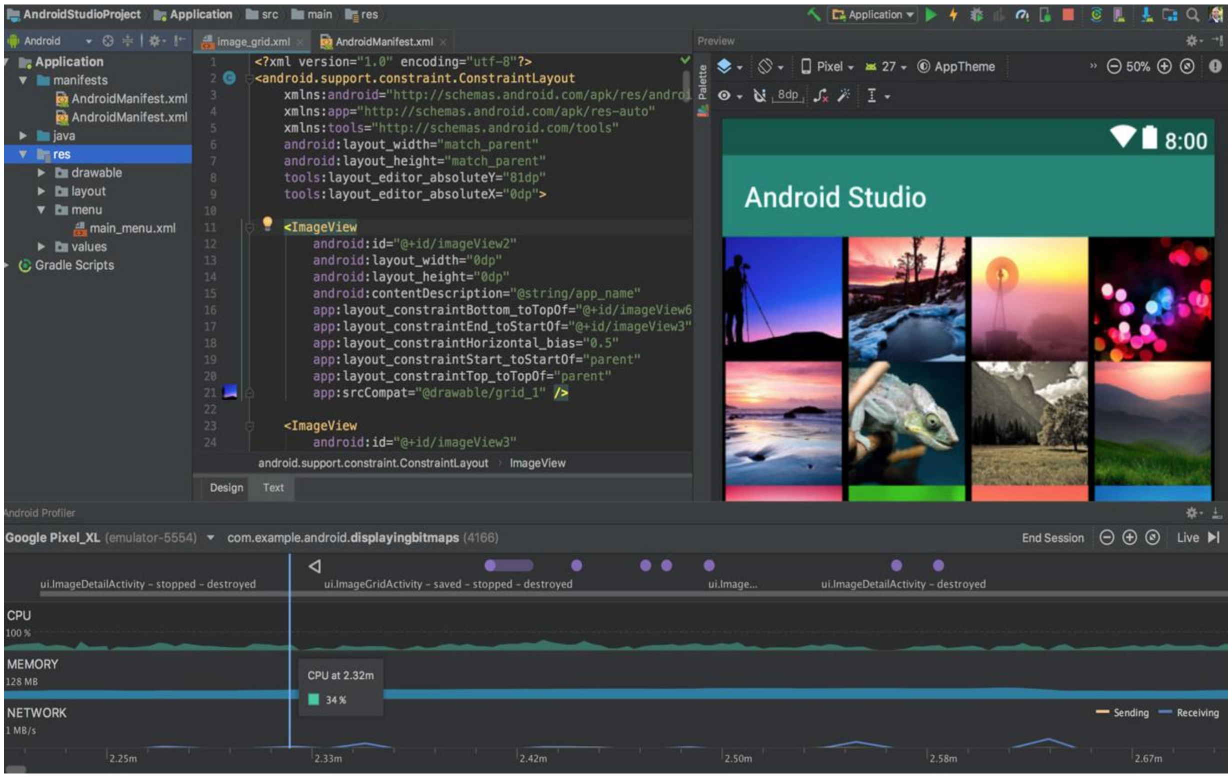
Java is a general purpose programming language. It is used for several applications, and in this case, for developing an Android application. Android is an open source software platform and Linux-based operating system for mobile devices. These Android applications can be developed using the object-oriented programming language Java and Android SDK. There are some benefits that it can bring onto the table:

Object Oriented Programming

Java utilizes object-oriented programming, which helps to reuse objects in other programs, it prevents errors have hiding that particular object’s information that should be accessed, it makes programs preplanned and organized, and offers legacy code modernization and easy maintenance. In addition, object oriented programming offers a simple syntax and a mild learning curve, which makes a language easy to read, write, and maintain. There are also standards for enterprise computing, which supports many libraries, and has plenty of integration support.

Reduced Security Risks
Although Java may some vulnerabilities, as seen in the news recently, there are some security features that it implements: it doesn't have pointers, which reduces unauthorized memory access and it has a security manager, which controls the application where it can specify access rules, which helps to run Java applications in a 'sandbox' which reduces any outside security risks.

Development Platform - Android Studio
Java applets can run anywhere (known as Write Once Run Anywhere) and once you write this application and compile it into bytecode, you can run it on any platform that has a Java Virtual Machine (JVM) included in it. The JVM serves as a buffer between the code and the hardware. To develop Android based applications, we can use Android Studio, which is an IDE (Integrated Development Environment) where we can save/edit projects along with application emulation support (Android Virtual Device) and their Real time Profiler, which allows to show CPU, memory, and network activity from your application, as shown in Figure 17 below. So, there are some drawbacks however - the program is a huge pain to install and setup, as the setup itself is nearly 1 GB! It also has limited documentation and support, despite the fact that it was launched by Google in 2013.



```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout_editor_absoluteY="81dp"
    tools:layout_editor_absoluteX="0dp">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:contentDescription="@string/app_name"
        app:layout_constraintBottom_toTopOf="@+id/imageView6
        app:layout_constraintEnd_toStartOf="@+id/imageView3"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/grid_1" />

    <ImageView
        android:id="@+id/imageView3"
```

Figure 17: The Android Studio User Interface

Swift
A mobile development language for iOS applications and a successor of Objective-C and C++, it is claimed that Swift is faster and better than Objective-C. They are designed to run on watchOS, macOS, and tvOS. There are three key areas that Swift addresses: cleanliness and intuitive language, readability and concision, and ease of maintenance:

Cleanliness and Intuitive Language

From a syntax standpoint - Swift looks like plain English, which makes it easier for developers to use the language - which means developers with previous language backgrounds such as from C++, Java, and Python would find it simple to transition to this language.

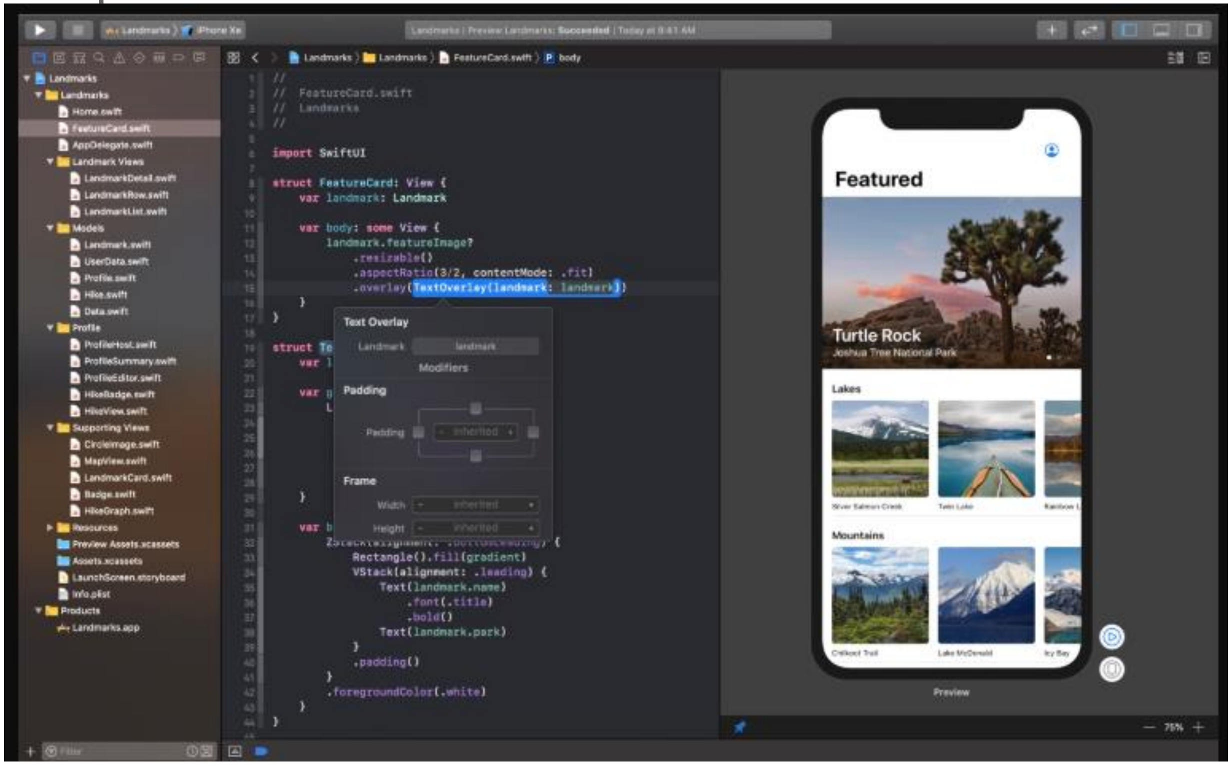
Ease of Maintenance

The file structure for Swift works with minimal interdependencies since the way it works is using a solitary code record which uses a .Swift file, consolidating an Objective-C header (.h file) and an execution document file (.m file). By using this code record, Swift has less code maintenance requirements.

Performance

Swift has a focus in performance. It is designed to outperform its predecessor. There is a 40% increase in performance compared to Objective-C. In addition, Swift ACR determines which instances are no longer needed in the program and gets rid of them on the user’s behalf. This allows the user to increase the CPU performance and there would not be any lagging.

Development Platform - XCode



```
//
//  FeatureCard.swift
//  Landmarks
//

import SwiftUI

struct FeatureCard: View {
    var landmark: Landmark

    var body: some View {
        landmark.featureImage?
            .resizable()
            .aspectRatio(3/2, contentMode: .fit)
            .overlay(TextOverlay(landmark: landmark))
    }
}

struct Te
    var l

    var g
        L



    }

    var b
        ZStack(alignment: .bottomLeading) {
            Rectangle().fill(gradient)
            VStack(alignment: .leading) {
                Text(landmark.name)
                    .font(.title)
                    .bold()
                Text(landmark.park)
            }
            .padding()
        }
        .foregroundColor(.white)
    }
}
```

Figure 18: XCode Development User Interface

XCode is an integrated development environment exclusive for IOS, macOS, watchOS, and tvOS applications. It is on the Mac App Store for free for macOS Mojave users. The user interface is shown above in Figure 18. With their application builder - SwiftUI there are some pros and cons for using the application language:

Pros

- Real-Time preview
- Simplified animations
- Simple Code using Declarative syntax
- UI Design Tools
- Open Source - plenty of community support

Cons

- Only on IOS platforms - meaning we would have to develop a separate code base for Android
- Relatively new language
- Poor interoperability with IDE/Third Party tools
- SwiftUI interfaces are basic

JavaScript with React Native Framework

Created by Facebook, React Native is an open source framework for developing cross platform mobile applications between IOS, Web, Android, and UWP. React Native eliminates the need for HTML, instead using JavaScript libraries for building user interfaces.

Pros:

- Only requires Javascript - no other languages needed!
- Fast to setup and start building
- No need to maintain multiple codebases
- Hot Reloading
- Fast Applications

Cons:

- Lack of custom modularity
- Not so smooth navigation
- Binded to Facebook rules

Maintenance and Updates

Maintenance with React native is somewhat a challenge, as since there can be separate issues on Android and IOS platforms, which can lead to two separate branches of the codebase in order to fix separate issues. Dependencies are also another issue as well - you have to make sure the React Native dependencies portion are updated, along with its other third-party dependencies as well, which can increase the likelihood of introducing bugs in the codebase. After making sure that the updates are applied, you also have to check that nothing in your code breaks as well when it comes to testing.

Performance

React Native applications work better when its CPU usage is optimized, however when it deals with graphics animation and processing, it struggles a bit. With this issue, Reactive Native went ahead and allowed the developers to embed native code into the application.

Development Platform
React is extremely simple to setup - just install the framework using NPM (Node Package Manager) through the terminal line, and after that you can run React Native commands from the terminal. In terms of project setup and configuration, the starter guide assumes that you already have the environment setup for IOS and Android and just jumps into creating a new project for IOS projects that is. For Android projects, there is no guide for creating one in its docs.

Support
Since its launch in 2015, React has been gaining traction ever since - there is an established community on Github and there are meetup conferences on React all over the world - the most recent one being React Native EU in Poland.

4.4.2. Back-End

For the database, selection is key in determining which database provided flexibility, cost, and ease of setup. We analyzed the three major database cloud providers on the market today - Amazon's AWS, Google Firebase, and Microsoft Azure.

4.4.2.1. Database Cloud Platforms

Amazon's AWS Amplify
According to Amazon's description of their Amplify service, it claims that it simplifies the process every application development - testing, building, and monitoring. It helps to easily add configuration of the application, such as push notifications, however additional features would have cost more. Its simple API helps you build a mobile application in about 10 minutes.
Their most prominent feature is their Amplify Console, which provides a workflow based on Github for hosting fullstack serverless web applications that have continuous deployment support. This helps to simplify backend and frontend deployment by simplifying the code repository (e.g. Github) and can be deployed in every commit of code, all in a single workflow. We can also connect a custom domain.
For pricing, the pricing is based on two features - deploy and build, and hosting and split into two pricing models:

Free Tier
- Build and Deploy- 1000 build minutes per month
- Hosting - 5GB stored/month, 15 GB served/month

Pay as you go
- Build and Deploy - $0.01 per build minute
- Hosting - $0.023 per GB stored per month, $0.15 per GB served

Google Firebase

Owned by Google Cloud, Firebase has many features such as: Test Lab (testing the application), Google analytics (unlimited and free application analytics), and Cloud Functions (run code without managing servers). According to some reviews some positive reviews are seamless user authentication API and management, is one of the cheapest cloud service providers, great documentation, and some support for integration libraries. There is real time data support as well.

There are some issues however, there is limited Javascript SDK support, as if you close the web/mobile application that you are working on, the data keeps going, which means you needed to develop a way to implement a cache system. There also is not a way to query your data properly, as it is only limited to basic filtering and pagination. So in the case if you need to develop a search functionality, you have to download the complete data and use it in the client through a server or a third party service.

For pricing, it is split into three tiers: Spark, Flame, and Blaze Plan. Since the Blaze Plan is a pay as you go service, we concluded that costs would have been higher than the Flame Plan, so we had to look at the Spark and the Flame Plan themselves (cost will be per month):

Spark Plan - Free

- Hosting for up to 1 GB
- 100 simultaneous database connections with 1 GB of storage
- 5 GB of storage
- 10 tests/day
- Phone Authentication for 10k/Month

Flame Plan - $25 a month

- Hosting for up to 10 GB
- 200k simultaneous database connections with 2.5 GB of storage
- 50 GB of storage
- Phone Authentication for 10k/Month

Microsoft Azure

Azure has excellent compatibility with Linux and Windows operating systems. With some features of Azure come with scalability - we can easily build websites with PHP, Node.js, or ASP.NET and quickly deploy in seconds. There are also simple backend solutions for mobile applications, such as user authentication, and a structured storage system in a snap. Some other features associated with Azure are cloud services, caching, virtual network, CDN (content distributed network), and business analytics just to name a few. Unfortunately, Azure has a high cost associated with, so it is better suited for enterprise organizations.

When it comes to authentication, there are some considerable options. Using Azure Active Directory, they can be used B2B or workplace applications. If working on the consumer side, they provide support for some third party sources such as Google and Facebook's API.

For pricing, they are split into 6 tiers: free, shared, basic, standard, premium, and isolated. Since this application was not going to be server heavy, we were looking into free, shared, and the basic tiers. The charges were based on per hour, but converted into per month for a responsible comparison between the other database costs. There is an initial $200 credit for Azure services however.

Free Plan

- 1 GB of Disk Space
- Up to 10 Mobile/Web/API apps

Shared Plan - $0.013/hour = $9.50/month

- 1 GB of Disk Space
- Up to 100 Mobile/Web/API apps
- Supported custom domain

Basic Plan - 0.075/hour = $54.75/month

- 10 GB of Disk Space
- Unlimited Mobile/Web/API apps
- Up to 3 Maximum instances
- Supported custom domain

Table 30 addresses the pros (+) and cons (-) of using each database cloud service.

Amazon AWS Amplify	Google Firebase	Microsoft Azure
+ One of the cheapest so far out of three databases + Cloud performance is very strong + Cloud solutions are beginner friendly + Excellent documentation + Real-time Database	+ Comparable low IT cost + Simple to implement, using cloud functions + Autoscaling built-in + Robust APIs for Javascript + Built in authentication support	+ Strong focus on security + Extremely scalable + Allows you to use any framework + Automate many tasks + Build a hybrid infrastructure
- Geared towards large teams - Can cost you more if you don’t manage it right. - Slight learning curve, as there are a large variety of products	- Database doesn’t provide relational data - Difficulty to query larger datasets - Locked into using it if you choose to scale - Vulnerable to downtime	- Can get very expensive - Requires management - Needs Platform expertise

Table 30: Database Pros and Cons

Decision Matrix

Table 31 is the decision matrix to decide on a database cloud service for our app.

(1: Lowest, 3: Highest)

Decision matrix	AWS	Firebase	Azure
Cost	3	3	1
Ease of Use	3	3	2
Performance	2	3	3
Support	3	2	3
Updates and Maintenance	1	3	2
Total	12	14	11

Table 31: Decision Matrix for Databases

4.4.2.2. Server-Side Development

In order to use the database and make it work with our application and hardware device, we need to make several API calls. An API call is the process that takes place after the API is set up and ready to go. With the API call, the information is sent and processed back to the user. We make calls to the server with these API calls.

There are a few languages that we will be considering for this section in order to make our API calls. We will be considering PHP, Node.js, and Python. For each language we will explore the pros and cons and make a decision matrix to determine which server-side language will be best in order to correctly have our API calls and functions done.

PHP Language

PHP is one of the oldest and most used scripting languages in the industry. It is used by several companies. PHP is fast, reliable, and well-known. If we are considering PHP, we are considering a language that is well supported by many developers. In Table 32 and 33, we will discuss more in detail the pros and cons of using PHP. In regards to the team knowledge, some of our teammates already have experience with PHP from previous projects and this language has been used for many professors as well.

Pros	**Description**
Low Barrier of Entry	Getting started with PHP is simple. We didn't need much knowledge in order to use PHP for our project. It has been a great tool and one of the oldest ones.
Huge Ecosystem	The ecosystem is big for PHP. This is due to the popularity with Wordpress and Magento. There are lots of training videos and tutorials online in order to master this language.
A Multitude of Pluggable Frameworks	Flexibility for the libraries being used across different frameworks. It has become easier and less painful to manage different frameworks with PHP.
Automation Tools	Automation tools are available for testing and deploying for PHP applications.
First Class Debugging	It is easy to debug remotely with First Class Debugging when it comes to PHP.

Table 32: Advantages of PHP

Cons	**Description**
Interpreted Language	Just like Python and Ruby, PHP is an interpreted language that is compiled down to Opcode.
Threaded Execution	There are scalability issues when it comes to PHP that plague all of the threaded runtimes for other languages such as Java Servlets, Python WSGI, or Ruby Rack, or Microsoft IIS. You are basically limited by memory to the number of connections you can support.
No IoT	Most interpreted languages, except by Node.js, do not handle IoT technologies. With PHP, there is not much support when it comes to this part.

Table 33: Disadvantages of PHP

Node.js (JavaScript)

Node.js is a runtime environment. Almost any popular editors now support Node.js, therefore, it is possible to use this environment almost anywhere. Nowadays using a middleware with Node.js is a common practice that makes development simpler for any developers. In Table 34 & 35, we looked into some of the pros and cons of using Node.js.

Pros	**Description**
Robust Technology Stack	Getting benefits such as better efficiency and overall developer productivity, code sharing and reuse, speed and performance, easy knowledge sharing within a team, and a huge number of free tools.
Fast-processing and event-based model	Node.js is fast. It has a great performance. Synchronization happens fast, which is helpful for event-based, real-time applications. Due to its asynchronous, non-blocking, single-threaded nature, Node.js is a popular choice for different types of applications such as online games, chats, video conferences and anything that requires updated data.
Seamless JSON Support	PHP and Ruby on Rails can use JSON format for communication, in our case, Node.js does it without converting between binary models and uses JavaScripts. This is great to build RESTful APIs for NoSQL database support.

Table 34: Advantages of Node.js

Cons	**Description**
Performance Issues with heavy computation tasks	The biggest drawback of Node.js is its inability to process CPU bound tasks. The problem is when Node.js receives a CPU bound tasks: whenever a heavy request comes to the event loop. Node.js would set all the CPI available to process that first and other requests will be queued. That would result in slow processing. Therefore, Node.js is not recommended for heavy computation.
Callback Issue	Due to the asynchronous nature of Node.js, this means it relies heavily on callbacks, the functions that run after each task in the queue is finished. There are a number of queued tasks in the background, and each with its respective callback, the result is a callback issue, which impacts the quality of the code.
Immaturity of Tooling	There is still poor quality or not properly documented/tested npm tools in the npm registry. It is not structured enough to offer the tools based on their rating or quality.

Table 35: Disadvantages of Node.js

Python

Python is one of the world’s most popular coding languages. Some of the top companies use Python in their technology stacks such as Instagram, Spotify and Dsiqus. In Table 36 & 37, we will be looking into the pros and cons of using Python in our project.

Pros	**Description**
Easy to Use and Read	Low entry barrier also applies to this one, as with PHP. It is very similar to the English language. The syntax is very simple to use. This makes it an attractive language to anyone who has never used a programming language before.
Asynchronous Coding	It is effortless to write and maintain asynchronous code using Python since there are no deadlocks or research contention or any other confusing issues.
Portability and Interactivity	Python has decent capabilities for dynamics semantics and fast prototyping, which is possible thanks to this. This can be embedded in a wide range of apps, even with ones that use different coding languages. Fixing new modules is easy with Python and it can also connect diverse components.

Table 36: Advantages of Python

Cons	**Description**
Speed Limitations	Python is an interpreted script language, which makes it slower than a lot of its competitors such as C/C++ or Java. Still, Python is used for a lot of software development teams.
Not Ideal for Memory-Intensive Tasks	Python is flexible with its data types. This results in fairly high memory consumption and unfortunately it makes it inconvenient to use for memory-intensive tasks.
Not Popular for Mobile App Development	Python is not a bad language to use for mobile development, it’s just that few companies use Python for that purpose. Many companies prefer native development for iOS and Android or React Native Development. It is not as popular.

Table 37: Disadvantages of Python

Decision Matrix

After understanding the advantages and disadvantages of our different options, it is time to make a decision. While many of these languages are great for our application, we will need to develop with only one. Table 38 will help us make a decision based on what we talked about above.

(1: Lowest, 3: Highest)

	PHP	Node.js	Python
Team Knowledge	3	2	1
Mobile Compatibility	2	3	1
IoT Compatibility	1	3	2
Performance	1	3	2
Community Support	3	1	2
Total	10	12	8

Table 38: Decision Table

In conclusion, Node.js was the best option when working with IoT technology and mobile development. It also will work great with our front-end which uses JavaScript.

5. Project Design

For the project design, we looked into several areas. Connecting the hardware and software was the most complicated part of the project. For that, there had to be a structure that allows the software and hardware to communicate effectively. From our research, we wanted to ensure we were using the frameworks, technologies, and hardware we felt the most comfortable in.

To start with, for the rest of this section, we are using the following colors to represent the reservation status on the hardware device:

- Available: Device is available and showing on the main screen of the app for the user to reserve. The color to represent this status is green
- Awaiting Confirmation: User attempts to reserve a spot and device is waiting for its button to be pressed. The color to represent this status is yellow
- Confirmed/Taken: User has pressed the button to confirm and the device verified that it is the correct user. The color to represent this status is red

In the following sections, we are breaking down the hardware and software into smaller areas. In these areas, we are talking about how the user interacts with our device, how the inside of the hardware looks like and how the user will use our application to reserve a spot.

5.1. Hardware Design

For the hardware design, we took into account our constraints listed above and try to meet with our specified requirements. We also tried to keep in mind how we had to build a design that can easily be taken apart and tested for errors. In order to make the initial debugging and experimenting process as easy as possible, we created a prototype on a temporary setup, which included jumper cables and protoboards. This allowed us to connect parts together without any permanent connection and allowed us to configure and update our prototype with ease.

5.1.1. Hardware Block Diagram

This section describes the overall flow and required components for this project. Since our project requires communication between hardware and software. We focused on the microcontroller, and Wi-Fi module in order to connect and transfer data from the software to the hardware and vice versa. Moreover, we paid attention to the interaction between the hardware components used and the microcontroller.

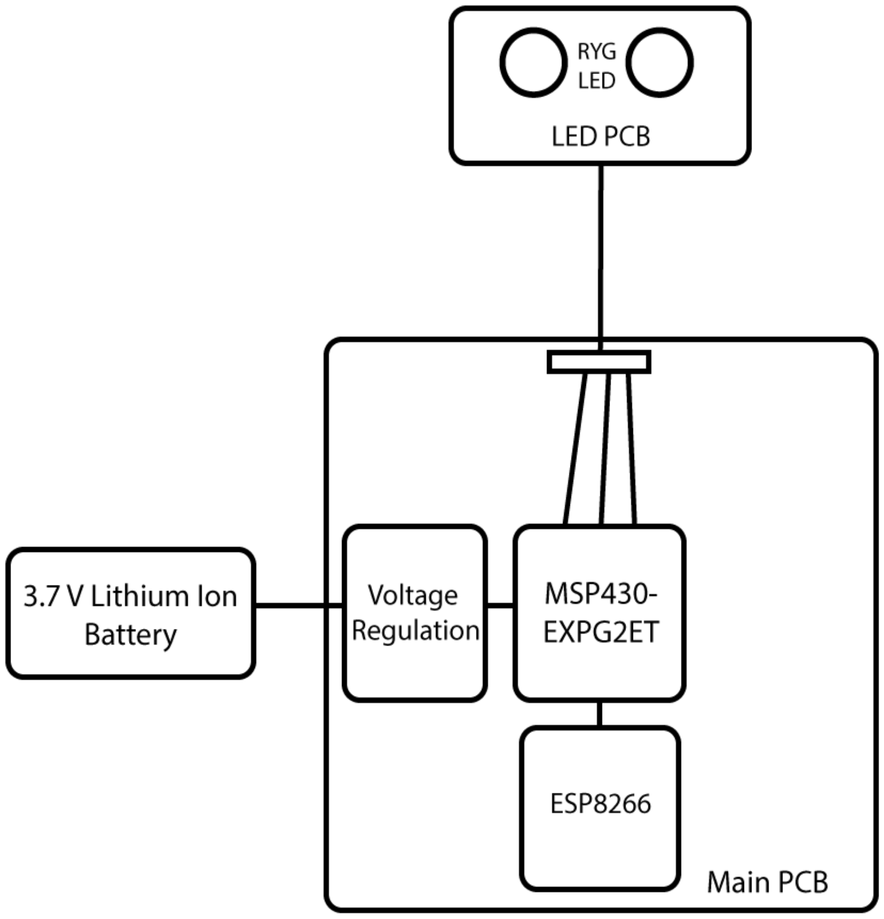


Figure 19: Hardware Block design

As shown in the figure above, the battery powered the main PCB where the microcontroller, Wi-Fi module and keypad are located. It has to go through a voltage regulator in order to keep a constant voltage through the system. The main PCB signals the LED PCB using connectors between them as well as the battery to the main PCB.

5.1.2. Power Management System Design

For the Power Management System, we needed to keep in mind all of the active components in this system, as well as how much voltage and current they require. Table 39 provides a list of the components we used.

Component	Supply Voltage	Current Draw
MSP430G2ET	1.8-3.3 V	0-400mA
ESP8266	2.5-3.6 V	80mA-200mA
LED	2.0-3.2 V	30mA

Table 39: Device input requirements

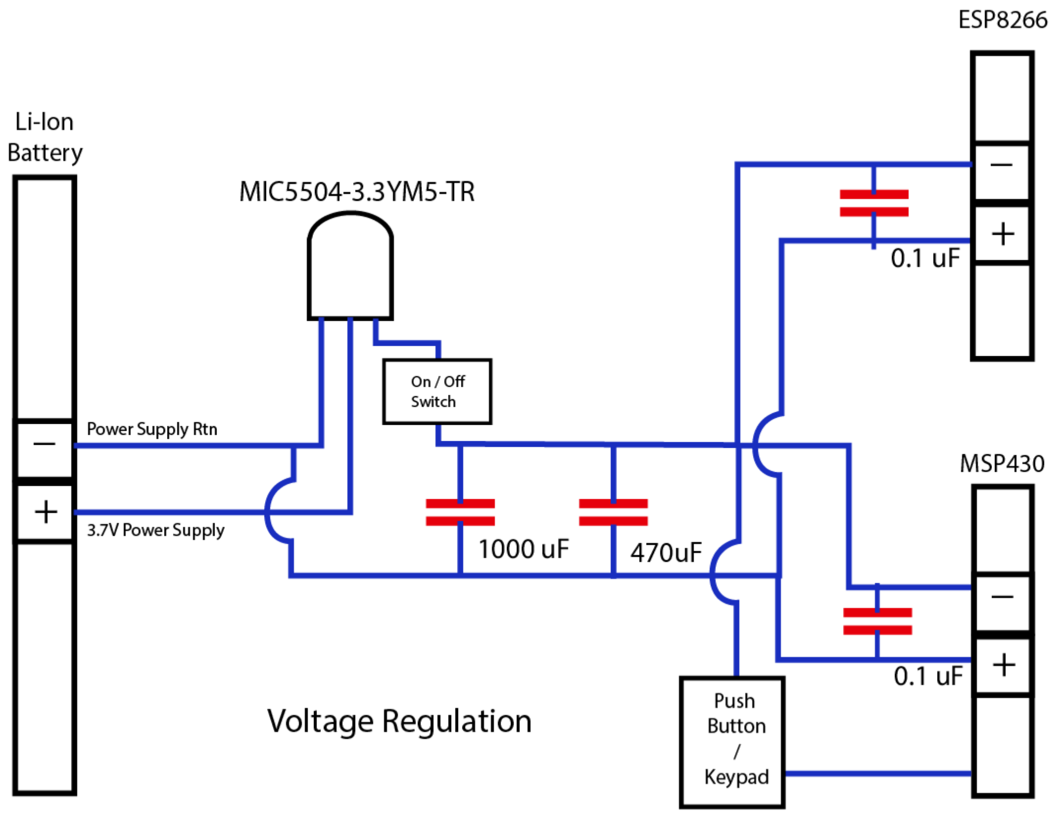


Figure 20: Power Supply Design

In order to provide the correct voltage to the MSP430 and the Wi-Fi Module, we utilized an LDO Voltage Regulator. This regulator was chosen to meet the current draw of both Microcontroller systems, as their peak current can reach 200-300 mA. However, to have a better power management system we are using an Atmega328p since it draws 15mA. Instead of the amount of current that MSP430 draws. Decoupling capacitors were added in order to assist with Voltage fluctuations due to the Battery or the ESP8266, which is known to cause Voltage peaks. There are 2 Decoupling capacitors on the main power supply lines and a 0.1uF decoupling capacitor next to each component’s VCC pin. They must be placed as close as possible in order to maximize their effect. Figure 20 above demonstrates this at a simplified level.

5.1.2.1. Regulator Design Concerns

MIC5504-3.3YMS-TR was used for this design. This component handles the current draw of the MSP430 and the ESP8266. This component has a dropout voltage of 380 mV at 300mA. Since our battery voltage is very close to the supply voltage of our logic devices, we needed to consider the dropout voltage as a risk factor. If the difference between the required voltage and the supplied voltage becomes too low, The voltage regulator ceases to function. However, if the battery voltage passes through the regulator, at that point it was enough for the logic devices to function properly. This mostly is the case when the battery reaches a low charge.

To stabilize the system when this occurs, we added capacitors in parallel. These helped smooth out any sharp drops in voltage. Also, since there is not much time for this project, we did not implement a battery charge level tester. This increases the risk for malfunction of the Regulator, but any fluctuations were diminished by the decoupling capacitors.

Another concern for the regulator was the current limit of the regulator, which is at 300mA. Once this limit is surpassed, an internal MOSFET shuts off the voltage regulator until the current returns. This could lead to sharp voltage changes that could affect the ESP8266, which can perform hard resets if the voltage varies outside of its range. However, the Wi-Fi module only consumes ~200mA when transmitting data, which only happens a minimal amount of times.

Since an Atmega328p needs a 5V to power, we are adding a second 5V voltage regulator, MIC5504-5.0YMS-TR, to power up this microcontroller instead of the MSP430.

5.1.2.2. Battery Design

In order to select the proper battery, it is crucial to take into consideration the intended operator of the system. Students will have some considerations that need to be made regarding the usability of the product. If disposable batteries are used in the system, that would require the user or administrator, for this particular case Universities, to purchase batteries continuously, which is unnecessary and not cost effective, especially with several rechargeable battery options.

After doing research and concluding a rechargeable battery as a final choice, the difficulty of recharging the battery should be considered. This is why the design of the battery should allow for the battery to be removable with relative ease. This battery contains a 2-wire JST connector which will have a mating female port in the main PCB.

This battery will be located at the bottom of the device for ease of access purposes and to prevent any interference/damage to other components if the circuit fails. The battery will be separated from the rest of the components using a casing which will be explained in the following design section.

5.1.3. Connectors

The system needed connectors to connect the two PCB designs and the battery. In this section, we will be discussing the connectors between these two PCB and the battery. We will also discuss the adhesive that connect our pieces together.

LED/Main

In order to connect the LED PCB into the main PCB we used cable connectors in the form of jump pins as shown in the schematic of the LED PCB. Using cable connectors (jump wires) increases mobility of the LED PCB in order to place in the desired location. It also allows for easy debugging and for simple replacement whenever an LED is determined to be faulty.

Main/Battery

The supplied battery terminates in a JST-PH connector. For the connection to the PCB to be detachable and easy to replace, we use a JST-PH socket. The battery we are ordering comes with a JST connector already integrated into the battery which makes it easier for integration.

5.1.4. Microcontroller

This section provides an overview of the connections and components necessary to successfully implement the design. Moreover, it illustrates the final circuit design, considerations and possible modifications. The final schematic design of the microcontroller can be found in the Main PCB schematic section.

MSP430G2ET

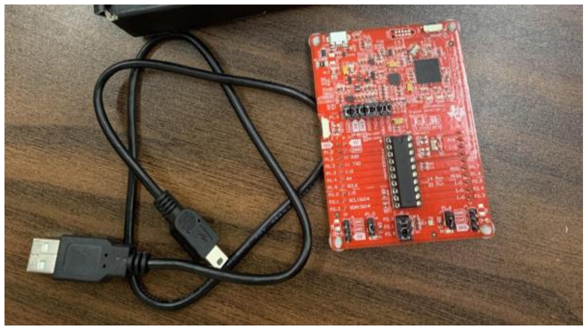


Figure 21: MSP430 Microcontroller

Figure 21 shows the MSPEXP430G2ET Microcontroller. It was going to be the main processor for all of the information processing. Figure 22 below demonstrates how the MSP430 was going to be connected to the rest of the components.

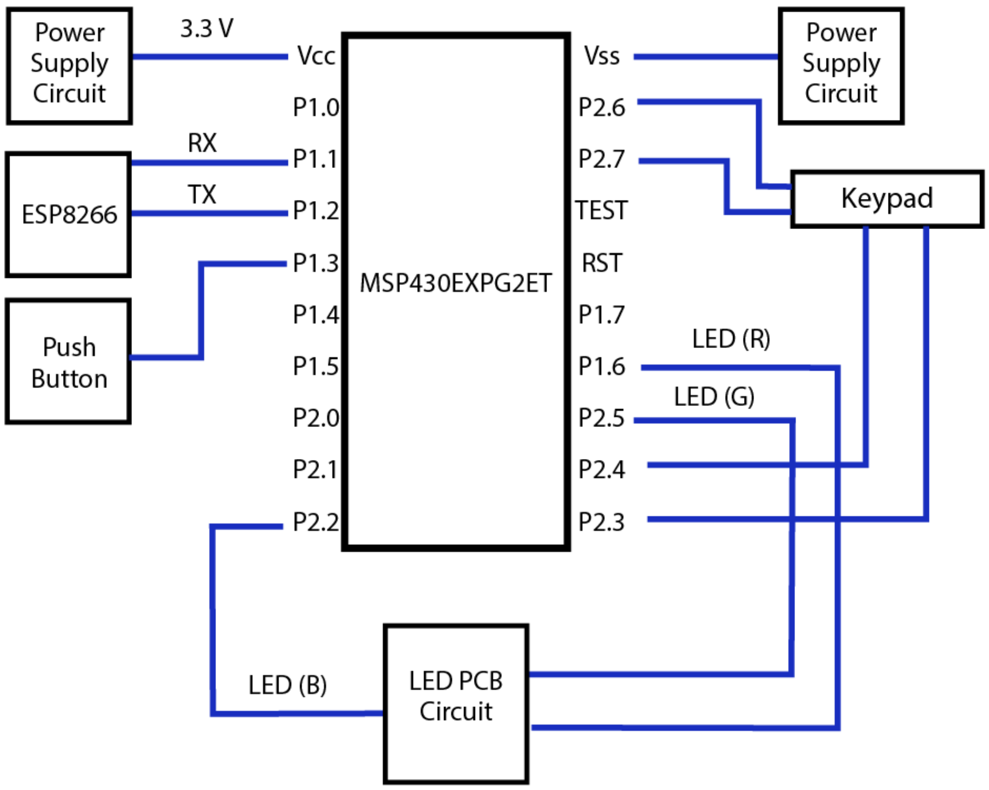


Figure 22a: MSP430 High-Level Wiring Diagram

The MSP430 was going to communicate to the ESP82666 via UART Serial communication. This was going to be done through pin 1.1 (RX) and pin 1.2 (TX). The RGB LEDs will require a Pulse Width Modulated signal, which was going to be done with Pins 1.6, 2.5, and P2.2. The Confirmation push button was going to connect to Pin 1.3 and the Keypad was going to be configured to Pins 2.6, 2.7, 2.4, and 2.3. Finally, the Supply voltage and ground was going to connect to the Power supply circuit described earlier. The MSP430 has a built-in crystal oscillator which it was going to use for the Watchdog timer and for the PWM Signal generation.

For prototyping, the development board was used to program and flash the MSP430. This board has an emulation portion that can be used to communicate with the host PC. By using this development board we can not only flash the MSP430, but we can also communicate with the ESP8266 Board.

This was the original design, however, after doing research and building the prototype, we decided to use the Atmega328p and came up with a new high level diagram of how this new microcontroller interacts with the rest of the components. We switched into the atmega328p because of better documentation accessibility, current draw is lower than other microcontrollers and also compatibility with Firebase. The same type of serial communication (UART) using TX and RX is used with this microcontroller which is connected to pin 4 and 5. The following figure represents our actual high level connection of the microcontroller.

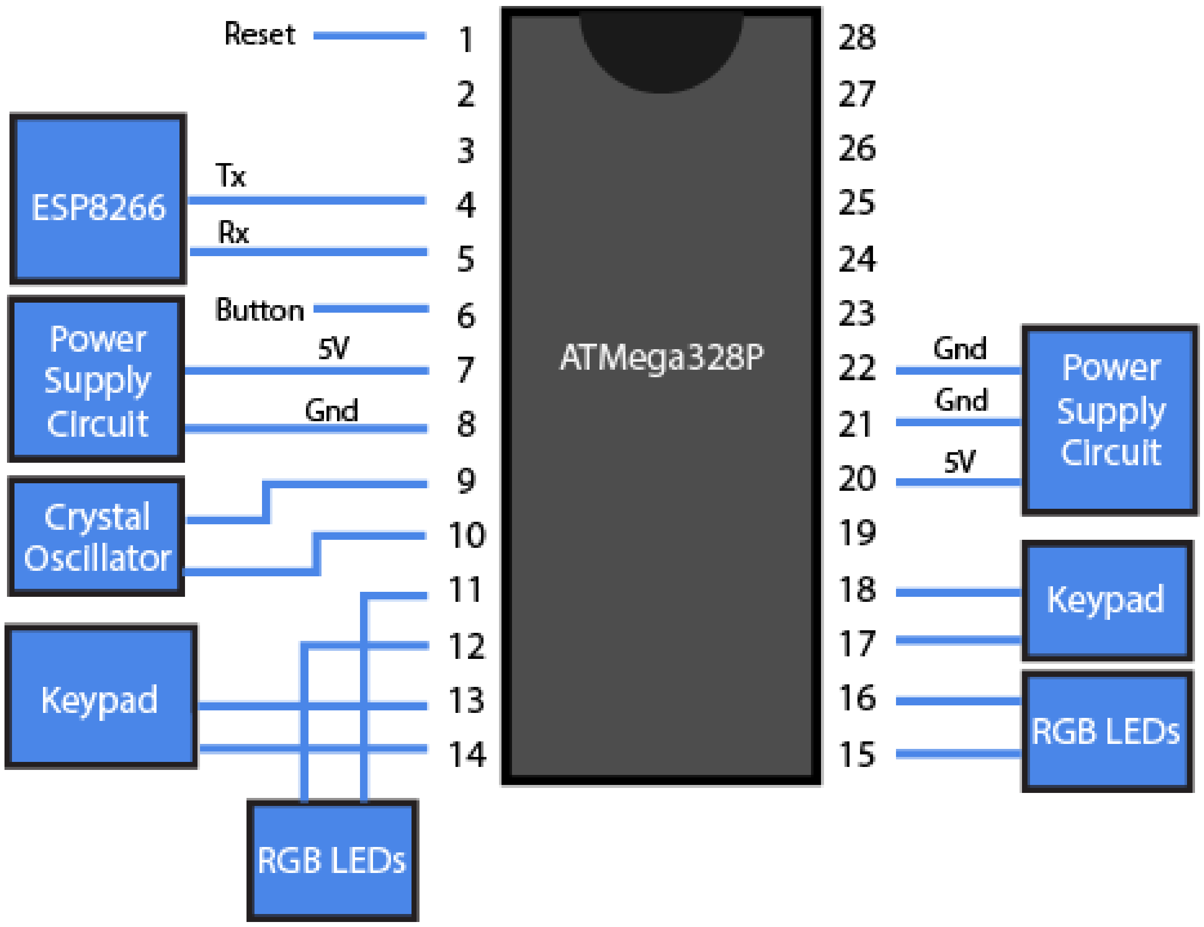


Figure 22b: Atmega328p High-Level Wiring Diagram

ESP8266

A Wi-Fi module is required for connecting the device via IoT. ESP8266 is compatible with the MSP430 series and also is the most popular and has more information and examples available. As mentioned in the research section, the WIFI module was programmed using the Arduino IDE which provides more libraries and it has more documentation online about libraries to connect to the web application. This WIFI module was implemented in the main PCB and was powered through the microcontroller using the battery chosen. It is another input of the microcontroller. The Rx pin connects into the P1.1 port of the microcontroller while the Tx pin connects to the P1.2 port of the microcontroller. This Wi-Fi module includes front-end module to export the data. It was required to transform into an IoT solution. Figure 23 shows the ESP8266 component acquired.

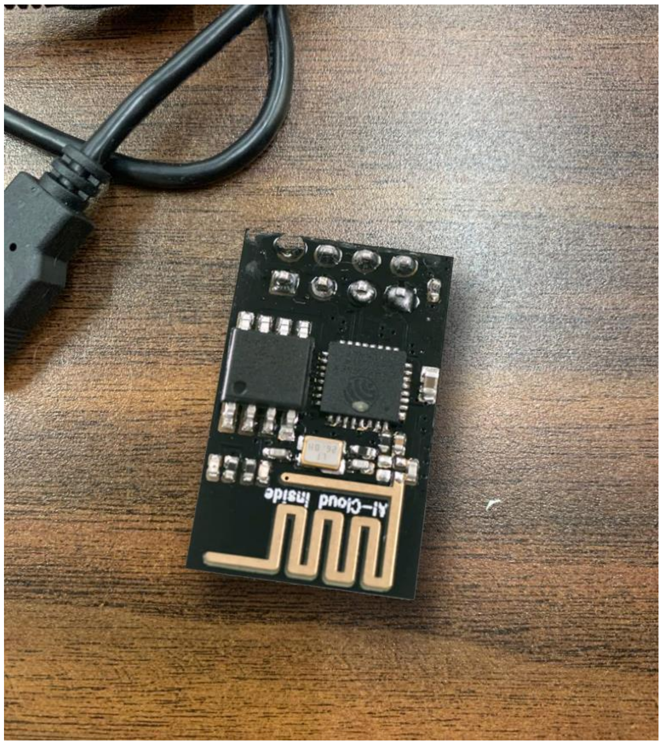


Figure 23: ESP8266 Wi-Fi Module

We are using the latest model of the ESP8266 which includes 8 Pins for serial communication, Power, Ground, Reset, Flash Configuration, Channel Enable, and GPIO Pins. This design does not use these GPIO pins as we are relying on the MSP430. However, for this new design it is relying on the Atmega328p instead. The Chip enable connects to the VCC with a 10k Ohm Resistor.

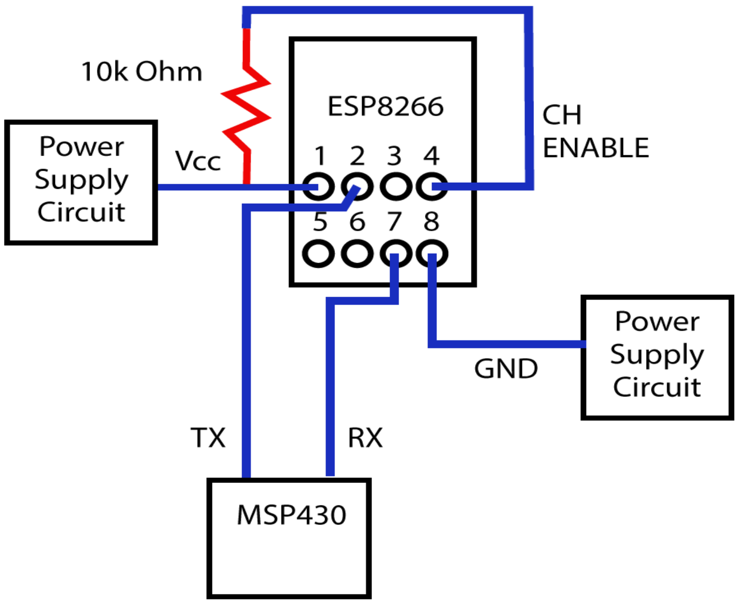


Figure 24: ESP8266 Connection Diagram

The board we are using had been broken out in order to create an easy experience for users to begin to learn to program the board, and while we began prototyping using this board, our ultimate goal is to implement this chip fully into our project. The following circuit is available on the Wiki site for the ESP8266. This served as the foundation for when we implemented the ESP8266 into the final design.
Ultimately, we did not have enough time to make these changes, so we implemented the ESP8266 as a CCA that mares to the PCB Board.

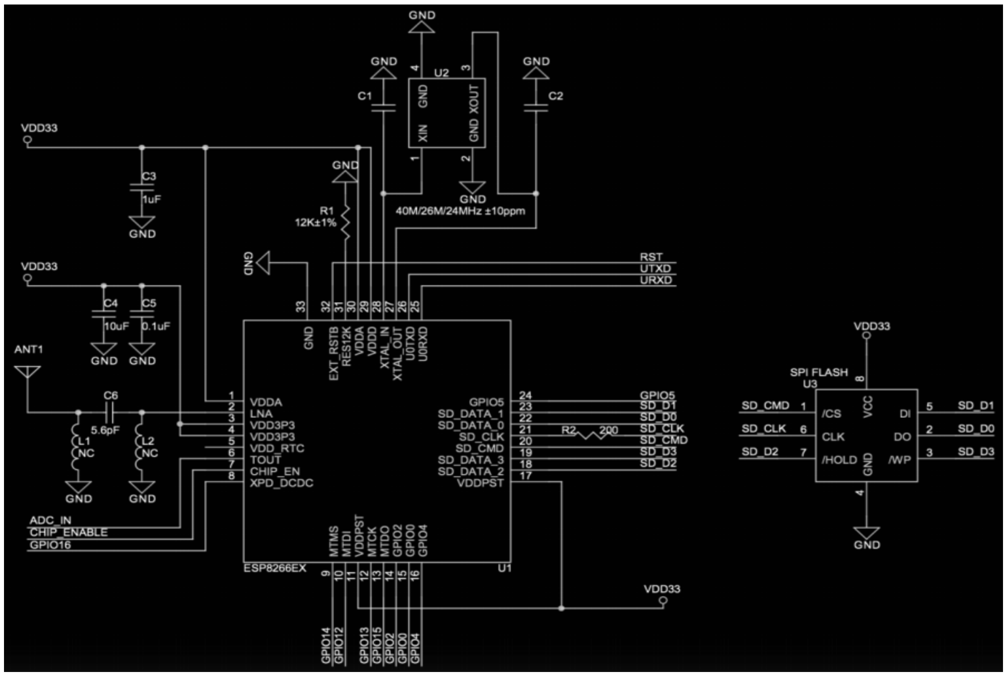


Figure 25: ESP8266 Dev Board Schematic

5.1.4.1. LED PCB

The LED has its own PCB for simplicity and organization purposes. Moreover, since the LED needs to be in a specific location, the transparent opaque stripe, having a separate PCB will make it easier to put it in the proper place. For the design of this PCB, a four pin header was originally used in order to connect the LED PCB with the main PCB.

In Figure 26a, the schematic of the circuit is shown. As shown in the schematic, since we are using a RGB LED we will need three pins for each color (red, green and blue) and another pin for the ground. For the red pin, a 43 ohm resistor will be used and for the other two colors, green and blue, two resistors will be used using 3.3 ohm resistance based on the specification of the LED chosen.

However, we decided to use two RGB LED boards which needed 4 PMW pins because of the limited number of pins left on the Atmega328p. Because we only needed Red and Green pins for the purpose of the device, we decided to not use the last Blue pin and did not connect it to the microcontroller. The updated LED PCB schematic is shown in fig 26b.

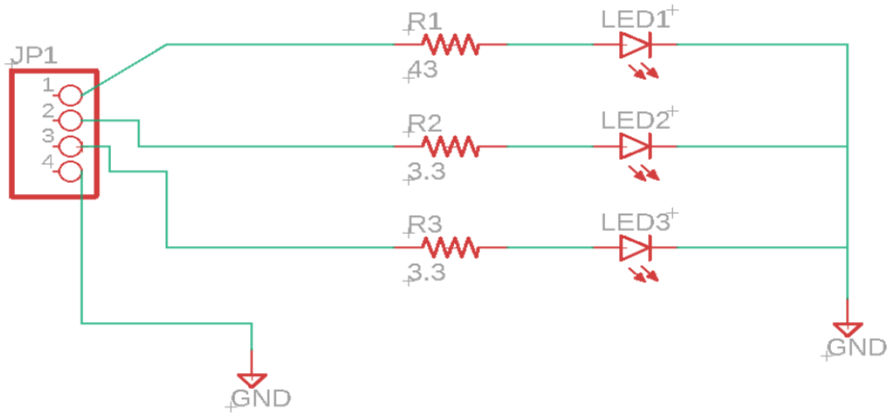


Figure 26a: Schematic of the LED PCB circuit

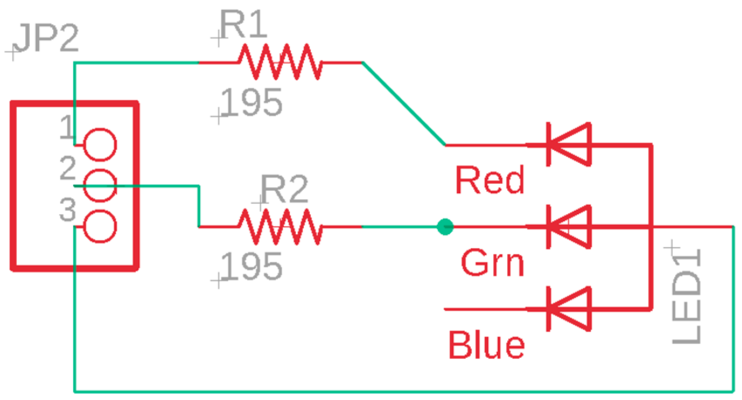


Figure 26b: Final Schematic of the LED PCB circuit

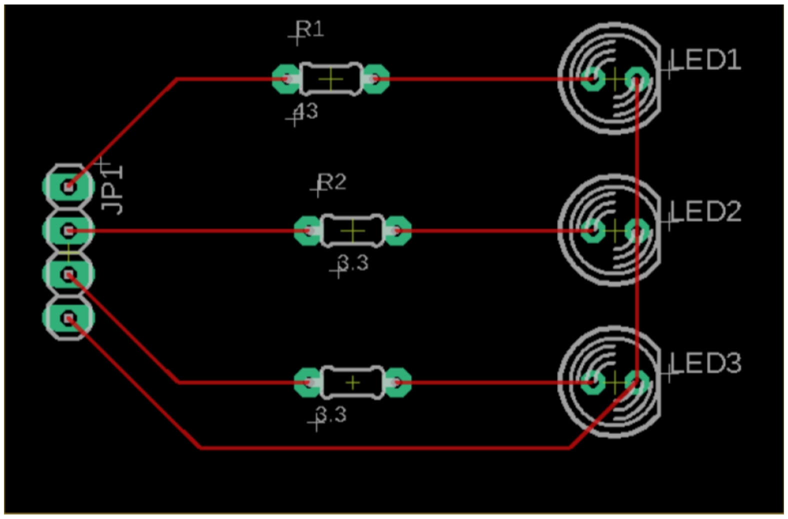


Figure 27a: Schematic of the LED PCB board

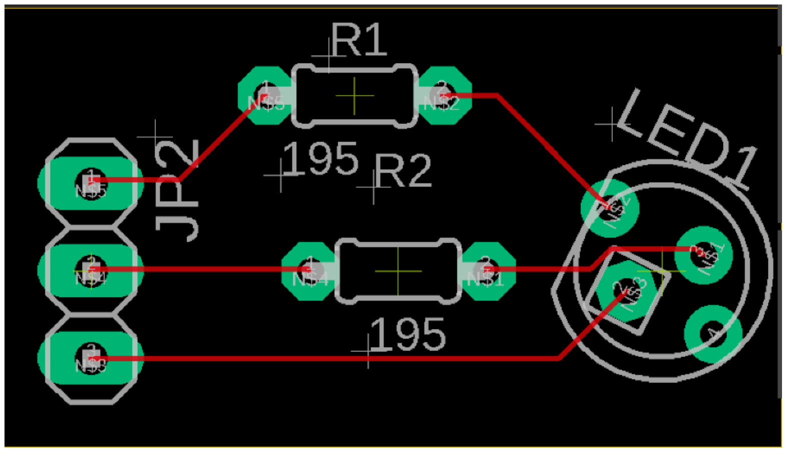


Figure 27b: Final Schematic of the LED PCB board

Figure 27a above shows the connections and how the PCB board would have looked like with the traces included. The PCB board is still small enough to fit the required components in order to keep the design within the budget limit. This board was printed separately from the Main PCB board and was attached to it using jumper wires. This allowed for easy debugging and replacement in the future. Figure 27b above shows the actual board with all the connections used in the device.

5.1.4.2. Main PCB

The main PCB shown in Figure 28a and 28b houses most of the components. It is comprised of the power supply circuit, The MSP430 Circuit (which is now replaced with the Atmega328p), the ESP8266 Circuit, and connectors to the battery and LED PCBs. It also contains the Keyboard, Switches, and Buttons used in the circuit. Table 40 serves as a guide for the component values of the capacitors/resistors.

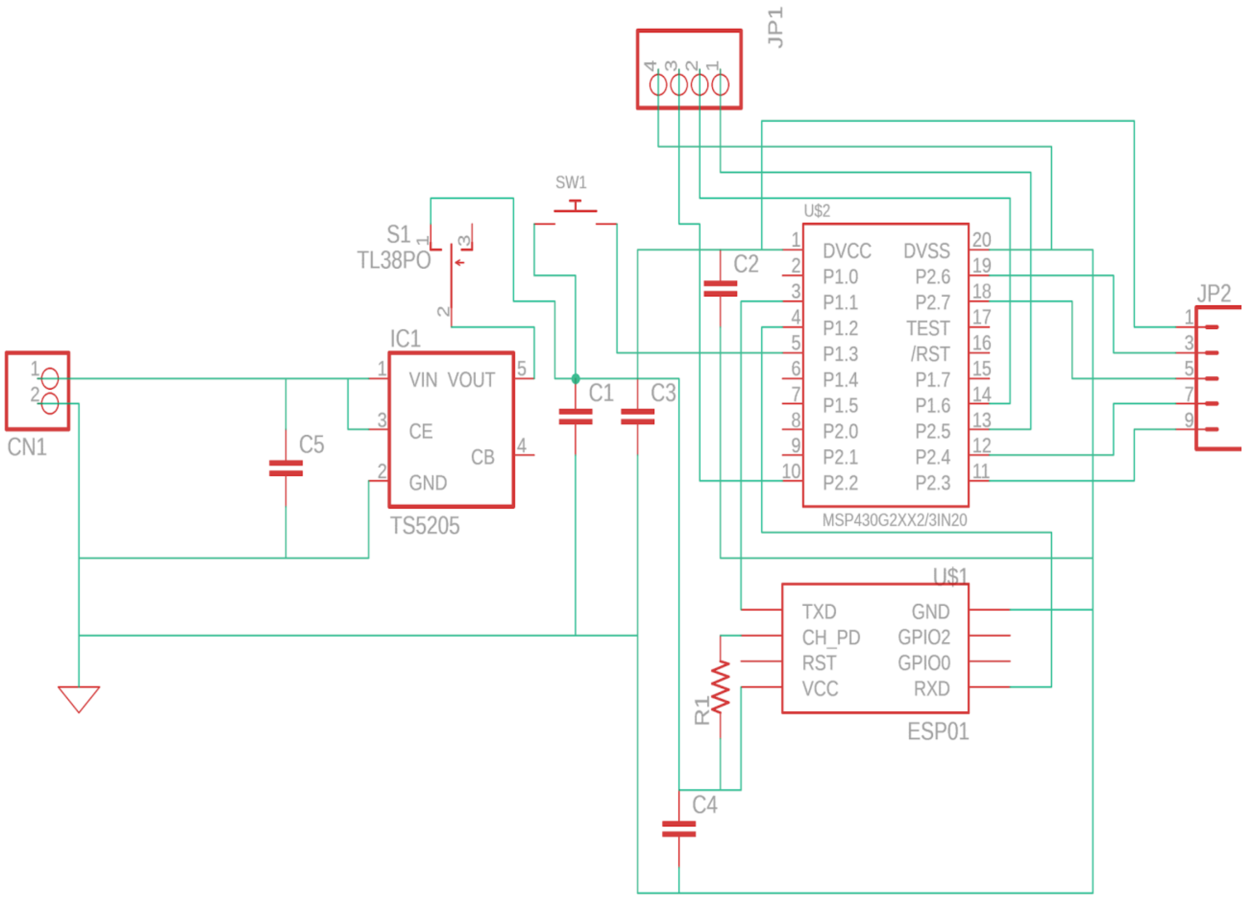


Figure 28a: Main PCB Schematic

Label	Component	Value
C1	Capacitor	1000 uF
C2	Capacitor	0.1 uF
C3	Capacitor	470 uF
C4	Capacitor	0.1 uF
C5	Capacitor	100uF
R1	Resistor	10 kΩ

Table 40: Component Value Table

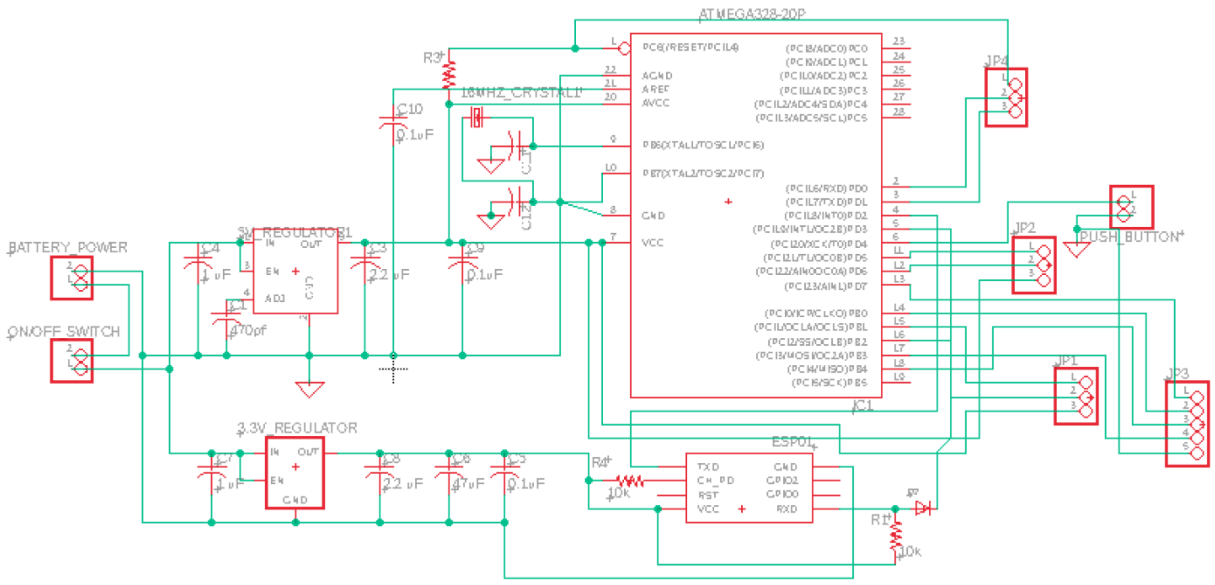


Figure 28b: Main PCB schematic updated

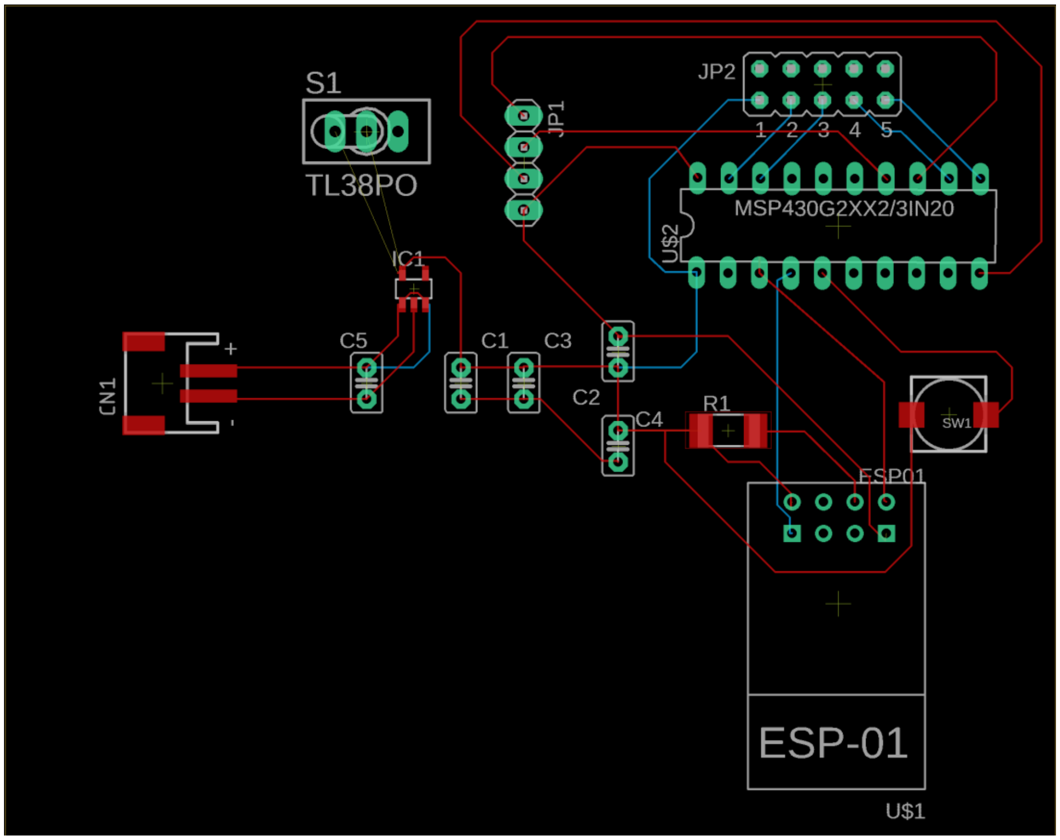


Figure 29a: Main PCB board design

Figure 29a shows the board design for the Main PCB. After positioning all of the components, it was found that the connections would be realizable using only 2 layers.

This minimizes the cost and complexity of the designs. The components were spaced apart as are the connectors to allow for easy debugging and manipulation. There was a large amount of space in the bottom left corner of the board which can serve as a ground plane.

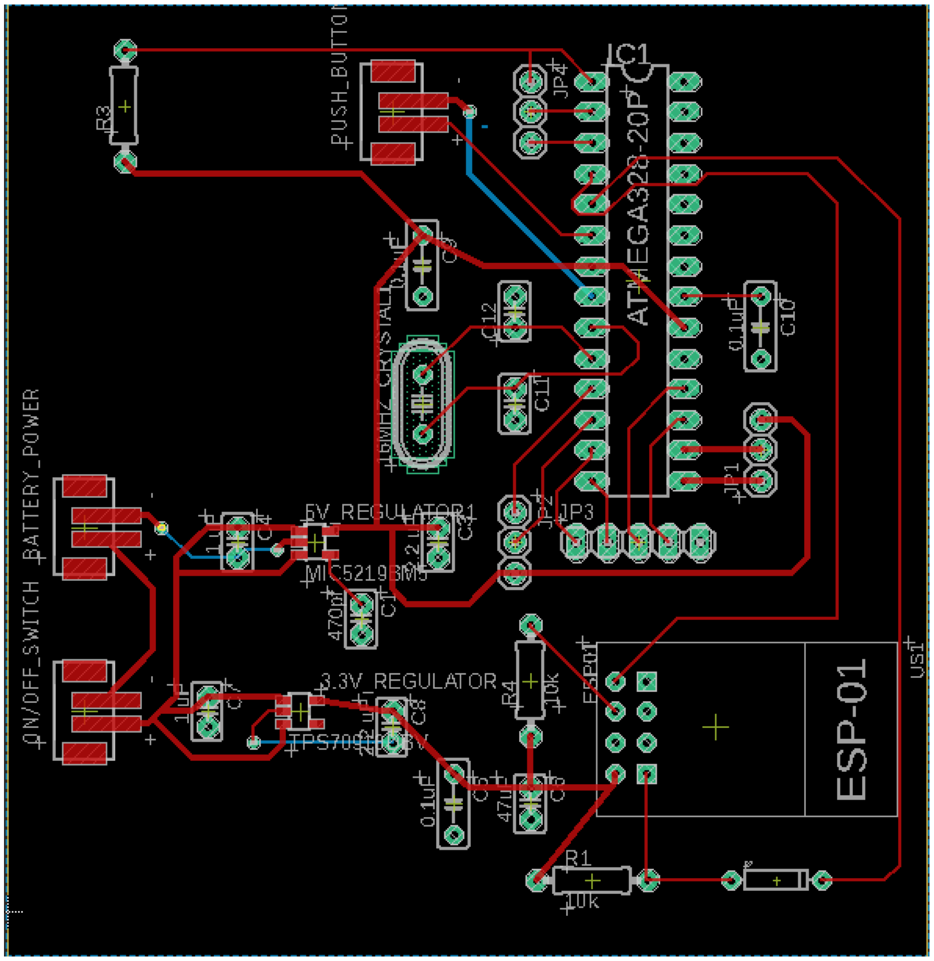


Figure 29b: Main PCB board design updated

The main PCB Board is 10cm by 10cm. The main components used are shown here. Connections to the Keypad, LED PCBs, and ESP Wi-Fi module are done using female jumper pin headers to allow for easy removal. The pushbutton, battery, and on/off switch are attached using JST-PH 2 pin connectors. On this PCB we have also included test points for debugging and programming purposes. We use through-hole capacitors and resistors for quicker assembling. However, the two voltage regulators are surface mounted since these were the regulators that met our requirements, making sure that the enable pin of the regulators go to the input. When designing the board, we made sure to avoid 90 degree angles on our traces and to increase the width of traces carrying supply voltage. We also created a ground plane on the bottom layer, which all of our through hole DGND points will connect to.

5.1.5. Enclosure Design

In this section, the design of the casing device is discussed, as well as the reasons to choose this specific design. Moreover, we discuss possible difficulties that came up when splitting the design into three sections and how would the 3D printing was affected.

5.1.5.1. Overall Design

As discussed previously, the enclosure is a cylindrical design manufactured out of ABS Plastic. Some initial mockups of what our design would look like are shown below in Figure 30.

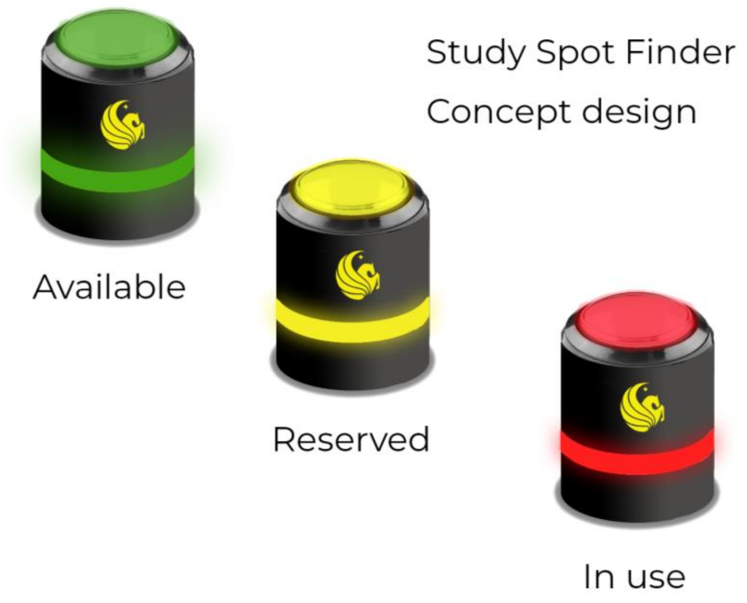


Figure 30: Concept Art for Study Spot Finder

The idea of the device is to appear modern and sleek, while not drawing too much attention to itself. Anyone who sees the device should know that it belongs to the school and that it has some purpose with the table. The universal color of the LEDs and the school logo on the device help portray these things.

This device was divided into three different casing designs: the bottom, middle, and top casing. We considered several designs to make the device easier to fix. Our device design allows the bottom to open as a cover which will be the bottom casing. This part of the device will house all the components and these two parts will be screwed together.

5.1.6. Outside Casing Design

The device uses an outside casing that fits over the inside design, which houses the main components and battery. The cover does not contain any components besides the main confirmation button and the 4 digit keypad, which connects to the main PCB Board through flexible, detachable connectors.

The outside design has two holes on the side at the bottom so it can be put together with the inner casing design. The inner design has threads tapped into the bottom in order to accommodate a screw. This screw is mostly a specialized screw so as to prevent tampering. The outside casing was decided to be made of black ABS plastic. Figure 31 shows the outside dimensions of the outer casing.

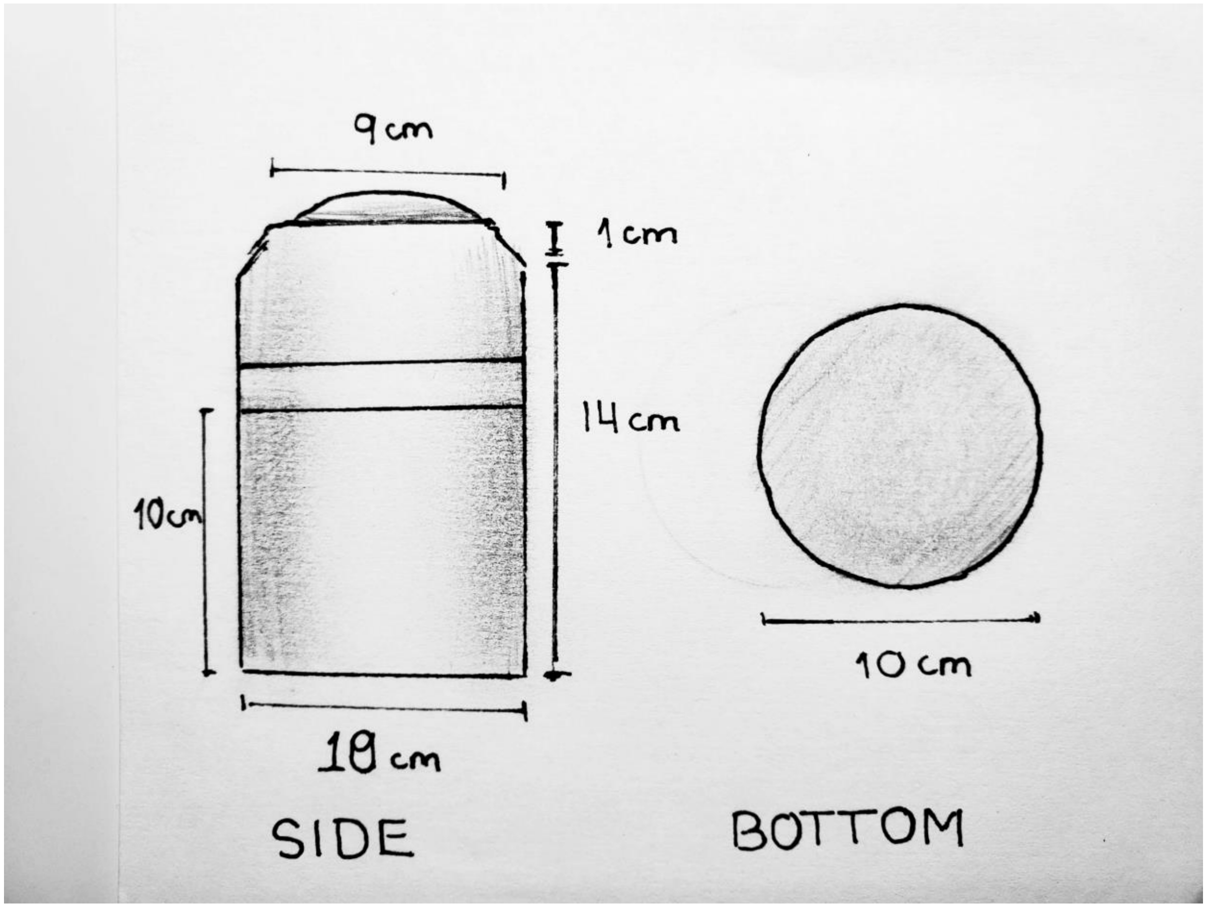


Figure 31a: Outside Casing Design for the Finder

The process of constructing the case was done in 3 separate parts. The top of the case was designed to fit the button we used. The top connects with the middle ring, which opaque to allow the light from the LEDs to shine through. This connects to the bottom part of the outer casing, which has no top or bottom in order to slide over the inside casing. The three pieces combine together using Plastic Weld. The ABS plastic is able to melt at high temperatures, and the plastic weld creates a chemical reaction that would have bonded the pieces together and create a smooth, seamless piece.

The top will be susceptible to the harsh environment of the library, so it would be wise to cover it in a clear coating to prevent fractures in the plastic or from students. The plastic layers from 3D printing are not as strong vertically as they are horizontally, so adding a clear coat would improve its resistance and would create a nice shine to give off a nice appearance.

The original designed was not able to be built due to some manufacturing and design restrictions we had on the 3D printing. We kept the design of the device, however, the process and the material used were not the same. Instead, we used two plastic PVC tube of 10 diameter for the top and the bottom piece where the main PCB and components are hold. For the LED PCBs, we used clear polypropylene tube so the LED can still be seen through it.

5.1.7. Inside Casing Design

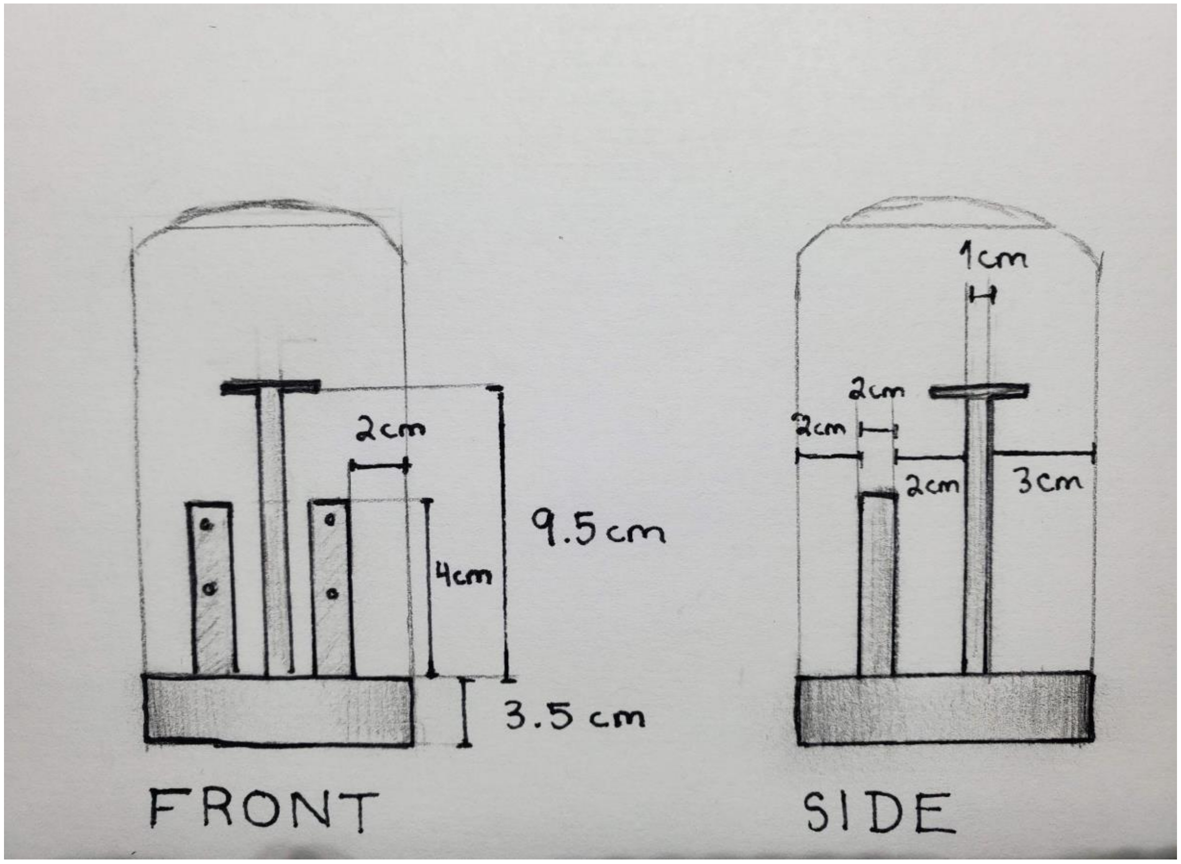


Figure 31b: Inside Casing Design for the Finder

For the inside casing design, There was going to be 2 fixtures; one fixture was going to be for the main PCB and another fixture was going to be for the LED PCB. These fixtures are shown in Figure 31b. The fixture for the main PCB was going to have two parallel blocks that was going to have slots on the insides in order to hold the main PCB in place without it moving around. It also was going to allow for the PCB to be taken out and worked on or replaced with relative ease. The fixture was going to hold the main PCB sideways because there was going to be many connections being made to the board and there is a big chance they were going to need to be connected and disconnected for maintenance.

The second fixture was going to be for the LED PCB. It was going to be placed higher up because it needs to be visible through the clear ring, which is installed at 10cm high. This fixture was going to be connected only to the bottom and not the sides because the outside enclosure was going to need to fit over and come off. That is why we decided to use a single pole that will support the PCB. It is thin to minimize weight but also thick enough to support structural integrity. Yet due to a world pandemic (COVID19) affecting manufacturing and labs operating hours, we had to change the inside and outside design and improvised a handmade casing with tools that were available to us. The main layout can be seen below:

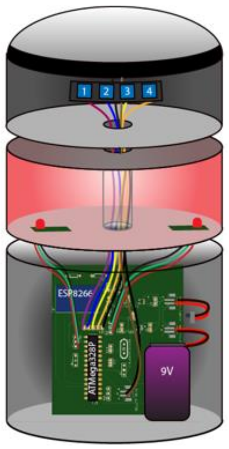


Figure 31c: Inside Casing Design for the Finder Final

5.1.8. Battery Casing Design

The battery casing design on the bottom of the device was designed as a separate compartment where it was going to hold the battery. This allows for the battery to be quickly accessible for repairs and replacement. Out original design was to have a hatched lid which would be able to swivel open on a hinged joint.

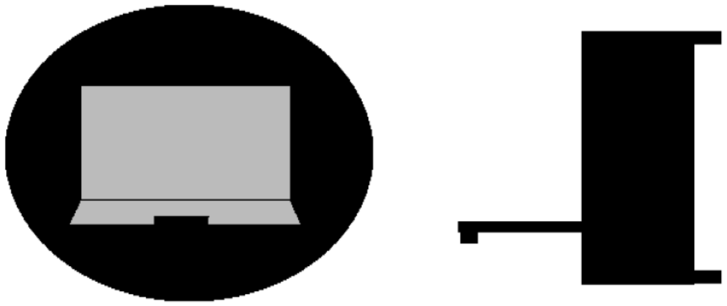
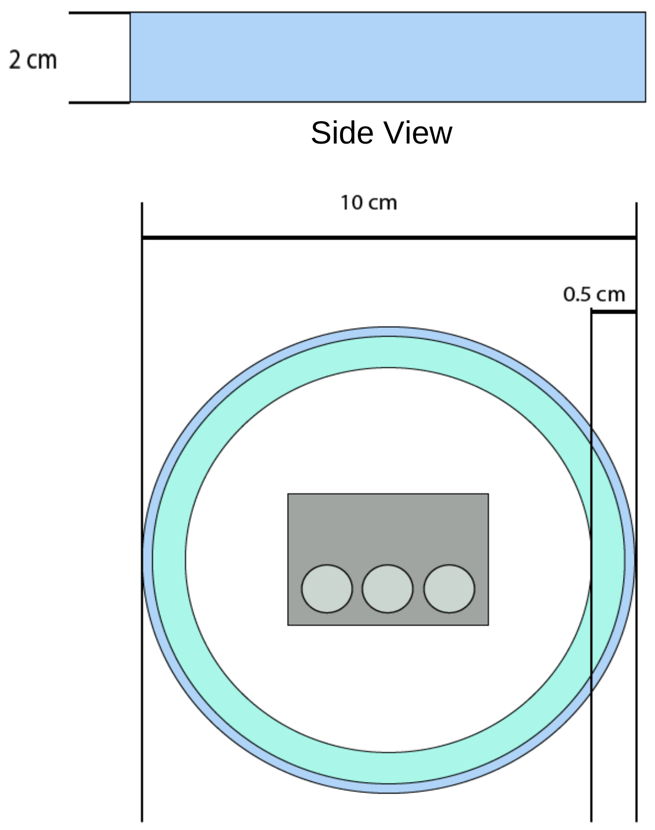


Figure 32: Design of the bottom angle and side for battery casing

Figure 32 shows the bottom side of the device on the left where there would have been a compartment for the battery casing if battery needs to be replaced for any reason. On the right of the Figure, it shows from the sides where it would have contained two plastic sticks coming out from the top in order to screw in the outside casing of the device. However, this was not possible since our design was modified and the battery is held within the bottom case, along with the main PCB.

5.1.9. LED Ring casing

The LED ring needed to be transparent enough to allow the light to shine through, but also opaque enough to not reveal the inside of the device. The ring needed to be in the form of a ring with a thickness equivalent to the thickness of the outer case in order to give the device a flush appearance. The LED ring consisted of a clear ring coated with an outer sheet of diffusive material. Figure 33 below demonstrates the size and dimensions of the LED ring from a side view and a top down view.



Top View
Figure 33: LED Ring Casing design

5.1.10. Adhesive

Seeing as the device comes in separate parts, they needed to be assembled together. The outer casing was made up of 3 parts:

- Top Portion (Push Button)
- LED Ring (LED PCB)
- Bottom Portion (Main PCB)

The adhesive held these three parts together. The bottom casing consisted of two main parts:

- Base
- Battery Casing

There was going to be adhesive and possibly a screw holding these parts together. The outer casing was going connect to the inner casing via two screws at the base where the two parts meet. However, since we are not using battery casing and we are using PVC tubes we decided to hold it with silicon glue.

5.2. Software Design

5.2.1. Software Development Process

When it comes to developing software, we had to implement an organized method in order to get our application and its related components finished by the end of the senior design project. There were several software development lifecycle models we could have been implemented - some of the most notable ones are Waterfall and Agile development, but there are also some other models such as iterative and spiral model. In general, the software development lifecycle (SDLC) usually has these core components:

Planning

The first and most critical stage of the development lifecycle, planning required gathering requirements needed for this project - we gathered information from the customer, surveys, experts, etc. This stage also defined the risks associated with this project and to see if the project was feasible.

Defining

After we have identified the requirements for this project we had to document it and make sure that it is approved by customer (in this case Dr.Richie/senior design panel). We also had to identify what features that we needed in our product that were required towards the end of deployment (known as minimum viable product) and which ones were stretch goals.

Designing
The design of the product comes down what is implemented in the requirements section and turns it into the design specification - the stakeholders went over it and suggested any possible changes for it. This was the pivotal point of this process - it determined if a project can take off or burn to the ground.

Development/Building
This was the stage where the product was built - developers must adhere to coding guidelines by their associated organizations. There were different programming languages that might have been used in this process, and it came down to what software was being developed.

Testing
The testing stage were where all the product defects were retested, tracked, reported, and fixed - until the standards were up to par defined in the requirements.

5.2.2. Deployment

We covered three major software development cycles for consideration in this project - waterfall model, agile model, and iterative model for getting software shipped out.

Waterfall Model
A classical software development model implemented in government agencies and in many companies, the waterfall model places an emphasis on planning in order to make sure design flaws don’t crop up during development. There are some basic principles associated with this model, it is divided into sequences, with allowable overlap and splash back between stages. There is also emphasis on planning, deadlines, and budget as well. A visual diagram is shown in Figure 34.

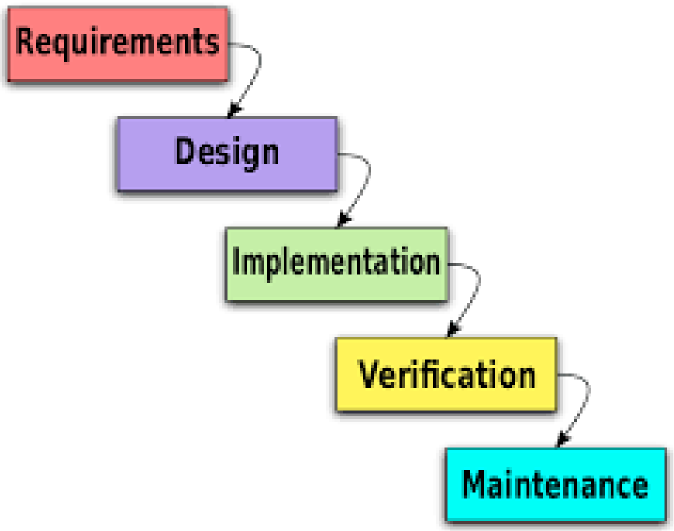


Figure 34: Waterfall Model

Iterative Model

Unlike the waterfall model, the iterative model does not place an emphasis on requirement gathering - it starts with already implementing and specifying part of the software, which can be later be reviewed at a later stage in order to develop further requirements. This is then repeated, creating new versions for the software. As said by Dr. Richie, this was implemented in Senior Design 2, as the requirement needs kept changing and evolving. Figure 35 below shows this concept:

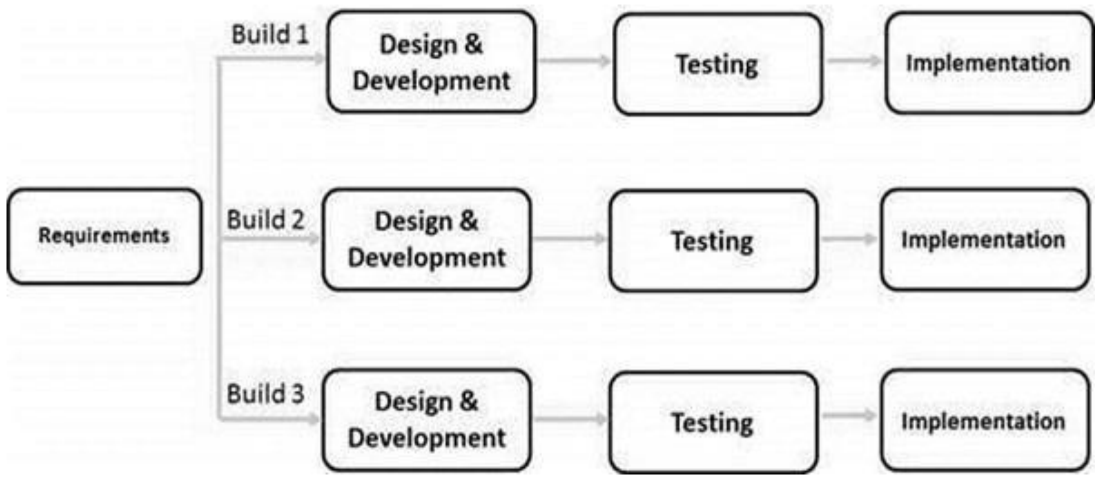


Figure 35: Iterative Model

As seen in the model, the stages were built into small parts, which allows the development team to give feedback and improvements earlier in the process. It does show some similarities to the Waterfall Model, as each build is similar to a mini Waterfall Model.

Agile Model

More focused on customer satisfaction and currently implemented in new tech companies and startups, Agile development is starting to gain more traction. Recently developed in the 1990s, it was later known to become the ‘Agile Manifesto’. The breakdown of this model is as follows:

1. They begin as kick-off meetings (‘sprint kick-offs’)
2. Requirements take precedence and is understood
3. We estimate which requirements take certain point values (time estimation)
4. Design using simple diagrams
5. Development process begins with plenty of team interaction
6. Code is tested more frequently
7. Code maybe refactored

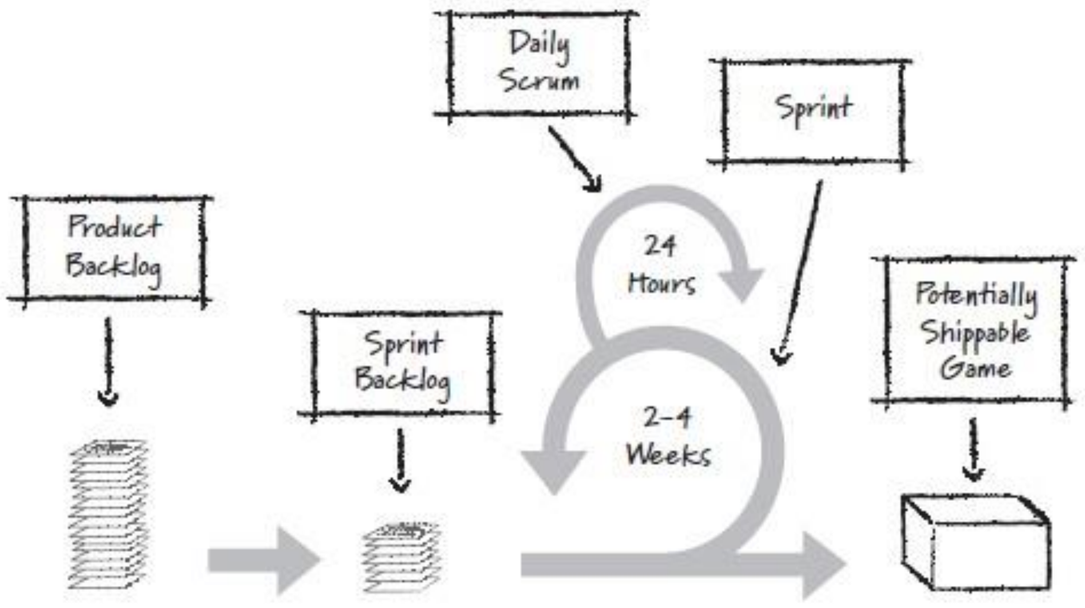


Figure 36: Agile model

Figure 36 shows how agile is organized - it starts with the product backlog, as determined by the product manager, who takes input from stakeholders, customers, etc, which is a list of requirements that gets compiled into a product backlog. The requirements then get refined - putting them into focused sections called sprints which happen every 2-4 weeks, with daily standups at the same time. At the end of the sprint, we hopefully have gotten a shippable product. Table 41 below shows the pros (+) and cons (-) of the software processes.

Waterfall	Iterative	Agile
+ Simple and easy to implement + Phases are done one at a time + Easy to manage + Good for small projects	+ More flexible than waterfall model + Works well for small/moderate projects + Documentation can be reduced	+ Can incorporate changes and release a product in less time + Suited for web based apps in correcting bugs over time
- Adjustments can kill a project - High amounts of risk - Software is not produced until the end of the life cycle - Poor model for complex projects	- Activities performed in parallel are subject to miscommunication - Unforeseen interdependencies can create problems - Milestones are ambiguous	- Limited documentation - Very difficult measurement - Fragmented output rather than one cohesive unit.

Table 41: Pros and Cons for Software Processes

Conclusion
There were some areas in the models that appeal to us, which was where we have decided to implement a combination of Agile and iterative development during the entire process.

5.2.3. Front-End

For the interface between the user and the device, a mobile application was used. This mobile application is a React App. This app is compatible with any mobile devices using Android or iOS, and is written with JavaScript and uses the React Framework.
The mobile app has different interactions for the user/student and admin/staff to use during the reservation and set-up device processes.

5.2.3.1. General User/Student and Admin Flow Sequences

Before we jump into each individual component, we would like to give a general idea of how our application works for both user and admin. This is a general overview of how the application works. The next sections will explain in detail how each individual component is working.

The first diagram (Figure 37a) describes the user flow sequence when they enter the application - they are first greeted with a user login screen that prompts for a user or admin login. If the person is a user, in this case a student, it will take them to a different page that shows all the available spots.

The second diagram (Figure 37b) describes the admin flow sequence. If the user is an admin, it takes them to a separate panel that gives them the ability to register the device and enter any additional properties associated with it, such as outlets and what type of table it has (e.g. group or individual). It will also allow them to modify and delete devices.

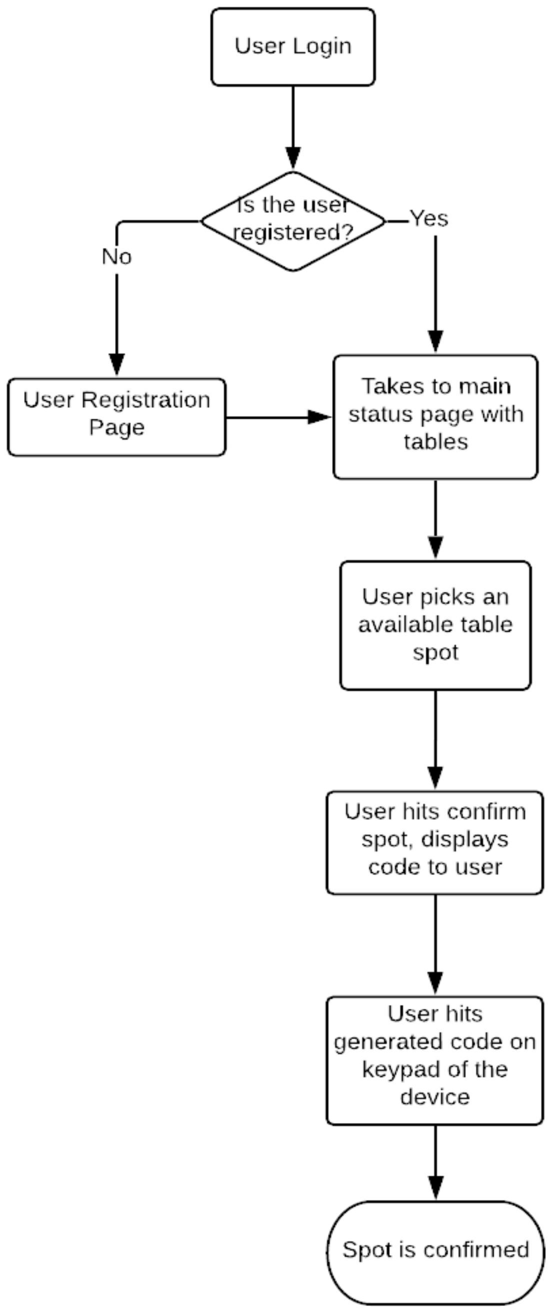

User Login
Is the user registered?
No
Yes
User Registration Page
Takes to main status page with tables
User picks an available table spot
User hits confirm spot, displays code to user
User hits generated code on keypad of the device
Spot is confirmed

Figure 37a. User flow diagram of user login page on application

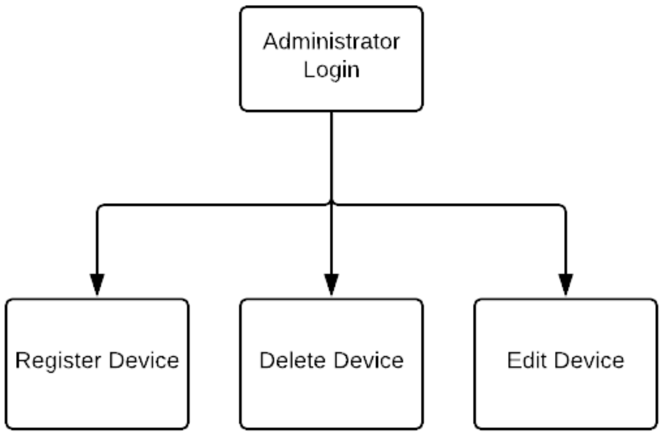


Figure 37b. Admin diagram flow

5.2.3.2. User (Student) First Interaction

In this section, we will be going over the entire user/student journey in order to understand how they are using the application. This part of the design allows us to not have to worry about how the app looks like. With the following designs, we were able to build the entire interface with React and created different components based on each different screen for our application. We will be explaining each screen and how the user is interacting with each one.

Login and Registration Screens

In Figure 38a, the user is able to have their own login information and password to keep track of their reserved spot. If the user is a first-time user, they are not able to login into the app yet. Therefore, they have to register themselves in the app.

On the next Figure 38b, the user is able to make an account if they have not created one yet. In this screen view, the user inputs general information such as name, school email, phone number and university name. It is important to collect the University ID # because we want to make sure the student does not use different accounts to get the best shot at reserving the spot. This way, it makes it fair for all students to have a solid chance.

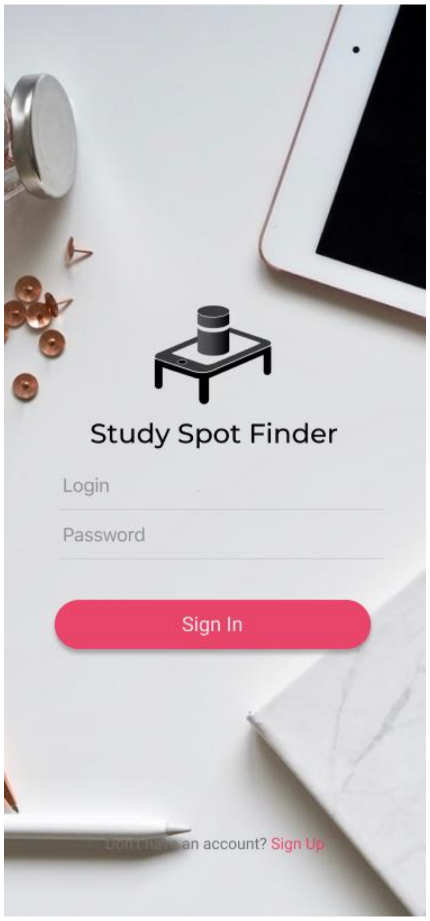


Figure 38a: Login Page

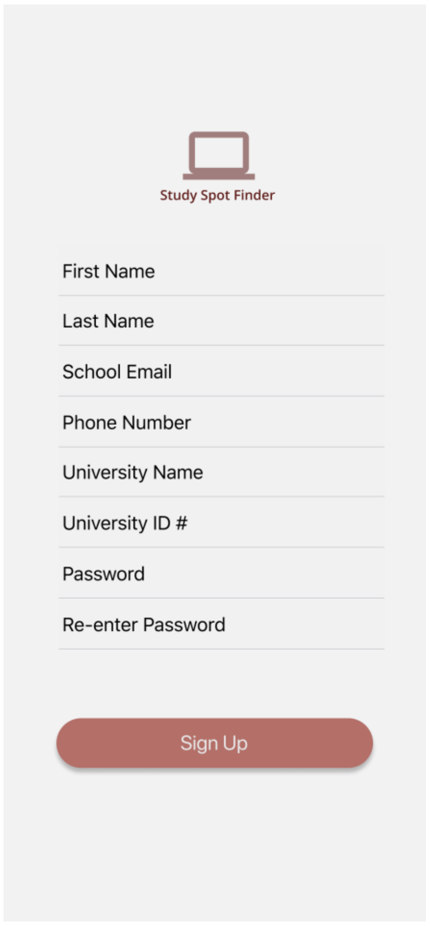


Figure 38b: User Creation Page

Login and Registration Flowchart

We illustrate this process with the following flowchart (Figure 39) as well. The user starts at the very top by entering the Login Page. Then, the user can either login to start using the app, reset their password and/or create a new account. After going to the login page, if the person forgets their password, we initiate the password reset, which sends out the email to the user with the link to change the password. The user changes the password, and the password gets updated and the password reset process is finished. However, if the user doesn’t have an account we prompt them to create an account. We ask them to fill out their information, and once that is finished, we ask them to login to the application.

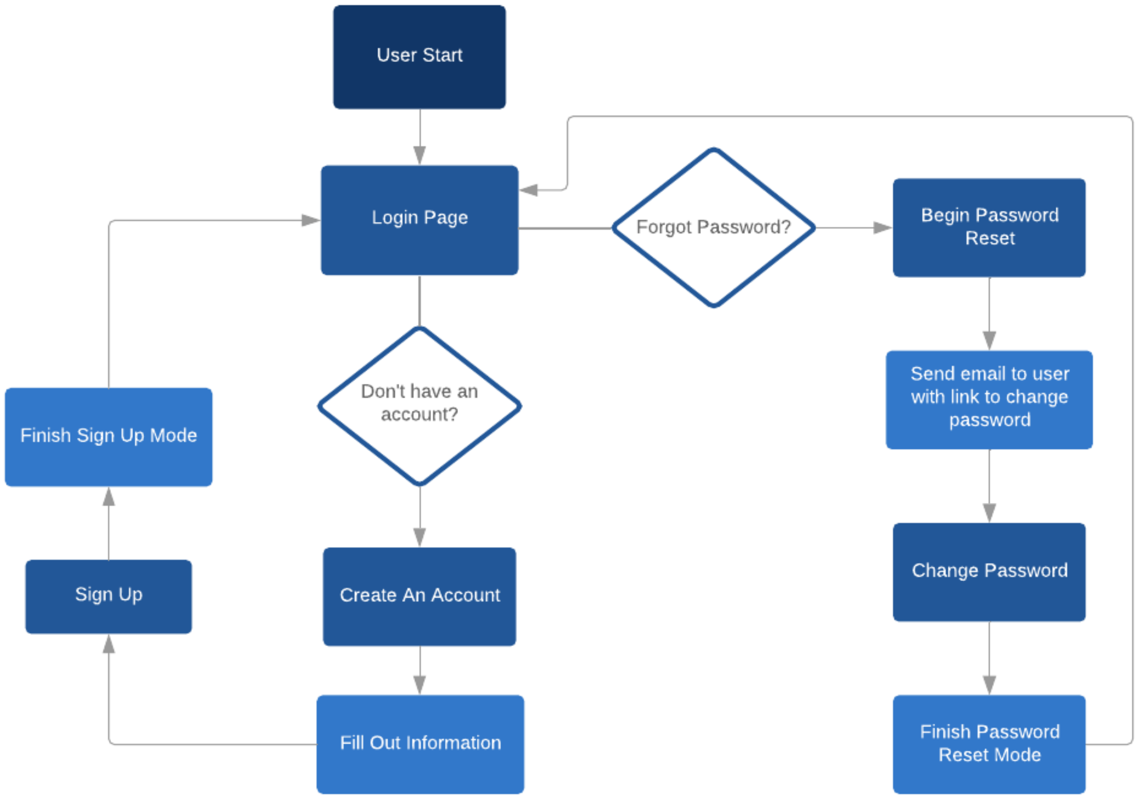
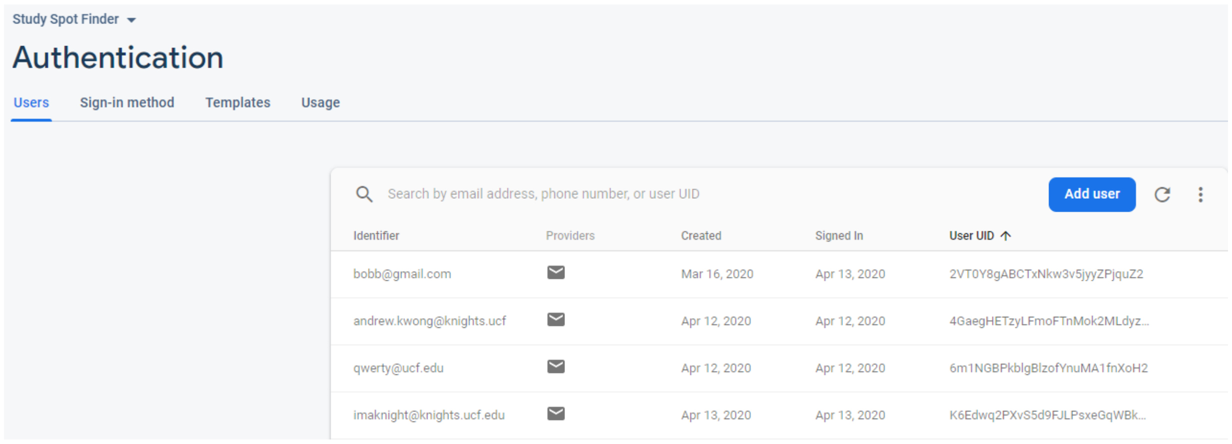


Figure 39a: User flow diagram when the user tries to login in

If the user attempts to create an account that has already been made in the system, Firebase shall check if there if a user has been created already by their email identifier, as shown below:



Identifier	Providers	Created	Signed In	User UID ↑
bobb@gmail.com	✉	Mar 16, 2020	Apr 13, 2020	2VT0Y8gABCTxNkw3v5jyyZPjquZ2
andrew.kwong@knights.ucf	✉	Apr 12, 2020	Apr 12, 2020	4GaegHETzyLFmoFTnMok2MLdyz...
qwerty@ucf.edu	✉	Apr 12, 2020	Apr 12, 2020	6m1NGBPkblgBlzofYnuMA1fnXoH2
imaknight@knights.ucf.edu	✉	Apr 13, 2020	Apr 13, 2020	K6Edwq2PXvS5d9FJLPsxeGqWBk...

Figure 39b: User flow diagram when the user tries to login in

5.2.3.3. Reserve A Study Spot

In order to reserve a study spot, the user must go through a series of steps in order to reserve, confirm, and use the spot. In this process, several users can reserve a spot as well. It is always important to keep accurate information for all available spots, since users are coming in and out to the app. The following screenshots illustrates the process on how one reservation can be made.

Available Spots and Spot Information Screens

In Figure 40a, the user can find all the available spots to pick from. In this screen, the user can search for available spots by putting the location they prefer, or just searching the name of the table if they know it already. The next part of this screen shows all the current available spots. These spots display basic information for the user to decide whether they want to click on it for more information or not.

The information includes:

- The name of the study spot
- The location (building) of the study spot
- The floor in which this study spot is located
- The keycode that will be generated by the user
- Symbols that gives an overview of what the study spot holds such as:
    - Group Icon to represent if the spot is for a group of people
    - Individual Icon to represent if the spot is for one single user
    - Outlet Icon to represent if the study spot has outlets or not
    - Desktop Icon to represent if the study spot has a desktop computer to use or not

In Figure 40b, the user clicked on a study spot to find out more information. Once the user clicks on this, it displays the actual number of outlets available and capacity if it is a study spot for a group.

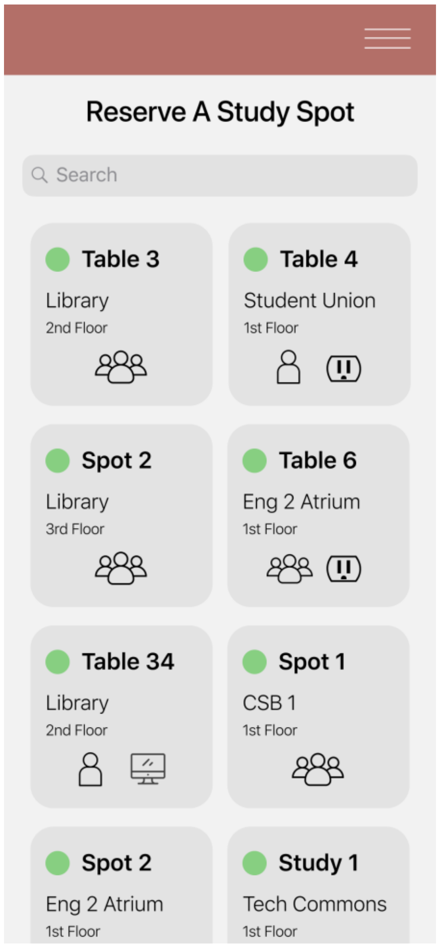


Figure 40a. Dashboard of App

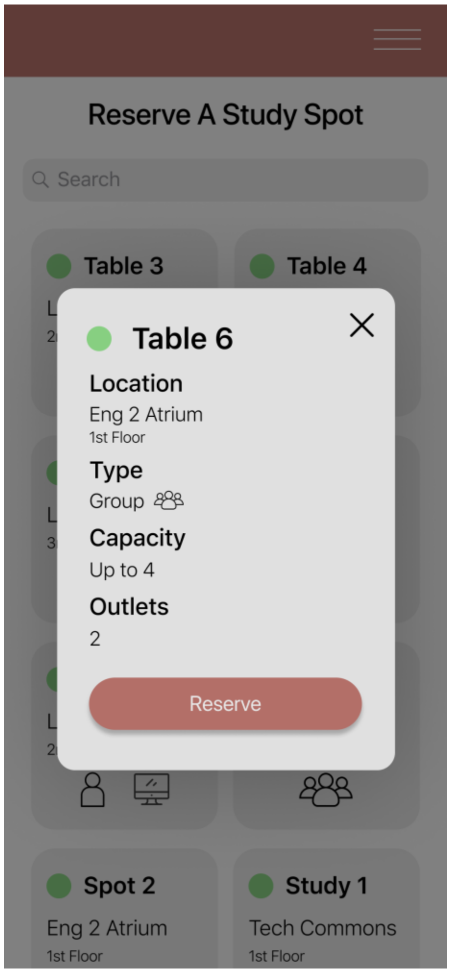


Figure 40b. Study Info Popup

As you can see in Figures 40a and 40b, the user can decide to reserve the spot by tapping on the “Reserve” button. If the user is not ready or does not like the spot, they can simply exit this screen by tapping on the X in the top right corner.

In our final application, we managed to implement this part with a different UX/UI design. We changed the theme colors and did not have a hamburger bar on the top. We also deleted the “Search” option since our application did not have as many study spots available in the database.

Loading and Reserved Screens

In figure 41a, the user makes a decision and decides to reserve the spot. The waiting time for the user is 10 seconds. This time is allocated because we need to ensure there are no other users trying to reserve a spot at the same time. This allows the device to update with the latest information and ensures the user is able to reserve the spot selected. The user also has the option to cancel this action in case they changed their minds in respect to this reservation.

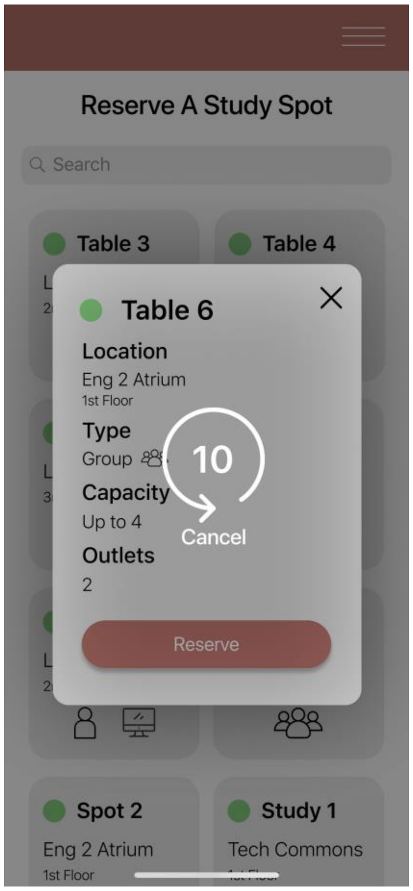


Figure 41a. Timer popup when the user clicks on an available spot

In Figure 41b, the user gets a “Success” pop up if the reservation is successful. If the reservation is not successful, another screen pops up saying that there was an error and that they should try to reserve a spot again. The “Success” pop up shows brief information on the spot and it also gives the user the generated code they must enter in the keypad of the device to confirm the spot. User have exactly 10 minutes to confirm this spot. This should be enough time to claim a spot since the purpose of our project is to provide real-time available spots - that is, the user should be at the university in order to claim a spot.

In our final application, due to time constraints and prioritizing, we decided not to include this waiting time for the user and the 10-minute time limit for the user to get to the spot. These features are not essential to the function of the application, and only serve to either press the user to claim the spot or give the device time to confirm the reservation when it checks the database. Instead, there is a popup that appears once the reservation has been placed, which asks the user to manually refresh the main page. This allows for the status to be updated as reserved while the device updated from the database as well. Once both the application and the device recognized this reservation, they will both wait until the user has arrived and performed the reservation validation, which both platforms update to reflect this change.

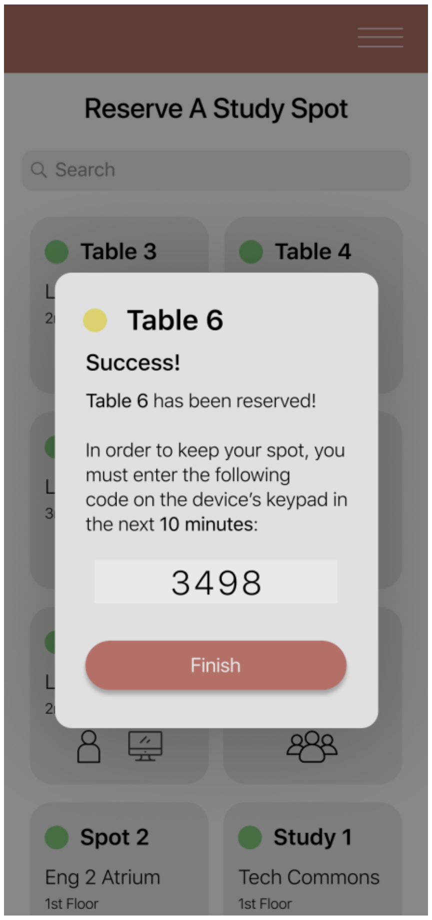


Table 6

Success!

Table 6 has been reserved!

In order to keep your spot, you must enter the following code on the device's keypad in the next 10 minutes:

3498

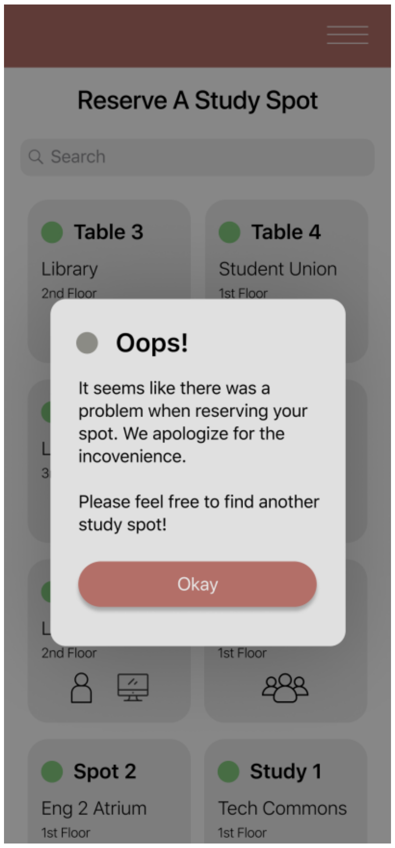
Figure 41b. Successful table reservation from the mobile app

Once the user gets the Figure 41b screen, then they are able to tap on the "Finish" button and proceed to the next step of this user journey. At this point, the user is responsible for confirming the spot by going physically to the spot and clicking the "Confirm" button after inputting the generated code shown in the "Success" page.

Error Screen

As mentioned in the Loading and Reserved screen sections, there could be a possibility that multiple users might be trying to reserve a study spot at the same time. For this, we were able to come up with a process that allows the hardware device to get back to us with the right information before proceeding to reserve. This solution gives 10 seconds for the user to reserve the spot. In this time, the device should update with the right information and also the back end should be aligned that there were no changes made before this reservation was taking place.

In Figure 42, you can see an example of how this error message appear for the user. It is simply letting the user know that there was a problem reserving the spot and that they should try reserving one again.



It seems like there was a problem when reserving your spot. We apologize for the incovenience.

Please feel free to find another study spot!

Figure 42. Error when user tries to reserve a spot

Reserving a Spot Flowchart

For the flowchart diagram (Figure 43) below we start with the user login and we first check if the student is registered (the main user). If it is not, then we make the user create an account. If the student is registered, then we prompt them to the main dashboard to look for a study spot. In this area, the user selects the spot they prefer. If they do not select anything, then just stay in the same dashboard showing other spots.

Now, if the user reserves the spot, then they must wait for 10 seconds to reserve a spot. They can also choose to not reserve the spot by clicking the X button and coming back to the dashboard again. If they decide to wait for 10 seconds, the user has 2 options: they can either wait till the spot is reserved or click on cancel. If they click on cancel, then they just come back to the dashboard again. If they decide to wait, then they have a successful page follow-up. If the request was not successful, it displays an error message and prompt them to the dashboard again to find another spot.

However, as mentioned above, time restrictions and prioritization has brought us to move forward without this feature, and simply allow users to arrive at their destination as soon as possible.

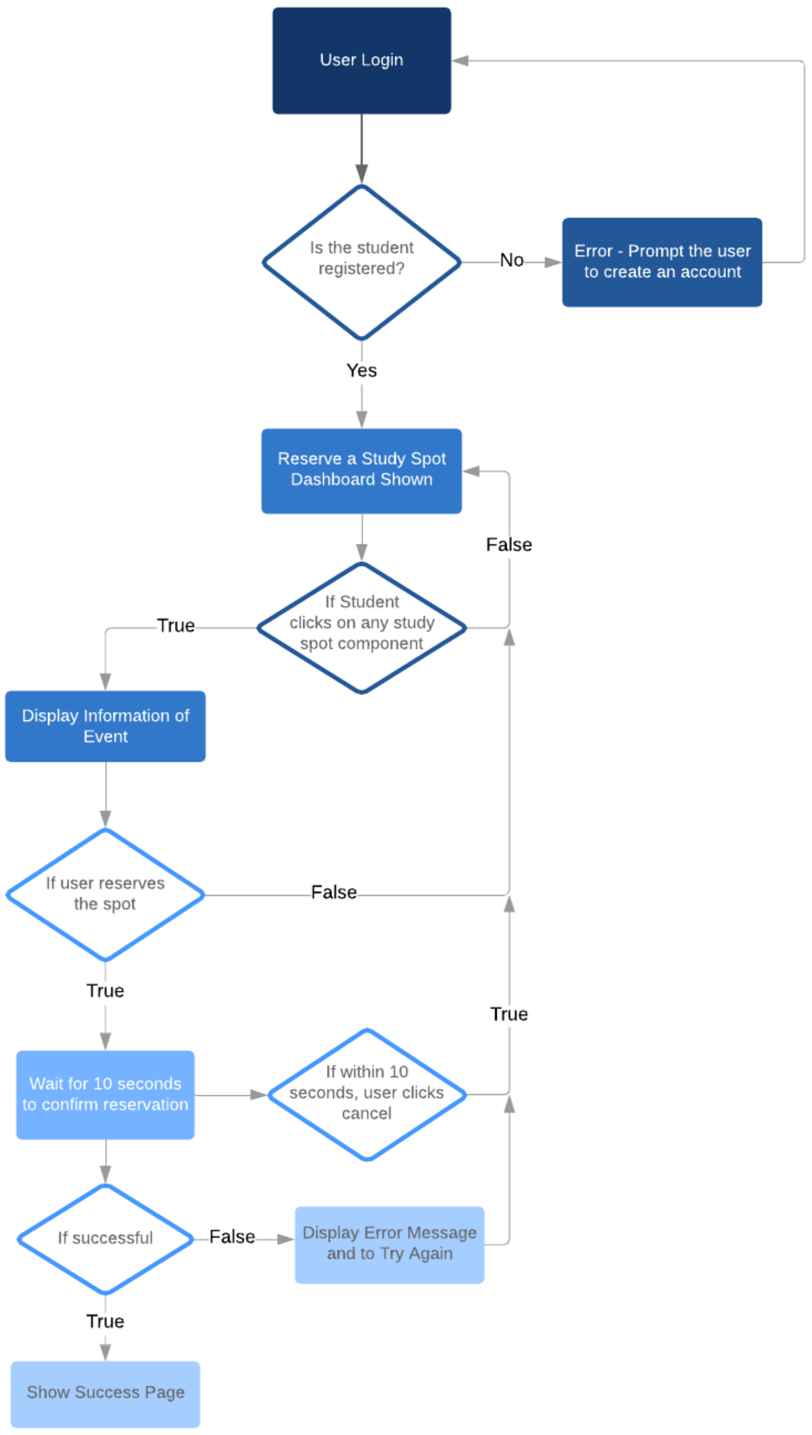
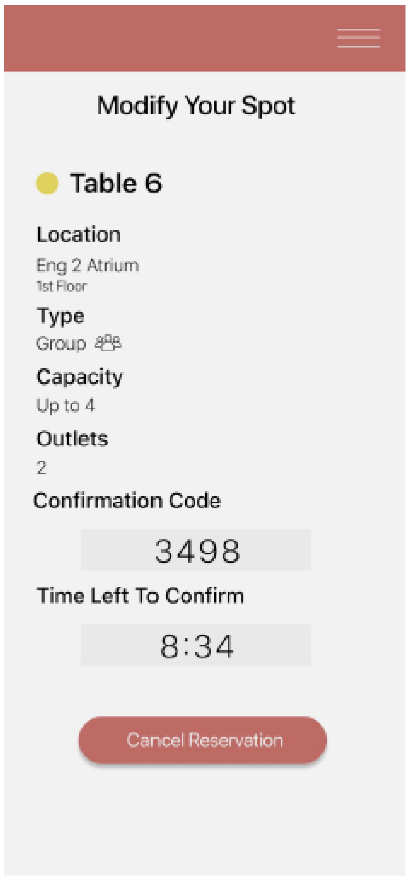

User Login
Is the student registered?
No
Error - Prompt the user to create an account
Yes
Reserve a Study Spot Dashboard Shown
False
If Student clicks on any study spot component
True
Display Information of Event
If user reserves the spot
False
True
True
Wait for 10 seconds to confirm reservation
If within 10 seconds, user clicks cancel
If successful
False
Display Error Message and to Try Again
True
Show Success Page

Figure 43. User flow diagram when they reserve a spot

Reserved and Confirmed Spot Screens


Modify Your Spot

● Table 6

Location
Eng 2 Atrium
1st Floor

Type
Group

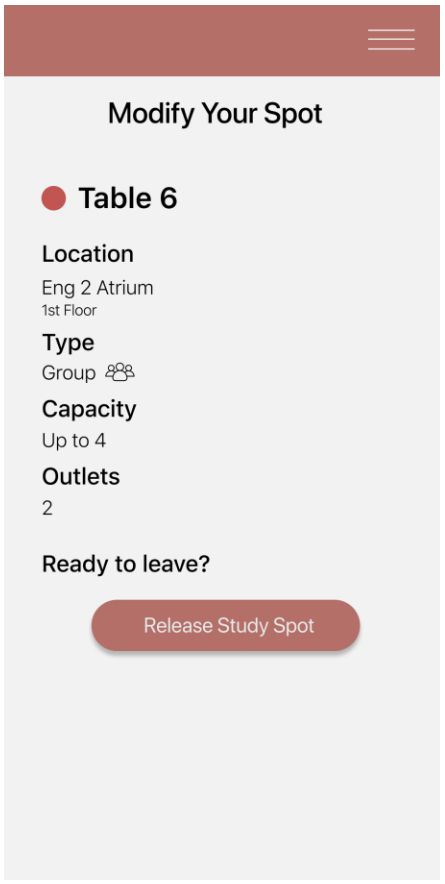
Capacity
Up to 4

Outlets
2

Confirmation Code
3498

Time Left To Confirm
8:34

Figure 44a. Spot Reserved


Modify Your Spot

● Table 6

Location
Eng 2 Atrium
1st Floor

Type
Group

Capacity
Up to 4

Outlets
2

Ready to leave?

Figure 44b. Spot Officially Confirmed

In Figure 44a, the user can modify the study spot. This screen shows the same information as the one displayed when the user selected the spot in the Available Spots screen. In addition, we put the information of the confirmation code here in case the user does not remember it. This helps the user find out how much time they have left before the spot is released for someone else to use. In this screen, the user has the option to also cancel the reservation in case they do not want it anymore.

In Figure 44b, the user already managed to get the physical study spot and entered the code in the device's keypad. This screen is to show that the study spot is confirmed for the user. The user also has the option to release the study spot when they are ready to leave. All these color changes are also being reflected in the actual device using an LED.

This is the end of the user journey regarding reserving a spot. Again, a user can reserve only one spot at a time. When it is a group, the one reserving the spot is essentially the group leader and they should always remember to release the study spot when they are done. This applies to individuals as well when it comes to releasing a study spot.

In our final application, we included the screens above into the first pop-up for easy access. Since we did not have time to include the time limit, we did not have a need for having a separate page for it. We end up combining the “Cancel Study Spot” button and “Release Study Spot” button into one single button called “Release Spot”. This gave the option to the user to modify their reservation on the same pop up. We thought this may be easier for the user since they would not have to go back and forth.

Reserved and Confirm Flowchart

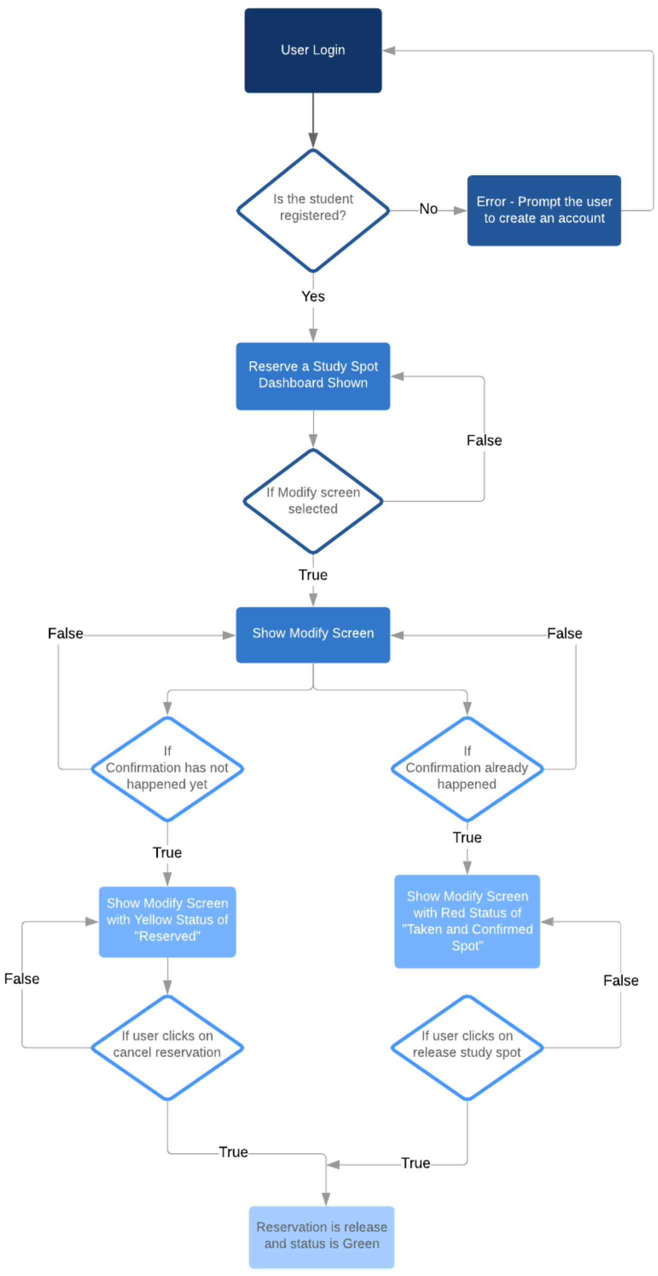

User Login
Is the student registered?
No
Error - Prompt the user to create an account
Yes
Reserve a Study Spot Dashboard Shown
False
If Modify screen selected
True
Show Modify Screen
False
False
If Confirmation has not happened yet
If Confirmation already happened
True
True
Show Modify Screen with Yellow Status of "Reserved"
Show Modify Screen with Red Status of "Taken and Confirmed Spot"
False
False
If user clicks on cancel reservation
If user clicks on release study spot
True
True
Reservation is release and status is Green

Figure 45. Logical flow diagram for the user application

Figure 45 above illustrates the user login system for the Study Spot finder and checks if the user is registered or not. If the student is not registered, we prompt for the user to create the user to create a new account. We then have the user logged in to the main dashboard, and we give them the option to reserve a study spot with two options - one to show or not show the modify screen. Once the modify screen is shown, we check for one of two scenarios: if the confirmation has already happened or not happened yet.

If the confirmation has already happened, we show to the user the modify screen with a status of taken and a confirmed spot. On the other hand, if the spot is unconfirmed, it gives two possible scenarios: if the user clicks on canceling the reservation, the reservation is released and the status is green, while if not, the yellow status still pertains. If the user clicks on release study spot - the status turns green, and if not, the states does not change to green and persists to red as the status that is already taken.

5.2.3.4. Admin Interface

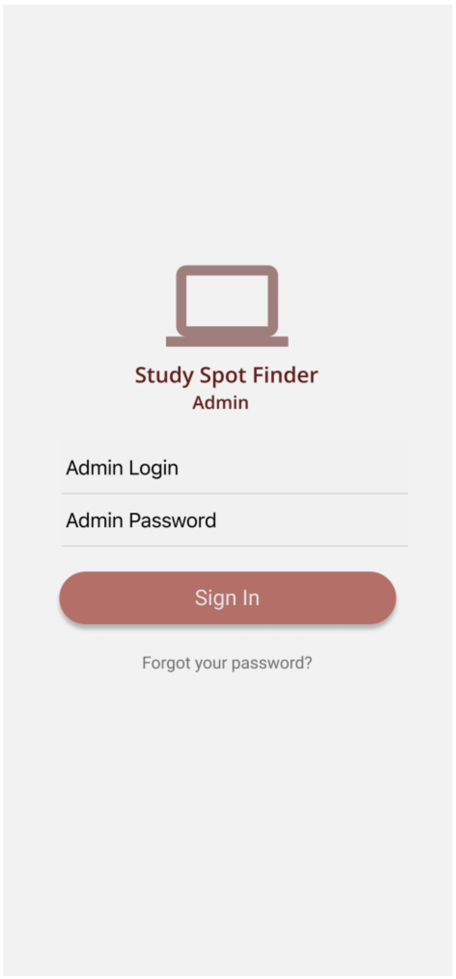


Figure 46a. Admin sign-in page

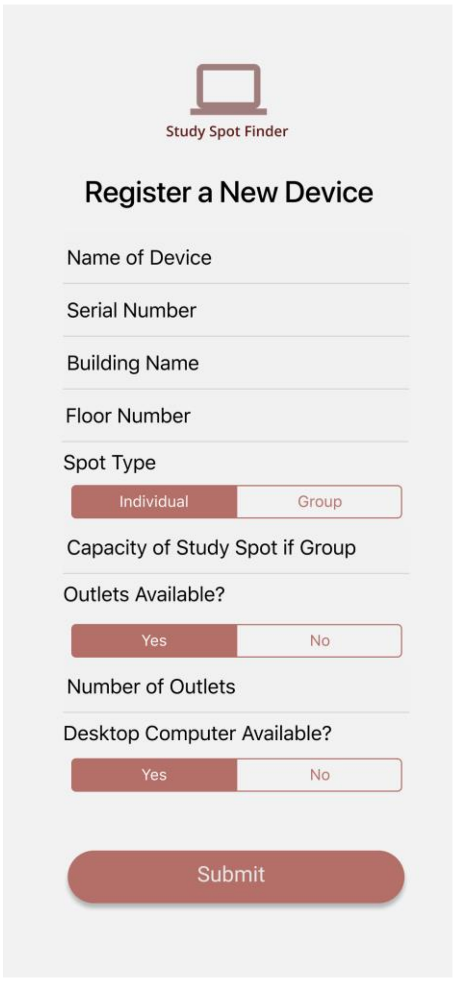


Figure 46b. Admin adding a Finder

For the admin, there is a different area in which they can register a new device. There is no way for the admin to make an account, but it is under the assumption that admins will be the IT department employees, and credentials are given to them to have special privileges when registering a device. Admins have a different sign-in page than the users.

Admin Login Screen

As Figure 46a shows, the admin are able to login with their own credentials that are provided to them since admins could be IT Department employees. No other user are able to access this device except for the admins. A generated password is given to them to sign into their account, once they take the first step, they can change their password so it is more secure.

Register a New Device Screen

For the following sections, we were looking into the different fields to register a new device. This action can only be done by the admin.

Name of Device
In Figure 46b, the admin is able to register a new device. This screen walks the admin through the steps of setting up a new device for the study spot. First, admin must enter the name of the device. This could get a bit tricky depending on how they want to show these names to the users. Examples include “Table 1 - Student Union 2nd Floor” or “Spot 43 - Engineering 2 Atrium”. This part depends entirely on how the university wants to structure their devices based on the infrastructure of the buildings and floors.

Serial Number Field
The next step is adding a serial number for the device. Each of our devices come with a serial number ready to be entered. This number makes each of our devices unique.

Location/Specific Area Field
For the next step, the admin must enter the location/specific area. Again, this is completely up to how the university wants to set this up. Part of our desirables include a map of where the study spot would be; therefore, we would have to be more cautious with the location if that was the case. Since we are not doing a map of the study spots for now, then we are leaving this to the university regulations.

Outlets Fields
Next in the list we had two fields for outlets. We also liked to consider the fact that students might want a study spot that might have outlets for their personal computers. That is, we cannot assume that every student needs an outlet either. We give this option for the admin to choose if the study spot where the device is being installed has outlets or not. This information appears for the user when they are selecting a spot.

Desktop Computers Fields
Following up from information being shown to the user, we also want to inform them if the study spot has desktop computers. We also have added what programs installed are in the desktop computers and what operating system these computers run.

Capacity Field
Part of our mission is to offer a study spot for every type of college student. These study spots can be for individuals or for groups. For the next field, the admin can enter the capacity for this spot. If it is an individual space, then the admin can enter 1. If the table is enough for four people, then they can add 4 in this field.

One more factor we considered is the fact that not all study spots might always have chairs next to the table. Sometimes students can take chairs from other tables to add them to their own table. Admins have the ability to enter the capacity they think might fit best. For example, if the chairs are not attached, then they can enter the capacity to be “up to 4 people”.

Modify Device Screen
The admin is able to modify the device whenever they think it is necessary. In screen Figure 47a, the admin has a similar interface as the user/student in the sections above.

In this case, the admin is not able to reserve spots but they are able to click on a spot to see more information and start the “Modifying” process. The user is able to tap on the Modify button and it takes them to the next screen.

The next screen is Figure 47b. In this screen, the admin is able to update any information on the device. This screen is never presented to the user/student, only an admin with special privileges is able to make these changes.

They also have the opportunity to cancel this request in case there is no more need to modify the device. Finally, they are able to delete the device as well in case it is not needed anymore.

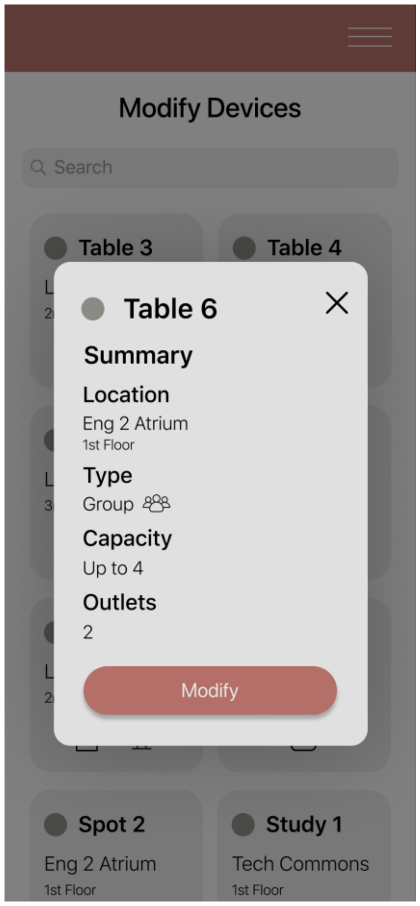

Modify Devices

Table 6

Summary

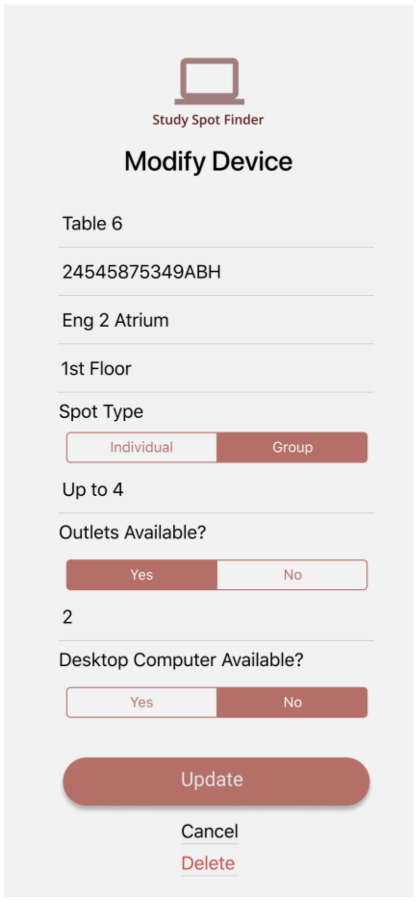
Location
Eng 2 Atrium
1st Floor

Type
Group

Capacity
Up to 4

Outlets
2

Figure 47a. Admin Spot Popup


Study Spot Finder

Modify Device

Table 6

24545875349ABH

Eng 2 Atrium

1st Floor

Spot Type

Individual [✓ Group]

Up to 4

Outlets Available?

[✓ Yes] No

2

Desktop Computer Available?

Yes [✓ No]

Figure 47b. Modify Properties

In our final admin platform, we decided to have all the fields to be text fields. In the image above, we can see that the admin can make a few selections such as Spot Type and Outlets Available. We decided to make it easier for the admin to create/modify a spot by making it simple text fields they can write on.

5.2.4. Back-End

To be able to handle the status of different devices and communication between the user and the device, a database and API(s) were used.

Google Firebase were used as the database. This software tool allowed us to handle data requests/process in real-time, using JSON format as the middle communication between the database and the device. In addition, Firebase has cloud functionality, in which changes could be made to the database in real-time. Firebase also works optimally with React, which is the main technology being used for the front-end.

Figure 48 shows how the database interacts with the microcontroller and with the front-end application:

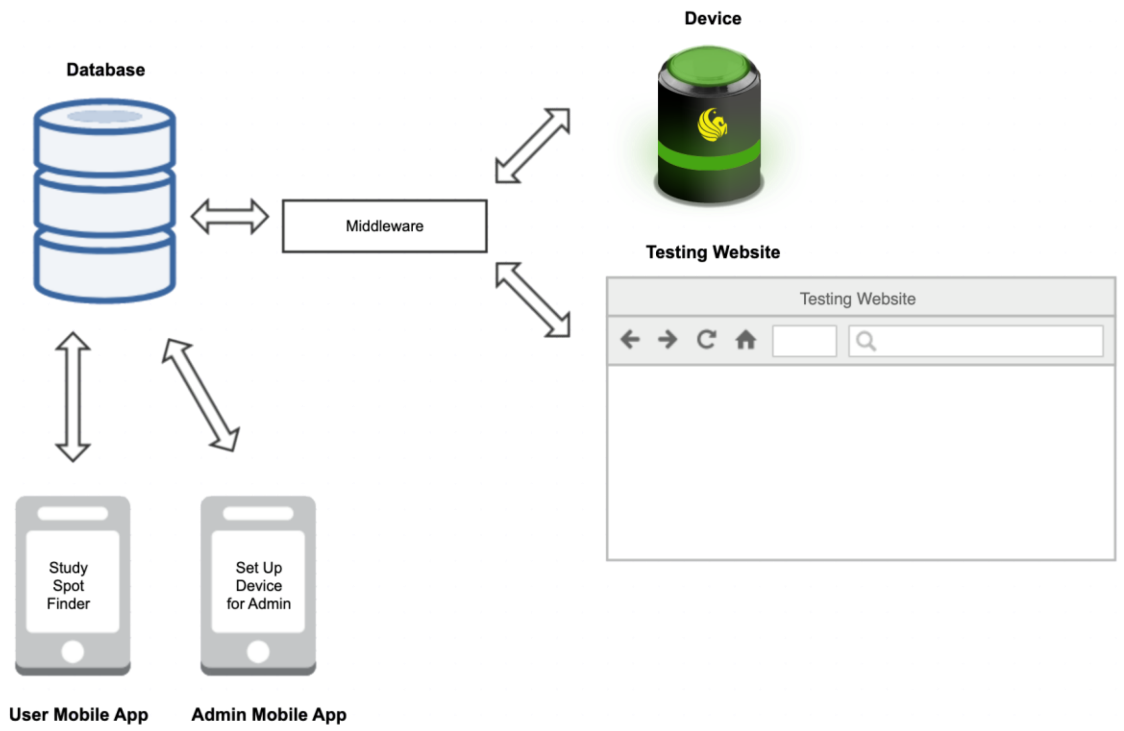


Figure 48: Back-End Diagram

As seen in the diagram above, Firebase interacts with the microcontroller using a middleware through an API (Application Package Interface) interface. The ESP 8266 Wi-Fi module is strongly being considered for it to relay the data to the Firebase, as well as any API support.log

5.2.4.1. Database

For the Database, we used Firebase made by Google. We used Firebase in order to communicate with all our platforms.

Entity Relationship Diagram

In order to understand how we are keeping track of all users, admins, and devices' information, we needed to create an Entity Relationship Diagram. This diagram allowed us to create different tables in the database. This way, we can ensure we are keeping track of all records and the application are functioning logically.

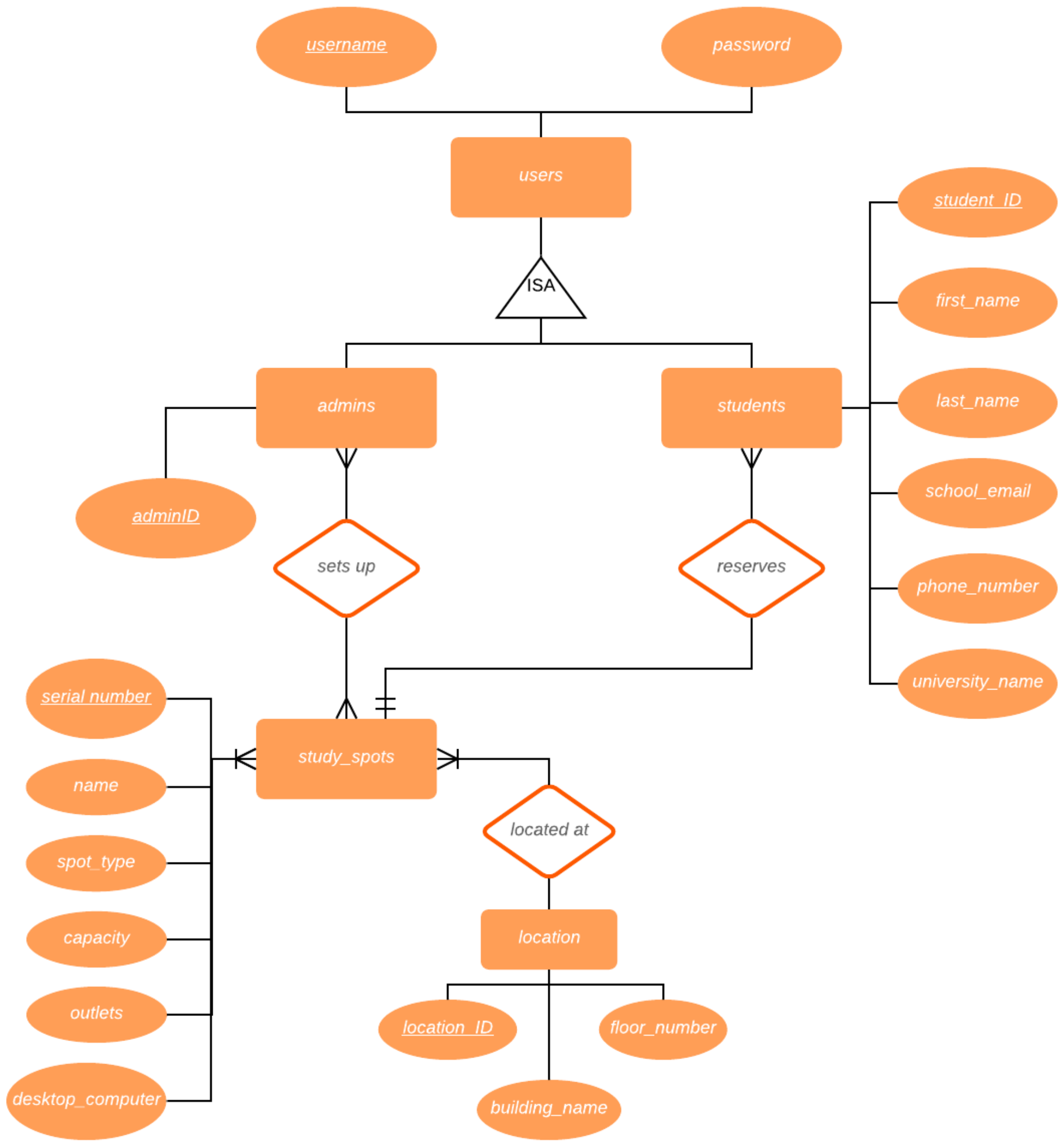


Figure 49. Entity Relationship Diagram of Study Spots, Students, and Users

In Figure 49, we can look at how the database stores information from all users. Admins have a username and password as well, but they do not have a profile like students do. Admins' mission is to set up the study spots with all the relevant information that might be beneficial for the student. The location was separated in case we had the time to include our map desirable feature. This way, we would be able to use the location and use (for example) google maps to provide a location for the study spot. This becomes accurate when it comes to latitude and longitude, which is what we would use if we were to provide a map to the student.

In our final application, we add the following fields to Study Spots:

- Status
- Change serial number to device ID name
- Location
- Floor

We delete the location entity since we are not using a map at the end of the day due to time constraints. We decided to merge these fields into one and have them all under Study Spots.

The database holds different values to keep track of different states within the application. Table 42 shows the descriptions of the values you can see in the Figure 49.

Values	Description
Sign-In Information	Database holds the usernames and hashed passwords from each individual user. A user can be a student, but a user cannot be a super admin.
Super Admin	Super admins have their own login credentials. They cannot register as if they were a student. Super admins are given their credentials and the option to change their passwords once they login.
Status	For the user of reservation system, we have a status field per device. Each device has their own statuses and based on the interaction from the user with the device, this status field changes. It sends new information to the device for it to use and collects information from the application.
Devices	The main part of our project is to hold all the devices we have, assuming we have a lot of them. Since we were not be able to create several hardware devices, we had the database populated with “devices” that serves the purpose of simulating the hardware devices.

Table 42: Table of Values

5.2.4.2. Middleware (API)

The middleware (API) is essentially the area in which we handled the requests from the user, from the device, and from the database. With this area, we are constantly asking the database to give us the most updated information or we are sending the most updated information to the user. These are called API calls. They are in charge of communication among all entities within the application.

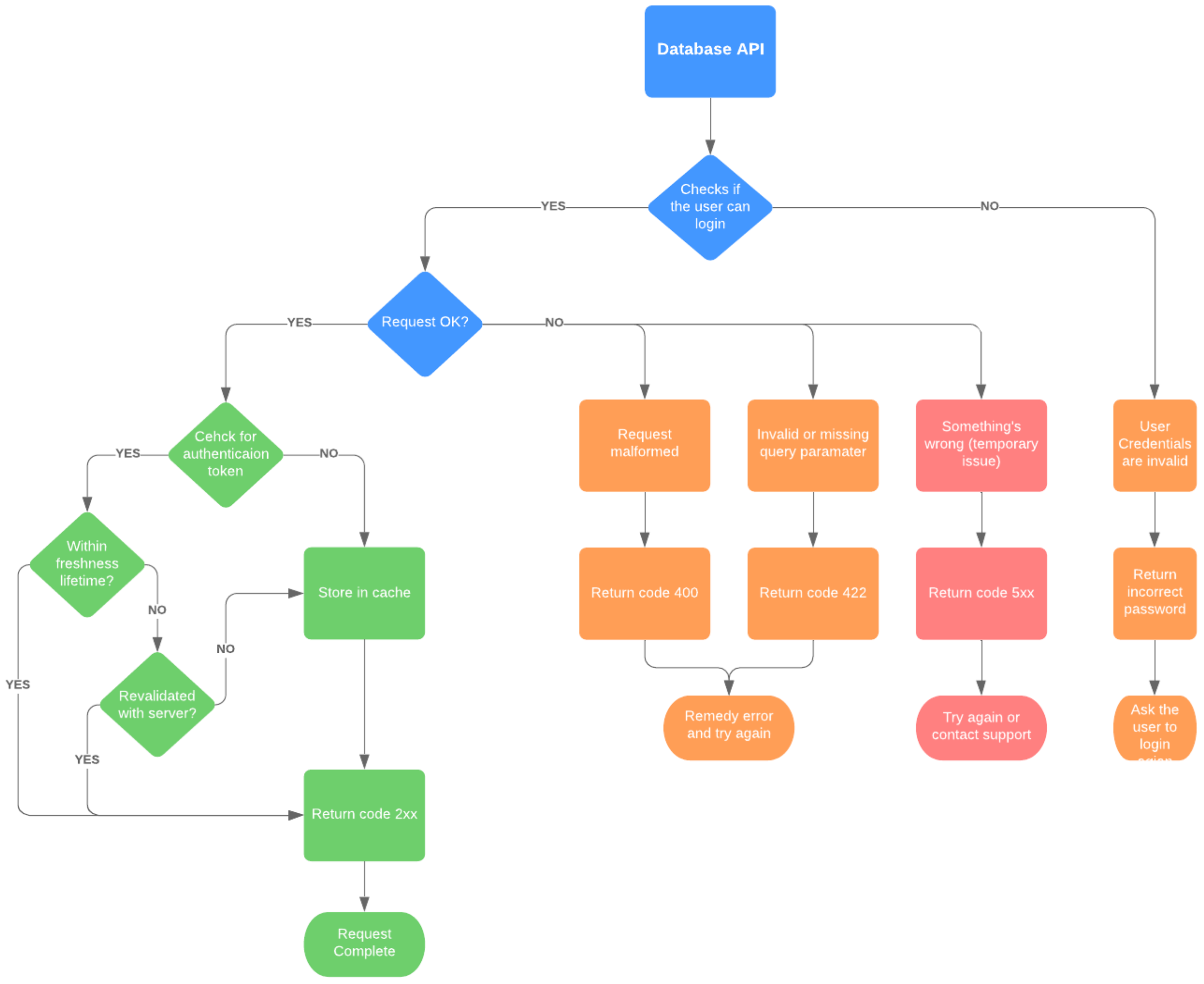

Database API
Checks if the user can login
YES
NO
Request OK?
YES
NO
Cehck for authenticaion token
YES
NO
Within freshness lifetime?
NO
YES
Revalidated with server?
NO
YES
Store in cache
Return code 2xx
Request Complete
Request malformed
Invalid or missing query paramater
Something's wrong (temporary issue)
User Credentials are invalid
Return code 400
Return code 422
Return code 5xx
Return incorrect password
Remedy error and try again
Try again or contact support
Ask the user to login

Figure 50. API Diagram

In Figure 50 above, this checks for user authentication on the mobile application - if they cannot, it redirects them to enter their password again. However if it a success, then it validates their request - if that request is improper - it can happen in one of three ways: a bad request, a missing parameter, or some random unknown issue. For the first way - if the request is malformed, it returns a 400 error code and we specify the user to retry inputting their password again. If we get a query parameter issue, it means the hash in the authentication got corrupted, and the temporary solution was to inform the user to reenter their password again - and posts a return code of 422 to the server. If there was an unknown issue that happened during login (sometimes there are issues with servers that we do not know about) we also post an error code of 5XX, and ask them to try again.

However, if the request is okay, we check for the if the authentication token is generated during the authentication process. If one is not generated, we store the credentials in cache or the database in this case (not recommended), as it is in plaintext, leading to possibilities of attack, and returns a return code of 2XX, and the authentication process is complete. If one is generated, it checks the lifecycle of the token, and if the lifecycle is expired, store the credentials in the database, and do the same as before, however, if the lifecycle is still active, post the return 2XX code and the authentication process is over, letting the user in the system.

We are using RESTFul API or Representative State Transfer, in which the server transfers to the client a representation of the state of the requested resource. The state is in JSON format. Some of the most common HTTP Format is GET, POST, PUT, and DELETE, which is described below.

Study Spot Admin Page

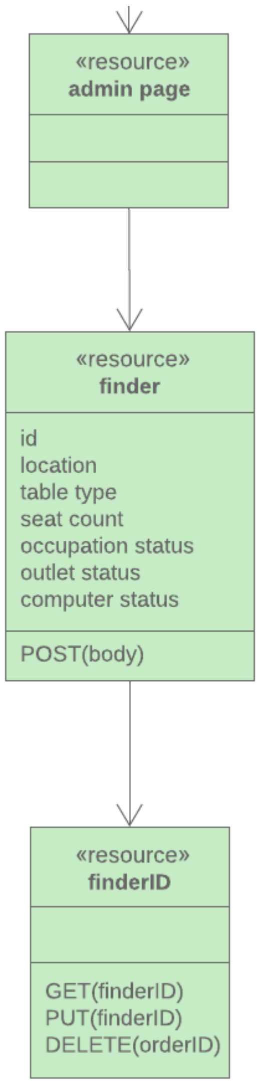


Figure 51: API diagram of the admin page

Username Diagram

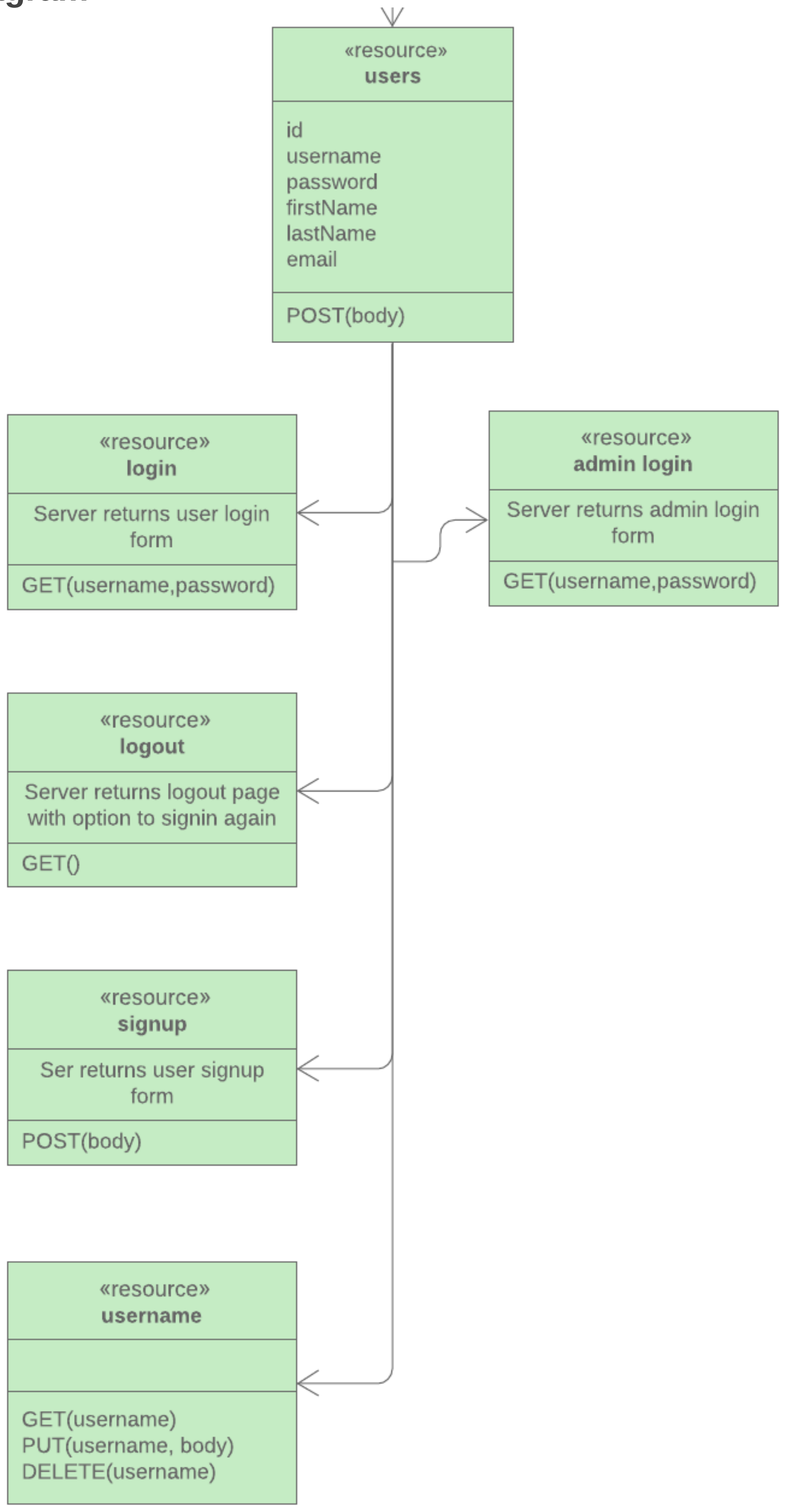


Figure 52: API diagram of the user login system

Main Logic (Steps) - User

Step #	User Action
1	When the user sees what spots are currently available, the app is fetching the data (devices with green statuses) from the database to show on the main screen what devices are currently ready to use
2	As soon as the user presses the “reserve” button, the system will take approximately 5 seconds to ensure the hardware device is ready to change the color status from green to yellow. The database is updated accordingly for this device and change the status now to yellow
3	The database communicates with the device to change the color status and reflect this recent change
4	The device waist for 10 minutes for the user to press the “confirm” button
5	Once the “confirm” button is pressed, this data sends back to the database so it can update the status from yellow to red

Table 43: User Steps for Reservation

Table 43 demonstrates the User Steps for Reservation, and how it relates to Firebase. Since these changes are being done in real-time, other users would expect to be able to claim a spot if the spot has not been confirmed by the correct user yet. After 10 minutes, the device turns green again and the status is reflected on the database for the next user to use.

6. Testing

In general, in order to ensure the software and hardware are working properly, we needed to look into testing each area individually. There are different ways to test software and hardware, but the main part was being able to connect them both together in order to make the whole project work. In the next sections, we will be providing in detail what was needed in order to ensure our project was delivered in time.

6.1. Hardware Testing

In this section, required and necessary test cases are presented in order to demonstrate the full functionality of the hardware section of the project and integration with software. Most of the hardware components were tested separately in order to verify their proper functionality, then assembled and tested again to verify their compatibility. If the device gives any issues as far as the hardware is concerned, these tests provided a useful framework to decompose the source of the problem.

6.1.1. Microcontroller Testing

The microcontroller oversees reading and interpreting data from the reservation system database, as well as reading inputs from the buttons, keypads, and sensors in order to confirm if a spot has been occupied or not. The microcontroller tests included:

Stand-Alone microcontroller circuit

This test verified that the microcontroller was not shorted internally, in order to ensure that turning on the microcontroller did not damage any external parts or cause further damage to the microcontroller. This was accomplished using a multimeter, by performing a resistance test between the GPIO pins and verifying that the resistance is >1 MΩ

Voltage level

For the voltage level test, we simply measured the level of the active GPIOs to verify that the correct voltage necessary to run the device was being supplied. This was done using a multimeter. The pass criteria for this test would be that the voltage of the GPIO meets or exceeds 3.3V.

Current level

The current level test verified that the external devices are functioning correctly and pulling the correct current from the Microcontroller. This was done with a multimeter measured in series. The probes were inserted before the power was turned on, and the microcontroller circuit test was done prior to this step.

Basic imported code

For this test we tested the basic functionality of the microcontroller without any additional components attached to it. For this test, a basic ‘hello world’ was conducted with an LED. The LED was tested on a simple resistor circuit to verify that it works. The ATMega328P was placed on the development board and a test program was flashed onto the board. This code turned on the LED, waited 2 seconds, and then turned off the LED.

This test verifies that the GPIO Pins are working and that it can connect to and receive code from a serial port. It also verified that the correct voltage level is being applied. This test was repeated for any GPIO pin assumed to be faulty. Here the pass criteria would be that the LED turns on and off with a 2 second delay.

The microcontroller circuit was tested in different ways to ensure the accuracy of the circuit design. Moreover, during this section the performance of the voltmeter and ammeter which demonstrated the efficient performance of the device. However, the MSP430G2ET did not pass testing with the firebase, thus we decided to switch to the Atmega328P. We performed the same test cases with this chip and we concluded that this had a satisfactory performance in current draws and firebase connection.

6.1.2. Wi-Fi Module Testing

The Wi-Fi module testing was performed by sending AT commands to the ESP8266 and see if it responded as expected. These commands were sent from the terminal to the module. The Arduino UNO development board was used to connect this Wi-Fi module to the PC. Another option was to use a USB serial cable which can be sent from any serial terminal program. To achieve this testing, we used the following settings for the serial terminal:

- NL & Cr: send a newline and carriage return char at the end of command
- baud rate needs to be set to 115200.

6.1.3. Battery Testing

The rechargeable Lithium Ion battery is the only power source for all the hardware components required for this project. The battery was tested by using a multimeter to read the voltage when it is fully charged and discharged. The results obtained from this test should be similar to Table 44 below.

Measurement	Expected Value
Fully discharged	3.0V
Fully charged	4.2V

Table 44: Battery Voltage Levels

We ended up changing the battery for the final design, so our expected values shifted to 8.2V and 9.0V for discharged and charged, respectively.

6.1.4. Push Button Testing

Component Testing

In order to test the push button, a simple testing was done in the beginning. We used LED in order to perform a test case. The LED was attached to a simple circuit with a reasonable power supply and a properly rated resistor. The push button acted as the switch which completed the circuit and light the LED. The pass criteria was that the LED lights up whenever the button is pressed.

Integrated Testing

Once the button was integrated into the microcontroller, there was a test program that verified that the microcontroller is recognizing the button press. This was necessary to ensure that any failures are not being caused by the inability of the microcontroller to recognize that the reservation button has not been pressed. This test was completed after the previous test in order to rule out the possibility that the button is faulty.

6.1.5. Keypad Testing

Component Testing
The keypad was tested by using a multimeter to measure continuity. The common node was measured for continuity between each port that corresponds to a single button. The pass criteria was that each button closes the circuit to the common node for its particular port.

Integrated Testing
We connected the keypad to the corresponding pins on the ATMega328P and flashed the Keyboard Test program. This program verified that the ATMega328P can recognize button presses and differentiate between the different keys pressed.

Functional Testing (Keypad and Button)
The Code Verification Test program also allowed us to test how the MSP430, in this case the ATMega328P, handled the inputs and verified it with a given 4 digit code. It was able to keep the last 4 values input into the keypad for verification purposes and compare those with the given stored value once the push button is pressed.

6.1.6. Voltage Regulator Testing

Component Testing
The voltage regulator was tested in order to get the desired output voltage and current with appropriate tolerances. We tested the voltage regulator circuit on its own. It was attached to a power supply and the other end was attached to a load. The power supply swept in value from 12V DC to 0V to measure the dropout voltage and whether regulation was occurring outside of those ranges.

Integrated Testing
The voltage regulator also needed to be tested with a load to see if it can handle the power draw necessary from all of the components. The circuit was assembled as it is in the final design and each device performed its full functionality. Then the voltage regulator was monitored with a DMM to make sure that it can sustain the current draw from the devices.

6.1.7. Switch Button Testing

Component Testing
Similar to the push button testing, we performed a simple test using LED in order to verify the function of the push button. We used the battery as a power supply without voltage regulator or microcontroller.

Integrated Testing
Another test was using the switch in order to control the microcontroller since the switch button was used for turning on and off the device. The switch button was connected to

the microcontroller and be toggled in order to check if power was sent to the microcontroller.

6.1.8. LED Testing

Component Testing

In order to verify that we have a functional LED, it had to be tested separately. We used a simple circuit consisting of a power supply and a proper resistor value. Since this LED can change colors, it has multiple inputs, each of which require different currents. We took this into consideration as we tested each input and verified that all the colors are working.

PCB Board Testing

To verify the PCB board was set up correctly, we used a multimeter to verify that the jumper wire header pins were connected to their respective cathode pin on the LED pinhole. The anode of each LED was connected to the common ground wire of the jumper wire header pins. The resistance of each resistor was verified as well before connecting the LEDs in order to prevent damage. Table 45 illustrates this test step

Steps	Description
1	Use a multimeter to read the resistances to verify that they are correct
2	Turn the multimeter dial to diode setting and connect it to a breadboard and connect the red and black probe to the proper location to ensure the LEDs are functional
3	Evaluate the brightness of the LED

Table 45: LED Testing

6.2. List of Test Programs

Table 46 lists the programs that were created to flash onto the Microcontroller whenever a debug was necessary. These programs singled out a specific component to test to verify that is working. They can be used in the overall main program that was flashed into the final version of our device, but kept separate to use as needed.

Part to Test	Name	Function
Microcontroller	Hello World	Blinks an onboard LED on a timed delay in order to verify that the microcontroller can receive and execute code.
GPIO	Hello World 2	Blinks an external LED on a proto-board circuit, to verify that the pins function correctly
Push Button	Button Test	Triggers an interrupt when the button is pressed and toggle the on board LED. Should contain preventative measures for button debouncing.
Keyboard	Keyboard Test	Reads the inputs from the keypad and transmit the received buttons to a terminal app. Should properly decipher between each button
KeyBoard/Push Button/MSP430	Code Verification Test	Reads and stores the last 4 inputs given before the confirm button is pressed. Compares the final value to the given stored value. Lights up green LED if correct, red LED if incorrect

Table 46: Test Programs for Hardware

7. Software Testing

In order to test how our mobile app was responding to the device, we needed to have multiple devices available. Since there were only 1-2 hardware devices being built, there was no way for the software side to figure out if the app was working properly. This was a real-time application and the database was expecting to have multiple devices available.

To solve this issue, we had a testing website which would simulate how the user would interact with our hardware device in real life:

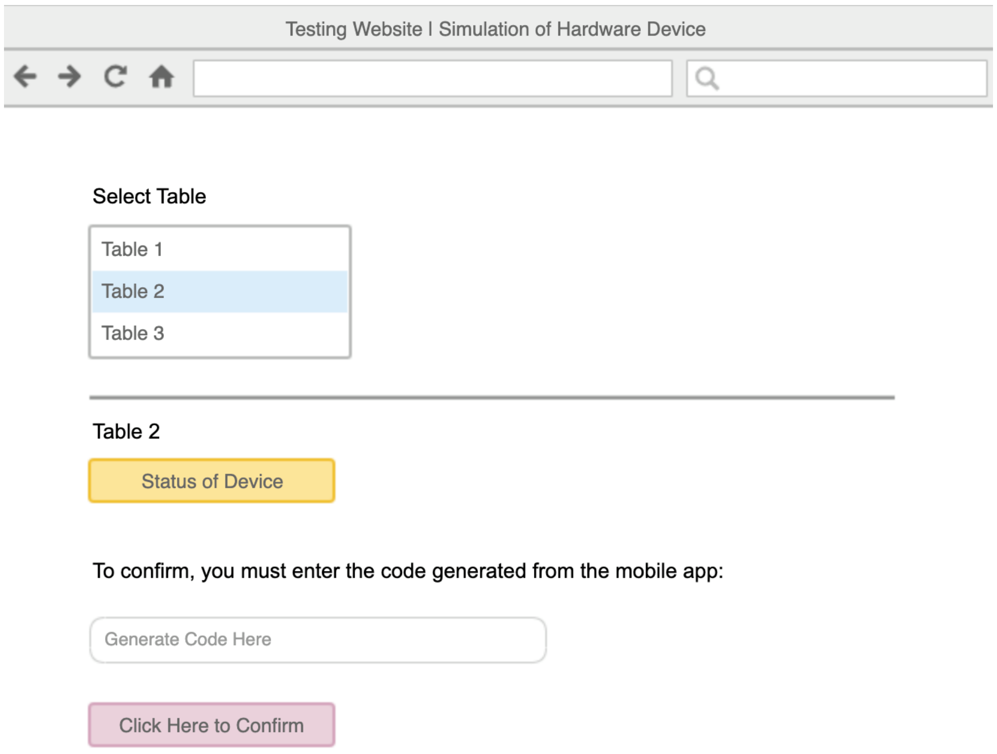

Testing Website I Simulation of Hardware Device
Select Table
Table 1
Table 2
Table 3
Table 2
Status of Device
To confirm, you must enter the code generated from the mobile app:
Generate Code Here
Click Here to Confirm

Figure 53: Figure of Testing Website

As Figure 53 shows, there were different factors being considered here in order to simulate how the hardware device would operate.

Select Table Field

In this dropdown menu, we can select which "device" we were using. This area was simulating the user approaching the study spot and sitting in the table. These names depended on what the admin set up from the "Register Device" page. In other cases, the university is to label different tables so it is easier for students to differentiate since a map option might not be available for them.

Simulation

Once the table has been selected by the user, a button "Status of Device" followed up after. This button changed colors depending on what the user wants and the database sends to the device. In the following figure, the "Status of Device" is yellow. That is, the user already reserved the table and the device is waiting to be confirmed. Potentially, we added a randomly generated code that would take care of the authentication since any other random person could press the button in real life and mess up the reservation. In order to prevent this, authentication between the device and the user is needed to figure out if the person confirming is the correct one. This code would be randomly generated when the user reserves the spot. Then, the user would enter this code in a keypad in the device and press the "Confirm" button. To simulate this part, we generate a code within the app and then add it in the “Generate Code Here” field on the testing website and click “Confirm”. Once this button is clicked, the “Status of Device” button should change to red to reflect that it’s taken.

This testing website was abandoned in the end due to time restrictions and the realization that the devices could be created through the firebase real-time database. Since the hardware devices read and write to the real-time database, the software was required to operate there as well. Google’s Firebase allows authorized users to access the real-time database and modify/add fields, which is the same process that would have been undertaken by this test site. Therefore, it was abandoned for simplicity purposes.

Test Cases

There were also a few additional test cases for making sure that the device communicates back and forth between each other, shown in Table 47.

Test Case #1 - Adding the device
Action: Add the device from the admin control panel **Reaction:** The device’s JSON should show up in the database, and the led on the study spot finder should flash green three times to confirm
Test Case #2 - Removing the device
Action: Delete the device from the admin control panel **Reaction:** The device’s JSON should disappear from the database
Test Case #3- Creating User Profile
Action: Create a user profile from the user signup form **Reaction:** Database should show updated user information in JSON format
Test Case #4 - Code generation and verification
Action: Application should generate a random code and when person enters said code. **Reaction**: User is confirmed for their spot, by changing the table status to red
Test Case #5 - Delete User Profile
Action: Admin should be able to login to the control panel and delete the user **Reaction:** User information should be gone from the database

Table 47: Software test cases for the Study Spot Finder

7.1. Security Concerns

In this section, explanation about software and hardware concerns are being discussed and possible problems that the design might run into and how to prevent it.

7.1.1. Software

In regard to software security concerns, we looked into different ways that someone might be able to steal our information, API keys, information from users, etc.

Table 48 Outlines some of our software security concerns:

Security Concern	Reason
Database Hack	One of our main concerns should be the database. If someone gets access to our database, they basically have the information to all of our devices and can easily change and customize information. This would not be beneficial for the Study Spot Finder app since several users might be depending on accurate data. Besides the devices, hackers could also steal students' information such as school and student ID. Several identities can be affected by this if not built properly.
API Keys Stolen	In order to get to the database, there is also API keys information that could be stolen. With this, hackers can grab whatever information they might want for who knows what. They might use this information to their advantage and to steal information from our users.
Multiple Reservations Per User	Our app only allows one reservation per user. There could be a possibility that a hacker might be able to create different identities and different IDs in order to create multiple accounts, giving themselves and advantage when it comes to reserving a spot.

Table 48: Software Security Concerns

Possible Overall Solutions

In order to keep our application safe, there are a few protocols and precautions we can do in order to prevent attacks from possible hackers.

Table 49 explains some possible solutions to the above concerns. Even though these are possible solutions to our concerns, there is still a possibility that hackers can find their way into our application.

Security Concern	Possible Solution
Database Hack	For this concern, we should be careful when it comes to creating a new database for our project. We should understand that there is no public way to put a database out there, even though this is still something people can do if they want. In our case, we have to ensure we build a strong database, but with the appropriate security.
API Keys Stolen	For this concern, there could still be issues when it comes to someone stealing our API keys. This could happen anytime. Perhaps a git push of a code file has these pre-set and our github is public. That is just one example, but hackers can always find their way to hacking our API keys. The best solution is to make sure these are kept in a safe place, even outside of the online world, but in a piece of paper. This way we would never have to deal with online hacking.
Multiple Reservations Per User	For this concern, it might be a bit harder to solve, but not impossible. As for our prototype, it would be impossible to tell if the Student ID is real or not since we do not own or have a copy of all the student IDs available at the universities. The best way to approach this if our app was scalable is to have an agreement with the university to have a copy of all the possible IDs at UCF, of course, making sure they trust our application to hold them. Another solution would be creating a separate system that will just “Read” the IDs from UCF’s database of IDs and then it would never had to interact with it. This would be the optimal solution since we will always have the latest updates when it comes to new IDs and old IDs.

Table 49: Software Security Concern Solutions

7.1.2. Hardware

In regard to the hardware of this project, we might be running into several security concerns as well since our product is physical and it is subject to the use and of many students and users in a single day. These concerns are critical to the proper function of the device and so we introduced solutions to handle all of our concerns.

Table 50 outlines some of our hardware security concerns:

Security Concern	Reason
Open Case	If a student manages to remove the screws that hold the outer casing to the inner casing, they would be able to access the inner electronics and de-rail the entire system. (Not a problem anymore since screws were not implemented in the final design. However, a pressure cap was used that can still provide the same problem)
Remove any external component	In the case of a student removing any external components such as the keyboard or the push button, the user would have no way of confirming a spot. This would render the device useless since it has a time-out function that would cancel the reservation if no code is placed.
Unplug Battery	Finicky students may also try to access the battery case and remove the battery in order to sell it or use it for their own benefits. This would cut all power to the system, making it unresponsive and leaving the status in the database locked forever. This problem could snowball into something much worse, as the app would be waiting for a response that is never coming so the spot could either never be confirmed or would stay reserved forever until it was back online. We would also need protective measures for the device to automatically start itself up and resume its normal status

Table 50: Hardware Security Concerns

Table 51 Below addresses these concerns to provide solutions:

Security Concern	Solution
Open Case	The best way to impede regular students from opening the case, however unlikely that they are carrying around a screwdriver with them, would be to use screws that require a special tool to open them, such as a torx bit or a security screw
Remove any external component	A proper solution for this would be to figure out how to use the microcontroller to self test its components and verify that they are indeed working. However, this may be outside of the possible range of the microcontroller. Ultimately it would be in our best interest to simply rely on student feedback. If a student realizes that their device is not working, the staff can simply take the device out of service by turning it off, thereby preventing other students from experiencing the same problem. Although this is not an ideal solution, it is also very unlikely a button will be removed or unplugged so this case is very unlikely.
Unplug Battery	Since the microcontroller and the Wi-Fi module loses power, all communications are terminated. The best way to combat any miscommunication is the verification process. The software waits for a response from the device before securing a reservation. If the device is active, it confirms the status and the reservation will be placed. If the device is off, the status does not be confirmed and so that reservation becomes unavailable, giving the user an error message to try again with another spot.

Table 51: Hardware Security Solutions

8. Administrative

All engineering design and production required planning in order to have a successful outcome. Each member acted as an administrator and kept track of the progress of each other. In this section, the following subjects will be covered:

- Finances and budgeting for the Spot Finder
- High-level milestones and detailed milestones.
- Deadlines for each part of the project to ensure the project stays with the time constraints

8.1. Project Budget

Since the project did not have any sponsors, the total cost of the project came out of pocket from the team members, split evenly by 4. Money should not be an issue for this project, as some of us are well funded from internships, and the hardware is relatively cheap for this project, as listed in Table 52 below:

Item	Cost
Plastic Weld	$5.00
Wifi Module	$3.80
Microcontroller	$13.00
LED	$1.00
Protoboard	$5.00
Solder/ Electric Tools	Free
Plastic Spool for 3D printing	$5.00
Button	$3.00
Circuit Components	$30.00
Total	$62/Device

Table 52: Cost Table

8.2. Bill of Materials

In this section, the list components used to build one device are listed with the relevant characteristics needed. This bill of materials only covered the device in its ‘Essential’ stage, meaning that other less important features not used in the final design are not included.

8.2.1. Bill of Materials Essential

Table 53 below represents the component list to achieve the essential or must have goals for this project.

Level	Description	Manufacturer	Part No	Quantity	Unit Cost	Total Cost
1	Microcontroller	Microchip	ATMega328P	1	$9.99	$9.99
1	Wifi Module	Espressif	WRL-13678	1	$6.95	$6.95
1	Battery	Adafruit	353	1	$29.50	$29.5
1	Regulator	Microchip	MIC5504-3.3YM5-TR	1	$0.89	$0.89
1	Regulator	Microchip	MIC5504-5.0YM5-TR	1	$0.89	$0.89
2	LED	Jamecu Value Pro	2228957	2	$0.39	$0.78
2	PVC Pipe	Lowe’s	MMUULTCG	1	$9.99	$9.99
1	Connectors	JST PH 2 pin	VUPN924	2	$2.90	$5.80
1	Button	Jiu Man	Y35-l265	1	$9.99	$9.99
1	Switch	Judco	J-188A-1	1	$1.85	$1.85

Table 53: List of components required to build this project

Parts Acquisition
In this section, we considered online locations of where we purchased our sensors and our microcontrollers with judgements based on quality and price: DigiKey, Mouser, and Adafruit. Table 54 shows each in detail.

Part	Description
Digikey	Cheap components and relatively fast shipping, also has a good search where you can look up any part quickly, also has a huge inventory and consistent pricing, their labels are much more better than Mouser, as they are easy to read, but they have dodgy packaging
Mouser	Lower priced than Mouser, but their search is a pain to use at some extent, no free shipping, and their inventory lags behind Digikey. Mouser's packaging is excellent on the other hand and takes care of their shipping.
Adafuit	Kind of pricey components, but very good for specific projects for the Raspberry Pi or Arduino, since they offer many ready-to-use components, such as breakout boards and kits.

Table 54: Parts Acquisition

8.3. Project Milestones

In order to create a product by the end of Senior Design 2, we created an estimated timeline for the individual tasks that we encountered along the way in Senior Design 1 & 2. These deadlines were not solid and were in progress in progress at the time Senior Design 2 Started. However, the project is being completed and fully functional meeting all requirements we set in the beginning of the design. We faced some challenges and changes throughout the process which was indicated in this paper.

Senior Design 1				
Number	Task	Start Date	End Date	Status
1	Brainstorm Senior Design Ideas	08/26/2019	08/30/2019	100%
2	Project Selection	08/30/2019	09/02/2019	100%
3	Goals and Objectives for Project	09/02/2019	09/06/2019	100%
4	Initial Project Documentation	09/02/2019	09/20/2019	100%
5	Table of Content	09/20/2019	09/22/2019	100%
6	Research Hardware	09/22/2019	10/21/2019	100%
7	Research Software	09/22/2019	10/21/2019	100%
8	60 Page Documentation Draft	10/21/2019	11/01/2019	100%
9	Hardware Design	10/30/2019	11/15/2019	100%
10	Software Design	10/30/2019	11/15/2019	100%
11	100 Page Documentation	11/14/2019	11/15/2019	100%
12	Develop Test Plan	11/15/2019	11/25/2019	100%
13	Order Components	12/03/2019	12/10/2019	30%
14	Final Documentation for Senior Design 1	25/11/2019	12/04/2019	100%

Table 55: Milestone Table for Senior Design 1

Senior Design 2				
Number	Task	Start Date	End Date	Status
13	PCB Layout/Print	12/09/2019	12/20/2019	50%
14	Assign Programming tasks	12/05/2019	12/10/2019	100%
15	Initial Hardware Testing	12/16/2019	01/02/2020	20%
16	Power Supply	01/02/2020	02/02/2020	10%
17	LED circuit	01/02/2020	02/02/2020	0%
18	Housing Prototype	01/02/2020	02/02/2020	0%
19	Database Set Up	01/02/2020	01/20/2020	2%
20	Static Application	01/02/2020	01/20/2020	0%
21	API requests	01/02/2020	01/20/2020	0%
22	Front-End Connections	01/20/2020	02/15/2020	0%
23	Back-End implementation	01/20/2020	02/10/2020	0%
24	Initial Software Testing	01/25/2020	04/20/2020	10%
25	Build Prototype	02/05/2020	03/02/2020	0%
26	Test & Debug Prototype	03/04/2020	04/16/2020	0%
27	Final Documentation for SD2	04/20/2020	04/20/2020	0%
28	Finalize Project	04/17/2020	04/17/2020	0%
29	Final Presentation	04/19/2020	04/19/2020	0%

Table 56: Milestone Table for Senior Design 2

Task Delegation

In order to accomplish the planned timeline shown in the two tables above, the team delegated tasks to each member of the team. The following table describes which team was responsible for which task. However, the design choices needed to be reviewed and approved by the entire team. Table 59 and 60 show the responsibilities of each team member.

Team Member	Task
Andrew	Primary Task: Responsible for the development of the front-end section. Researches the software tools needed to build the best suitable one for this project as well as programming language, libraries, documentation and environment for integration with the hardware components and back-end aspect of the project.
	Secondary task: Responsible for communication with the back-end person in charge and the hardware team. Moreover, responsible for the prototyping of the app. Creates and updates group meeting plans.
Perla	Primary Task: Responsible for the development of the back-end section. Researches the software tools needed to build the best suitable one for this project as well as programming language, libraries, documentation and environment for integration with the hardware components and front-end aspect of the project.
	Secondary task: Responsible for communication with the front-end person in charge and the hardware team. Moreover, responsible for the prototyping of the app. Proofreads the project report constantly to ensure continuity.

Table 57: Software Task Delegation

Team Member	Task
Maria	Primary Task: Responsible for the LED pcb layout. Research the hardware components needed as well as integration with the software team. Moreover, in charge for the schematics and device design.
	Secondary Task: Requests progress updates for each team member to ensure everyone is on track and every section is done within the deadlines. Also, ensures that the chosen components are bought and delivered prior to the timeline milestone
Darwin	Primary Task: Responsible for the power generation, wifi module and microcontroller. Research the hardware components needed as well as integration with the software team. Moreover, in charge for the schematics and device design.
	Secondary: Responsible for the printing of the main pcb design and the soldering of the electrical components. Furthermore, Verifies every section, table and figure is labeled and numbered correctly.

Table 58: Hardware Task Delegation

9. Conclusion

The Spot Finder project involves existing technology previously designed and implemented on the market; however, this device has both software and hardware implementation in order to provide an advantage to students who would like to get a study place without wasting time looking for it. Past technology was either meant for a business environment which had similar features, yet some extra features no required for an academic environment and missing features needed on the academic environment. Another technology had only the software aspect of the project, however, this device provides easiness and simplicity through interaction with the machine and web application planned.

Even though, the team wanted to implement more feature to the device to make more interactive, automated and simpler, due to time constraints and budget constraints, we have decided to implement only the essential requirements to the device. For future version of the device, the team plannned to implement more hardware features such as sensors and recording, as well as improving the application platform. Regarding the completion of this project, we were on time with our project based on the timeline planned and ahead with ordering the components as well as building the software static prototype.

During this semester, the team did extensive research about new software technology as well as hardware components and a way to integrate them. Then, the team decided what features can be included in order to complete the project within the time constraint. Also, research helped which hardware components were more accessible to integrate with the desired software environment used. The goal for the full completion of the project by the end of next semester was guaranteed with the essential specification requirements functioning.

There have been certain challenges that the team encountered during the design of the project for integrating the Wi-Fi module with the software design. The challenges were overcome, and we came up with solutions to every problem we encountered. While this group is formed by three computers engineer the hardware team and software team has been divided into two equal teams where programming skills are required in both sides. By the end of this project, the team have developed new skills in both software and hardware levels. For the hardware aspect, the team have gained programming skills, hardware design involving power supply, hardware components integration, PCB schematics, design and assembly. On the software aspect, the team have gained new programming languages, as well as new database environments, libraries and software development skills. Furthermore, the entire team gained managerial skills and teamwork.

The main goal of this project was that each team member applied the knowledge of what they have learned during years of college about circuit design and software development.

10. Appendix

10.1. Bibliography

"The 16 Most Important Pros and Cons of Using Python for Web Development." *Django Stars Blog*, 17 Sept. 2019, djangostars.com/blog/python-web-development/.

"5V 40 LEDs Digital WS2812B Programmable Pixel LED Light Ring." *Kutop International Limited*, kutop.com/5v-40-leds-digital-ws2812b-programmable-pixel-led-light-ring.html.

"Accelerometer Sensor." *Accelerometer Sensor - an Overview* | *ScienceDirect Topics*, www.sciencedirect.com/topics/engineering/accelerometer-sensor.

"American National Standards Institute." *ANSI*, www.ansi.org/.

"Code of Ethics." *Code of Ethics* | *National Society of Professional Engineers*, www.nspe.org/resources/ethics/code-ethics.

"Detection Based on 'Ultrasonic Waves "What Is an Ultrasonic Sensor?" *KEYENCE*, www.keyence.com/ss/products/sensor/sensorbasics/ultrasonic/info/.

"DIP LED." *Visual Led*, 26 Apr. 2019, visualled.com/en/glossary/led-dip/.

"ECFR - Code of Federal Regulations." *Electronic Code of Federal Regulations (ECFR)*, www.ecfr.gov.

Gamma, Smart. "What Are the Pros and Cons of Using PHP?" *Medium, Medium*, 1 June 2016, medium.com/@smartgamma/what-are-the-pros-and-cons-of-using-php-490553ed8ff2.

"The Good and the Bad of Swift Programming Language." *AltexSoft*, www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/.

"The Good and the Bad of .NET Framework Programming." AltexSoft, 28 June 2019, www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-net-framework-programming/.

"IEEE Standards." *IEEE*, www.ieee.org/standards/index.html.

"LED 101: Identifying Different Types of LEDs." *Electronic Products*, 19 Apr. 2018, www.electronicproducts.com/Optoelectronics/LEDs/LED_101_Identifying_different_types_of_LEDs.aspx.

"LED STRIP LIGHTS." *Everything You Need to Know About LED Strip Lights* | *Waveform Lighting*, www.waveformlighting.com/led-strip-lights.

Marrakchi, David. “Top 5 PCB Design Guidelines Every PCB Designer Needs to Know.” *Altium Resources*, 30 Sept. 2019, resources.altium.com/pcb-design-blog/top-pcb-design-guidelines-every-pcb-designer-needs-to-know.

“Native vs. Cross-Platform Apps: The Startup Dilemma.” *Skelia*, 25 Sept. 2019, skelia.com/articles/the-startup-dilemma-native-vs-cross-platform-apps/.

Scully, Taylor. “7 Things to Know Before Buying and Installing 12V LED Strip Lights.” *LEDSupply Blog*, 1 Nov. 1969, www.ledsupply.com/blog/7-tips-before-installing-led-strip-lights/.

Smith, W.A. “ESP8266 Testing.” *Starting Electronics, Electronics for Beginners, Hobbyists and Beyond*, startingelectronics.org/articles/ESP8266-testing/.

“SPST PUSHBUTTON, PUSH ON/PUSH OFF.” *All Electronics Corp.*, www.allelectronics.com/item/pb-13/spst-pushbutton-push-on/push-off/1.html.

“Ultrasonic Sensors to Detect Human Presence.” *MaxBotix Inc.*, 22 Nov. 2019, www.maxbotix.com/ultrasonic-sensors-detecting-people-156.htm.