Initial Project Document

Facial Recognition Locker

University of Central Florida
Department of Electrical Engineering and Computer Science
Senior Design I

Group 4
Bryce Dere - Electrical Engineering
Che' Baptiste - Computer Engineering
Julian Boaz - Computer Engineering
Ryan Wiegman - Computer Engineering

Table of Contents

List of Figures

List of Tables

1. Executive Summary

With the explosion of companies like amazon and uber, more and more items are being delivered to your door than ever before. However, with more things being delivered to the door, porch pirates are becoming more of an issue than ever. In turn people are now starting to purchase lockboxes to help prevent their purchases from being stolen.

Though this may work some of the time, these boxes do not even lock and some of them only lock with a simple padlock that can be easy to cut. Now, unless you want to spend a fortune on a state-of-the-art lockbox; our project intends to help reduce the possibility of porch pirates stealing your items at a more affordable cost. We intend to create a fully safe facial recognition lockbox.

Our lockbox, as described, will use facial recognition to lock and unlock users. The focus will be to use a camera that is mounted to the lockbox, to scan the face of the person in-front of it and if it is a registered face in the system, unlock the box. The locking mechanism will also be hidden to prevent porch privates from just cutting the lock.

Another perk of our project will be an application that you will be able to download to your phone. This application will be used to register faces for the box to unlock, notify users when packaging is delivered and provide any security alert to the user.

Another big aspect of our lockbox will be for food delivery services. Along with facial recognition, we will implement a heating system to keep food warm for the owner if they are not home yet. Once the delivery driver closes the box there will be a user screen where they will be able to select if the food will need to be kept warm or not. If they press yes, the heating system will turn on, allowing the food to stay heated. If not, the system will maintain its internal temperature.

One real constraint for this project is how it will be powered and where it can be utilized. The Facial Recognition Box will be powered through an outlet. Due to this, there is a big limitation on where we can put the box. Not every home has an outlet on the outside of their house, let alone on their porch. Despite this, we plan on keeping the dimensions of the box to a microwave-like size so it can be placed anywhere without taking up too much space. The camera and user device will also run in an idle state when no one is around, only activating when the camera picks up movement in front of it.

Our goal is to make an affordable and safe lockbox that implements the technology of the 21st century to best keep the items the user purchases out of the hands of porch pirates even when the owner is away from home.

2. Project Description

Project Background:

As technology continues to develop, things are becoming more automated and more safe than ever before. Machines have been developed in order to manufacture the chips that go into our computers and phones, driving assistance AI have been incorporated into newly made cars to help keep drivers in their lane while driving. These are just a few of the many examples where technology has not only made our jobs easier, but has made tasks safer to perform.

One of the more popular trends that has come up in recent years are food delivery applications, such as Doordash, Postmates, Uber Eats, etc. They deliver your food and leave them at your doorstep, and in theory this is fine; however, with this comes the opportunity for someone else to come and steal your food. Our proposal is to make a device that can only be opened by the delivery person and the user or owner so that nobody else can steal their food.

While this locker's sole purpose is not to hold food, it will be small enough for it to be placed outside of one's house or apartment. It will be a small box that will incorporate a warming mechanism, and be about the size of a microwave.

The box will have a camera on the top of the box that will only be active when there is motion in front of the box. When the camera is activated it will start looking for a face that matches that of the picture of the user or of the delivery driver. If the facial recognition decides that the face in front of the camera and that of the picture match, then the box will open. Otherwise it will stay closed. If however, the person decides to try to break into the box, an alarm will be set off and a notification will be sent to the user.

The companion application that was briefly mentioned earlier will only be available to the user. It will contain information like the location of the box as well as the ability to upload an image of the driver's face. Additionally, from the application you will be able to tell if the item that will be placed in the box is an item that needs to be cooled. If it is, when the item is placed in the box, the cooling system will kick in and keep the item cool. The temperature that the box will be at will be a set temperature, so the user does not have to worry about what temperature they need to

set the box to. The application will also serve as a way to notify the user when the food has been placed in the box. This will be done through weight sensors in the box, and when the microcontroller registers that something has been placed in the box, a notification will be sent to the user.

The companion application that will go along with the device will allow the user to be able to communicate with the device from anywhere they so wish. The only time they will have to touch the box is when they go out to grab the food. Additionally, the application will feature a way to open the box from the device in case of a face recognition failure.

The purpose of the box will not only serve as a way to temporarily store your food, it will serve as a way to help drivers identify your house. This is especially the case for those who live in apartments, as it is very easy for drivers to get lost in complexes where all of the rooms look the same. With drivers having an easier time identifying your room, they will be less likely to call and ask for help and/or directions, making their job easier as well. In addition, so that the delivery driver does not get confused, when the box senses motion and confirms that the person in front of the box is the delivery driver, the box will say "Please place the food here," and open the door for the food.

Additionally, the device will also have a few anti theft features in order to prevent stealing. Since the box will share its location with the user, the user will be able to know if someone moves the box and where it is at all times. Additionally, if someone tries to forcefully open the lock or break into the box, a siren will be set off.

The overall audience for this device is for any household that gets food delivered often. More and more people are using food delivery applications and when the food is left outside of your door, it is very easy for someone to just walk by and take your food.

Additionally, many consumers have made comments about delivery services with a penchant for keeping orders; it is not unheard of that some delivery people will simply take your food without delivering it. With this device, it will help deter anybody from stealing your food.

The device will have some physical constraints. For example, it will not be very portable or versatile in its usability. By adding a battery pack to the Facial Recognition Locker, it makes the scope of the project too complex for a group of college students to complete in the given time frame. It raises a lot of questions that we do not know the answers to, leading us to believe it is

not a feasible feature to implement in the project. Instead, the Facial Recognition Locker will be plugged into a wall outlet, with the ability to turn the device on and off.

Project Motivation:

The primary motivation behind this project is to build a device that is not meant to be expensive or exclusive in any way, but simply a way of securing your food or items using cutting edge technology. The objective of this device is to simply provide users with a safe and efficient way of securing their food. We want to make it a relatively painless and stress free process of getting your food. This is especially the case when the user might not be able to get their food as soon as it gets delivered.

While things continue to become more automated in our lives, we must take into account the risks that come along with it. The risks that come along with getting your food delivered is that someone might steal it, and our device seeks to combat and reduce said risk for all users.

Project Design:

The design of our Facial Recognition Locker will be based on using an electrical outlet for its power supply. With the amount of appliances each box will have that involve power, having the Locker run off a battery source is not logical. The size and cost of the battery alone will drastically increase the total cost of the entire project. Since one of our main goals is to make a cost effective Locker, having the system be battery powered is not possible. This also would require replacement cost of the battery once it has died.

By using an electrical outlet, the locker will have a consistent and constant supply of energy. As mentioned early, the locker will have quite a few appliances that will constantly be using energy from the heater, LCD, microprocessor and the camera. Using an outlet also decreases the risk of the Locker losing power and the user being unable to open it.

The lockbox will also be designed in such a way that it could be easily used to hand off small or medium-sized packages that are not food items; the project viewed through this lens simplifies slightly in scope to simply being a porch pirate prevention lockbox that uses facial recognition as the authentication procedure. The required constraints surrounding security, efficiency, performance, etc. do not change, but the requirements related to safe food holding/handling are not considered in this case.

For the software component of the design, our team has decided to use Amazon Web Services. There are a variety of reasons we ended up choosing Amazon Web Services to use for this project. One of the big reasons is for academic purposes. By utilizing Amazon Web Services we allow ourselves to get accustomed to and gain experience with a tool that is very commonly used in the real world. Big companies like McDonalds, Moderna, and Finra use Amazon Web Services, and Amazon Web Services experience is commonly asked for on job applications. By constraining ourselves to use a cloud service like Amazon Web Services, we gain an invaluable experience that will only help us, as engineers, ease our way into the workforce.

We decided to use a cloud service, and ultimately Amazon Web Services, due to the nature of our project. We want the application to be able to communicate with the Facial Recognition Lockbox from outside of the local network the both are connected on. Due to this, a cloud service was necessary as it allows communication to those who have access to the server from virtually anywhere. A cloud server will store the necessary files and components of the project that will be necessary for both the Facial Recognition Lockbox and the application to be able to communicate. After doing some research on the many cloud services available, we ended up settling on Amazon Web Services.

Amazon Web Services provides us with the perfect environment to tackle our project. Amazon Web Services allows us to utilize cloud storage. With this cloud storage, we will be able to store all of our important files in one place that can be reached from anywhere. This includes the files and information that the application will have to access and utilize. Amazon Web Services also supports artificial intelligence programs being uploaded to their cloud. Therefore, our artificial intelligence will also have an easier time being able to access the files and communicate with both the application and the Facial Recognition Lockbox.

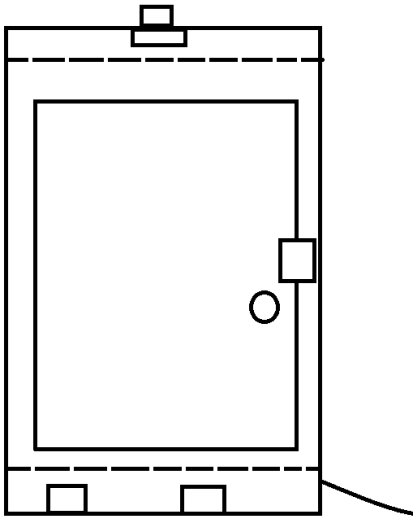


Figure 1 - Preliminary Diagram

Illustrated above is a rough sketch of our preliminary diagram for the Facial Recognition Lockbox. We have not decided on the exact dimensions of the box we are going to use, that will depend on the box we end up purchasing, which depends on a variety of factors, mainly price. Nonetheless, that does not mean we cannot plan out how we are going to incorporate the technologies into the project.

As you can see, there are a variety of unlabeled boxes on the body of the illustration. The two boxes at the very bottom are where the heating and dehumidifying components for the Facial Recognition Lockbox will go. Seen right above those two boxes is a dotted line, which will create a false floor. In order to do this, we will have to use a piece of wood or something solid of the sort to keep the user from being able to access these components. Additionally, there will need to be a false back to the box in order to run the cables from the Raspberry Pi to the outlet that is unable to be illustrated through the diagram.

In the middle of the box, you can see another box located directly above the door handle. This box will be where the locking mechanism will go. The locking mechanism, along with the heating and dehumidifying mechanisms at the bottom, will communicate with the Raspberry Pi in order to regulate the temperature, as well as when the box will be able to be opened.

At the very top of the Facial Recognition Lockbox is where the Raspberry Pi and the camera will be located. As of now, since we do not have an extension for the camera, the camera and the Raspberry Pi have to be close together. As a result, we plan to have the Raspberry Pi mounted on the roof of the Lockbox. Again, the lockbox will have a false ceiling, as to not have the Raspberry Pi exposed to the user. This will also serve to protect the Raspberry Pi, as it will not

be able to be exposed to the varying temperatures that the lockbox will be producing. If the Raspberry Pi were to be exposed to these temperatures, we risk damaging the Raspberry Pi, which will not only eventually destroy the Raspberry Pi, but ruin the project as a whole since we would have to purchase another one.

Once again, the inside of the Facial Recognition Lockbox will be covered in insulation to better keep the temperature within the cooler the same. This will help put down on energy costs which is also an important factor that must be taken into consideration.

Requirement Specification:

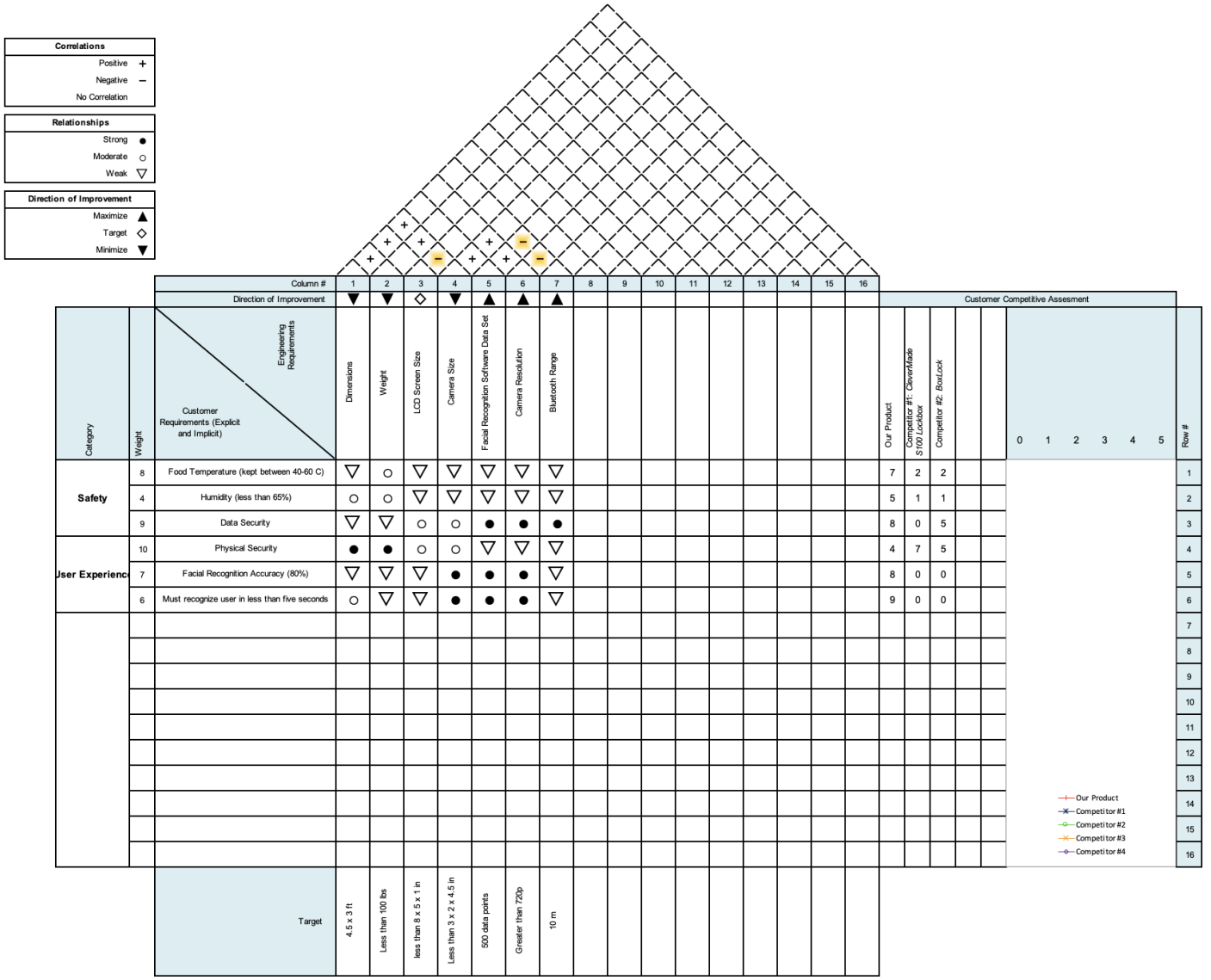
1.	The system shall be no taller than 4.5ft
2.	The system shall be no wider than 3ft
3.	The system shall not weigh more than 100lbs
4.	The system shall contain a lock box, Electronic Latch, LCD screen, facial recognition camera, AI facial recognition software, storage database, Wi-Fi Connector, a wall plug, a back-up digit code lock.
5.	The Electronic Latch shall not exceed $30
6.	The LCD screen shall be no larger than 8 x 5 x 1 inches
7.	The LCD screen shall not exceed $20
8.	The facial recognition camera shall be no larger than 3 x 2 x 4.5 inches
9.	The facial recognition camera shall not exceed $30
10.	The AI facial recognition software must have a minimum of 500 data points per user
11.	The AI facial recognition software must be able to identify a user with 80% accuracy

12.	The AI facial recognition software must be able to generate a file to reference for future evaluations
13.	The AI facial recognition software must go through a training period to properly evaluate users
14.	The AI facial recognition software must have an execution process between 3 to 7 seconds
15.	FRL supports Wi-Fi connection
16.	The Wi-Fi sensors must be compatible with the FRL application
17.	The FRL application will work with AI facial recognition and storage database to log user's facial ID
18.	The FRL application will notify users of deliveries to the system
19.	The FRL application will notify users of attempted theft or breach of the system
20.	The FRL application database allows up to 5GB of storage
21.	The FRL application database shall not exceed more the $0.02 per hour
22.	System Microcontroller shall not exceed $50
23.	The system shall maintain a maximum temperature of 140° F
24.	The system shall regulate average temperature of 73° F

25.	The system shall maintain moisture at acceptable levels (<65% humidity)
26.	The power supply shall be capable of powering multiple sensors and accessories
27.	The power supply shall be capable of powering the heating system and accessories
28.	The total cost of the system shall not exceed $300
29.	The system will have a 6 digit code lock, in case of facial recognition failure.

Table 1 - Requirement Specifications

House of Quality Analysis:



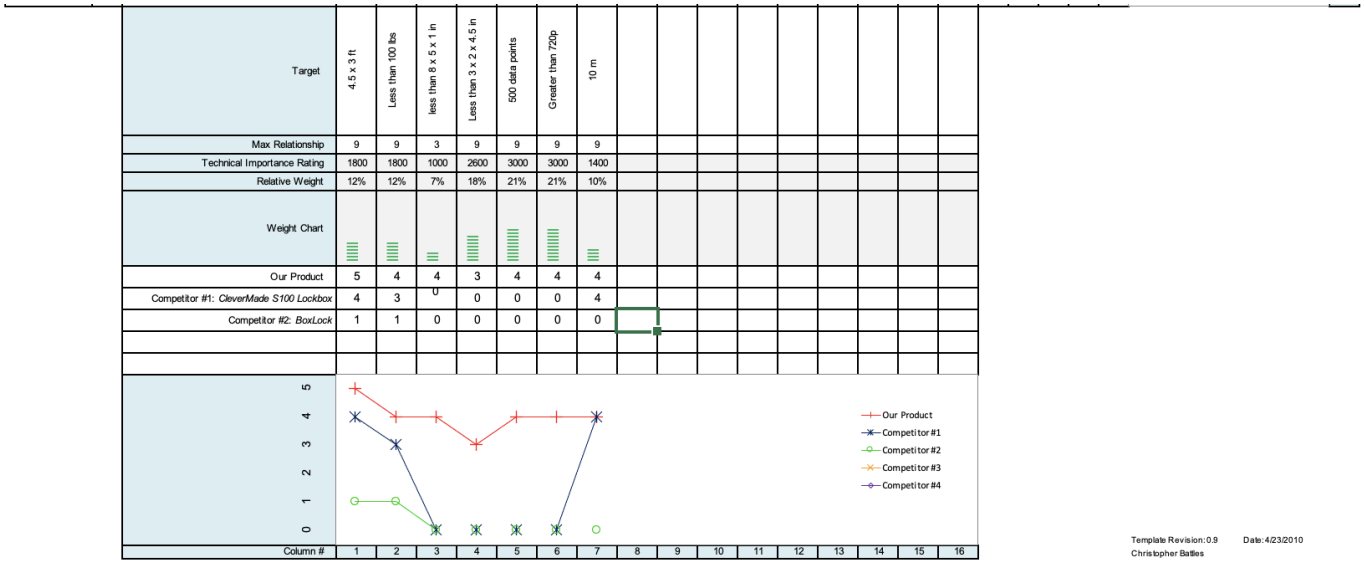


Figure 2: House of Quality Analysis

3. Research Related to Project

In 2021, an estimated $870.8 billion dollars were spent on e-commerce sales. With an increase of 14.2 percent from 2020 [1], this rise shows that more and more people are shopping online. As online shopping grows, so do the thefts that accompany at home deliveries. A study done by C+R Research in 2020 showed that more than 40% of Americans have been the victims of package theft and more than 60% know somebody who has fallen victim [2]. While the average cost of stolen packages was $136, home owners have spent more time and energy to avoid these thefts than any one package was worth.

Ranging from buying preventative security measures to online shipping companies, implementing more descriptive dropoff measures, like allowing delivery drivers access to your house or garage, consumers, producers, and distributors are trying to prevent porch pirates and other pesky package thieves. However, it is fair to assume most people do not feel comfortable allowing a stranger access to their home.

Another strategy at-home buyers are trying is installing security cameras that look over their porch or driveway. Now this might help prevent some thefts from happening and even catch the individuals who did steal the package, but it does nothing for porch pirates who wear masks, making it impossible to correctly identify them. Even Frank Albergo, president of the Postal Police Officers Association, has expressed his worry about mail theft being at an all time high [3].

Now, with Porch Pirates more prevalent than before, preventing stolen packages, and any other goods procured with the ease of at-home delivery, is a must. That is where the Facial Recognition Lockbox comes into play.

Existing Products:

The "smart" lockbox is not a new concept, in reality. In 2018, Prevent Package Theft released an article listing the top 5 best parcel boxes on the market [4]; when faced with an adversary it is only human nature to find a solution. These products range from as expensive as $1,000 to as cheap as $300, each providing unique security options for each customer. While they all differ from installation to external and internal acessability, they all incorporate smart technology in one way or another. This section covers a plethora of designs that combine smart technology and an anti-theft design to ensure package safety and customer satisfaction.

However, when researching products for facial recognition lockboxes you won't find many options out there. There are some lockboxes on the market, ranging from highly integrated technology to some that do not even lock.

A prime example of this is a smart lockbox by a company called Eufy Security. This lockbox is one of the top smart lock boxes currently on the market. This Smart box, like many other designs, is equipped with an application, granting the customer round-the-clock protection and notifications whenever a package arrives. It allows its users to communicate with any courier through a two-way audio system, and its frontal panel is installed with auditory instructions to assist delivery services unfamiliar with its system. Its multiple access points allow a range of options, from singular usage through PINs and the application to manual access through keys; whatever the preferred method, this design has something for everyone. This lock box is an untethered design, its main charging port being via USB, and it operates primarily through its battery pack [5].

Another popular option in today's market is Yale Access's Yale YRBB CB1K Brighton Smart Delivery Box. However, the smart features this box contains are quite limited, only featuring a smart keypad and an application, downloadable to your phone. Unlike the previous design, this lock box does not provide its user with any features to ensure the safety of the package other than the box itself. The application allows for remote access and the ability to easily change or delete passcodes. This lockbox also has an untethered design running on a battery, with a battery life of one year.

As the world changes, technology adapts with it; this is increasingly true in the case of security, specifically that of smart security. Almost every electronic device, from cellular phones to computers, has implemented a form of user specific security, moving away from the fingerprint reader of past years and into the realm of facial recognition.

Despite its rising popularity, facial recognition has yet to make its way into smart lockboxes. Without any smart technology options external security is left to the whims of codes and manual locks. The only devices used for external at-home security integrated with facial recognition software are designed to attach to your front door, leaving their use very singular and their design uncustomizable. This design will be similar to what our Facial Recognition Lockbox will be implementing. This front door lock uses an instant facial recognition that will unlock your door in under 0.5 seconds. It comes with a 120 degree, super infrared night vision video live feed to work at all hours.

This lock also comes with an integrated mobile application and a built-in keypad for alternative modes of access. Using a Wi-Fi connection, this application allows the user to remotely unlock the device, and change multiple features on the online interface such as their email and backup security pin. Along with remote access, the application will also send the user any alert notification about the device. Much like the previous devices, this lock is powered by a rechargeable lithium battery.

Relevant Technology:

Figure 3. Camera and button Accessories [20]

In the figure above it shows one lockbox implementing the use of a camera and several buttons for the user to press. This design is similar to what our Facial Recognition Lockbox will have. Although this design just allows the owner to see anyone who is in front of the container, our

design will have the camera scan the face of the individual. The AI will connect to our database, relaying the information from one to another. If the data received from the AI matches the database's information for acceptable users, the locking mechanism will be released and the box will open.

Our system will have a similar user interface allowing food couriers to activate heating within the system.

Relative Database:

In this age of technology when almost everything has some integration of an application or browser or user account, there are millions of millions of bytes that are needed to process. This is where having a strong and secure database for an application becomes increasingly important. Now, depending on the project and the goal of the team, creating a database from scratch may be the best option.

However, there are a plethora of user-friendly, easy to integrate, databases out there; Kintone is one of these databases. Kintone already comes with pre-made databases, application templates, and easy to add fields for the database. Kintone is one of the most user-friendly database builders out there, implementing a drag-and-drop interface to create the database; this accessibility allows users of any skill level to curate the database that fits their needs without having to go through the hassle of creating one from scratch. Kintone is deployable on mobile and web-based devices, giving users access controls and permissions, data replication, database conversion, supports multiple programming languages and relational control in multiple developmental formats for ease of access.

Technical Research:

After some time and consideration, we have decided to use the Raspberry Pi 4 as the “brains” of the hardware component. The Raspberry Pi will communicate with Amazon Web Services and send and receive commands from there. The Raspberry Pi will send the camera feed to the AI that will be hosted on Amazon Web Services where the AI will in turn tell the Raspberry Pi if it should unlock the door for that person. The Raspberry Pi will be responsible for locking and opening the door as well as deploying the camera.

There are a variety of reasons we ended up choosing the Raspberry Pi 4, the main reason being convenience. At the time of this project, there is a chip shortage around the world. As a result, microprocessors and anything that requires these chips are not only short in supply, but also more

expensive than they have been previously. This does not just affect a small portion of microprocessors, but all machines that utilize chips to function. Luckily, one of the members of group 4 has access to a Raspberry Pi 4. By using this Raspberry Pi, we will be able to save a significant amount of money and time as we will not have to pay for this expense. With this we will be able to allocate the funds to other components and might be able to add more features if time permits.

In this section, we will take a deeper look at the Raspberry Pi and its hardware specifications to discuss the pros and cons of the microprocessor. Additionally, we will look at some of the components necessary for the Facial Recognition Lockbox to operate and compare the microcontrollers to each other.

NVIDIA Nano:

One other option that was considered was the NVIDIA Nano, a microprocessor in a similar vein to the Raspberry Pi series that is developed by the graphics card designer NVIDIA. This microprocessor was designed for embedded applications that want to run neural network or graphically based simulations on an embedded platform at relatively low power and space costs.

One of the factors making the Nano more appealing is its specific intent towards deployment of Artificial Intelligence software. NVIDIA has specifically tailored their development environment towards decreasing the complexity of Artificial Intelligence programming and deployment, as well as modifying and training neural networks on the fly.

The main limiting factor in wanting to utilize the NVIDIA Nano is the cost: a full Developer's Kit that includes the same equipment as a Raspberry Pi Developer's Kit comes in at almost twenty times the price. An individual part cost of more than six hundred dollars greatly exceeds our target project cost of three hundred dollars, ultimately making this system a poor choice for this application, regardless of the potential benefits.

MSP 430:

An embedded system controller we have become very familiar with over the course of our degree programs has been those in the MSP-430 series of microcontrollers. These controllers are lightweight, have low power requirements, contain connections to many different forms of digital communication, including Universal Serial Bus, and have an onboard LCD and buttons for interacting with an end user.

For all of these educational benefits, however, they are ultimately not particularly useful for practical applications. Deploying software to them involves using external devices to flash the memory over a wired connection, which is impractical when connection disruptions may require setting configuration onsite.

The lack of connectivity with a keyboard and mouse or ability to deploy an Operating System that can more easily support our application means any microcontroller from the MSP-430 series will ultimately be a poor choice for this application.

Raspberry Pi 4:

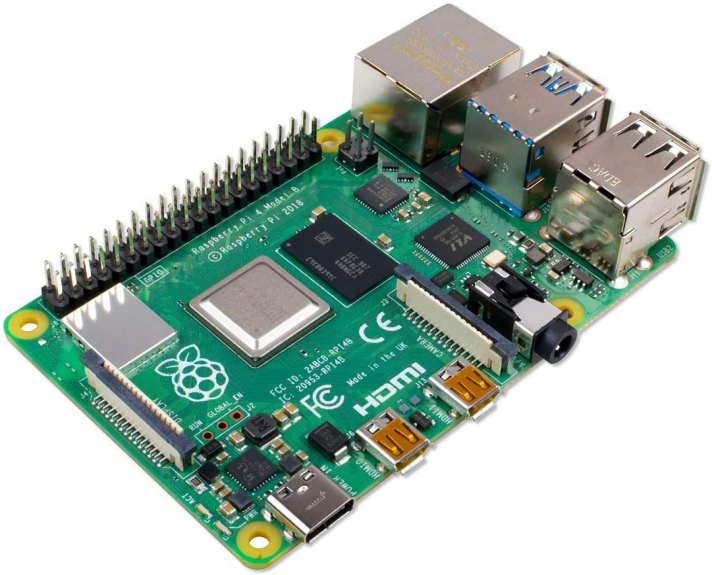


Figure 4 - Raspberry Pi 4 [22]

The Raspberry Pi 4 is a powerful microprocessor and is often referred to as a mini computer. This is because the Raspberry Pi can function completely on its own once an operating system is saved onto a micro-SD card. The Raspberry Pi comes equipped with multiple USB ports that allow it to support the use of monitors, a keyboard, and a mouse. Additionally, it can be hard wired to the internet through the use of its ethernet port and features a 64 bit dual-core processor.

The Raspberry Pi is an amazing tool that can perform a variety of functions. It has the ability to download software through the operating system and can be utilized for a variety of tasks. Additionally, since the Raspberry Pi is its own environment, it can store a lot of information that can be utilized for the Facial Recognition Lockbox. This is important because we want to have one microcontroller in the Facial Recognition Lockbox that can do a variety of things. Not only does the box need to undo the lock, but the camera must be deployed.

Additionally, we are considering adding a 6 digit access code for the Facial Recognition Lockbox. This means that we will also have to add some sort of input output device for someone to interact with. The Raspberry Pi is able to handle all of these tasks and more, making it a very strong contender for our project.

Furthermore, the Raspberry Pi is capable of adding more support to its arsenal. This is especially important because we want the microprocessor to be able to act without needing to be directly connected to the internet. Without any extra costs, the Raspberry Pi is able to connect to the Wi-Fi with just a simple command. Lastly, the Raspberry Pi has a few different ranges of RAM that are available for purchase. The Raspberry Pi can come with 1, 4, or 8 GB of RAM. The option we are considering for our application is the 4 GB of RAM due to the fact that we will not have to worry about the unit being overworked with this amount of RAM.

Additionally, the Raspberry Pi has a USB-C power supply port, meaning it does not come equipped with any sort of battery pack installed on the device. This means that the Raspberry Pi needs a constant supply of electricity in order to function, no matter how big or small the task. Despite having a fairly common power supply port on the Raspberry Pi, it is not safe to just use any USB-C power supply for the microprocessor. This is due to the fact that the Raspberry Pi simply cannot take that much electricity.

So, in order to combat this issue, the makers of the Raspberry Pi have made their own power supply for their device. By looking on the official website for the Raspberry Pi, we can see the specifications for the power supply that the Raspberry Pi uses. The most notable features of the power supply is that it has a 5.1 V / 3.0 A output and a 96 - 264 Vac operating input range, meaning that the Raspberry Pi needs about 5 V in order to function. For reference, the msp 430 requires 3.3 V in order to function, while the nvidia nano requires 4.75 V in order to function.

Nonetheless, the Raspberry Pi does not come without its constraints. The main constraint worth talking about is the fact that the Raspberry Pi uses a lot of power. The Raspberry Pi uses a minimum of 700 mA of current and 5.1 V of electricity, meaning it takes a lot of power to support itself. This is an important consideration since we want the Facial Recognition Lockbox to be able to run without needing to be plugged into an outlet.

Though, it makes sense that the Raspberry Pi uses a lot of power, as it contains a graphics card for example, it means that we will have to account for time constraints of the box or make it so that the Facial Recognition Lockbox is plugged in at all times.

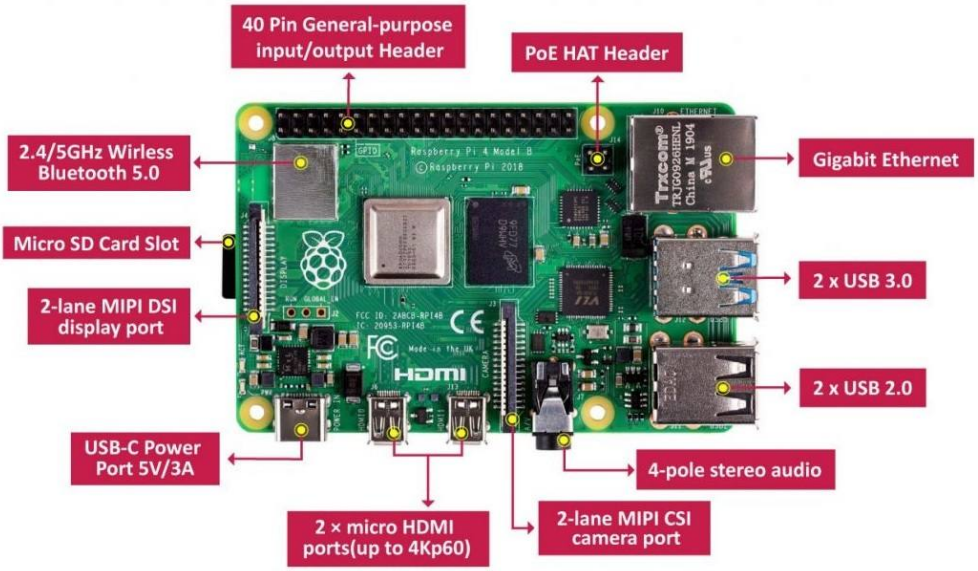


Figure 5 - Raspberry Pi components [28]

As you can see in figure 4, the Raspberry Pi is equipped with a variety of tools that we are capable of utilizing for the Facial Recognition Lockbox. The Raspberry Pi comes equipped with a 40 general purpose pins for the electrical connections needed for the project. Additionally, the micro HDMI ports are perfect for displaying what the user needs to see in order to know they have access to the Facial Recognition Lockbox as well as for the 6 digit code we will implement. The USB ports on the Raspberry Pi makes it easy to plug in a number pad and for the Raspberry Pi to easily read and display said input. With the wifi component we are able to wirelessly connect to the wifi, or if the user has an ethernet port nearby they can use an ethernet cable. Although unnecessary, if the user wants to have a faster connection then that is also available to them. Both the power port and the camera port can be seen in the picture, both are necessary for the Raspberry Pi to function and use the camera respectively. The camera port vs a camera through the USB port will be explored later in the report. Lastly, an important aspect of the Raspberry Pi is the micro SD card slot that can be seen on the left of the Raspberry Pi. Although this component is not important for the user, it is very important for us as this is where the brain of the Raspberry Pi is loaded. The micro SD will have to be preloaded in order for someone to utilize this project. The micro SD will first have to be formatted then you will have to install Raspbian, the operating system specifically made for the Raspberry Pi. After that, you will have to turn on the camera settings for the Raspberry Pi, as it is not turned on upon the first boot of Raspbian.

Can the Raspberry Pi even handle these requirements?:

One of the first things we need to look into when conducting our research on the Raspberry Pi is if it is even capable of facial recognition. As the Raspberry Pi is akin to a mini computer, it is able to do a large variety of things with the right commands. Not only does the Raspberry Pi have its own camera built into the system, it has a few USB ports available to use. By using the USB ports we are able to use a camera that is not only better in quality, but is more flexible in where we can place it than the Raspberry Pi. By using a camera that is connected with a cord, we will not have to expose the Raspberry Pi to the outside, where it can either be damaged by animals or the weather, or stolen. Throughout this section we will go over the Raspberry Pi's capabilities in relation to the Facial Recognition Locker.

The 2 main features of the Facial Recognition Locker is both the ability to recognize and to take user input through a numeric keypad. The Raspberry Pi has 4 USB ports, 2 are USB 2 ports while the other 2 are USB 3 ports. This means that the Raspberry Pi can handle up to 4 USB devices without needing any types of modifications to the system. This is perfect for the Facial Recognition Locker because, at the very least, we will need to utilize 2 of these USB ports. One for the camera that is needed in order to scan a user's face and another for the numeric keypad needed in order to type in the 6 digit passcode. The Raspberry Pi will not only be able to use both of these input devices simultaneously, but be able to handle the input without a problem. The USB camera we are looking at uses at least 502 MB of RAM and 2 GB of RAM at most for HD recording.

Although it is not needed for this project, if we wanted to record and take pictures in HD, we would still have 2 GB of RAM left to work with on the Raspberry Pi in the worst case scenario. Since keyboards don't use RAM, the remaining RAM can be used to communicate with Amazon Web Services. The process may not be the fastest due to the limited RAM, but it will definitely work. Alternative camera options will be explored in further detail in a later section.

First off we have to test that the Wi-Fi module that is built into the Raspberry Pi works properly. Luckily, since the module is built-in, there is no need to install any software and it can be tested and utilized immediately. This means that the Raspberry Pi should be capable of communicating with Amazon Web Services, which, by extension, also means that we will be able to communicate with the Raspberry Pi through a mobile application. The image below shows the Raspberry Pi logging into our Amazon Web Services account through a local Wi-Fi connection.

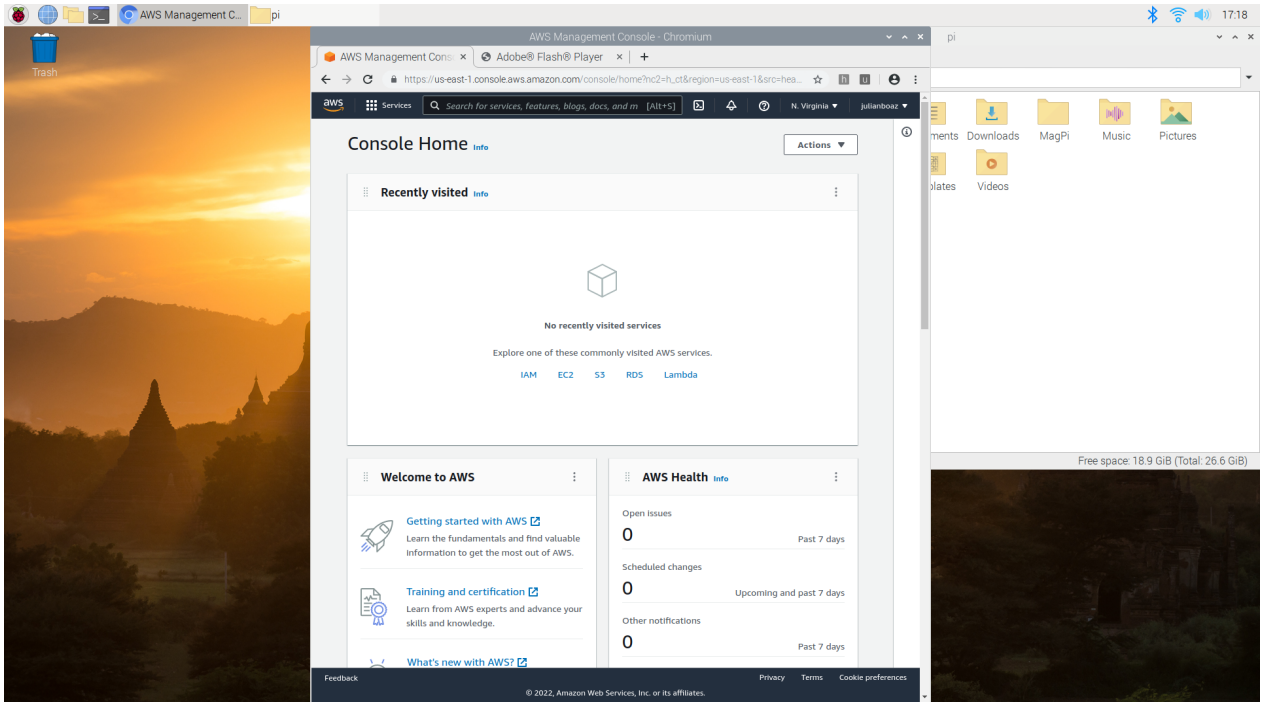


Figure 6 - Raspberry Pi login to Amazon Web Services

Additionally, we have to test the options for the cameras of the Raspberry Pi. As stated earlier, the Raspberry Pi has multiple USB ports that we can use for both a camera and a keypad. Since the Raspberry Pi does not immediately support USB cameras out of the box like regular computers do, some installation is necessary before we can actually test the camera. The software for the USB camera will be installed onto the Raspberry Pi for the sake of testing, however, the steps will not be shown as it is not relevant to the project.

On the other hand, just like the power supply, the company that creates the Raspberry Pi also has an official camera and related companies who supply said camera modules. These camera modules differ from the USB video camera in that they can directly connect to the Raspberry Pi. This is important to note because by connecting directly to the board through the camera port, this camera has less of an impact on the CPU than the USB camera does. Additionally, since this is a camera specifically made for the Raspberry Pi, there is no additional installation needed for the camera to work. All that is needed is for us to go into the settings of the Raspberry Pi and simply enable the camera setting for the device and reboot it. This camera is the most akin to the official camera made by the company who created the Raspberry Pi themself.

Lastly, there is another camera module that is able to auto focus. This camera module can also be directly wired to the camera port on the Raspberry Pi, however, unlike the other cameras, this

one cannot take videos. Instead, it uses a third party software in order to attempt to focus on the object in front of the camera in order to take a picture.

Nonetheless, there is a lot to consider when comparing the three different types of cameras, so both pros and cons as well as the specifications of the cameras will be compared in the tables below. All of the information for both the USB video camera and the camera modules have been taken from their respective distributors, which is Amazon for all components.

	Camera Module	**Autofocus Camera Module**	**USB Video Camera**
Price	$10	$15	$18
Max Video Resolution	1080p	0	720p
Still Picture Resolution	2592 x 1944	2592 x 1944	Not Specified - Can assume to be 1280 x 720
Size	25 mm x 24 mm	25 mm x 24 mm	9.85 x 4.94 x 6.45 inches
Weight	.02 ounces	Not specified	4.6 ounces
FPS at Max Resolution	30	0	30
Angle of View	54°×41°	54°×44°	60°
Focus Type	Fixed	Fixed	Fixed
Connectivity	Infrared	Infrared	USB
Cable Length	15 cm	15 cm	5 ft
Autofocus	N	Y	N
Auto Light Correction	N	N	Y
Image Stabilization	N	N	N

Table 2 - Camera Comparison

	Pros	**Cons**
Camera Module	Small and Light	Very Short Cable
	Designed to Take Both Videos and Pictures	Smallest AoV
	High Quality Videos	
	High Quality Pictures	
	When User Chooses Lower Resolution, FPS Increases	
Autofocus Camera Module	Small and Light	Very Short Cable
	High Quality Pictures	Cannot take videos
	Can autofocus	
	Decent AoV	
USB Video Camera	Decent Quality Videos	Large and Bulky
	Auto Light Correction	Most Expensive
	Long Cable	Not Designed for Pictures
	Largest AoV	Lowest Picture Resolution
		Uses More Power

Table 3 - Camera Pros and Cons

It can be seen through the comparison of these 3 devices that each has its own benefits and drawbacks. Based on our research alone, however, it is very unlikely that the USB camera will be the best for our project. Despite being able to take high quality videos, having an auto light correction feature, as well as coming with a long cable, its weaknesses are apparent and much more detrimental than the others. Not only is it the most expensive, but it does not take the best pictures or videos when compared to the other options. Most importantly, since the other cameras can connect directly to the Raspberry Pi's graphics card, it uses less power than the USB camera. Overall, the cons outweigh the advantages for the USB camera.

Looking at the camera module, there are very few disadvantages we could think of in regards to Facial Recognition Locker. Not only does it take a higher quality video than the USB video Camera, but it also takes high quality pictures. Additionally, the camera is very small so it would also not take much to keep the camera secure and somewhat hidden. An unfortunate aspect of the camera module is that it has a short cable length, however, that can be remedied by purchasing an extension for the cable. Not only is this camera the cheapest, but it seems to be the camera that will give you the best value for the money you have spent on it.

If it ends up being necessary for the performance of the artificial intelligence to include a small video then this camera will most definitely be used. Additionally, if the autofocus feature of the other camera module ends up being less than desired, we will most likely use this camera module.

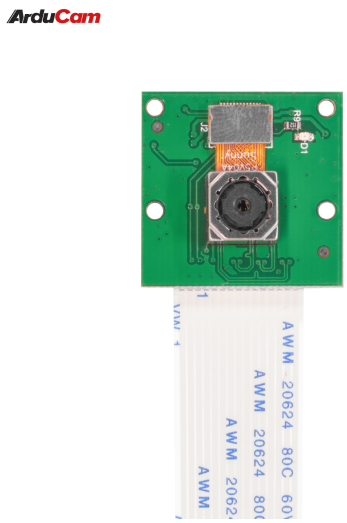


Figure 7 - Autofocus Camera [27]

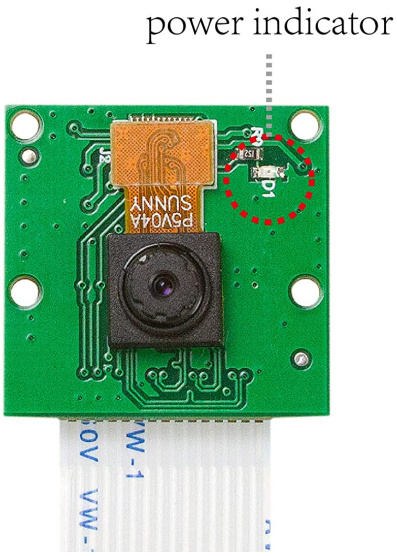


Figure 8 - Camera Module [26]

Lastly, we have the autofocus camera module as shown above. This camera module is very similar to the regular camera module in terms of the specifications of the hardware. The main difference is that this camera module cannot take videos. Instead, it exchanges the ability to take videos with the ability to autofocus the camera. Due to the nature of this project, taking videos is not a necessity, therefore this autofocus camera module is in the same playing field as the regular camera module. The ability to autofocus and how well this camera module can autofocus will be discussed in a later section, however, due to this capability, we are leaning toward the autofocus camera module to be utilized for this project. We believe that the capability to autofocus will result in higher quality pictures that will allow our facial recognition software to be more

accurate. Since the accuracy of our facial recognition software is the most important aspect of our project, high quality images are imperative.

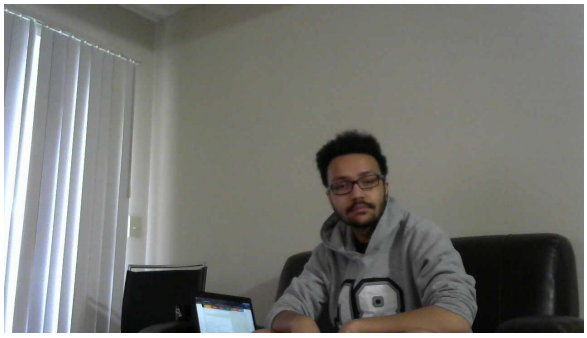


Figure 9 - Picture by USB Camera

Figure 10 - Picture by Camera Module

Figure 11 - Picture Taken Manually with Autofocus Camera Module

Figure 12 - Close up Picture by Autofocus Camera Module

Figure 13 - Long Distance Picture by Autofocus Camera Module

As can be seen in the pictures provided above, the quality of the pictures vary for each camera. By far the worst quality picture taken was with the USB camera. This is to be expected, not only based on our research, but also since the USB camera is not directly connected to the Raspberry Pi's graphics card like the other two camera modules are. Nonetheless, as a result of both the research and the quality of the pictures, the USB camera will not be an option for the project.

The second best quality picture taken was with the autofocus camera. The pictures taken in figures 9 and 10 were taken roughly around the same time, however, due to Che having to hold the camera it is possible that his hand movement caused the picture to blur. Additionally, it is important to note that figures 11 and 12 did not have natural sunlighting when those pictures were taken, reducing the quality of the pictures. Nonetheless, the autofocus camera module worked as intended. Somewhat clear pictures were taken of Che no matter where in the room he was.

In order for the autofocus camera module to function, a python script was run in order to take the picture. The script is available to view in Appendix B, number 6 and can also be found on the website of the manufacturer for both the autofocus and regular camera module. The script correctly identified when an object was close to the camera and when it was far, as shown in figures 13 and 14 respectively below.

Lastly, the highest quality picture taken was with the regular camera module. It provided an extremely clear close up picture of Che in both natural and unnatural lighting during the day when the pictures were taken. As we continue to develop the Facial Recognition Locker, we will swap the cameras to see which camera best works with our artificial intelligence in practice. For now, however, we will use the regular camera module as it provides the clearest pictures. Since we own both cameras, it will not cause any problem to switch out the camera that the Raspberry Pi is using to take the pictures.

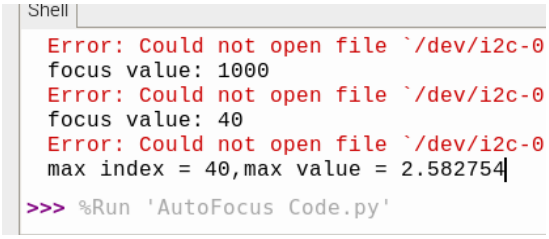


Figure 14 - Close Object

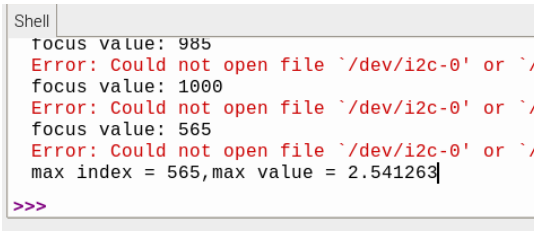


Figure 15 - Distant Object

Now that we have tested that the Raspberry Pi can take pictures as well as connect to Amazon Web Services, let's attempt to put these two things together. At this current moment, however, we cannot upload to Amazon Web Services. This is because as soon as we do, Amazon Web Services will begin charging us for using the account. To avoid having to use more money than necessary, we will be avoiding directly uploading to the database. Instead, we will upload to another cloud service, Google Drive. This test is to simply illustrate that the Raspberry Pi is capable of not only connecting to a cloud service, but also interacting with it.

If this test is successful then, theoretically, the Raspberry Pi will be able to upload and take images from Amazon Web Services. We will consider this test successful if the Raspberry Pi not only uploads an image to Google Drive, but also downloads an image from it.

In order to test this, we will first have to login to a Google Drive account through the Raspberry Pi. For the sake of ease, we will use Che's account since he owns the Raspberry Pi. Since we have already taken multiple pictures through the Raspberry Pi, we will be uploading one of those while also taking a pre-uploaded image of the Raspberry Pi logo found online.

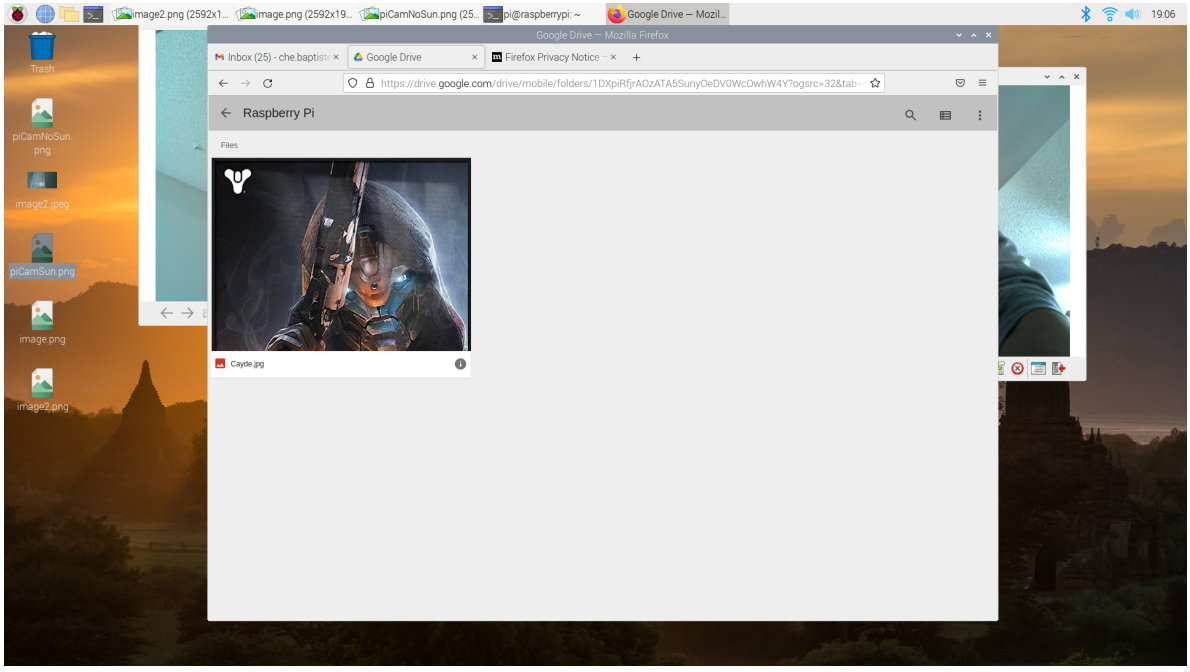


Figure 16 - Google Drive, before Download

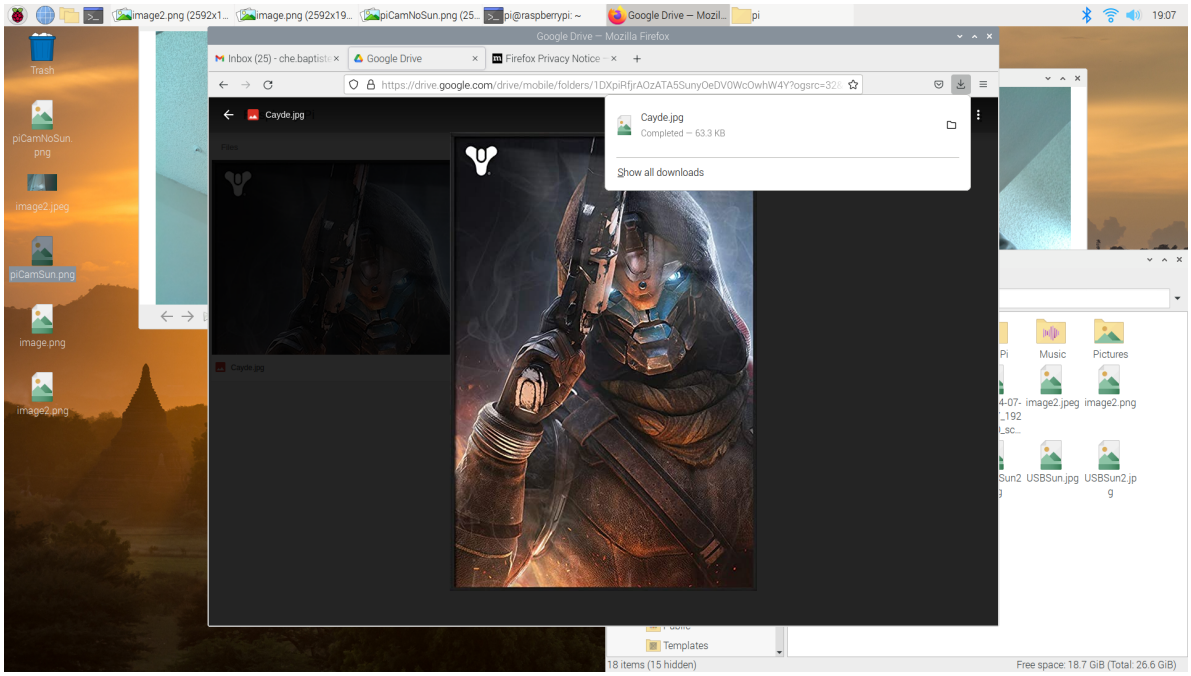


Figure 17 - Google Drive, After Download

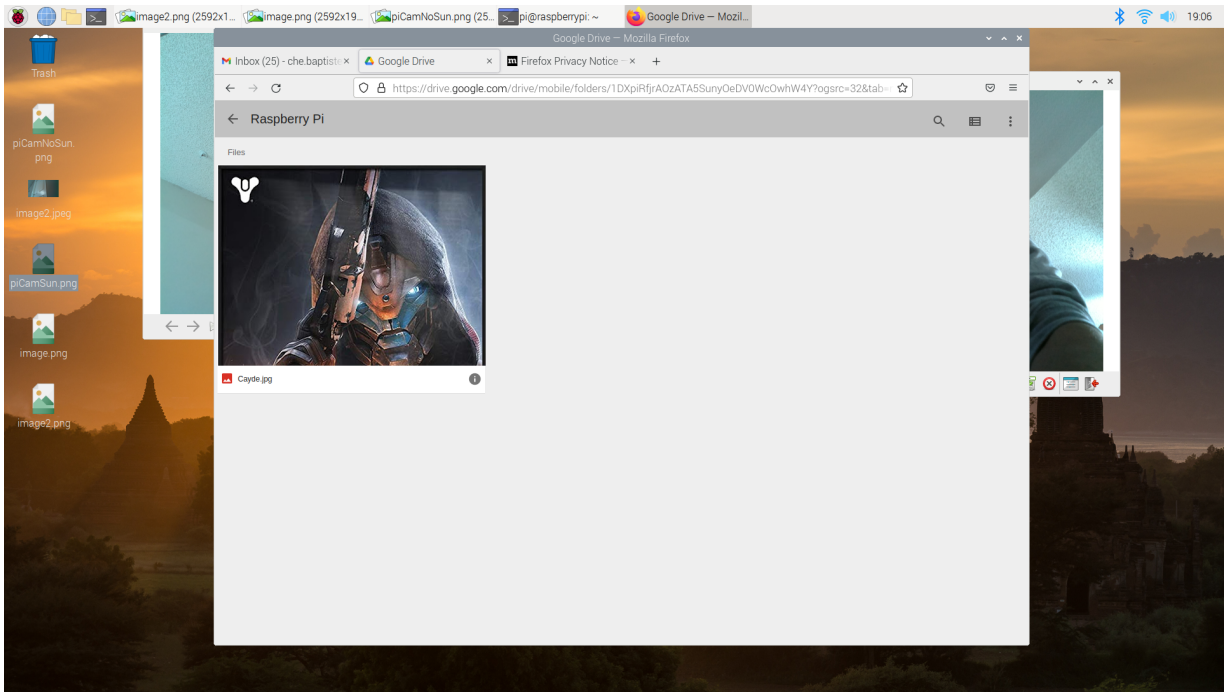


Figure 18 - Raspberry Pi, Before Upload

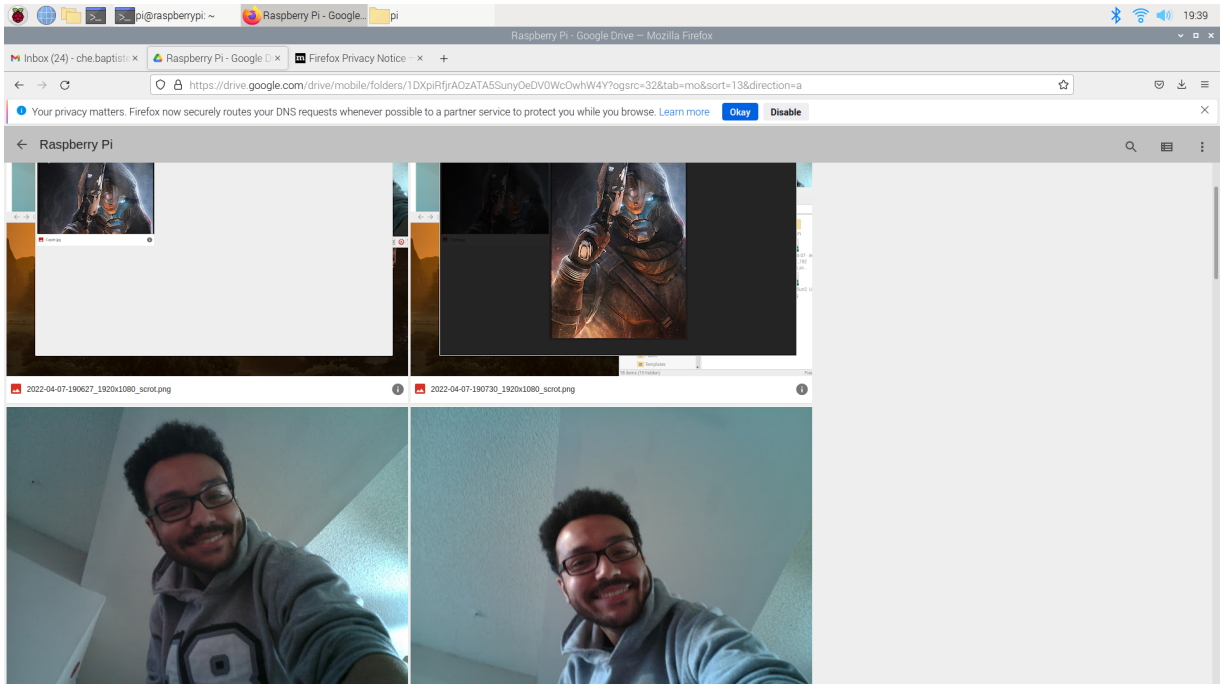


Figure 19 - Raspberry Pi, After Upload

The pictures shown above are both images taken before and after running these tests on the Raspberry Pi. The images are taken from Che's Google Drive as well as the Raspberry Pi's local storage. As is shown in the images, the Raspberry Pi has no trouble uploading an image to Google Drive as well as downloading an image from the platform as well.

It is a bit troublesome to see, however, for Figure 18 you can see the popup showing that the image was downloaded successfully, meaning that the Raspberry Pi was able to successfully download the image. With this, we can consider the test for the Raspberry Pi a success and can assume that the Raspberry Pi will have no trouble uploading and downloading images from Amazon Web Services as well.

It is important to note, however, that this process will need to be automated for the project to be a success. After doing some research, the Raspberry Pi has the capability of keeping a folder in the device synced up with a Google Drive.

Although this test was a success, there will have to be further testing on how to automatically upload the image from the Raspberry Pi to google drive at a later date.

Artificial Neural Networks:

Artificial Neural Networks (or ANNs for short) are an extremely powerful tool for making predictions. There are multiple types of prediction problems that ANNs can be applied to. Generally, these problems are Classification and Regression. Classification is like taking a bunch of pictures of fruits and asking the ANN to predict what fruit is depicted in a picture, while regression can be used to predict values, like the price of stock.

Artificial Neural Networks work by having layers of neurons, however, they are more commonly known as nodes when related to ANNs. These nodes have weights corresponding to them that is a numeric value. While passing in each training data point through the model, each node's weight is adjusted based off of the label that is given to the datapoint, and by the end of the training set, each node's weight should have minimized loss in the predictions. While this is a high level view of a neural network, it is very useful for identifying how they work.

As one can expect, there are many adaptations of Artificial Neural Networks that are optimized for different use cases. One of these that is relevant to our project is the Deep Convolutional Neural Network. While the main idea of a neural network is maintained, Deep Convolutional Neural Networks (DCNNs) will be most useful for image classification because of its ability to break down large data points into bite-sized points, which can minimize the amount of nodes required to go through the images.

The way this operation is performed is by taking a convolution of the image through pooling. Pooling can be configured to take the maximum weight that was identified while training a group of pixels whose size was defined by the user, or take the average of the weights of that group of pixels. This will help with our AI for this project immensely as the size of files containing portraits of people can become quite large, and a traditional forward feeding ANN will require a much higher amount of nodes to be effective in this field.

The Convolutional Neural Network can also help reduce overfitting while training relative to the traditional forward feeding ANNs. Overfitting happens when a neural network ends up learning patterns of the training data more so than real data. So it might seem like our model is performing well when training on a dataset, however, in reality it's just really good at the particular data points it already has the answers to.

A simple yet effective method of reducing overfitting is to reduce the amount of data coming into the network. Convolutional Neural networks can achieve this with the methods described above with pooling where it effectively reduces the amount of pixels per image that affect weights.

Containers versus Virtual Machines:

There are a few reasons that we would consider running these two different apparatuses for deploying our software. We need to have version control of packages in order to ensure our application runs on all different hardware that we deploy it on.

The main differences between a virtual machine and a container is that virtual machines partition hardware and can run whatever operating system the user desires thus making it more secure, while containers must use the kernel to develop the container but since it doesn't have to partition hardware it is much more lightweight, this can allow it to be opened up to several vulnerabilities. Both of these methods would obtain the results that we desire by allowing us to have identical package installations of Python and Python's AI libraries, what it really comes down to is determining whether we would like separation of hardware and more security or a more lightweight implementation and less security.

For us, since all of our sensitive data will be held in the cloud, the obvious choice would be to use containers since we can deploy it on the microprocessor without worrying about resource allocation, and also the ease of setting up new lockers.

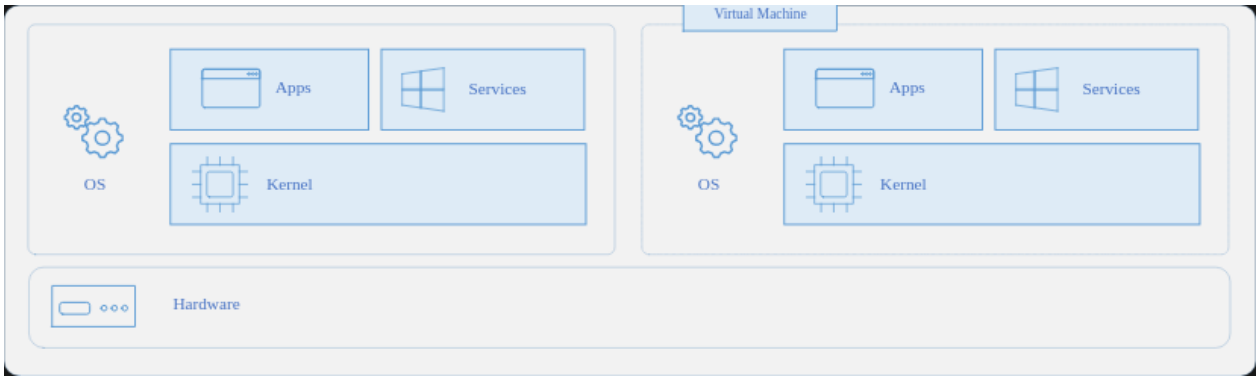


Figure 20: Virtual Machine Breakdown

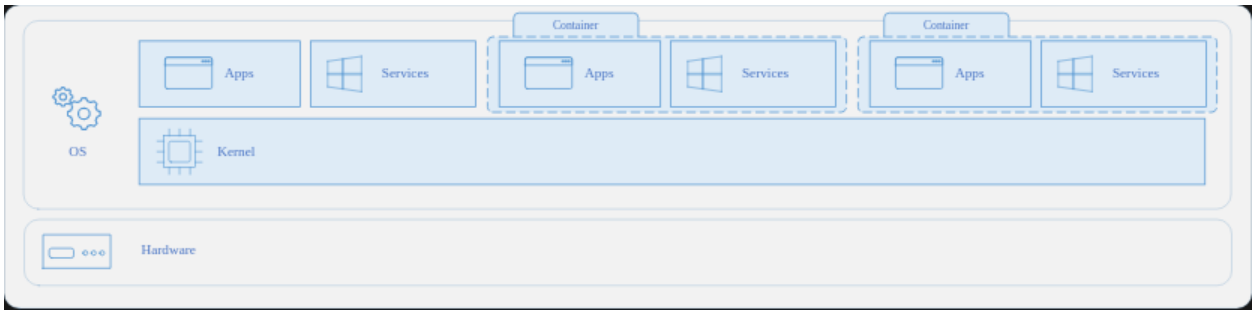


Figure 21: Container Breakdown

4. Standards and Constraints

Project Constraints:

Due to the nature of the project, there will be a number of constraints in order for the box to successfully complete its functions with limited resources. One of the constraints for this project as a whole is time. This is especially concerning due to the fact that since we plan on finishing the project during the summer semester, we lose 2 weeks compared to if we were to finish it in spring or fall. Additionally, all members of the group are full time students. This means that all members of the group must split their time between other classes as well as spending time on this project.

Additionally, we don't have a sponsor for this project. This means that we have to pay for all of the components for the project with our own money. As college students, there is not an ample amount of money we can spend, therefore it is a factor we need to take into consideration. For example, the box may not be as accurate as we aim it to be, because we won't be able to afford the best quality camera available.

Space is also a constraint that needs to be considered when planning and making this project. This is especially the case when considering what the box will be used for. We aim for the box to be small and be able to be placed in both homes and apartments, something that you can set out on a table.

However, as non mechanical engineers, it is beneficial for us to set aside some of our budget to purchase an anti-porch pirate box rather than build one ourselves. As a result, the box we aim to build might take up more space than we might have hoped; however, with both time and money being a concern for this project, this is most likely the best course of action. One of said boxes available for purchase can be seen in Figure 21.

Figure 22. Purchasable Anti-theft Box [25]

Additionally, the places where this box can be utilized is also something of concern. This box can only really be utilized where people live, as companies or schools will not have small objects delivered to them often.

Food is also an existing restraint for the project, as food safety standards are also an important factor to consider. As a result, how long food can stay in the box and if it needs to remain cold could be limiting factors as well.

Another constraint for this project is that we, for academic purposes, have to use Amazon Web Services. Amazon Web Services provides us, as students, an amazing opportunity to use a tool widely known and utilized across a variety of companies. By using Amazon Web Services, we will be able to say we have the ability to use these services and have even implemented them in our own way.

Power Constraints:

With our current design of the Facial Recognition Lockbox, we do not plan on having it be powered on battery alone. Due to the fact that it will hold a few different components such as, the Raspberry Pi, the heating device, and the camera, each requiring a significant amount of electricity in order to function, we decided to use an outlet connection. There are a few constraints that accompany this decision, all of which must be followed for the device to function

properly. The main constraint is the outlet itself; because American outlets have a standard voltage production of 120 V, with a frequency at 60Hz [18], we must design the Lockbox to be able to fully function within this voltage.

One concern that we have since the device will always be plugged in, is we do not want the devices and circuitry to become damaged from overuse. We plan to combat this by having an idle mode that the system can operate in until necessary. It will become fully functional again once it detects a user in front of the camera or the lock pad is accessed. With this functionality implemented, our hope is that the power consumption will drastically decrease, and make the device's lifespan longer, avoiding any heavy maintenance in the short term.

Social Constraints:

Since anti-theft boxes are already a common feature in many people's lives, we do not expect upgrading the pre-existing technology to incite many issues. We also intend for both the Lockbox and application to be user friendly: something simple that anyone can just pick up and be able to use with ease. The electronics will all be hidden from others using the box as well, so it should do no harm to the look or feel of the location it is set in.

Social Concerns:

There is the potential social concern that may arise in the form of the user's electric bill. Many concerns about electrical overages have arisen as products like Tesla cars have become more and more popular; many consumers are wary of products that require a consistent energy source, and we are proactive in believing that concern may relate to our product.

EnergySage conducted research to show the difference between a Tesla's charging port and the amount consumers spend in gas each month; while these figures are not pertinent to our project, the monthly electrical increases reported by Tesla users, is. EnergySage reported that, according to the Department of Transportation's Average Annual Miles per Driver by Age Group Report from 2018 [35], the average miles driven was about 1,100 per month [34]. They calculated that, if the consumers drive around that average and charge their Tesla's at home, consumers can expect their "electricity bill to increase by about $50 each month" [34].

With our design, we are attempting to follow Tesla's suit, and create a power system that does not raise its consumer's electric bill exponentially. We understand that because the Lockbox is plugged into an outlet, that concern is bound to follow. Our plan to combat this is by implementing an idle state where the power consumption is decreased drastically when compared to its fully functioning state, as stated in the Power Constraints section of this document. Based on our estimates, with the decrease in power consumption, the Lockbox should

not exceed the electrical cost of a small mini refrigerator, averaging anywhere between 27 dollars to 47 dollars every billing cycle [19].

Political Concerns:

As for political constraints, there is the singular issue of facial recognition software being a major concern for data privacy. An article written by NPR highlights some of the issues surrounding facial recognition as it was being used with the social media platform, Facebook, last year. The article dove into the reasons the Meta platform was shutting down "its face-recognition system and [deleting] the faceprints of more than 1 billion people," [31] which it was using as a tag-your-friend feature among other experimental services. The company was interested in seeing the positives and negatives of such a service, but it was implied that after Frances Haugen, a former employee of Facebook, brought concerns before the Senate regarding Facebook's blatant disregard for its consumer's safety, the social media platform was intent on playing it safe [32].

So, what does it mean that Facebook, arguably one of the biggest social media platforms currently operating on the internet, is rescinding its facial recognition software? This shift reflects concerns that many consumers have; concerns that are crucial for us to address due to our use of the same software. According to the United States Government Accountability Office's Report to Congressional Requesters from July 2020, newer functions within facial recognition software have attracted many businesses to the ever-growing market. However, with this expansion, the concern for privacy has grown right along with it. Many consumers fear the lack of anonymity while in public if there are facial recognition cameras implemented on streetlights or other traffic law enforcers [33]. They also fear the use of facial recognition software without their consent, and the subsequent storage of this data [33].

Our team understands the above data concerns and take them into account with the utmost sincerity while developing this product. Unlike the databases used by cities or large media conglomerates, our Facial Recognition Lockbox operates on a closed network. While we have a bluetooth compatible application that the users will use to access the Lockbox's functions when away from home, the database that holds their information is a closed one. Many of the consumer concerns revolve around the purchasing and sharing of data within the private sector [33], however, there will be no backlog of information with the Facial Recognition Lockbox. All of the data will be backed up in-house, within the same network, without the use of a cloud or separate network service.

We specifically chose a closed network to ensure the highest level of security for our consumers. Amazon Web Services was chosen as our circuit host because they hold data security above all

else. We understand the external concern, and expect this project to assuage some fear of the technology.

Physical Concerns:

We do not anticipate the physical design of the Lockbox or internal system to have any effect on government or political factors. We intend to illustrate how current technology can be improved while implementing it onto a device that is already woven into the fabric of our everyday lives. The goal of this project is to make it easier for the owners and users of the Lockbox to keep their packages safe and interact with the interface that is maintaining this level of security. The physical structure of the Lockbox should, therefore, have no political impact, as we are not creating something new, and, subsequently, should not invoke any political constraints on this project, either.

Additionally, our team will follow the highest standards possible when producing the Facial Recognition Lockbox. The safety of not only the people who will use the Lockbox, but the environment in which it is in place, is also important to consider. It is the responsibility of us, the developers, to ensure that the Facial Recognition Lockbox can not and will not harm any person or object as a result of using it.

Manufacturability and Sustainability Constraints:

Manufacturability and Sustainability constraints are also vital to include on the list of constraints that need to be considered when making the Facial Recognition Lockbox.

Everything throughout the entire project, from accumulation of supplies to the actual execution of the Lockbox, and therefore there should be no inconsistencies with manufacturing. As students and civilians, we do not have access to any hazardous, potentially dangerous, or extreme materials that cannot be found in stores or online. Because we will not be dealing with products unavailable to the general public, acquiring the products, and possible reproduction of this project should not be a constraint.

Sustainability should also be no issue as we plan on keeping the parts in safe keeping. The most important part will be the Raspberry Pi, or the brain of the Lockbox, which will be kept safe in a case with a fan to keep the microprocessor cool. It will also not be inside the insulation therefore the only thing we have to worry about is the outside weather. Since we live in Florida, cold weather is not usually a factor that needs to be considered, so the Raspberry Pi will have enough fans to make sure that it is cool and able to run for extended periods of time.

Relevant Standards:

Following is an explanation of the relevant standards mentioned in the above sections: the project specifications and constraints submit many different standards, and these are the justifications and details of those standards.

1.1: Food Heating

In short-term food storage applications where food is meant to be kept at a serving temperature, such as in a catered event or a buffet, the Food and Drug Administration recommends that hot foods be kept at an internal temperature of 120-140° F (50-60° C). This temperature ensures that bacterial growth will not be able to begin in the time that the food is left unattended.

Figure 23: 120V / 500W Warmer Element [30]

Keeping the food at the proper temperature can be accomplished with a warming element such as the one shown above. Since a powered element such as this does require a relatively large amount of power (compared to a typical 50 Watt lightbulb), this element is another good reason we are planning to have the lockbox be connected directly to wall outlet power.

This standard may be more or less relevant depending on the specific consumer use for this application; safe holding times as indicated by the FDA is two hours at room temperature, so users who plan on picking up their food from the lockbox in times significantly below that are less impacted by the possibility of the food cooling beyond safe levels. That being said, that same safe holding time is reduced to one hour in environments exceeding 90° F, making specific arrangements for Floridian weather more relevant.

1.2: Food Humidity

Different sources are generally written with different food items in mind; many sources that talk about safe storage for food are talking about storage of produce over a period of three days to two weeks and not storage of a prepared meal for fifteen minutes to four hours!

The rules concerning moisture in food are relatively simple with some delineations made: vegetables should be kept moist and fruits should be kept dry. Many foods that don't fall into either category, such as starches like potatoes, are recommended to be kept in storage that is not climate controlled at all, and only well-ventilated.

The standard that seems most relevant to apply to our application is that of foods meant to be kept dry. This is because many prepared foods that might be delivered with this application (such as sandwiches, fries, rice, etc.) are liable to become soggy or stale if kept in a sealed container with a humidity content that is too high.

Most official sources conclude that any dry food will start to attract moisture content over time if the humidity level remains above 60-70%. Therefore, we will include some type of dehumidifying component in the design of this application.

Figure 24: Purchasable Delvalle Dehumidifier Electrical Enclosure [29]

One such dehumidifying component that we are currently planning on including is a dehumidifier such as the one depicted above. This dehumidifier is aimed at maintaining a low level of moisture inside electrical enclosures such as server towers and electric boxes, and is rated at the lowest level for IP55 when in an outdoor application. This designation means the device has Ingress Protection from water jets produced by a 6.3 millimeter nozzle from any protection.

Ingress Protection Number	First Digit Meaning	Second Digit Meaning
0	No solid protection assured	No liquid protection assured
1	Protection is assured against solid objects that are over 50 mm (e.g.	Protection is assured against vertical drops of water (or drops of water from direction parallel to the surface of

	hands, feet, other large extremities).	installation)
2	Protection is assured against solid objects that are over 12.5 mm (e.g. fingers, toes, other small extremities).	Protection is assured against vertical drops of water when the device is tilted up to 15 degrees from its normal position (see previous note)
3	Protection is assured against solid objects that are over 2.5 mm (e.g. larger wires, smaller tools)	Protection is assured against direct sprays of water at any angle up to 60 degrees
4	Protection is assured against solid objects that are over 1 mm (e.g. smaller wires)	Protection is assured against water splashing from any angle
5	Protection is assured against all solid objects and partially protected from dust ingress; Not fully dust tight	Protection is assured against water projected from a 6.3 mm nozzle from any direction
6	Protection is assured against all solid objects and is fully dust tight	Protection is assured against water projected from a 12.5 mm nozzle from any direction (equivalent to a jet nozzle)
7	N/A	Protection is assured when the device is completely submerged in water up to a depth of 1 meter for up to 30 minutes
8	N/A	Protection is assured when the device is completely submerged for periods of time significantly longer than 30 minutes and at depths over 1 meter
9K	N/A	Protection is assured when the device is subjected to high-pressure water jets (e.g. from a pressure washer) or steam cleaning

Table 4: Ingress Protection Standard Comparisons

It can be safely assumed that if this device is able to safely protect from ingress and negate the moisture effects of an active water jet, it should also be able to effectively remove a significant amount of ambient humidity from the air inside the container.

2.1: Lock Security

The lock specifications for this application mostly revolve around deterrent; the lock must be strong enough to stop any unprepared would-be thieves from simply breaking into the locker to steal whatever may be inside, but not so strong that an emergency responder would not be able to forcibly open the container with a simple tool and disable any electronics or remove a potentially harmful object.

ANSI Grade	Latch Cycle Requirement	Door Strike Requirement	Weight Holding Requirement	Hammer Strike Requirement	Typical Applications
Grade 3	200,000 cycles	2 door strikes	150 pounds	2 hammer strikes	Front door with deadbolt in residential building
Grade 2	400,000 cycles	4 door strikes	250 pounds	5 hammer strikes	Higher-end residential and some “light” commercial applications (e.g. industrial equipment locks)
Grade 1	800,000 cycles	6 door strikes	360 pounds	10 hammer strikes	Highest grade security applications

Table 5: Lock Security Standard Comparisons

Therefore, the standard being used is the lowest one publicly available from the American National Standards Institute, which is Grade 3. Grade 3 locks are locks meeting the generic level of security that would be found on a residential home. They are generally not considered secure enough for commercial building use, but this lower level of security would be ideal for this application.

Grade 3 locks must be able to handle 200,000 lock cycles (locking and unlocking of the deadbolt), 2 door strikes (the number of strike plates the latch or deadbolt passes through parallel to the door; a standard door lock has two strike plates, one on the side where the latch or deadbolt is anchored in the door and one on the side it slots into in the frame), and a 150-pound weight test (where the 150 pounds of weight is applied with a hammer).

ANSI standards for locks include separate standards for the latch versus the deadbolt, but we will only be looking at the lock/latch standards, since the electronic lock will be functioning as the latch in this application.

3.3: Wi-Fi

Wi-Fi is the common name for the entire 802.11 standard, a standard that was originally created in 1997 for wireless internet communication. Since then, a variety of other similar standards have been built on top of the original 802.11 standard, only some of which may be useful for any particular application. Some options have better resistance to interference from other devices, but are more easily obstructed by physical obstacles, while other options have higher data throughput, but are more expensive to implement.

The most appropriate Wi-Fi standard for this application would be one that operates well over a relatively short range (anywhere between 5-20 m), is at least somewhat resistant to obstacles like exterior building walls, and has a data throughput in excess of 16 Mbps (Since each facial recognition profile is expected to be at least 1 MB in size, and we would like to be able to transmit the photo for recognition and have it be returned in less than one second of data transmission time.).

Standard	802.11ac	802.11ax	802.11n
Radio Frequency	2.4 GHz & 5 GHz	2.4 GHz & 5 GHz	2.4 GHz & 5 GHz
Advertized Total Throughput (in megabits per second)	200-1000 (2.4 GHz); 400-4000 (5 GHz)	200-1200 (2.4 GHz); 600-8000 (5 GHz)	10-300 (2.4 GHz); 80-600 (5 GHz)
Cost of Implementation	~$5	~$10	~$4

Table 6: Wi-Fi Standard Comparisons

Giving consideration to the above factors, the best standard to use will likely be 802.11ac, otherwise known as “Wi-Fi 5”. This standard supports higher transmission speeds, is somewhat more resistant to physical obstruction than other standards, and is one of the most affordable standards to implement.

The main drawback of using this standard would be possible interference or obstruction, as this standard uses unregulated frequencies, increasing the chances of interference from other devices, as well as mainly utilizing higher frequency transmission bands, bringing the possibility with it of possible connection loss in settings where there are multiple walls and other devices being used.

However, in the bulk of uses for this application, the somewhat limited range may end up not being a problem: in residential uses where the application is directly outside of a person’s home, the Wi-Fi implementation will only need to penetrate one exterior wall (or door) and reach the nearest router, which in most single family homes will not exceed 15 m, the lowest distance modern 802.11 routers are reported to reach indoors.

In more commercial applications, such as in a workplace or apartment building, the implementation will not have to penetrate any exterior walls, only needing to possibly penetrate interior walls.

This setting would, however, likely have more wireless devices to receive interference from, but as long as the overall distance needed to reach the router was still approximately less than 15 m, this implementation would function fine in almost all uses.

Database Standards:

Naming Guidelines: Amazon Web Services DynamoDB objects, tables, and attributes must be named and have a concise and unique name. Those standards are as follows [13]:

- All names must be encoded using UTF-8, and are case-sensitive.
- Table names and index names must be between 3 and 255 characters long, and can contain only the following characters:
    - a-z
    - A-Z
    - 0-9
    - _ (underscore)
    - - (dash)
    - . (dot)

- Attribute names must be at least one character long, but no greater than 64 KB long.
  The following are the exceptions. These attribute names must be no greater than 255 characters long:
    - Secondary index partition key names.
    - Secondary index sort key names.
    - The names of any user-specified projected attributes (applicable only to local secondary indexes).

DynamoDB also contains a list of reserved words that are pre-programed into DynamoDB that are specific to function calls and operations. It is highly recommended that when creating the database, that users do not use these keywords to prevent errors within the program. If the user must use one of these keywords it is advised to define an expression attribute name to use in the place of the reserved word [13]. Along with the reserved list of words, DynamoDB also has a few restricted characters that already have a defined definition; including the "#" character and the ":" character. The full list of reserved words will be listed in the appendix.

Data Types Guidelines:

Numbers:

Numbers can be positive, negative, or zero. Numbers can have up to 38 digits of precision. Exceeding this results in an exception. Those standards are as follows [13]:

- Positive range: 1E-130 to 9.99999999999999999999999999999999999999E+125
- Negative range: -9.99999999999999999999999999999999999999E+125 to -1E-130

String:

The minimum length of a string can be zero, if the attribute is not used as a key for an index or table, and is constrained by the maximum DynamoDB item size limit of 400 KB.Those standards are as follows [13]:

- For a simple primary key, the maximum length of the first attribute value (the partition key) is 2048 bytes.
- For a composite primary key, the maximum length of the second attribute value (the sort key) is 1024 bytes.

Binary:

Binary type attributes can store any binary data, such as compressed text, encrypted data, or images. Whenever DynamoDB compares binary values, it treats each byte of the binary data as unsigned.

The length of a binary attribute can be zero, if the attribute is not used as a key for an index or table, and is constrained by the maximum DynamoDB item size limit of 400 KB. Those standards are as follows [13]:

- For a simple primary key, the maximum length of the first attribute value (the partition key) is 2048 bytes.
- For a composite primary key, the maximum length of the second attribute value (the sort key) is 1024 bytes.

Morality and Ethics of User Data:

Dealing with user data comes with a great amount of responsibility to the clients that use this product as well as their clients that the product interacts with. In our project we're dealing with many user inputs that are sensitive. This sensitive data includes locations, full legal names, pictures identifying the user, as well as contact information such as email and phone number. We as a group have an obligation to keep this information both secure from attacks and bugs.

In terms of protection from hacking or attacks, we have decided that hosting the information on an Amazon Web Service database would be the best practice. Amazon Web Service provides encryption through their services that will be much easier for us to implement into our product as we do not have to develop an API with encryption for a database that we have to host, all of this is managed for us. Through Amazon Web Service, we can use their API to make calls onto data with a key that is specifically for our database. And so the question is, where would our vulnerabilities be?

The main concern for our product in terms of vulnerability would be where we make the API calls. We want to make sure that an individual cannot simply access someone else's data by typing in a user's name or phone number and drop a table with their last known location, or perhaps even an image with their face on it. So we must be careful to design our application with good quality functionality, but also to protect the user from potential vulnerabilities.

Another thing to consider is how the neural network can be used and how that affects our users. Consumers always worry about privacy and rightly so. So we must be cautious about how the AI model is deployed. The AI must be used solely for a quick verification that the right user is

attempting to open the locker and nothing more. This AI could potentially be used to identify people walking through public transport stations, different businesses, government buildings, and all sorts of other places, which could lead to a massive breach of privacy.

Imagine the model being able to identify you walking into a courthouse and it logs the time and date you were in the building. This could potentially be used maliciously by an entity to blackmail you if the case was criminal, and likely for other purposes that are too sinister to imagine. It is paramount that the AI cannot be used for "Big Brother" scenarios as that is not the purpose of our model.

Including the encryption from the Amazon Web Service database, we will be encrypting the passwords that the user chooses inside of the application using an MD5 algorithm. This will create a 128 bit hash value that will be used to represent the password! So if for some reason the Amazon Web Services account is compromised, the attack cannot uncover the actual password and will not be able to log into that user's account and/or uncover critical data. While MD5 isn't the most secure (there are algorithms that produce 256 bit encryption), MD5 is quite easy to implement and will be an extra layer of protection for clients of our product.

Ethical, Health, and Safety Constraints:

Health and safety constraints as well as ethical constraints are very important factors that need to be looked at since the Facial Recognition Lockbox will primarily be used to deal with food. The Facial Recognition Lockbox will be able to hold food for a certain period of time by keeping the area cooled or heated. It will not be able to keep frozen items frozen, and this is due to the economic constraint of the project. Frankly speaking, a cooling unit is too expensive for the scope of this project.

However, even if it is impossible to keep frozen food frozen, we can keep the food safe by installing insulation into the box. By installing insulation, we will be able to keep the temperature of the food closer to what it was when it was entered without it potentially damaging the hardware. The safety standards of food are talked about more in detail in the Relevant Standards section of the paper.

Due to the nature of this project and the installation process, there are certain guidelines that should be followed. As producers, we intend to follow them, but recommend any user planning on installing this item themselves to also follow lifting and carrying guidelines presented by UNC's Institutional Integrity and Risk Management:

> "Get as close to the load as possible. Try to keep your elbows and arms close to your body. Keep your back straight during the lift by tightening the stomach muscles, bending at the knees, keeping the load close and centered in front of you, and looking up and ahead. Get a good handhold and do not twist while lifting. Do not jerk; use a smooth motion while lifting. If the load is too heavy to allow this, find someone to help you with the lift," [14].

We also recommend the users follow the UNC Institutional Integrity and Risk Management's carrying guidelines as well, which states:

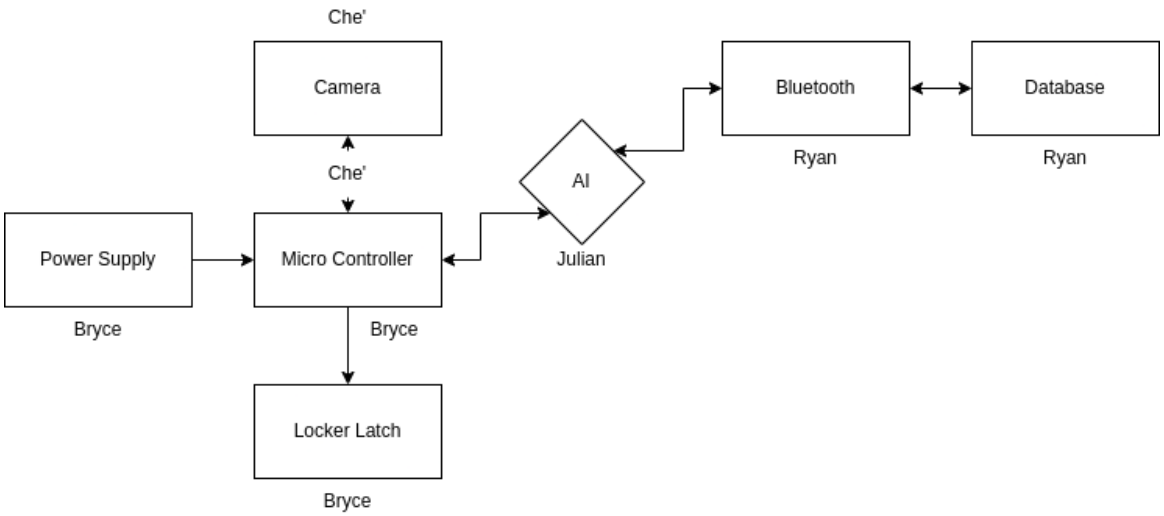
> "Do not twist or turn the body; instead, move your feet to turn. Your hips, shoulders, toes, and knees should stay facing the same direction. Keep the load as close to your body as possible with your elbows close to your sides. If you feel fatigued, set the load down and rest for a few minutes. Don't let yourself get so fatigued that you cannot perform proper setting down and lifting technique for your rest," [14].

5. Project Hardware and Software Design Details

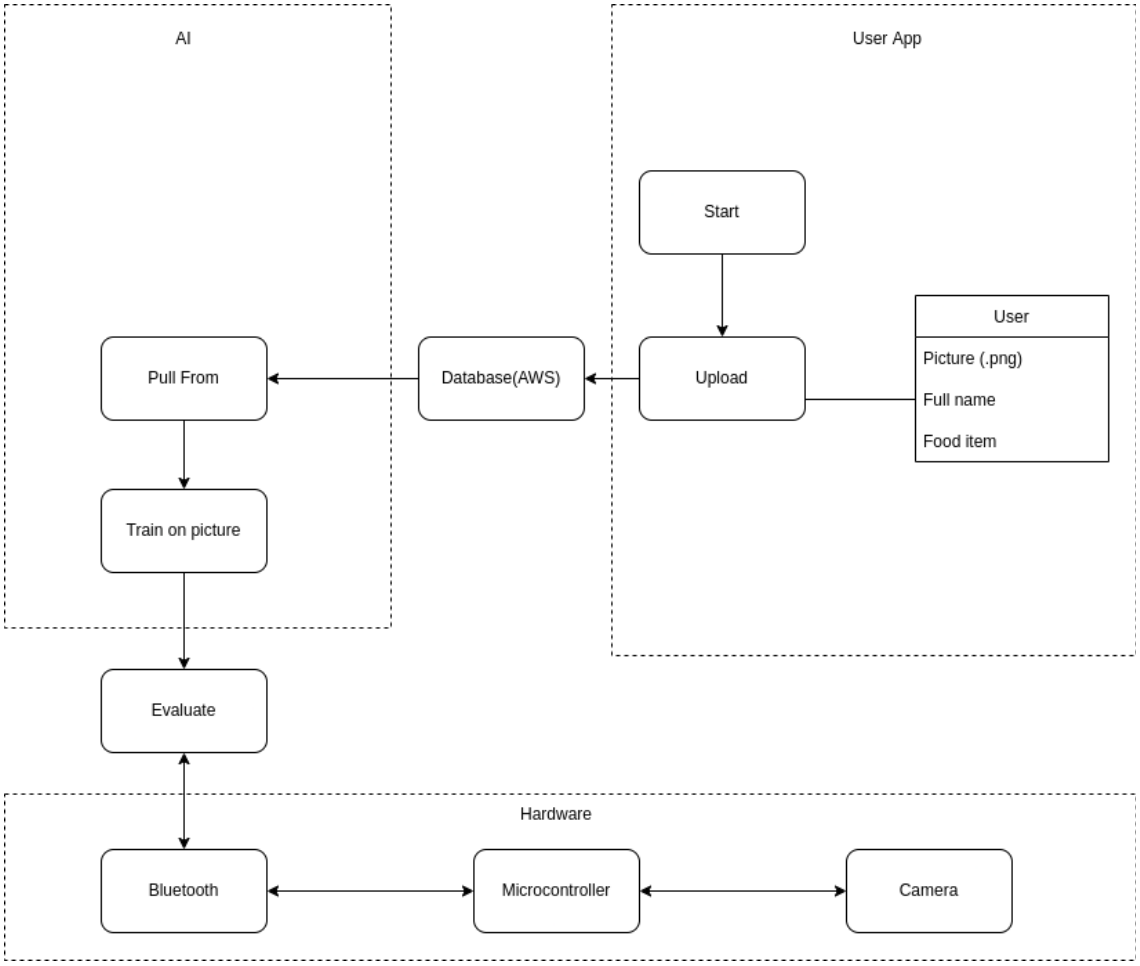
Project Block Diagrams:

This first diagram is to describe how all of the hardware will interact with each other. So the microcontroller is the heart of our system. Bryce will be working to connect a bluetooth module and camera module into the microcontroller. The bluetooth module will be able to send and receive data from the server that contains all of the faces of the users that have orders, one of which will be selected of course, Ryan will be working on this. The camera will be continuously polling to see if the current object on screen is actually the user waiting to pick up the food, and Che' will be working on this. Julian will be working on the AI that works on taking in data from the Camera and matching it to the user that has been selected. Once the AI has determined that the object on screen is the user, the microcontroller will unlock the latch on the locker, allowing the user to take the food inside.

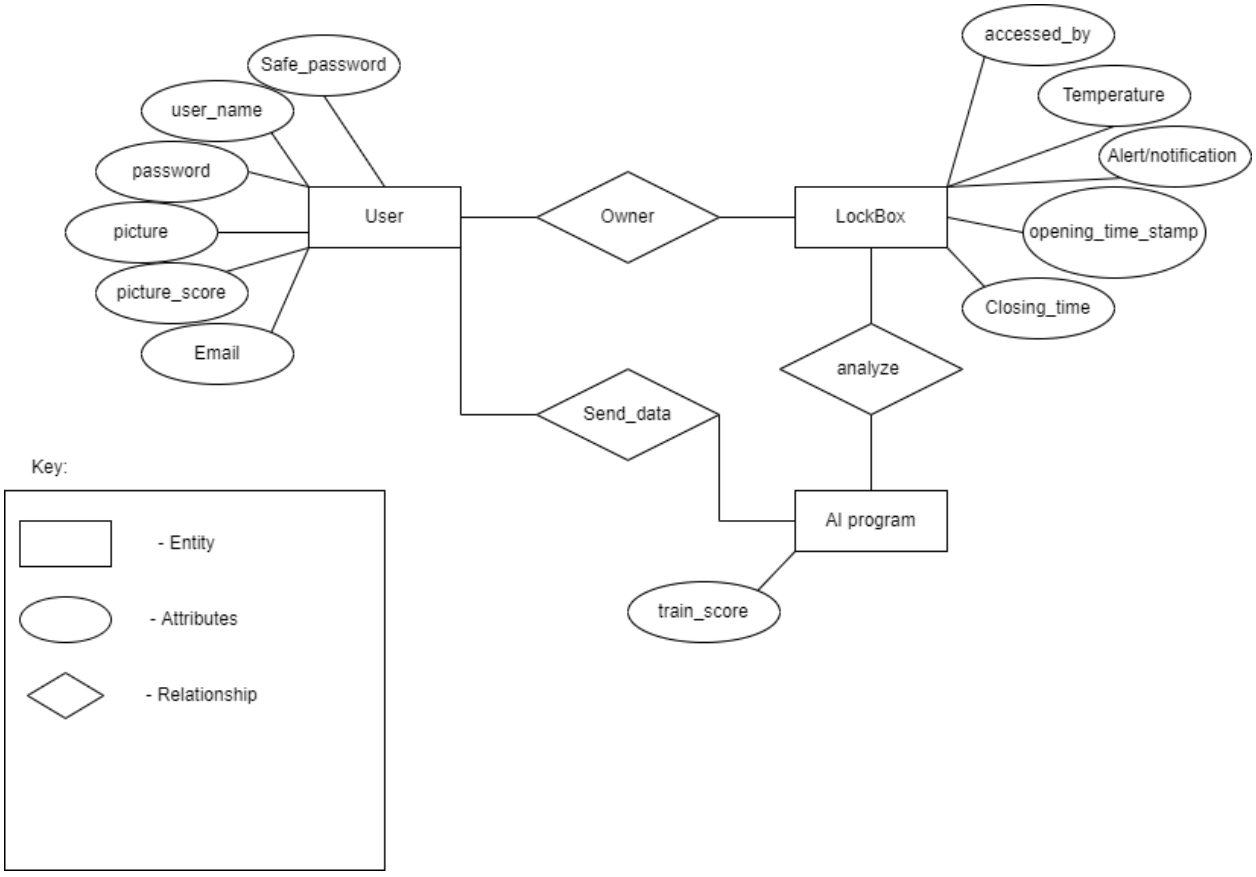
Hardware Diagram (Figure 24):


Che'
Camera
Che'
Bluetooth
Ryan
Database
Ryan
AI
Julian
Power Supply
Bryce
Micro Controller
Bryce
Locker Latch
Bryce

Software Diagram (Figure 25):



ER Diagram (Figure 26):



An ER diagram is a great way to show the relationship between entities in a database. It contains a host of options that allow users to show different relationships. These options include: one-to-one, one-to-many, many-to-many, and much more. It's a simple diagram design that lists all the created tables and their internal attributes. It also clearly depicts how each entity will be connected with one another and what way that connection is made.

Step Motor:

In order to unlock the box, we will need a mechanical device to move a dead bolt in order for the door to open. In order to accomplish this, the Raspberry Pi will need to be able to control a step motor accurately based on whether a user has been authenticated. The best way to accomplish this will be to have a step motor that is controlled via a Python library since Python is natively installed on the Raspberry Pi. Python can also deploy the application to send and receive data from the Amazon Web Services so the step motor API can be used within the same application which will reduce complexity.

One solution we've come up with is to use the NEMA 17 step motor along with the ADAFRUIT Stepper Motor. They're relatively cost effective, and will work quite well with the Raspberry Pi.

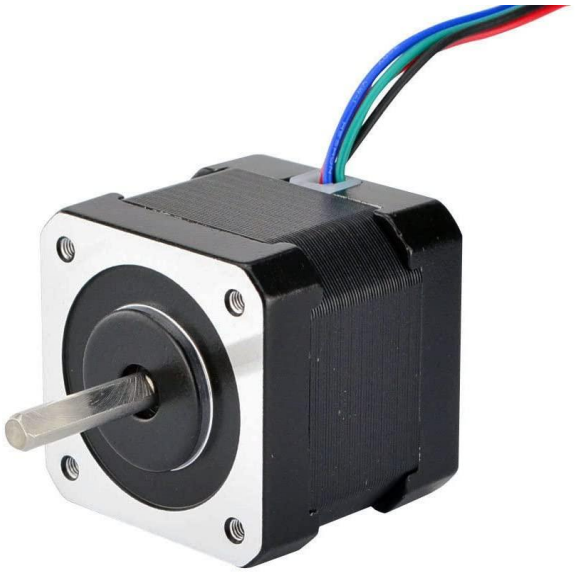


Figure 27: NEMA 17 Step Motor []

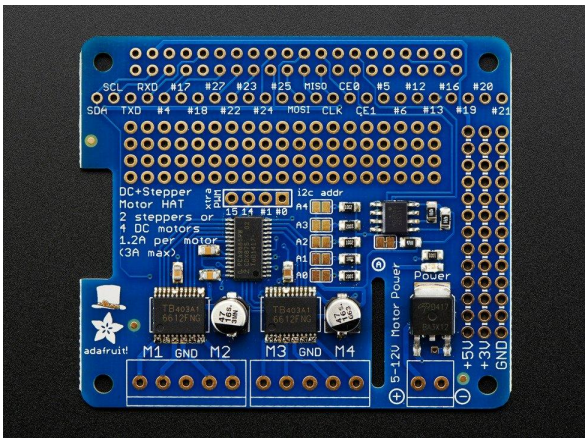


Figure 28: ADAFRUIT Stepper Motor Hat []

Using this combination of hardware will allow us to efficiently control the step motor via Python libraries. There is a github repository that is open source that hosts the libraries needed to control the stepper that is made to be run on the Raspberry Pi.

Github Repository Link: https://github.com/adafruit/Adafruit_CircuitPython_MotorKit

The actual Step Motor can be acquired from Amazon for $12.99. The ADAFRUIT Stepper Motor Hat can be acquired directly from Adafruit, however, finding it in stock could be an issue, so alternatives might be necessary.

The ADAFRUIT motor hat will have to be soldered onto the Raspberry Pi, so the cost of soldering will have to be considered in our final pricing analysis. Bryce will be soldering the motor hat onto the Raspberry Pi and testing to ensure the connection is functioning properly.

One last thing to consider when using this combination of hardware is the amount of power that it will consume. The manufacturer recommends having the Raspberry Pi plugged into the wall with a standard power adapter rather than being battery powered. Logistically this makes sense for this product as we would like to reduce the amount of time that the product is unusable as well as reducing points of failure. Having the locker be battery powered could reduce the effectiveness of the step motor or even not allow it to work at all, thus a wall power adapter will be used.

The stepper motor will be used to reciprocate a deadbolt lock into the door of the locker that has a bracket attached to it. When the deadbolt is moved inside of the bracket, the locker will be unable to be opened. With one API call within the Python script, the deadbolt can be reciprocated. The step motor as well as the deadbolt lock and bracket will be on the inside of the door to of course maintain security as well as ensure that the electronics are not affected by weather.

Artificial Intelligence Model:

In order to detect the user's face, we will be using an artificial intelligence algorithm. The most effective way of doing so will be using Python. Python has many useful libraries for facial recognition, however, OpenCV seems to accomplish the task required. Ideally we will have the model train itself on each new user's face within the application that will be integrated with the lock box, and thus there will be a separate model for each user' face. Once the user goes to pick up their item from the lock box, the model that was trained on their face will be selected to perform its classification on the face that is within the camera's bounds, unlocking the box if there is a match.

Something to note is that OpenCV is a facial recognition software, but it is not an AI model. OpenCV will be used to detect faces that the camera is looking at, and the model will be used to identify who's face it is.

Another option to be explored and developed will be using the built-in artificial intelligence tools that are provided through the Amazon Web Service. Amazon Rekognition is the primary tool to be used when developing an AI pipeline through Amazon Web Services. Amazon Rekognition provides *pre-trained* models that are optimized to process data through customizable computer vision tools. Amazon rekognition can detect many different things through images such as faces, text, celebrities, and other useful things.

One of the things that their marketing has been pushing is the fact that it can detect PPE equipment to ensure proper safety measures are taken in the workplace. So needless to say, it is a very powerful tool for facial recognition. It could be incredibly useful for this project as this tool has been optimized over and over from a company that can throw millions of dollars into research and development. So using this will likely be beneficial.

The Amazon Web Service can store all of the data needed to run the artificial intelligence model. This includes all of the users and their faces, the metadata associated with the faces once the model has trained on them, as well as the labels that have been associated with each face. The model will use the metadata for each face to make predictions on faces that have been sent to the Amazon Web Services from the microprocessor. The Amazon Web Service will also store the two factor authentication code to be paired with each user.

Something to consider about using facial recognition with artificial intelligence is that the model will almost never be one hundred percent certain that its prediction of the user that is on camera is correct. And unfortunately, the certainty of its prediction goes down when the amount of users in the database increases. And so optimization of the algorithm will likely be needed once more users are added, or even using a different algorithm altogether.

To familiarize our group with how well these models would predict faces, Julian wrote a model to predict faces using the packages that Che' suggested (OpenCV) and Keras and TensorFlow in Python. The model was able to predict Julian's face with one hundred percent accuracy. However, with five different faces added to the database, the accuracy of the model quickly dwindled to lower than ninety percent, even with over five hundred faces per user to train on.

While it is likely that the Amazon Web Services Rekognition model is much more accurate overall with much less data, this is still something to consider when dealing with a large number of users that could affect a lot of things. Namely security. Even though it is unlikely that this will be penetrable due to two factor authentication, it's possible that a picture of a user could pass as being the actual user instead of that user being there in person which is a huge security risk.

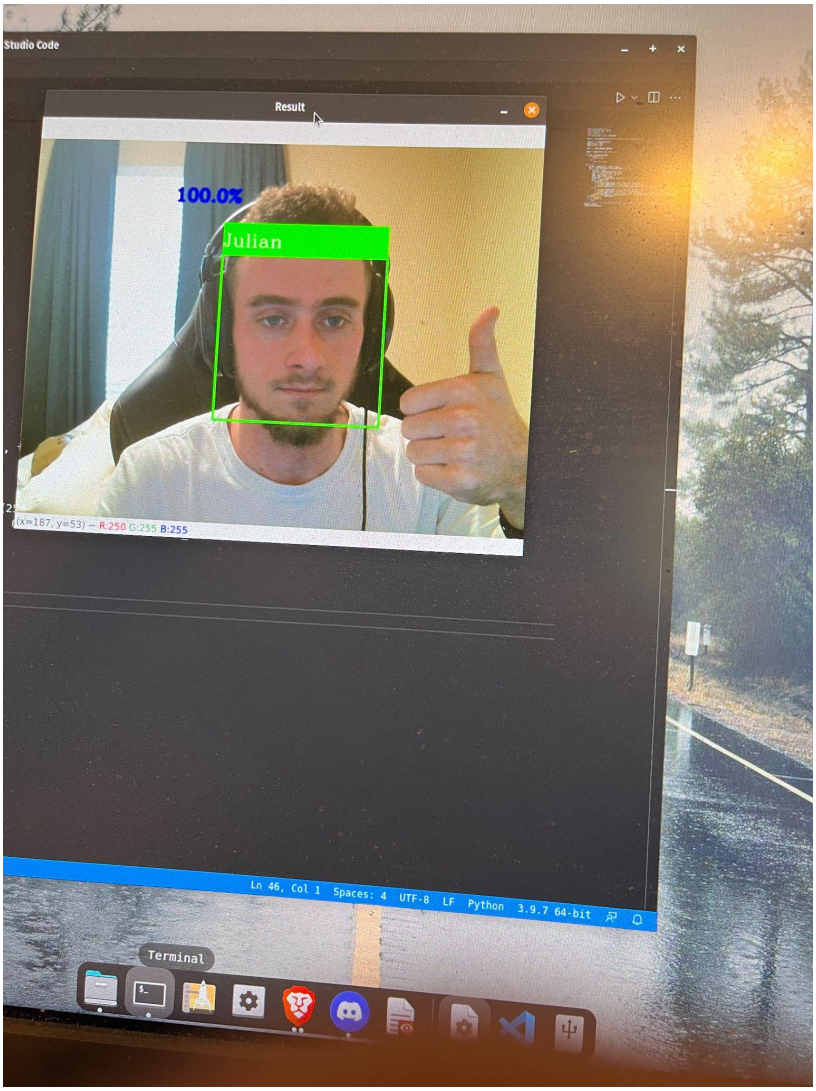


Figure 29 - Identity Verification

In order to use Amazon Rekognition for identity verification, we can create a collection. This will be used to store faces for the model provided. We can actually accomplish everything we need to do within Python using a package called Boto. We can use boto to create this collection which doesn't just store the photo itself, but also information for all of the known faces as well which will be used by the model in order to make predictions. Collections in general can store all sorts of information, but applies to this problem perfectly so we can adapt them to work with our data that we will be feeding it.

When adding a face to a collection, we can actually use OpenCV to take a single photo of the user and push it to the collection. These collections store metadata inside a JSON object which can host all sorts of useful information, like Gender, if the user is smiling, if they're wearing glasses, and what kind of emotions they're expressing. We can do all of this by using a method

provided by the Boto package called *detect_faces* which can translate all of the information acquired into that JSON object mentioned earlier. All while pushing this information into the collection.

Once this user has been added to our database, we can use a method provided from Boto called *SearchFacesByImage* which will compare the photo passed in to all of the photos in the collection and determine if there is a match. These matches are given based on a similarity percentage which we can set a threshold for. For example, if the certainty that a match has occurred is below seventy percent, then we can assume that that prediction isn't correct and the user isn't in the database. Unfortunately, determining the threshold for which a match has occurred will likely be the trickiest part in optimizing our software to run smoothly for every user given that all user's photos will be taken with varying quality and lighting.

AI Specifications:

Sp1	There must be a minimum of 500 data points per user for training
Sp2	The model must be able to identify a user within 80% accuracy
Sp3	The model must generate a model file that is retrievable for future evaluation
Sp4	The model must be able to train on photos received from the camera
Sp5	The model must train itself on the data points within 5 seconds

Table 7: AI Specifications

These specifications are subject to change in the future while research is being done. Another thing to note about AI is that the model can only be as good as the data, meaning that the quality of each photo will affect the accuracy of the model. Depending on whether we use infrared cameras or a digital camera to take photos of the user will change how the model is developed as well. It is also possible that 500 photos of the user will not be enough to have an accurate model, or perhaps more than enough in which we could reduce the amount of photos taken to save memory. However, the ability to have a model saved for the future will be required as well as taking photos from the camera (even if the photos are a matrix of dot projections from an infrared camera).

The specifications that describe integration with the other mechanisms of the project will be discussed in the future once more research has been done. In terms of cost, the research and development of the model will be free.

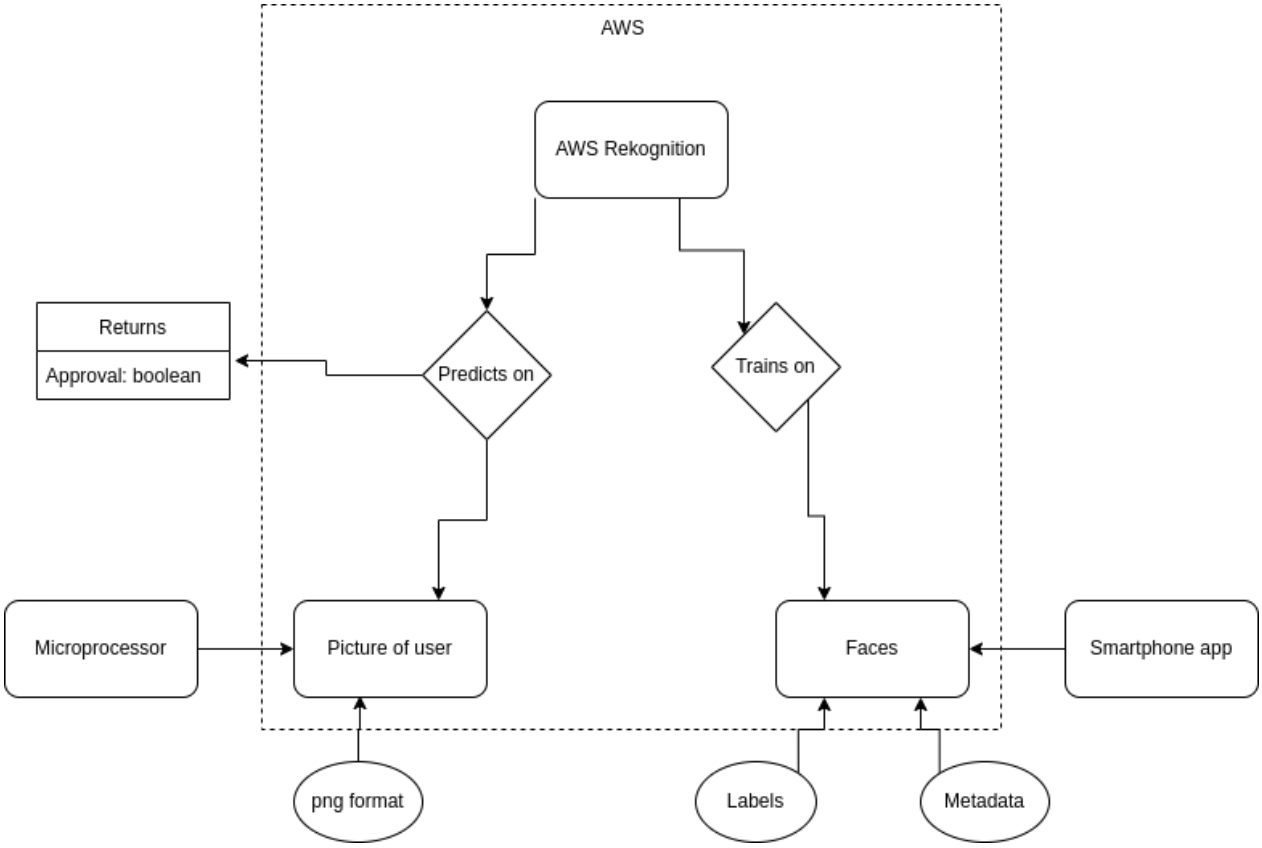


Figure 30: Amazon Web Services block diagram

Application:

The application will be on the user's phone and it is used to train the model on the user's face and also facilitate the interaction with the hardware to ensure the user is the correct user. This entirely depends on whether we can integrate python within the application. The application will be written with React Native so it will be compatible with both iPhone and Android products, without having to write the application in two different languages. The application will have to be able to communicate with the hardware through bluetooth. The data that will be sent will likely just be data about the user and what they ordered.

And as mentioned in the previous section, if the model is trained within the application, a reusable model should be transferred as well. Ideally the user's application will store their own

model within their device in order to save space on the lockbox's Raspberry Pi, and the React Native application will retrieve the model from memory and pass it to the Pi in order to evaluate the user's face. Otherwise, all models will be trained and stored within the Raspberry Pi.

Cost for this could vary depending on how we store data. If we use the Amazon Web Services servers to store data, the cost is written above, however, everything could potentially be stored locally, allowing for free storage and management as well as a simpler and more secure program.

Security:

There are a couple of things we are going to use in order to ensure the user is the only individual that is capable of opening the locker. Firstly, we will be storing the model that is developed to identify faces on the Amazon Web Service database. This will ensure that the model isn't able to be attacked locally. Otherwise we will have the two main methods of identifying an individual. The technique we are using falls into the category of Two Factor Authentication.

For Two Factor Authentication, there will of course be facial recognition software, but also a code that only the user has access to that will be needed in order to prompt the camera to start polling for a face. Neither of these authentication processes can solely allow the user to access the locker and thus will create a higher level of security simply by adding an additional layer.

Essentially what will happen is the user will walk up to the locker and type in their code, in which the locker will then check the Amazon Web Services database to see if the code is associated with the user. If it is confirmed to be associated with their account, then the camera will simply take a small amount of pictures of the user currently on screen and send those pictures to the database where the AI will then process its predictions and send the predicted labels back to the locker. If the locker detects that the labels sent back from the AI matches the user data that it has a parcel inside the locker, then it will unlock, allowing the user to take their parcel. Once the door has closed, it will relock and clear the code, as well as the user data associated with the previous order.

As stated earlier, we will also use MD5 encryption when storing passwords. While MD5 is still vulnerable to brute-force cracking, the hash would only be uncovered if the attacker were to gain access to the Amazon Web Services account and manually log in, which is unlikely, given the Amazon Web Services account has two factor authentication as well. So there are plenty of layers of security an attacker would have to go through in order to penetrate the database and uncover sensitive user data. This is not feasible in the practical world. Due to this we are confident that the user's data will be unreachable by most attackers.

Deployment of Software:

The bane of existence for most software developers is the fact that the software that they're testing works on their computer but not on the test machines or their team machines. This can become a time consuming task as it takes the developers away from development and forces them to set up each individual test unit to have the same packages available as well as the same versions of each package. For this, developers typically will choose between creating a Virtual Machine or containerization.

Each of these different methods have their own benefits. With virtual machines (vm), you have higher security. This is due to the fact that the vm partitions hardware resources and runs on its own personal operating system. The operating system can be completely different from its host operating system, so there's lots of flexibility with vm's. Storage tends to be easier to manage as well.

However, containers will be the clear winner for our project as it is much more lightweight. Containers build off of the kernel of its parent operating system and don't partition hardware resources. This will be extremely beneficial for our project as the host will be a deployed Raspberry Pi that will have limited resources to share, and will likely be pushed to its limits.

The current method of deployment for our project will be maintained through Github, and hosted on Docker Labs to containerize the application. This way, deploying the prototype or finished product will be extremely simple, and can be done remotely. Having the program hosted on Docker Labs in an image format will allow us complete control on maintaining dependencies as well the packages and versions of those packages that we deploy each time. Since the image contains all of the packages and dependencies and versions required to run the program, the program will run the exact same each time it is run, and on each machine it is installed on.

Realistically, we would like to have a native install eventually as that would be even more lightweight, but for prototyping and development, containerization will be utilized.

In order to streamline the development progress, a Github Actions pipeline will be utilized. Our Github Actions pipeline will consist of a linter, which will run through each file and compare them to each language's standard styling method (PEP8 for Python), as well as looking for any errors that could have been missed by a user. The pipeline will also automatically push the entire

program to Docker Hub where an image will be built and hosted in a repository for easy access and deployment.

While we will have applications on mobile devices and on the locker itself, we will be using Amazon Web Services to host the AI. Hosting the AI on the Raspberry Pi would require the pictures to be low resolution, as well as the model being more simple, and if we hosted the AI on the application, the user would suffer from having a model that is training and updating on their phone frequently. So the obvious choice is to host the AI on Amazon Web Services. This way, there is one central place to store user data as well as the models that are built upon the user data, and API calls to the Amazon Web Services server will be simple .

Realistically, for our budget, an Amazon Web Services application is also going to be the most cost effective method of transferring data. There's not enough storage on either device to store these models, especially if it consists of hundreds of faces, the model could potentially be huge, and the processing power to analyze a face by retrieving the model from the cloud and running it locally would be too great, and so all the calculations will likely be done on Amazon Web Services servers as it will be cheaper in the long run.

It is important to note that the deployment of the mobile application will be different from the deployment of the microprocessor application. The mobile application will have to be native since it will be hosted on the Apple Store as well as the Android play store. Luckily, the deployment of software through those two hosting methods are very similar; it will be simple to stay congruent throughout the packages used to write those applications and maintain version control. Keeping the mobile application and the microprocessor application separate will allow us to simplify the development process, use different languages to develop each, and isolate issues within each program as well.

While it is possible to utilize containerization in order to deploy mobile applications, it is a little difficult given the kernels that are available among these devices. Even though Android is the largest mobile platform in the world (not in the US), Apple products run their own proprietary software which doesn't have the linux kernel embedded within it, not allowing a container to be deployed on both devices. There would actually have to be two different containers for Android devices, which do have a linux kernel, and Apple's proprietary kernel, which defeats the purpose of using containers for the mobile application. Either way, using containers to deploy to a mobile device would be overkill for our application and thus using Docker for just the microprocessor will be sufficient.

To put it into perspective on how important Docker is to the continuous integration pipeline of this project, a popular linux distribution has already been updated during the writing of this

document, which has subsequently changed the native version of Python. If we were to use the updated version of this distribution on our Raspberry Pi, then we would have to change a bunch of dependencies and packages and initializations of functions in order to properly deploy the software again, or, if we wanted to use an updated version of Python and hadn't upgraded the operating system of the locker (which is much more likely), than the container could just be upgraded and redeployed without any hassle in terms of compatibility. Now unfortunately we will not have this same luxury with the mobile applications, however, there are only two operating systems that we will have to worry about when developing them and making sure that all of the packages used are up to date at any given time.

In summary, Docker will be incredibly important for maintaining our locker. It will greatly cut down on labor in regards to bug fixes, further updates and development. Using the pipeline to create the container we can also fix issues ahead of time before even pushing changes to the delivered product to ensure that our deployment of the software is smooth and always working.

Mobile Application Environment:

A major component of our project is offering the user an easy to use way of having access to the information as well as the features of the box remotely. Since most people have a smartphone, it will be very easy for most people to have access to the application, making it the most accessible platform. Another reason we chose to go with making an application the main form of communication between the user and the box is that the smartphone is almost always with the user, both inside and outside of the home. When you go outside, you will not always have access to a computer for example, so making the smartphone the main form of communication is ideal for updating the user while allowing them to make decisions in regards to the box.

Lastly, there are a lot of things to consider when designing a mobile application for the user, these things include how the mobile application will look as well, the platforms it will be available on, and the functions available to the user.

We have decided that the best platform to develop the application would be on Android. This is simply due to the fact that Android is a lot friendlier for novice application developers as well as it is free. As a group, we are able to utilize Android emulators in which we can both test the application as well as develop through it. There are a variety of different ways to code and develop the application, as there are a lot of good IDEs available to work with.

We have chosen to use Visual Code Studio with flutter. This is due to the fact that the software developers of the group have some experience with these programs in terms of application

development, so we will have an easier time developing the application and will be able to allocate more time into properly integrating the other, more important aspects of the application.

Although we plan on testing and developing the application through Android, all members of this group own an Apple smartphone. We plan on the application being available to both Android users and Apple users, so once the application is finished we will download the application to both an Android phone and an Apple phone to make sure it works on both platforms.

Visual Studio Code allows for an easy to use way of working in any language, including Dart, the programming language we plan on using for application development. Although it is not possible to see the changes you are doing as you are typing like with Android Studios, another IDE that we looked into for application development, Visual Studio Code, allows you to incorporate an emulator with the program. By running the program with the emulator, and by saving your code, the emulator can immediately update to reflect the changes you have made in Visual Studio Code. The emulator allows for the developer to do everything the user will be able to do when the application is downloaded to their smartphones. So, we are able to see what happens when you click a certain button as well as seeing if the functions of certain aspects of the application work properly.

It is important to note that we do not plan on putting this application on either Android's Google Play Store or Apple's App Store, so when the application does eventually move from the IDE and emulator, we will have to transfer the necessary files from a computer to the phone.

A legitimate concern that may be brought up when considering the development of the application is if a mobile application developed through the Dart programming language can even connect to Amazon Web Services. Luckily, there are many resources online to look into this, some even published by Amazon themselves. One such link is provided in appendix A [15]. The link provides a step by step installation of Amazon Web Services packages into the application development environment, making it able for not only the application to connect and utilize Amazon Web Services, but the emulator as well. This makes it perfect for testing as well as deploying to a real phone. Che' plans on using this guide in order to properly install Amazon Web Services into the mobile application as it is crucial for the Facial Recognition Lockbox to function.

Once installed onto the emulator, some testing will need to be done. The main tests that will need to be performed are for sending information and making sure that Amazon Web Services properly receives them. In order to properly make sure that the mobile application is accurately

connected to Amazon Web Services, we need to make sure that you can send text to Amazon Web Services.

Upon verifying that, we will then need to move onto verifying that you can send images to Amazon Web Services. Subsequently, we will then have to make sure that the application can send these objects to the proper file location. These same tests will need to be done after moving the mobile application from the emulator to an actual phone. Once this is done, the only thing left for the mobile application is making it easy for the user to use.

Lastly, something that needs to be discussed further is Amazon Web Services Amplify. Amazon Web Services Amplify, or just Amplify for short, is an amazing tool that can be utilized on mobile applications. The reason this needs to be brought up is because this is the specific tool that Che' is looking into using for the mobile application. He is interested in Amplify for multiple reasons.

One reason is that this is also an Amazon tool, meaning it will be easy to use Amplify to communicate with the Amazon Web Services server, where all of our information will reside. Another reason is that Amplify has a method for verifying users, which is an important tool we will need for the mobile application. The tool is imperative for new users to use the application and create an account.

Additionally, Amplify also allows for existing users to login. It is the perfect tool for allowing the mobile application to not only communicate with Amazon Web Services, but also making it easy for us to implement the login and signup into the mobile application. The login works as expected; it looks at the users in the Amplify Services and sees if the information matches. If they do then the user is able to move on to the next page, otherwise they can not. Signing up is interesting, however, since Amplify Services are able to send an email verification to the new user. This leads to a secure feature of the mobile application, where we will be able to verify that this is indeed the correct user by sending them a verification code. By sending the new users a verification code, we will then be able to verify said code in the mobile application.

Other aspects of the application that we will be able to use as a result of Amplify is resending a code and changing an existing user's passcode. All of these are important features to have in any mobile application, and are tools we will implement in ours as well regardless of what service from amazon we utilize.

Similar to the regular Amazon Web Services, Amplify also has their own installation guide on their website. That installation guide can be seen in Appendix B [16]. They also have a guide specifically for Flutter, the software development tool we are using. With this, Che' will have an easy time installing the software into the mobile application and use it for testing.

We will test both methods in the emulator setting and see which works best for us. Ideally, we will use Amplify for logging in an existing user and signing up a new user and the regular Amazon Web Services for communicating back and forth with the database. However, if Amplify or Amazon Web Services proves to do both aspects well then we will simply use one or the other. We can compare whether Amplify is better than regular Amazon Cloud Services by not only performing the tests earlier stated, but also testing to see which is easier to install and use. Although it is completely subjective, we feel it is best to use whichever program gives us the least trouble. However, we also want to ensure we are using the program that will still be capable of performing all of the tasks as they are needed for the Facial Recognition Lockbox to work. If they are not able to perform these tasks, they are rendered useless for the project, and we will have to move on to something else.

In terms of the standards for the mobile application, there is not much to note. There are no economic, environmental, social, or political constraints due to the mobile application not having anything to do with the economy or the environment. It does not affect any social or political settings. It also does not have any health, ethical, manufacturability or sustainability constraints as the application does not affect or relate to one's health or ethics and since we are solely focusing on the mobile application, it will be transferable to any device meaning manufacturability and sustainability are not a problem.

There is a time constraint for the application, though. This constraint revolves around when the project is due. The mobile application needs to work and communicate with Amazon Web Services by the time the project is due at the very latest. It also needs to be fully accessible by any external users by this date as well.

Another big concern is the safety constraints of the mobile application. Not for a user's physical safety, but for their information safety. Since we will be dealing with some personal information from the user through the mobile application, their private information needs to stay private. Luckily, Amazon Web Services is made by one of the biggest technology companies in the world, Amazon. This, in turn, makes Amazon Web Services highly secure as a result of that. One of the main goals for Amazon Web Services is to keep the user's information secure. This makes it easier for us to develop the project and mobile application since we do not have to worry about finding a way to keep the information secure since the cloud service does that for us.

Mobile Application Development:

The language we are going to use in order to develop the mobile application is Dart. Dart works well with the program flutter, which is also a tool we are using to develop the application. While

using both of these programs together, we are able to incorporate many useful aspects that will help to keep the user and their information safe.

One of the first things users will need to do is be able to login and sign up. Thus, both of the basic layouts for these pages are done. Once users login they will be taken to the main page where they will be able to access all of the features that the remote Facial Recognition Lockbox has to offer. Here they will be able to edit their account information, upload images, and more with just a couple button presses.

An important feature to note of the sign up page, however, is the email verification. Email verification is a necessary component of the sign up page as it will help with keeping all of the user information better sorted in the Amazon Web Services. By verifying the user email, Amazon Web Services will be able to document that email and keep it in their secure cloud service. In that cloud service, we plan on sorting things in regards to that email to have an easier time accessing the user information. With this, things like preferences and images will be sorted in a neat fashion in a single place, on a single server.

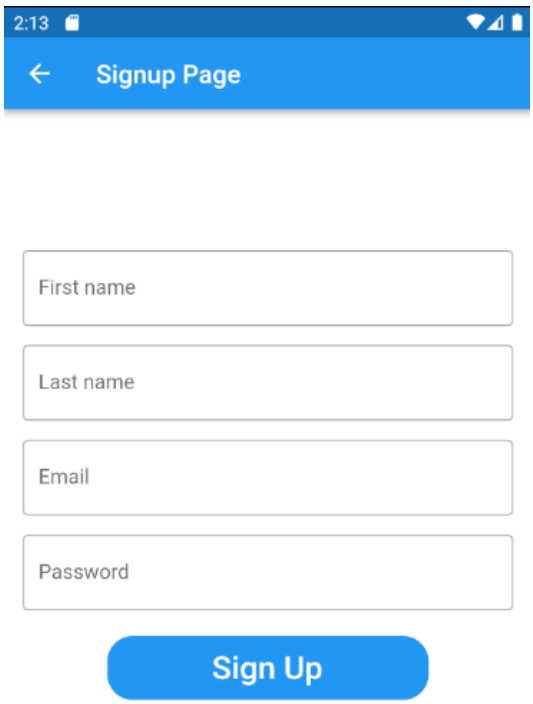


Figure 31 - Sample Sign up Page

As previously stated, there are a variety of features that we want to incorporate in the application. One of the most notable features is the Amazon Web Services. Although we plan on using Amazon Web Services, none of the members of the group have any experience with it. However, that does not take away from how important this component of the project is. Amazon Web Services provides us with a cloud service that is imperative to the project as it will be able to not only host the data of all of the users, but keep it secured as well. This takes a huge load off of our, the developers, shoulders as Amazon and Amazon Web Services is far more secure than anything we could produce.

Additionally, since Amazon Web Services is a cloud service, it will have no problem being able to communicate with both a smartphone and the Facial Recognition Lockbox from anywhere as long as the Facial Recognition Lockbox is able to form a solid connection with a Wi-Fi source. Amazon Web Services will serve as the secure storage facility for all the information needed for not only the application to run, but for the Facial Recognition Lockbox to successfully fulfill its purpose.

The other features that will be incorporated into the application will be available for the user to utilize. All of these features will be easily accessible through the application's home page. The home page, as of now, will have 5 main buttons: Incoming Delivery, Edit Profile, About Us, History, and Log Out. The Log Out and About Us buttons are pretty simple; the Log Out button will take you out of the home page and back to the login screen, while the About Us button will just have a brief description about the developers and links to all of the important documents. The main bulk of the application’s purpose comes with the other 3 buttons.

First, the Incoming Delivery button. The Incoming Delivery Button will allow you to upload an image of the person who is coming to make the delivery. That image will then be sent to Amazon Web Services where it will be stored and used for a set period of time that will also be set by the user. The user will have the ability to set how long they want to use that image when they upload the image. Although they will not be able to change the time once it is set, they will also be able to end the image user prematurely if they so wish. After that, they will have the opportunity to upload a different image.

As of now, once the delivery is set, we do not plan on allowing the user to be able to edit any information. So, if they wanted to change any information they would have to end the current delivery and go through the process again. These images will then be stored in a folder in relation to the user in the Amazon Web Service.

When the user presses History, they will be able to use images they have used previously. We feel this will make using the application more convenient for the user as they will not have to access their gallery repeatedly in order to upload the same image. After pressing the image, they will be asked to set a time with the image already loaded and things will proceed the same as they were if you were to click the Incoming Delivery button.

Last, the Edit Profile button will serve several different purposes. One is to do what the button suggests: edit your profile. This will allow the user to edit their account settings, things such as name, password, location, etc. However, you will also be able to do important tasks like changing your preferred picture time usage as well as clearing your history. Changing your preferred time will change the time that appears when you upload an image, so the user will not have to constantly change the time every time they upload an image. When a user clears their history, it will delete all of the images stored in Amazon Web Services.

We feel the aspects mentioned above are all important factors, not only for the Facial Recognition Lockbox to work properly, but to provide the user with a friendlier experience.

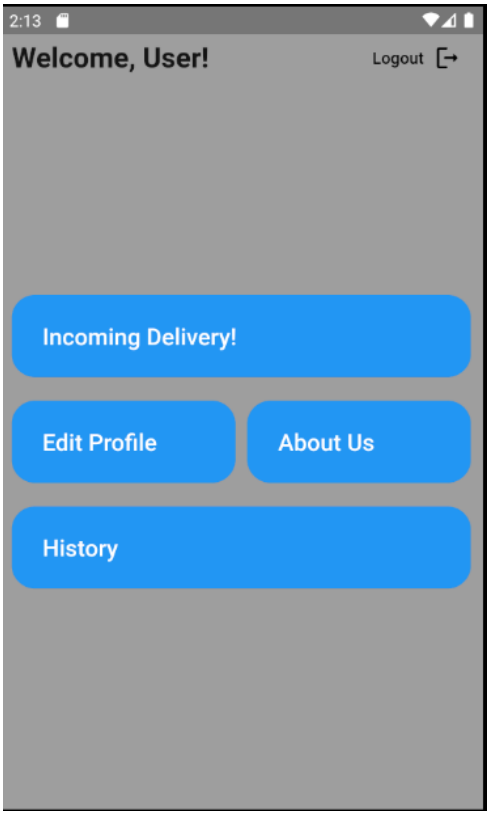


Figure 32 - Sample Home Page

Database Design:

Although there are database builders out there similar to Kintone, our Facial Recognition Lockbox database will be built using Amazon Web Services or Amazon Web Services. Amazon Web Services offers users a broad selection of databases, and several pre-built options; these options range from key-value, relational, document, in-memory, and ledger based databases [8]. Amazon Web Services also automatically scales well with the size of the database as it grows, having its relation database perform three to five times faster than other popular database options [8]. Along with its automatic scaling, Amazon Web Services also has a self-healing storage that keeps database clusters running [8].

The main structure that the Facial Recognition Lockbox will be around is Amazon Web Services's DynamoDB; this is one of the main databases that Amazon Web Services offers for handling key-value pairs. The Facial Recognition Lockbox's database will be structured with this key-value pairing; the key would be the user's username *(possibly change)* and the values would be all the attributes that belong to that user. This will hold the user's password for the application, passcode for the safe, facial recognition score, email, and name. One of the biggest advantages of using Amazon Web Services DynamoDB is it provides users with fast and predictable performance and, as mentioned before, allows scalability within the program.

1.1 Amazon Web Services using DynamoDB:

Amazon Web Services is one the world's most comprehensive cloud services out there. It hosts a range of development platforms, databases, security, several integrated coding languages, storage and much more. Amazon Web Services will be used throughout this entire project, being the host of the Facial Recognition Lockbox's storage, security, and database. The main focus of this section will be what kind of database is implemented for the Lockbox. Amazon Web Services has a plethora of databases constructed for users to implement into their program, specifically one called DynamDB.

The structure of most databases nowadays are written with SQL, which is a standardized programming language that allows users to implement various operations on a database [9]. However, DynamoDB has created it in a way to eliminate SQL. Providing fast and predictable performance, DynamoDB was created to be as user-friendly as possible, taking most of the administrative workload such as scaling, setup and configuration, hardware provisioning, software patching, and clustering out of the hands of the user [12].

Along with its automated help with administrative properties, DynamoDB also includes added security for your programs. An included encryption service, that is already implemented so users

can skip creating one themselves, protects sensitive data within the database. DynamoDB also allows for on-demand backup of any database. This will allow for full, long-term backups of all the tables. This service also includes a point-in-time recovery, which helps protect databases from accidental write and delete operations [12]. The point-in-time service will allow users to restore a previous database to any point from the past 35 days.

Although DynamoDB is noSQL, it does not diminish the amount of control you have over the tables in the database. Allowing you to easily create and monitor tables, and retrieve any amount of data and serve any level of request traffic [12]. Using the Amazon Web Services management console, users can monitor resource utilization and performance metrics. This also gives the user the ability to scale the database up or down, depending on the table's capacity needs. DynamoDB has implemented delete functions that will automatically clear up the database by deleting expired inputs, saving storage space within the database's tables

With any software program, having a set of rules to guide naming and data type conventions is crucial to the workflow progress. This allows a group of programmers to work separately, moving towards the same goal, without overlapping or overwriting other code. All of these guidelines should be in place before the start of the project, and are in place to cohesively connect the other programs together. The goal of implementing a naming convention is to keep name changing to a minimal and variable naming across platforms; this does not mean diminishing naming convention or limiting user naming. As the scale of a program becomes larger, these guidelines are in place to help the programmer identify what and how an object should be initialized and declared.

Data type guidelines are just as important to the workflow as naming guidelines. These guidelines deal with the declaration of objects, variables, and attributes of the program. Much like the naming guidelines, these should be predefined before the software gets implemented. Following these data guidelines are crucial for the program to run. With the amount of data that is going to be transferred from the Lockbox to the application to the database, if one of these programs does not follow the data type rules, the chance the data will be corrupted is quite high. This also will lead to possible crashes throughout the program if the data types do not match up.

Amazon Web Services DynamoDB has three different data types it can handle: scalar types, document types, and set types. The Facial Recognition Lockbox’s database will mostly contain scalar types, which are defined as one value types such as numbers, strings, binary, boolean, and null values. With some databases, you are required to have a defined name and data type for each column. However, with Amazon Web Services DynamoDB, and since we are using a key-value relation, only the primary key attributes need to be defined when creating the table.

The full list of naming and data type standards will be listed in the standards section of this document.

1.2 Database Storage:

One of the most convenient aspects of Amazon Web Services is its ability to scale with the program as the amount of data increases. Since this product is going to be individualized to each customer–by having an account for each user and the possibility to have multiple users connected to one box–we require a large amount of storage space. The exact amount of storage is still unknown due to the product still being in its experimental stage, however as we move into the development stage, this number will solidify.

1.3 Database Structure:

The structure of the database will be quite simple as most databases, at their core, are simply tables holding specific information. With the Facial Recognition Lockbox's database it will be split into three separate tables. One table will be for the user account information; one table will be for the lockbox generated information; the last table will be for the AI program results. All of these tables will be interconnected, using a relationship connection. The AI table and the Lockbox table will both have a dependency back to the user table. The Lockbox will be connected to the owner's account, also known as the primary user. The AI program has to generate the facial recognition score for each user and send that back to the correct user's account. The Lockbox and the AI program's table will also have a dependency with one another. This dependency is for when the box is using the AI to use the facial recognition program to open the box to the user.

With the Facial Recognition Lockbox's database, its main focus will be on storing user information. Each user will make an account through our mobile application, and, through this sign-up process, the user will input an email, a password, and a username. Once the user has created an account, they will have access to the account page where they will be required to add a user photo and a back-up lockbox code in case the facial recognition fails or there is a loss of power. The user photo will be analyzed by the AI program to generate a number. This number will also be stored with the user's data to be used in conjunction with the facial recognition software. All of these listed items will be saved inside a table inside the database.

The Lockbox itself will also have a table included in the database. Its main attributes will consist of a temperature reading that can be used to ensure the device is not overheating, as some packages will not require this function at all. The database will also have a notification, or alert, column with a history of the notifications sent to the user. This will also include an open and close timestamp allowing the user to keep track of the exact time a package is delivered and

when the device is opened; this feature will also allow the primary user to know which account was used to access the box.

Finally, the AI program will have a table to keep track of all the scores it generates from its training on photographs. As discussed in more detail below, in the Artificial Intelligence section, the AI program will train on at least 1,000 user faces and be able to correctly predict a score to be sent over to the microcontroller. This score will then be sent over to the user's account to be attached as an attribute to the user and stored inside the database. Using this score is crucial in correctly identifying the user's face, and unlocking the Lockbox.

Facial Recognition Lockbox					
Primary Key	Attributes				
Username	Password	Email	Safe Code	Picture	Picture score
user1	Password	test@test.co	123456	file.png	1304
...	...	...	...	...	...
userN	Hi!	email@emai	654321	file.png	1255

Figure 33: Key-Value Table

There are many different database styles on the market, each with their own specialities. Some work better with different data than others.The facial Recognition Lockbox's database will be based on a key-value relationship pairing. As seen above, the user will have a primary key, their username, which will be unique to each user. From the primary key, all the other attributes attached to that user will be accessible for reading commands. Any other operation, such as a change to a password, email, safe code, picture, and picture score, will also be available from the primary key.

1.4 Database Operation:

Although the behind-the-scenes may seem convoluted, operating the Facial Recognition Lockbox will be quite simple. To lock or unlock, the process follows typical facial recognition standards. The user will stand in front of the camera for about three to seven seconds to allow the camera to capture their face. While the camera is capturing the user's face, the AI program will be running in the background, analyzing the face and creating data points from it. This will generate a score for the program to use.

From there, the program will search through the database. The program will be looking for a registered user with the same facial recognition score that the AI program is producing. This is where the SQL language comes into play; by using its fundamental SELECT, FROM, and WHERE calls, the program will be able to search through the database and eliminate unnecessary data entries while searching for the correct one. If the program finds a match in the database, it will initiate the unlocking mechanism, releasing the latch and allowing the user to open the Lockbox.

If the program does not find a match it will signal the user to try again or manually enter a six digit code on the box. If the user trying to enter the box incorrectly enters a code three times in a row, the system will send an alert message to the primary user to notify them that someone is trying to unlock their Lockbox and has incorrectly entered the code several times. The same alert message will be sent to the primary owner if the camera has captured someone in-front of the box for an extended period of time and is unsuccessful in identifying them as a registered user.

1.5 Database Optimization:

With technology nowadays we want it to be efficient, quick, and for it to optimize its storage capacity. This is especially true with products that work over an application from a distance. Users do not want to be waiting for an extended period of time to achieve a simple task. Users also want their product to work efficiently for all users across the entire, expanding database.

As developers, we want a product that can compute large amounts of data accurately and efficiently so the user will have the correct information when retrieving it from the database. As mentioned earlier, the Facial Recognition Lockbox's database will be a group of linked tables. However, there are still quite a few things to keep in mind when constructing the database to keep it optimized. The most important attribute when optimizing this database is the indexing of the tables.

The tables must be indexed properly; without proper indexing, or with repetitive, unspecific indexing, the program will be slow and ineffective. Along with indexing, the type of data taken into the database is incredibly important as well. Having the database take in unnecessary data or

storing data that is already in the database, will slow the program down. This will become increasingly noticeable as more data gets stored into the database.

Since SQL will be the main language used with the Lockbox, there are a few situations that should be avoided. One being the implementation of the WHERE calls; because the tables will be linked together, we want to make sure the WHERE calls are sent to the program in a straightforward manner. This will reduce the chance of having the program fall in a constant loop. Constant loops will either cause a possible crash or create a major delay.

1.6 Database Security:

With the rising use of technology in our everyday lives, there is one thing that has to be taken seriously: security. It is the most important thing in any database and that is true for this project. With facial recognition software and the accompanying information being stored within the Lockbox database, we must be prepared to include firewalls and security that match the importance of what is being stored. Using Amazon Web Services allows us to do exactly that.

Amazon Web Services makes data protection its utmost priority, providing a continuous monitoring system equipped with threat detection, encryption, and key management [11]. No threat is left undetected due to the aforementioned continuous monitoring system and analysis of network activity. Similar to that of a spend analyzer on a credit card, Amazon Web Services will notify the user of multiple failed attempts, keeping them fully aware of possible security breaches. The user will also have an opportunity, should they desire, to implement a failsafe code into their Lockbox upon installation.

Amazon Web Services allows us to modify “identities, resources, and permissions at scale” [11], meaning the database is fully customizable to allow each feature to perfectly fit our Lockbox design. This flexibility lends itself to the compliance status of the database as well. Compliance checks can be automated to follow industry standards and Amazon Web Services best practices depending on the level of data privacy necessary [11].

However, we understand that most data breaches do not come from within the network, but rather from an external source attempting to breach. Amazon Web Services allows us to install high-alert security at vital control points across the database. It will also allow us to filter incoming and outgoing traffic, inspect suspicious requests, and maintain the highest form of security to ensure that the boundaries established around the host, network, and application are all maintained. [11].

6. Project Prototype Testing Plan

Equipment Needed:

- Soldering iron and material
- Multimeter
- General power tools and screws
- Raspberry Pi Motor Hat
- Keyboard, mouse, and monitor for debugging software on the Raspberry Pi
- Wi-Fi enabled environment
- Smartphone with the application installed on it
- Stepper motor hat

Testing/Debugging:

The most important aspect of a successful project is that it functions as intended when it all comes together. In order to make sure we have a properly functioning Facial Recognition Lockbox, we will have to spend some time testing the project and fixing any mistakes. Like most projects, we do not expect to get everything done correctly the first time around. There will undoubtedly be mistakes and bugs that need to be fixed in both the hardware and software aspects of the Facial Recognition Lockbox.

Three weeks have been allocated for testing and debugging alone.This is arguably the second most crucial stage of the project, behind actually building and putting it all together. Each component of the project will be tested and then practice run-throughs of the project as a whole will also be done. The following are the components that we will test individually as explanations on why each is important.

Testing the mobile application is imperative because that is what the user will interact with most of the time. There are a variety of small, but important things that need to be checked so that the user will not encounter any problems when using the mobile application.

In addition to testing aspects unique to the mobile application, we need to make sure that the mobile application also works properly in tandem with Amazon Web Services. This is mainly due to the fact that the sign up will require an email verification is sent, which is reliant on Amazon Web Services. The only aspect of the application that is reliant on Amazon Web Services in order to continue testing the mobile application is the sign up page.

Hardware Test Environment:

In order to test the hardware we will need to do so both inside and outside. The reason for this is because, ideally, the Facial Recognition Lockbox will be able to function for a long period of time in either setting. The conditions that the Lockbox will face indoors and outdoors are completely different, especially that in temperature.

In order to test these two environments, we will test the hardware in a group member's home. The Raspberry Pi will be checked in Che's home since he is the one who owns the device. Depending on who purchases what parts will change where each part will be tested. Regardless, each part will be tested in the comfort of a team member's home.

After testing it inside, we will test the components outside. Since there will be times of the day where the sun will be hitting the Lockbox no matter where the user places it, we will test each component under direct sunlight to ensure they function properly. After passing all of the hardware tests stated later in the report in both the indoor and outdoor environment, we will consider the parts fully functional.

Software Test Environment:

We will test the software in both the emulator and a physical phone. We will first test the software by emulating the program on a Pixel 2, which is the default emulated Android phone that is set up on Che's computer. After the mobile application successfully passes all of the tests stated later in the report, we will consider the mobile application viable to move on to the next phase.

That next phase is testing the mobile application on an iPhone by transferring the mobile application through a Mac product. Once we confirm that everything done on the Android emulator can be transferred to an actual, physical phone, we will consider the mobile application fully functional.

An important thing to note when transferring the mobile application from Android to iPhone, or even from phone to phone in general, is the dimensions of the phone. The dimensions of each phone can vary. If the application is made to fit a certain set of dimensions, it will not carry over well to a different phone. Therefore it is important to note that in order to make a proper application, it needs to be scalable. The application needs to be able to fit a phone of any kind of dimension so that it can reach the largest audience. If, for example, a mobile application you were using had black parts at the top and bottom, no one would find the mobile application visually appealing. Luckily, in Flutter we are able to take this into account and make scaling the dimensions to the screen of the phone possible.

When moving the mobile application from the Android emulator to the physical iPhone, everything will need to be tested again. This is to ensure that everything is capable for the user to do from the palm of their hands. Although testing will be the same, this will arguably be one of the most crucial aspects in developing the mobile application. Not because more tests will need to take place or more features will need to be added, but due to the possibility of something not working. If an aspect of the mobile application is working on the Android emulator, but not on the physical phone, we must contemplate why this is the case. As a result of this, more tests and transfers between emulator and physical phone will need to take place and more research will need to be done.

This is a natural process in any kind of development, especially when working with multiple platforms. This is why these testing environments play such a crucial role in developing a mobile application. Nonetheless, we felt it important to note how crucial it is to transfer the application not only to a physical phone, but to a different operating system as things do not always go as planned.

Artificial Intelligence Testing Environment:

In order to test that the Artificial intelligence is working correctly as well as all of the API calls that we make to the AWS server, we will be doing a similar operation that is described above. This will involve creating an isolated environment that will mitigate the risk of other factors contributing to a potential failure within the program. Not only will it mitigate those factors, but it will also allow us to analyze the model in a simplified system that will allow us to correctly debug our API calls as well as being able to check what has been correctly stored or retrieved from the database.

To set up this environment, we can just create multiple short Python scripts that represent each operation of the application that would be working on the Android device in order to analyze further what data would be generated and received.

The first operation and script that we will need to test is that the application can take a photo of the user, append it to the account of that user, and store it within the database for the Amazon Rekognition model to train on. The database for which the Amazon Rekognition model needs to train on the photos is called a collection. We can set up a collection in the database specifically for training to ensure that the full deployment of our application doesn't have false users in it while also providing the additional benefit of isolating issues which had been mentioned before.

All of these API calls can be within one script because we would like to make sure that we can sequentially issue these commands without issue with one click of a button within the application. If the photo of the user is associated with that user within the database, then we are

likely able to move onto the next step, if the photo is also within the collection in which the Amazon Rekognition model has access to it, then we can assume that the next set of API calls will run correctly (however they will be tested also). The steps of setting up these scripts will be very similar to one another in terms of development.

The next set of instructions and operations to be written in a Python script will be emulating the operation that will be called from the physical locker. The emulation is that the user will type in the code that they were given for their scheduled pickup, and the camera on the locker will then be probing for the face of the user, in which it will unlock if the artificial intelligence model detects that the user matches the current user that has scheduled a pickup. So the way that we can isolate this interaction is simply by manually inputting a fake user and order into the database with a six digit pin of our choosing, and a face of one of the developers.

Once this has been done (which is easy to verify through the Amazon Web Service development dashboard), we can then write a Python script that takes a six digit pin as input in the console, and makes an API call to the database to confirm that that pin is associate with the order, which can then use OpenCV to utilize the camera on the computer we're using to test everything with to snap a picture of whoever is in front of the computer. The script can make an API call to send that picture to Amazon Rekognition to predict whether it matches the user associated with the order. If it matches within a 70% accuracy interval then we can have the script return true, and otherwise it can return false.

These are realistically the only two operations that are required in order to test the Amazon Rekognition model within our application, but are the core functions of our system. Isolating these two operations are important because we can mess around with different elements of a photo and user data in order to see if we can break the API calls, or find flaws in our storage methods, as well as experiment with how expensive storing photos and user data can be for the duration of this product's lifespan.

Specifically for testing flaws within the photos of the user's face, we can alter certain factors that could be performance altering. This has been alluded to in other sections of this document, however it's important to note how lighting can impact an artificial intelligence model's ability to predict upon the data that has been presented to it. We can individually craft specific photos that we would like to test the performance of and manually alter the data in this environment in order to satisfy our testing requirements. By doing this, we can analyze the performance without having to use the fully deployed application where we would have to create new users, with unique names and emails. While we will of course be testing the ability to create new users, being able to manually input new faces into the database with these Python scripts that make API calls can dramatically decrease the amount of time and energy required to test our program's limits.

Another added benefit of isolating these scripts for testing is that we can create specific test cases for ensuring that our API calls are secure, and that the camera is used at the correct time. We don't want to allow the user to be able to use injection to issue commands to the database without specific permissions. We can avoid this by creating curated test cases with commands that can be used for injection. The test commands will ensure that there are no breaks in the database entries, and that users cannot view data within the database that would not be displayed otherwise.

Mobile Application Testing:

- Ability to open the mobile application from App Icon
- Ability to log in to existing account
- Ability to create a new account
    - Email verification is sent
- Ability to log in after email is verified
- Inability to log in if email is not verified
- Ability to reset password
- Ability to navigate to the home page
- Ability to click buttons on the home page
- Ability to navigate to respective pages
- Ability to add a new image
- Ability to set image use time
- Ability to see history of uploaded images
- Ability to clear history of uploaded images
- Ability to edit profile
- Ability to set preferences
- Ability to connect to Amazon Web Services
- Ability to click links in about us page
- Ability to logout
- Ability to call APIs
- Ability to store personalized 6 digit code
- Ability to randomize new 6 digit code

Listed above are all of the individual aspects of the mobile application we will have to test. In order to perform these tests, we will need some sort of phone. Ideally, we would use an Android phone due to the fact that it is easy to transfer an application developed on a computer to an Android phone. Unfortunately, however, all members of Group 4 have iPhones. Transferring a mobile application developed on a computer to an iPhone is not impossible; in fact, it is an easy task. In order to transfer the application from computer to mobile, we will need a MAC product.

Though, these 2 items, an iPhone and MAC, are all we need to test the mobile application. Luckily, Julian owns a MAC product, therefore making it possible to test our mobile application on an actual mobile device.

Prior to the actual test, though, we will be developing and testing all of the individual aspects shown above on a simulated phone. The simulated phone will be an Android phone, and will have the ability to connect to Amazon Web Services. Additionally, the mobile application will be developed in order to fit the dimensions of the user's phone, so this will have little to no affect on how the information is displayed on an iPhone.

We will also find someone who is willing to download the mobile application on their Android phone to verify that the application works for both Android and Apple. Che has the most experience with developing a mobile application, so he will lead this section of the development. The mobile application test will be considered a pass if the mobile application is able to complete all of these individual tasks consecutively in both the simulated and real world environment. We will consider it a failure if any of these tasks fail, and will investigate why this may be the case and fix it immediately.

Server Testing:

We need to test the server to make sure that both the Facial Recognition Lockbox and the mobile application are properly connecting to the server. If this part of the project fails, the whole project can come crumbling down. Making sure that the server works as intended is imperative to the success of this project.

- Ability to connect to mobile application
- Ability to receive images
- Ability to sort data based on verified email address
- Ability to hold preferences
- Ability to maintain AI
- Ability to be connected to Raspberry Pi
- AI functions as intended in environment
- Ability to respond to API calls
- Ability to compare 6 digit code
- Ability to store encrypted passwords under correct user
- Ability to retrieve unencrypted password

Listed above are all of the individual aspects of the server we will have to test. We will have to test Amazon Web Services extensively as it is the home for not only our database, but for our artificial intelligence and account information. It is important that the group not only tests this environment, but is extremely comfortable with it due to how imperative Amazon Web Services is to the project. Aspects like email verification, getting artificial intelligence running, and properly storing the data in an organized fashion are especially important to get done as soon as possible. The only thing we need to test Amazon Web Services is some sort of device that can connect to the cloud service. This device could be anything from a smartphone to a computer to the Raspberry Pi.

For the sake of ease, however, each member will use their computer for testing throughout the development of the project, while there will be a final test going through all of the aspects with the Raspberry Pi when the product is constructed. It is important that we validate that the server works through the computer in order to access the information and see what is going on in the cloud. The server testing will be considered a pass if the server is able to complete all of the individual tasks consecutively on both a computer and the Raspberry Pi. We will consider the test a failure if any of these tasks fail, and will investigate why this may be the case and seek to fix it.

Hardware Testing:

Lastly, we need to test the hardware and make sure that not only the individual parts work, but that they all work together. Although there are not many moving parts in the Facial Recognition Lockbox, they all need to work properly. Since we do not want anyone to be able to steal the components, they will all be hidden and components like the camera will only appear when necessary.

- Lock open and closes based on remote command
- Camera appears and disappears based on remote command
- Ability to open and close door
- Raspberry pi can communicate with Amazon Web Services
- Raspberry pi can connect to Wi-Fi
- Step motor hat can control step motor via Python library
- Ability to input 6 digit code

Listed above are all of the individual aspects of the hardware we will have to test. Although not stated, we will have to test to make sure that all individual parts are working individually before putting them all together. This includes things like testing the number pad, the motor controlling

the locking mechanism, the Raspberry Pi, and more. If any of these parts do not work as intended, we will have to purchase another one. It is important to test all of these individual parts before putting them all together to reduce time investigating a problem. If we put it all together and something does not work, we will have to spend additional time figuring out what is not working and why. Additionally, we will have to put together some circuitry in order for the electronics to work. We can use an oscilloscope in order to make sure that each wire is properly conducting electricity. In order to test all of the parts, we will also need a power source on top of the wires and oscilloscope.

After testing each part individually, we will have to test the parts after we connect them as well. This makes the oscilloscope even more important, because it will be able to tell us if there is any malfunction or mistake made when soldering. The hardware testing will be considered a pass if all of the individual tasks are completed successfully. We will consider the test a failure if any of these tasks fail, and will investigate if it is not working and aim to fix it immediately.

Artificial Intelligence:

Testing the artificial intelligence models will be pretty black and white. We need to ensure that it's correctly identifying faces compared to other objects, as well as correctly matching faces fed to the model and the user that the artificial intelligence model is trying to identify. Other than that the artificial intelligence model needs to be lightweight. And since there is no way for us to define what is lightweight in this, it will simply be that it is not hosted on the hardware (Raspberry Pi) itself, and rather be in the cloud.

- The model will be able to be fed a picture of the user
- The model must be hosted on an Amazon Web Services server
- The model must be able to correctly identify each user (any incorrect prediction fails this test case)
- Smartphone application must be able to make API calls on the AI
- Microprocessor must be able to make API calls to the AI
- AI must train and predict from within the Amazon Web Services server
- AI must be able to predict from at least a user base of 1000 users
- AI must be able to pass the label of the prediction back to the microprocessor
- Ai must be able to add new faces to the label database within 10 seconds of being received onto the server
- AI must be able to receive new faces/labels from the smartphone application
- Predictions of faces must be processed on the microprocessor from retrieval from server in 10 seconds

- Predictions of faces within the server must be made within 10 seconds
- The label database must be expandable if demand from customer base increases
- The two factor authentication code will unlock the locker if and only if the AI has correctly predicted the face fed to it
- If the prediction of the user results in an incorrect prediction or no prediction, the locker must not display that information in order to maintain privacy
- The AI must be able to pass information over Wi-Fi through API calls on the Amazon Web Server
- If the AI is inactive or unable to process new API requests, the hardware will not be able to allow the user to retrieve items from the locker

These test cases for the artificial intelligence model must be utilized in conjunction with the test cases of the smartphone application as well as the application running on the microprocessor.

Database:

With something as important and sensitive as a database, we want to make sure it is working properly and is performing every operation needed. Especially now, when the front end of programs are becoming more expansive, the size or amount of databases grows in parallel. No matter what the database is used for, it will follow the ACID properties. ACID properties is an acronym for atomicity, consistency, isolation, and durability; these guidelines are what every database must follow to be considered a functioning database.

Atomicity is the first rule a database must follow. Atomicity stands for an all-or-nothing clause during transactions. This means that either all the data passes or all of it fails. Even if only one byte fails, the entire transaction will fail.

The next rule for transactions is consistency. Consistency states that only valid forms of the database will be saved. Data scientists use these consistency rules to reject invalid or unnecessary information in the database. Implemented using constraint fields at a software level, this should help with error prevention and data corruption within the database.

Isolation is the next check the database will go through for transactions. Isolation makes sure the database can handle executions independently of one another, even if, realistically, the executions could be happening simultaneously. The state and results of the database will happen consecutively.

Finally the database needs to have durability. Durability states that once a transaction is completed and written to the database, no external event should be able to change the state of the

database. If the database is missing any of these properties, major failures can occur throughout the program.

Accessibility to objects and variables within the database is crucial for any user or admin. Not knowing what is in the database will leave the user at a loss, or, worse, label the whole database as useless. This is where the CRUD operations come in.

CRUD is an acronym that stands for: create, retrieve, update, and delete. These operations allow users to adjust objects within the database. Without the CRUD operation, the database is an empty table. However, for a database to be considered effective, especially a user-based one, it needs to have every CRUD operation implemented.This is the biggest element we focus on when testing the database, even when considering security. Here we will want to make sure that every user has the ability to create, retrieve, update, or delete their information straight from the application.

Data integrity is a big element to take into consideration when using the CRUD operation. A major issue that can arise when updating a current object in the database is to have the same object display old information on another screen. To combat this, we will run numerous tests implementing all the CRUD operations to make sure each function operates correctly on the database and completely saves across the database.

Since this is a user-centric application, another major thing to take into consideration is ensuring the database can withstand having multiple users accessing it at the same time. On top of this, it is important to make sure those operations are not affecting or overwriting one another. Even though the database and program will have these requests be multi-threaded, or happening at the same time, the database will effectively process them one at a time to avoid conflict.

- Create and store new users
- Hold primary key, foreign key, and attributes per user
- Unique entries for usernames
- Retrieve information from the database using the primary key from the user, and only showing information pertaining to that user
- Allow users to change any attributes pertaining to their account
- Updates will be saved across versions, showing the most recent version.
- Users are allowed to delete their account from the database
- Once deleted, the database permanently clears the user’s information

The first operation checked will be the create operation. Here we will have multiple users access the application and create an account. We will check the database of users to make sure each user was saved onto it, and that every other attribute the user enters is attached to their account. It is

important to ensure that, on top of the data being saved, the attributes were saved in the correct way. This means that the primary and foreign keys are holding the correct information and all other attributes are attached to the primary key. Another important thing to check during the account creation process is that the user has inputted data for each attribute that meets the requirement of the attribute.

The second operation that will be looked at is the retrieve operation. Mainly, we want to make sure that the users are receiving the correct information from the database and it is being displayed on the user interface correctly. This is where one of our first security checks happen; the check is to ensure only relevant information, related to the user, is on their user interface. We want to make sure other user's information, such as an email, is not being displayed, as that would be a breach of security and data integrity.

From there users will have access to update their accounts. There are two main checks that accompany this step: inputting valid information and overwriting current attributes. We want to ensure that the information being inputted is valid, meaning it is unique and meets length and character requirements. In order to ensure current attributes saved into the database are overwritten by the new information, we will review the user interface after the changes have been made. This review will make sure the updated information is being displayed and not the outdated information.

The final operation that would need to be checked is the delete operation. This one is quite simple and only would require one check. When the user chooses to delete their account, it is important to make sure everything in the database that is connected to that user's primary key is also deleted. From there, the user should no longer have access to any information or account page, and be prompted to create a new account on the application.

7. Administrative Content

Project Budget:

Unfortunately, we were not able to find a sponsor to help fund the production process. Because of this, all the fees including, buying the material, products, and any subscription services, are the producer's responsibility. Since we are all college students, each with our own bills and responsibility, and some of us without a steady income, we would like to keep the budget to 75 dollars per person and limit the total cost to around 300 dollars. However, as discussed below, pricing for some materials and services may vary quite drastically. The 300 dollar budget has been discussed, and each producer is willing to exceed this budget, if only minimally.

If pricing becomes too expensive, we may look into another build option or source of income. In the case that this happens, we plan to possibly reach out to either UCF's SGA or UCF's Engineering department for additional funding.

Item	Price
Microcontroller	~$50
Lockbox	~$50
Bluetooth module	~$10
Camera module	~$30
Electronic latch	~$30
Amazon Web Services Server	~$0.016 per hour
Power supply	~$10
Heating Element	~$30

Table 8: Planned Budget

This budget highly depends on two factors. The microcontroller and camera will be dependent on how we decide to integrate the AI. Our budget will have to be updated if we decide to use a Raspberry Pi and an Infrared Camera for facial recognition. We could save money by going with a cheap microcontroller, or we could have the Raspberry Pi which make for an easier integration of hardware. The Raspberry Pi's vary greatly in price, as well as camera modules. However, a standard camera module for the Raspberry Pi (RP branded) is approximately 30 dollars, and the Raspberry Pi could end up being over 100 dollars.

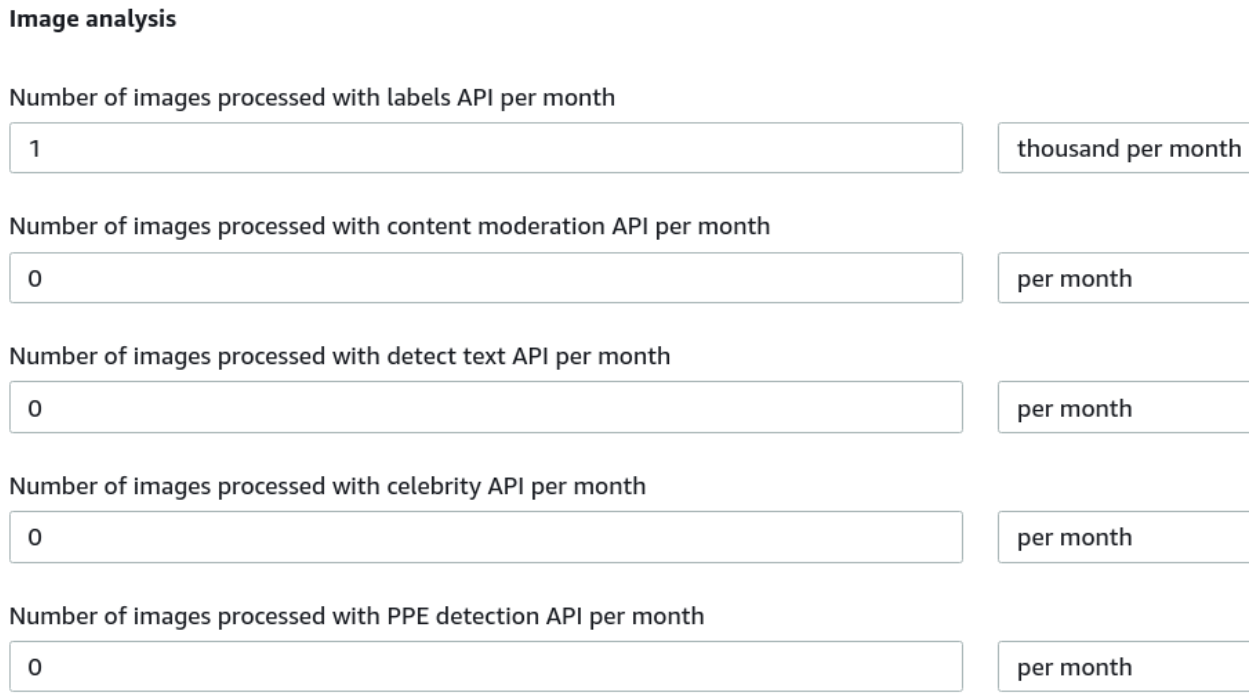
Another factor that could contribute to a budget change would be the price of the lockbox, due to this being strictly a Electrical and Computer engineering project, we've decided to buy a pre-built lock box which depending on brand could be anywhere from 30 to 100 dollars as well.

<u>Amazon Web Services Price Breakdown:</u>

Amazon Web Services will be extremely useful for our project with its powerful artificial intelligence tools such as Amazon Web Services Rekognition, as well as being able to store loads of metadata about faces without having to store anything locally on the Raspberry Pi. In terms of

account ownership, Julian will be the account holder for the Amazon Web Service, and will be responsible for paying for the usage of their services.

For pricing, we've used their built in calculator to determine the cost of having our artificial intelligence be hosted on their servers and they've given a price breakdown given our needs. The server that will be hosting us is the Amazon Web Server that is located in the East Region of the US in Ohio.



Due to our budget, we've determined we would like to limit the number of images that are processed with labels per month to just one thousand. This will still give us plenty of room for testing and initial deployment of our product for presentation, while keeping costs very low (final cost is shown after the breakdown).

We've also determined that we do not need any of the other services offered under Image Analysis, we simply need to be processing images with labels. We do not need to detect text, or moderate content, or the other services they offer as it is out of the scope of the project (even though this could be useful later in terms of analytics for determining the reach of our product).

Face analysis and comparison

Number of DetectFaces API calls per month
1 API call per image

1 | thousand per month

Number of CompareFaces API calls per month
1 API call per 2 images

0 | per month

As for facial analysis, we do need to detect faces, which we will match the amount of images processed. We do not need CompareFaces API since it'll be single faces on screen. It'll be one API call per image.

Face search

Number of IndexFaces API calls per month
Add all faces in an image to a face collection

1 | thousand per month

Number of SearchFaces API calls per month
Compare an indexed face against a face collection

1 | thousand per month

Number of SearchFacesByImage API calls per month
Compare face on an image against a face collection

1 | thousand per month

Number of faces stored in a collection
Each indexed face results in one face metadata stored)

50 | per month

This is where the Amazon Web Service gets a little more expensive. We will likely need to be using all of these services to identify the client that is picking up their object from our product.

So far, in this estimate we have limited the number of faces stored in the collection to 50 per month, as we're hoping to limit the cost of testing so we can spend it on hardware for other components.

Number of model training hours (one-time)

There is a cost for each hour of training required to build a custom model with Amazon Rekognition Custom

2.5

Estimated number of transactions per second

Defaulted to 10, depending on the complexity of your model you could expect between 1 and 15 TPS.

10

Number of images processed per month

1000

▼ Show calculations

2.50 training hours x 60 minutes x 0.01666667 USD per minute = 2.50 USD

Training cost (monthly): 2.50 USD

10 transactions per second x 60 minutes = 600 transactions per minute

0.06666667 USD per minute / 600 transactions per minute = 0.000111 USD per transaction

1,000 images processed x 0.000111 USD = 0.11 USD for processing images

Inference cost (monthly): 0.11 USD

2.50 USD + 0.11 USD = 2.61 USD total

Custom labels usage pricing (monthly): 2.61 USD

Now, the expensive part of this is the training and the inferences. To keep costs low, we've limited inferences and focused on training, which will allow us to test our product more efficiently. With the limited number of faces that will be stored for our product initially, we believe that the training of the model will not take much time at all, however, with each new face the training time increases, so this could potentially cost us thousands of dollars a month if this product is to become a popular solution for our clients.

As stated before, we believe the amount of images processed per month being one thousand is sufficient for the current stage of development and can easily be increased in the future to accommodate new clientele.

This leaves the total cost of using Amazon Web Services Rekognition per month at **7.61 USD**, which luckily is very inexpensive at this stage of development. However, modifying some of the

values in the breakdown from thousands of operations to millions of operations increases the cost into the hundreds of dollars up to a maximum of tens of thousands of dollars. This could be potentially drowning in the future of the product, so it could be worth the time to research lighter and more cost effective solutions when dealing with a very large amount of clients, however, for the current stage of development, Amazon Web Services is simply the best solution and most powerful as well as user friendly.

Amazon Web Services Storage Pricing Breakdown:

There's only two things that we will be needing to store on the Amazon Web Services in order for this product to work properly. We will be storing account information for our users. These accounts will hold a couple of things that are extremely lightweight in terms of storage size, like name, email, password, phone number, etc. There will be pictures attached to each account that can have quite a large storage size associated with them. Each user will have at least one good quality picture that is associated with the account, and the size of these photos depends on the quality of the camera on their device that they take it on, which for modern smartphones will have a large pixel density which will unfortunately cost us more money. The other thing we have to consider is the metadata that will be associated with each of these pictures that are associated with the accounts, which will be used by Amazon Web Services Rekognition in order to predict who is on camera at the locker. The pricing for storage is based off of the rate of the Amazon S3 simple storage that we will be utilizing.

S3 Standard - General purpose storage for any type of data, typically used for frequently accessed data

First 50 TB / Month	$0.023 per GB
Next 450 TB / Month	$0.022 per GB
Over 500 TB / Month	$0.021 per GB

As is seen here, the storage is actually very affordable. We will likely only be paying less than five dollars a month when running at full scale. The average file size of a 12 MP JPEG taken on an iPhone is around 15 MegaBytes, so for every ~67 users we will be charged around two cents from the Amazon Web Services to store the photos. So the cost for simply storing everything will be extremely cheap compared to the upfront cost of producing each locker.

For simply grabbing information from the Amazon S3 database, the SQL operations for producing the information is also very affordable and is actually trivial to the overall cost of this product. Given the nature of this product, it will be highly unlikely that we reach 1000 mySQL operations on the database, but if this product were to become an industry standard, it could possibly become a non-trivial cost. The following is an outline of these operation's costs.

	PUT, COPY, POST, LIST requests (per 1,000 requests)	GET, SELECT, and all other requests (per 1,000 requests)
S3 Standard	$0.005	$0.0004

So it is much less than one cent in order to process 1000 SQL requests on our database. However, it is important to note that this is only for retrieval of data for account information and does not reflect any predictions from Amazon Web Services Rekognition. These costs will compound with Amazon Web Services Rekognition and will be an on-going cost, so it is important to produce projections of our Amazon Web Services usage for every month to ensure we are charging an adequate amount for this service and product.

Milestones:

Initial Milestones for Both Semesters

- Gather materials
- Good accuracy with facial recognition
- Working, small scale lock mechanism
- Security feature, in case someone tries to break in
- Incorporate locking mechanism with facial recognition

Week	**Task**
1	Project idea
3-6	Initial project documentation (D & C I & II)
7-9	Project Research, Collaboration, and Selecting Parts
10-12	Testing Components
13-14	100 Page Documentation Draft
15	Review Documentation
16	Submit Documentation

Table 9: Senior Design 1 Milestones

Ideally, the most research we will have to do in order to complete this project will be on optimizing the AI model for any given dataset, the only other issue will be integrating all of the

software with the hardware, especially using bluetooth to transmit data from one unit to another. This will likely take most of our time in development, however, once our foundation is created, the rest of the work should follow. The prototype in the following weeks will be set up to handle primitive test cases, simply for establishing a proof of concept. Optimization of all systems will come once our test cases have been accomplished efficiently.

We have over committed time to testing to allow for debugging within communications and integration, as that has been identified to be the most cumbersome. We could likely continue to perform debugging on the software side once we're at week 13 and just commit new updates to it, to ensure that everything works smoothly. We will take a week to review the documentation we've written to make sure it is congruent. Otherwise the document will be submitted.

Week	**Task**
1	Retest Components and Double Check Design
2-7	Build Prototype and the Program
8-11	Test Prototype and Fix Bugs
12	Finalize Project and Presentation
13	Prepare for Presentation
14	Final Presentation

Table 10: Senior Design 2 Milestones

When we make our way to Senior Design 2, the first thing we need to do is make sure that everything still works. We need to make sure to do this immediately because if we lack a crucial component, the whole project could go down in flames. In addition to that, we need to make sure the design of the project that we want is good, and that we did not accidentally miss anything.

After those preliminary checks, we can move on to building the prototype and working on the code. We have the most time allocated to this part of the project in Senior Design 2 due to the fact that we want to make sure we have enough time to implement everything we want. Additionally, this will easily take us longer than any other part of the project. After building the project, we will need to test it and fix any bugs we might come across. We've allocated 3 weeks

to this, not only to have ample time to identify and to fix any bugs, but to have some room in case building the prototype takes too long.

After fixing anything that might be wrong with the project, we'll have to make sure to tidy up any documentation that comes with the class. Not only work on the documentation, but the presentation as well. We'll have another week to prepare for the presentation and practice for it, while the last week of summer is when the final presentation takes place.

User Manual:

Installing Software on Your Locker:

Firstly we will install the required software onto the host locker's machine. This will be the hardest part of the install, and while a technician of the company can use a Secure Shell to access the system and install all of the required packages, it is good practice for the company or individual utilizing the product to understand how to set up their locker for use. So as previously stated in this paper, Docker will be the method of installing and running the required application on the locker. Docker will download a file called an *image* and this will remain on the machine and will be used to run the container. This image can be easily updated later in time by the host when required, and will be described how to do this later in the manual.

The Raspberry Pi will be running an operating system called NOOBS, which is a debian distribution that is built for the Pi specifically. It was designed to be lightweight, and easy to use on the Pi for developers. Luckily the installation for Docker will be quite simple. Here they are in steps (in the terminal):

1. First remove all previous versions of Docker that might be present on the machine with this command:

```
~$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

This will ensure that we do not run into problems with docker versions in the future as well as ensuring that we do not accidentally use the previous version when running our program.

2. Next we will run some commands in the terminal window in order to allow the *apt-get* command to see the new indexes available as well as allow *apt* to use repositories through the HTTPS protocol. We can do this by running the following two commands:

```
~$ sudo apt-get update
```

```
~$ sudo apt-get install ca-certificates curl gnupg lsb-release
```

Now *apt-get* will effectively install the correct and most current version of docker when it comes time to in the later steps.

3. Now we will add Docker's GPG key. GPG is a method of encryption for messages and/or data. Docker uses this, so we must add their GPG key locally in order to use their service. We can do so with this command:

```
~$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Now for the Docker repository, there are three different options. There is the stable version, test version, and nightly version. Now it might seem obvious that we're going to choose the stable version, but this is extremely important for the maintainability of the product as the stable version will result in less crashes, less security risks, and less hassle. To set up the stable repo we run this command:

```
julianboaz@pop-os:~$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
 https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Now finally we can run the command in order to install Docker. Just in case, we should run the apt-get update command found in step 2 in order to have the completely updated version of the index, and then we can install the Docker Engine and Docker CLI and Containerd.io. So firstly run the command from step 2 here:

```
~$ sudo apt-get update
```

Then we will install the packages previously mentioned above:

```
~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Docker is now installed.

Updating Docker:

In order for the pull request (which will be covered later in this manual) to be successful, we must make sure Docker is up to date. Keeping docker up to date is the only maintenance that the locker host should need to do, otherwise that host should contact a technician.

In order to update Docker, we will update the *apt* index again, similarly to how we did while installing Docker. Use this command:

```
~$ sudo apt-get update
```

Once the index is updated, we can now simply use *apt* to update Docker Engine and Docker Engine CLI with this command:

```
~$ sudo apt-get upgrade docker-ce docker-ce-cli
```

Once this command has finished processing, we've successfully updated Docker!

Running the program on the host locker:

There are a couple of reasons the host might have to run the application on the locker themselves. Initial startup, system crash, reboot, and updates. Realistically, the most important will be initial startups and updates. The initial startup is self-explanatory. The updates can consist of a couple of things: Updates pertaining to the application itself, updates pertaining to Docker, and updates pertaining to the operating system. Both are required to keep seamless integration with the Docker hub where we will be grabbing the application to run! Running the program will be described below.

1. First we must check to see if Docker is up and running, we can do so by simply running this command:

   ```
   ~$ sudo docker info
   ```

   If Docker isn't running, we can reload the daemon and its configuration in order to restart Docker by running the HUP signal to the daemon. Make sure to run the previous command in the terminal to ensure that the daemon has been reloaded. Here's the command for that:

   ```
   ~$ sudo kill -SIGHUP $(pidof dockerd)
   ```

2. Once we have confirmed that Docker is running correctly, we can now go to installing the image of our application from Docker Hub. We can think of the image as being an application, and Docker Hub being an application store. The image of the application will contain all of the necessary resources, dependencies, configuration files, and versions of everything needed to successfully run the application. And it is done independently of anything currently installed on the host locker. For example, if the host locker has Python version 3.6, but the image has Python version 3.9, the Python version 3.6 is ignored (technically it's not even considered, as Python is installed within the image). This way, there's no chance the host locker can have a version crisis.

   However, instead of there being a specific place to go just click download on, we must pull the image from a repository located on Docker Hub, Docker Hub being similar to Github as it is a hosting website for all Docker Images. To grab the image from Docker Hub we can run this command in terminal:

   ```
   ~$ sudo docker run jpboaz/seniordesigntest
   ```

   Conveniently, the run command for Docker will both pull the image from the online repository as well as run the image. This process could take a few minutes, depending on the hardware of the given locker, as well as if there are other processes running alongside this. This process puts the application into something called a container which will run a script that was written specifically for the application that will run all of the commands on the host's behalf, so there's no additional setup required.

3. While it should be obvious that the programming is running, a step to ensure that the container hosting the program is actually running is to use the terminal to list all of the currently running containers. We can do this by running the following command:

   ```
   ~$ docker ps
   ```

   This will list all of the running containers and show their status. If there are no containers, the host will only see the titles of the columns of the table that is supposed to display all of the running containers. In this case, the machine should be rebooted, and processes one through three should be repeated.

Restarting the Application on the Host:

We can restart the programming whenever we'd like, but it's important to do it the correct way through the Docker CLI. What we should do is stop the container from running, and then re running it and we can do that by running these two commands:

```
~$ sudo docker stop seniordesigntest
```

```
~$ sudo docker start seniordesigntest
```

The following should not be done by an end user, but by a technician or sysadmin. However if you find that the image is damaged or is not serviceable, we can remove the image and re-add it to the host locker with a few commands.

This is realistically a last-ditch effort in the debugging process and will likely be unnecessary as a simple restart of the container should be sufficient. Since the maintenanced image is hosted on Docker Hub, we can simply remove the image from the machine with this command. The *-f* refers to forcing the remove command.

```
~$ sudo docker rmi -f seniordesigntest
```

And once this command has been processed we can go ahead and look back into the manual inside of *Running the program on the host locker* inside of *part 2* and copy that command to pull and run the image back from the Docker Hub repository. This will start the container running the program back up again.

Process of Using the Locker:

To start using the locker first the user must acquire our application for their smartphone. The application will be able to be acquired from the Google Play Store or the Apple App Store. The smartphone application is responsible for creating the user account as well as starting a new order.

Firstly, the user must create an account via the application which will require their name, email, password, and a picture in order for them to be identified by the locker upon pick up.

Once the user has created an account the user now has the ability to create an order. When the user creates an order, the user will be notified via the application once their item they ordered has been locked inside the locker and they will also receive a six digit code that is used for the two factor authentication. The user may then move to the locker.

Once at the locker, the user will be prompted to type in the six digit code they received earlier from the application on their phone. After the user has input the six digit code, the locker will then use the camera to probe for the user's face. The user should remain in view of the camera while the locker is probing for their face. The locker will then use the AI to confirm their identity and if it is confirmed, the locker will open!

Once this process has been completed, the six digit code will be recycled and the user will be able to create another order. Creating another order will generate a new code for them to use at the locker to ensure security. The picture taken of the user at the locker will also be recycled in order to reduce the probability that someone could use that same picture to authenticate at the locker.

Division of Labor:

With a project of this size, there are quite a few tasks that need to be completed. We have decided to split the workload between each team member based on comfortability and skill level. Our team is composed of three Computer Engineers and one Electrical Engineer, and we intend to focus heavily on implementing software with pre-existing hardware. This will allow the workload to be spread evenly, and avoid anything intensive falling on just one person.

To split the work in this nature, we have three different sections of our main software design: the application, the database, and the backend holding the AI training program. From there, Bryce, our Electrical Engineer, will have any electrical components in place that he needs for the design. A complete breakdown of the division of labor will be located below, in figure x.

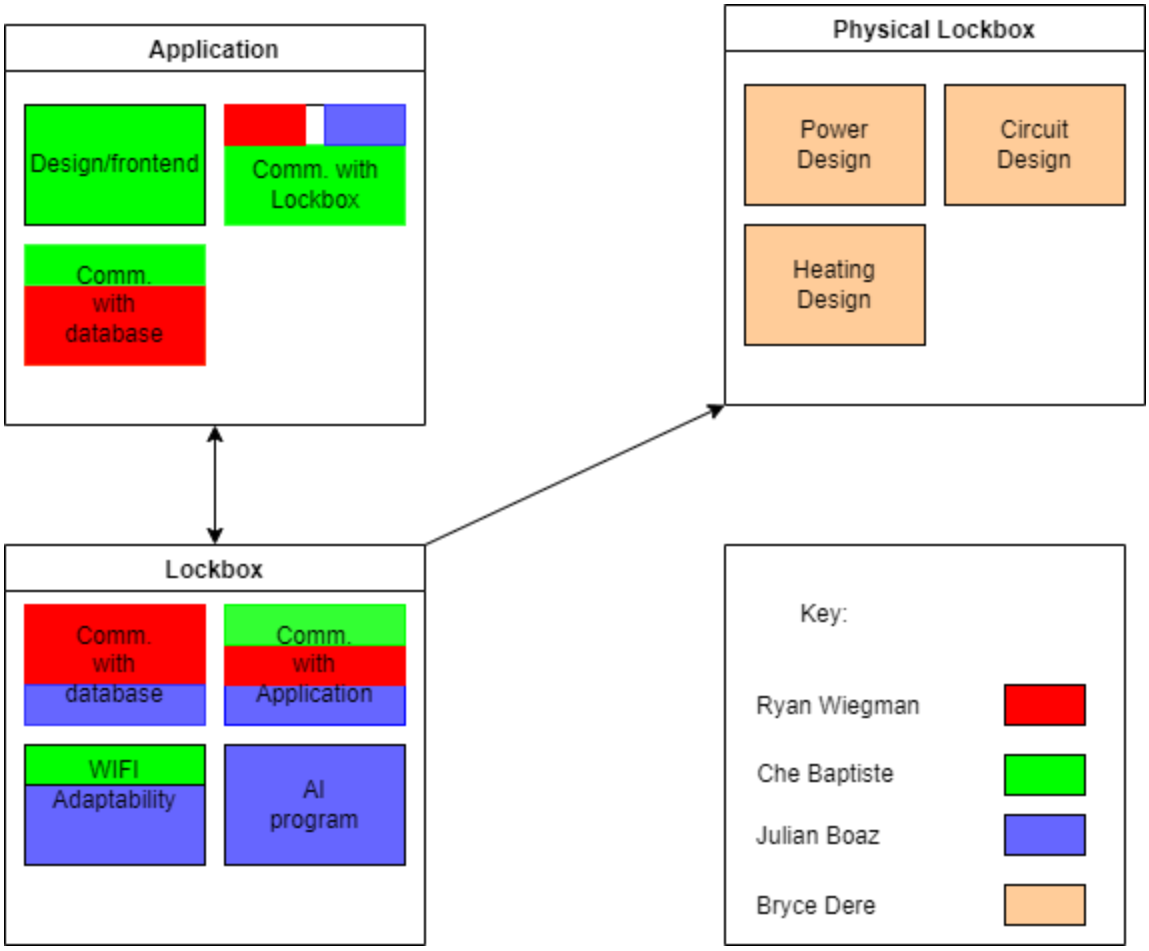


Figure 33. Division of Labor Diagram

8. Conclusion

With our Facial Recognition Lockbox, we want to design a product that is not only sought after by a pre-existing consumer base, but also a product that will fit perfectly into the rich tapestry of our growing society. We aspire to promote growth within this booming industry while adding our own personal touch in the form of this product. As a team we want to use the skills we have honed for the last four years of our academic careers to cultivate a product that we can be proud of. However, we also want to design something that challenges our already strong foundations, and build upon these skills to potentially use as a reference in future endeavors.

Our two main goals while cultivating this design we wish to accomplish with the final product are: to revolutionize the components of already existing products that we would implement into existing technology to create an item that has yet to be utilized in such a way, and to create an item that will deter porch pirates from perpetuating a culture of package theft. We want to create a product that exudes maximum security within an affordable budget that will optimize the importance of the security features without breaking the bank.

Although, on a very limited budget, we believe we are able to achieve these goals, even if there is still room to grow within the project. Though most of this growth is dependent on budget constraints and how we overcome them, like improved hardware and more storage capacity for training which would improve facial recognition speed, we believe they are attainable. With that said, this project is not without its current issues.

The two biggest issues our team currently faces are being able to find reliable parts to construct the Lockbox, and being able to have them shipped within a reasonable time period. With international delays in shipping due to the global pandemic that has severely impacted the shipping and production of hardware components including, but not limited to: microprocessors, circuitry parts, and many of the electrical components we require. Our main way of overcoming this is to order everything well in advance, as we are facing a fixed deadline at the end of the class.

The second issue our team faces that may cause final product delays, is the software we are using, namely Amazon Web Services. The service itself is relatively cheap, around one-hundredth of a cent per image stored, using this service to train an AI program to run through and read hundreds, or, ideally thousands, of training points to produce accurate facial recognition results, can become very expensive very quickly. This issue only increases when our budget is taken into account. As mentioned throughout our project, we are operating on an incredibly strict budget. The cost of training the model to read information multiple times to dispel any bugs or errors has the potential to be rather cost ineffective.

In an online market that is currently exploding, with millions and millions of people purchasing things online and having them shipped to their home, from clothes to tools to food, we believe that our Facial Recognition Lockbox can make a meaningful impact on not just the market, but the consumer’s lives. We fully believe the Facial Recognition Lockbox will be a product that online shoppers deem necessary for keeping their purchases safe, and making their lives easier and happier.

9. Appendix A: References

[1]
https://www.emarketer.com/content/us-ecommerce-forecast-2021

[2]
https://www.crresearch.com/blog/2020-package-theft-statistics-report

[3]
https://www.preventpackagetheft.com/the-5-best-parcel-boxes-to-buy-in-2018

[4]
https://www.preventpackagetheft.com/the-5-best-parcel-boxes-to-buy-in-2018

[5]
https://us.eufylife.com/products/t8790?utm_source=google&utm_medium=search&utm_content=sec_smartdrop&utm_campaign=us_security_seach_smartdrop&utm_term=%7B15386000138%7D_%7B131667887193%7D_%7B564676108718%7D&gclid=Cj0KCQjw0PWRBhDKARIsAPKHFGiDgzhNcU7EE94qQf07LL7cy1tB5OLCrv0UR7zERBS3RI0bCBNIm4YaAo0WEALw_wcB

[6]
Autofocus python script - https://www.arducam.com/raspberry-pi-camera/autofocus/


```
import cv2 #sudo apt-get install python-opencv
import numpy as py
import os
import time
import smbus
bus = smbus.SMBus(1)
try:
        import picamera
        from picamera.array import PiRGBArray
except:
        sys.exit(0)

def focusing(val):
        value = (val << 4) & 0x3ff0
        data1 = (value >> 8) & 0x3f
```

```
            data2 = value & 0xf0
            # time.sleep(0.5)
        print("focus value: {}".format(val))
       # bus.write_byte_data(0x0c,data1,data2)
            os.system("i2cset -y 0 0x0c %d %d" % (data1,data2))

def sobel(img):
            img_gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
            img_sobel = cv2.Sobel(img_gray,cv2.CV_16U,1,1)
            return cv2.mean(img_sobel)[0]

def laplacian(img):
            img_gray = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
            img_sobel = cv2.Laplacian(img_gray,cv2.CV_16U)
            return cv2.mean(img_sobel)[0]


def calculation(camera):
            rawCapture = PiRGBArray(camera)
            camera.capture(rawCapture,format="bgr", use_video_port=True)
            image = rawCapture.array
            rawCapture.truncate(0)
            return laplacian(image)


if __name__ == "__main__":
    #open camera
            camera = picamera.PiCamera()

        #camera.awb_gains=4
        #camera.exposure_mode='off'
        #camera.awb_mode='fluorescent'
        #open camera preview
            camera.start_preview()
            #set camera resolution to 640x480(Small resolution for faster speeds.)
            camera.resolution = (640, 480)
        time.sleep(0.1)
```

```
    print("Start focusing")

    max_index = 10
    max_value = 0.0
    last_value = 0.0
    dec_count = 0
    focal_distance = 10




    while True:
      #Adjust focus
          focusing(focal_distance)
          #Take image and calculate image clarity
          val = calculation(camera)
          #Find the maximum image clarity
          if val > max_value:
                max_index = focal_distance
                max_value = val

          #If the image clarity starts to decrease
          if val < last_value:
                dec_count += 1
          else:
                dec_count = 0
          #Image clarity is reduced by six consecutive frames
          if dec_count > 6:
                break
          last_value = val

          #Increase the focal distance
          focal_distance += 15
          if focal_distance > 1000:
                break

#Adjust focus to the best
    focusing(max_index)
    time.sleep(1)
    #set camera resolution to 2592x1944
```

```
camera.resolution = (1920,1080)
#save image to file.
camera.capture("test.jpg")
print("max index = %d,max value = %lf" % (max_index,max_value))
#while True:
#       time.sleep(1)

camera.stop_preview()
camera.close()
```


[7]
Rpi4 powersupply - https://www.raspberrypi.com/products/type-c-power-supply/

[8]
https://www.kintone.com/en-us/trial/?utm_term=kintone%20database&utm_campaign=kintone+-+Google&utm_source=google&utm_medium=cpc&hsa_acc=4164549067&hsa_cam=361704299&hsa_grp=31471075859&hsa_ad=380066541743&hsa_src=g&hsa_tgt=kwd-206695695419&hsa_kw=kintone%20database&hsa_mt=e&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAjwxZqSBhAHEiwASr9n9H4_t1JoAh-IkNGC-eY6rNvEfLUKHfx8XirhLVLTMhOY1jctHr-BXhoCqhwQAvD_BwE&order_id=03939703

[9] Usher, S. (2021). Free. Amazon. Retrieved April 1, 2022, from https://aws.amazon.com/free/database/?trk=c0fcea17-fb6a-4c27-ad98-192318a276ff&sc_channel=ps&sc_campaign=acquisition&sc_medium=ACQ-P%7CPS-GO%7CBrand%7CDesktop%7CSU%7CDatabase%7CSolution%7CUS%7CEN%7CText&s_kwcid=AL%214422%213%21548665196142%21e%21%21g%21%21amazon+cloud+database&ef_id=CjwKCAjwxZqSBhAHEiwASr9n9LXbJJmqEdzP5Emh0T3nyaSYHWZ7P9VG90p_3BHrbVhRMICSgxy4rhoCTz0QAvD_BwE%3AG%3As

[10] Loshin, P., & Sirkin, J. (2022, February 7). *What is structured query language (SQL)?* SearchDataManagement. Retrieved April 8, 2022, from https://www.techtarget.com/searchdatamanagement/definition/SQL#:~:text=Structured%20Query%20Language%20(SQL)%20is,on%20the%20data%20in%20them.

[11] Thumar, C., Author, Chirag Thumar Chirag Thumar is an experienced and innovative web developer at TechnoLigent, Chirag Thumar is an experienced and innovative web developer at TechnoLigent, & 2020) .شبکه نگهداری, January 7). *6 ways to optimize your SQL database*.

JAXenter. Retrieved April 2, 2022, from https://jaxenter.com/6-ways-optimize-sql-database-136448.html

[12] Perlitz, L., & Elliott, S. (2000). *The products*. Amazon. Retrieved April 3, 2022, from https://aws.amazon.com/products/security/

[13] *What is Amazon dynamodb? - docs.aws.amazon.com*. (n.d.). Retrieved April 6, 2022, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

[14] *Naming rules and data types - amazon dynamodb*. (n.d.). Retrieved April 6, 2022, from https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.NamingRulesDataTypes.html

[15] *Lifting and material handling*. Environment, Health and Safety. (2021, February 15). Retrieved April 24, 2022, from https://ehs.unc.edu/workplace-safety/ergonomics/lifting/#:~:text=Try%20to%20keep%20your%20elbows,do%20not%20twist%20while%20lifting.

[16]https://aws.amazon.com/blogs/compute/new-using-amazon-ec2-instance-connect-for-ssh-access-to-your-ec2-instances/

[17] https://docs.amplify.aws/start/q/integration/flutter/

[18] *USA - Power Plugs & Sockets: Travel adapter needed?* POWER PLUGS AND SOCKETS OF THE WORLD. (n.d.). Retrieved April 25, 2022, from https://www.power-plugs-sockets.com/us/united-states-of-america/#:~:text=In%20the%20United%20States%20of%20America%20the%20standard%20voltage%20is,the%20United%20States%20of%20America.

[19] realgastropub_po03iu. (2022, March 8). *How much electricity does a mini fridge use per month?* REAL gastropub. Retrieved April 25, 2022, from https://www.realgastropub.com/blog/how-much-electricity-does-a-mini-fridge-use/#:~:text=Energy%20Consumption,-Energy%20consumption%20varies&text=According%20to%20Consumer%20Reports%2C%20the,a%20year%20for%20conventional%20refrigerators.

[20] https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-reduced-schematics.pdf

10. Appendix B: Images

[20]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fus.eufylife.com%2Fproducts%2Ft8790&psig=AOvVaw1K83fwUakRRKZKHRpwmopu&ust=1648307727300000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCMCSyffG4fYCFQAAAAAdAAAAABAD

[21]
https://www.google.com/aclk?sa=l&ai=DChcSEwjV7dvOyeH2AhXI3sgKHTIBC7IYABAXGgJxdQ&sig=AOD64_0ihcA3s_ME_O6XBNRTbQP0kYSRxQ&adurl&ctype=5&ved=2ahUKEwj16sjOyeH2AhUnh-AKHYMvCqkQvhd6BQgBEJoB

[22]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.amazon.com%2FRaspberry-Model-2019-Quad-Bluetooth%2Fdp%2FB07TC2BK1X&psig=AOvVaw064yzgeei1JQ1FH8S_N5fS&ust=1648307537498000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCNiSipzG4fYCFQAAAAAdAAAAABAE

[23]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fdeveloper.nvidia.com%2Fembedded%2Fjetson-nano-developer-kit&psig=AOvVaw2Hl6nRaB6QMWyASfzRCrWZ&ust=1648307612422000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCOjz67_G4fYCFQAAAAAdAAAAABAF

[24]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.ti.com%2Ftool%2FMSP-EXP430FR4133&psig=AOvVaw3eyVCgngBX8evcr2EedY04&ust=1648307635822000&source=images&cd=vfe&ved=0CAsQjRxqFwoTCPjy_NDG4fYCFQAAAAAdAAAAABAH

[25]
https://www.amazon.com/Cosco-Outdoor-Living-88333BTN1E-Lockable/dp/B07XRS7TTB/ref=asc_df_B07XRS7TTB/?tag=hyprod-20&linkCode=df0&hvadid=385575019675&hvpos=&hvnetw=g&hvrand=9604317679225963375&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9011767&hvtargid=pla-824144697051&psc=1&tag=&ref=&adgrpid=77759961519&hvpone=&hvptwo=&hvadid=385575019675&hvpos=&hvnetw=g&hvrand=9604317679225963375&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9011767&hvtargid=pla-824144697051

[26]
https://www.amazon.com/dp/B012V1HEP4?psc=1&ref=ppx_yo2ov_dt_b_product_details

[27]
https://www.amazon.com/dp/B07SN8GYGD?psc=1&ref=ppx_yo2ov_dt_b_product_details

[28]
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.hackatronic.com%2Fraspberry-pi-4-specifications-pin-diagram-and-description%2F&psig=AOvVaw3CHFg788sdk2IbFODR2bfh&ust=1650909175903000&source=images&cd=vfe&ved=0CAkQjRxqFwoTCODf8IuirfcCFQAAAAAdAAAAABAI

[29]
https://www.delvallebox.com/en/outdoor-telecom-racks-19/enclosures-de-humidifier-for-electronic

[30]
https://www.globalindustrial.com/p/elements-34-1443-apw55959

[31]
https://www.npr.org/2021/11/02/1051504165/facebook-delete-facial-recognition-data-privacy

[32]
https://www.npr.org/2021/10/06/1043600094/ex-facebook-employee-frances-haugen-testifies-before-senate-panel

[33]
https://www.gao.gov/assets/gao-20-522.pdf

[34]
https://news.energysage.com/tesla-charging-cost-vs-gas/#FAQ

[35]
https://www.fhwa.dot.gov/ohim/onh00/bar8.htm