IntelliRack

Authors: Ryan Lin, Doreen Valmyr, Surafel Tsadik
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—This project introduces a solution for streamlining the coat checking process at large events and conventions. The storing and retrieval process is typically time-consuming and inefficient as it involves waiting in long lines and distributing tickets to event attendees for identification purposes. During item retrieval, attendees may misplace their tickets and attendants must sift through numerous items in order to locate a specific item, making it difficult to efficiently locate items. The aim of our project is to automate this process by making a system that integrates a facial recognition system with a physical hardware item stand.**

***Index Terms*—Arduino, facial detection, facial recognition, autonomous, OpenCV, dlib, Python, user interaction, facial embeddings, web application**

1 INTRODUCTION

At various events and conventions worldwide, attendees often need to check in their belongings temporarily. Whether for security purposes or simply to organize the event space, this process typically involves interaction with a personal item checking attendant. Attendees exchange their items for an identifying claim ticket, after which the attendant stores the items in a designated area.

There are several problems that can arise. For example, an attendee can lose their identifying ticket. This usually results in a long and irritating game of the attendee describing various features of their personal item while the attendant scrambles to identify it. With no clear identification, the attendee can also steal an item that may not belong to them. A more general problem is the sheer amount of time this process takes. Attendants, given a ticket, have to search through possibly hundreds of personal items, resulting in long queue times and frustration among attendees.

This project aims to address these challenges by implementing an automated system for personal item retrieval, utilizing facial recognition as a secure and reliable identifier. Such an approach has the potential to significantly streamline the personal item checking process, enhancing security and efficiency while ensuring attendees' belongings are safely managed.

Currently, there seems to be one notable competing product called CoatChex. CoatChex eliminates the need for a claim ticket by providing a kiosk where attendees can sign in using their phones. Our approach provides more functionality and is easier to use. Firstly, CoatChex still relies on numbering a personal item and manually placing them on a rack. It still relies on an attendant to search for an item, which could slow down the process as described earlier. Secondly, CoatChex requires the user to sign in on their phones which takes some time, while our project seamlessly uses facial recognition to automatically recognize a user and their personal item by simply walking up to the system.

2 USE-CASE REQUIREMENTS

2.1 Facial Recognition

To ensure that users can check in their items using facial recognition, our solution must incorporate an accurate face detection and recognition system. For facial recognition, our system must be capable of identifying users from a maximum distance of 0.5 meters within 5 seconds. This not only facilitates a seamless check-in process for users but also prevents bystanders walking in the background from inadvertently triggering the check-in process. Additionally, we aim to achieve an accuracy rate of at least 95%, erring on the side of false negatives. Accurately identifying individuals with high precision is essential to ensure that users receive their correct items, while also mitigating the risk of mistakenly assigning items to the wrong user.

2.2 Item Deposit/Retrieval

To ensure that the system can move onto to the next user in a timely manner, we want to implement our hardware such that users can interact with the system quickly. We also want to make sure that when a user has deposited or retrieved their item, that their changes are quickly read so our system knows what action to take next. We aim to use load cells to detect when an item has been placed or removed from the rack within 1 second, then propagate this information to the web application. We also plan to have the physical rack find the relevant position and rotate to it within 7 seconds.

2.3 Weight

We aim to enable users to place heavier items, such as backpacks, onto our rack without encountering any issues. Therefore, our objective is to ensure that our rack can withstand weights of up to 20 lbs on each hook without impeding the rotation of the rack. Considering that our rack will feature 6 hooks for hanging items, it must be capable of supporting a maximum load of 120 lbs collectively without compromising the stability of the rack. Additionally, we need to account for potential weight imbalances on the rack resulting from items being placed on or removed from it.

To address this, our system will incorporate a mechanism to manage these imbalances by determining the optimal rack position while considering the weights of the items.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

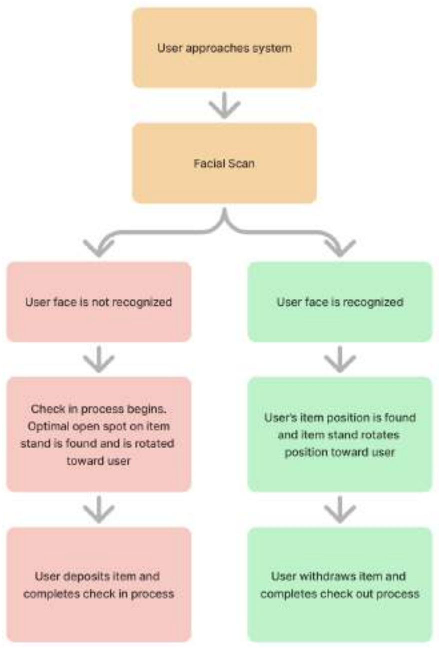

User approaches system
Facial Scan
User face is not recognized
User face is recognized
Check in process begins. Optimal open spot on item stand is found and is rotated toward user
User's item position is found and item stand rotates position toward user
User deposits item and completes check in process
User withdraws item and completes check out process

Figure 1: System Flow Chart

The user flow diagram is illustrated in Figure 1, and the system block diagram is shown in Appendix Figure 13. A user initiates interaction by approaching the camera and positioning their face within a distance of half a meter and at the center of the frame. Subsequently, upon user authentication, the check-in process commences. In the system's back-end, available open slot positions on the coat stand are assessed, and an optimal location for the user's item placement is determined. The rack then rotates accordingly, facilitating item placement by the user. Once completed, the user can proceed to their event activities. The check-out process involves a similar procedure, where the user repeats the initial steps but removes their item from the rack instead. The overall system remains the same as described in the Design Report, with the addition of new features like stealing detection and logs allowing staff to see who is currently checked in.

3.1 Facial Recognition

When the user approaches our system, they will scan their face to begin the facial recognition process. When their face is detected, its facial embeddings will be compared to existing embeddings. If we have seen the face before, our system will communicate to the rack which item belongs to the user. If the user is checking in then our system won't recognize the face, which is when their face information will be saved and compared against future check in or check out processes. They will then be prompted to put their item on the rack.

3.2 Item Stand

The rotation of the rack is made possible though the NEMA 34 Stepper motor, which will slowly rotate the rack until stopping at an empty location where the user can place their item. Once an item is placed, the readings from the load cell are passed through an ADC and processed by an Arduino Mega. These read weights will be stored in the memory of the Arduino Mega, and used to later compute the optimal locations for other users to place their items.

3.3 Engineering Principles

One of the most important considerations and engineering principles taken into account when implementing this system was safety. Our system incorporates a physical subsystem, the item stand, comprising various hardware components, including load cells, a NEMA 34 stepper motor, and power cables. Users must approach the rack to store their items, making it imperative for us to implement safety precautions to prevent harm. Firstly, we addressed the rotation speed of the motor. If the rotation is too fast and a user places items or fingers near the stand during rotation, they may be caught, potentially leading to breakage. To mitigate this risk, we initially set the motor to a very low speed during item rotation. However, such a low speed resulted in users having to wait approximately 1-2 minutes for the motor to come to a complete stop at their assigned position, which did not meet our efficiency requirements. We gradually increased the speed until reaching an optimal level, fast enough to rotate items to the user's assigned position, yet slow enough to minimize potential harm. Despite these safety measures, risks remained due to the moving components. Therefore, we incorporated LED indicators on the rack and provided instructions in the web application (as seen in Figure 2). Upon interacting with the web application, users are instructed to only place or remove items from the rack when the yellow LED is blinking. When the top of the item stand begins to rotate, the LED turns off, signaling users to refrain from taking any actions. Once the motor comes to a full stop, the LED begins blinking again, indicating to users that they may approach and place or remove their items safely. Lastly, electronic components are held within a partially closed area on the item stand, meaning that users would not be able to interact with such components. These features and design characteristics enhance safety during user interactions with the item stand.

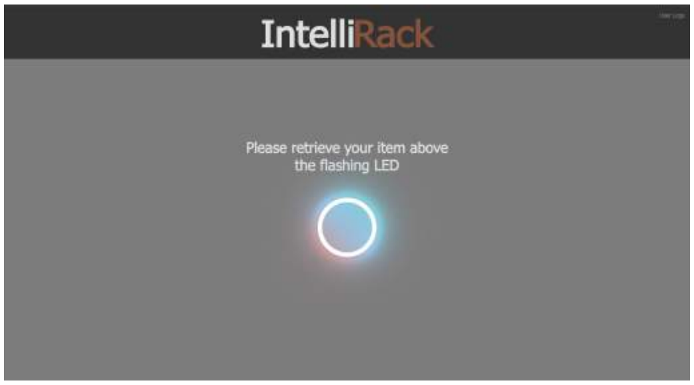


Figure 2: Check-out Instruction Page

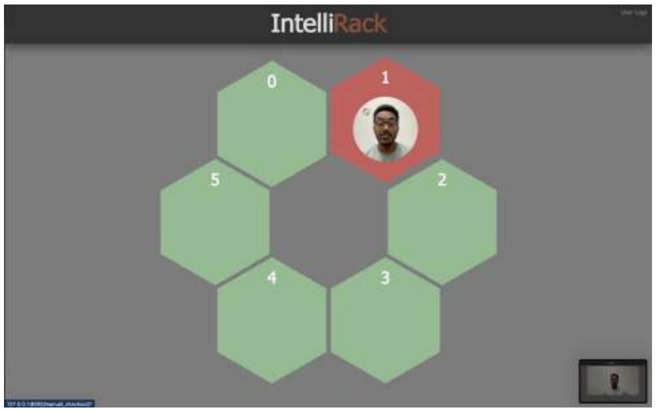


Figure 3: Web Application Log Screen

3.4 Scientific Principles

In developing our final product, we conducted extensive testing on each component. This aligns with the scientific principle of reproducibility. Testing each component of our system helped determine whether we had met specific functionality requirements, and it also provided a basis for others to work from if they continue our work. The experiments and evaluations conducted on each component were systematic, well-documented, and repeatable, allowing others to replicate and validate the findings. For example, others can examine the various tests performed on the facial recognition algorithms that our team implemented and compare their results to our own.

3.5 Mathematical Principles

In developing our automated personal item retrieval system, a key principle of mathematics that played a pivotal role in its design and implementation was probability theory, which underpins the facial recognition algorithm used to identify attendees. Our algorithm analyzes facial features and compares them to stored embeddings, calculating the likelihood that a given face matches a known identity. In order to correctly identify users, we had to reason about threshold parameters for determining matches ensuring that the accuracy of the facial recognition system was high. Furthermore, optimization theory plays a crucial role in optimizing the system's efficiency and performance. Mathematical optimization algorithms are utilized to optimize the facial recognition algorithm's parameters, such as thresholds and feature weights, to maximize accuracy and minimize false positive and false negative identification rates.

4 DESIGN REQUIREMENTS

We have established specific design requirements for each of our subsystems to enhance usability and minimize frustration for all potential customers, including event attendees and event managers.

4.1 Facial Recognition

The first design requirement pertains to our facial recognition system. Facial recognition is one of the key pillars of making the experience of interacting with our system easy and worry free. Our design goal for the facial recognition system is a 95% facial recognition accuracy rate. The aim for this high accuracy rate because we map faces to people's personal items and keeping attendees' items secure is very important.

While we want to have high accuracy, speed is very important as well. Our second design requirement is that a user's face should be recognized within 5 seconds. If a facial recognition system takes too long to recognize your face, a user can get frustrated. Extra time can also slow down the overall check-in process if dozens to hundreds of people have to check in their items. We choose 5 seconds because it is a relatively short time, was used by past capstone projects, and gives us flexibility with improving accuracy.

Lastly, the two previous design requirements should apply only to users that have come within 0.5 meters of the camera- outside of that nothing should happen. The system is designed for a fast and seamless user experience, but has to determine who is actively trying to use the system. Because the user has to place their item on the physical stand, we choose 0.5 meters as a reasonable distance a user would stand to check in their personal items.

4.2 Hardware Item Stand

The second set of design requirements revolve around the hardware item stand, which still store physical user items and interface with the facial recognition system for item retrieval.

The first design requirement is the quick detection items added to or removed from the system- within 1 second. This design requirement coincides with the general idea of reducing frustration for the user. The quick detection of physical changes can allow the rest of our system to know if it can move on to supporting the next person. 1 second is rather arbitrary, but is a very short time and entirely doable with our hardware.

The second design requirement is the quick display of users' items within 7 second, after the user's face has been processed (recognized so it is a check out process, or not recognized so it is a check out process). This requirement, along with the relatively fast facial recognition requirement, allows the user to retrieve their personal items as fast as possible and allows for the steady flow of users checking in or checking out their items.

We want to support a larger variety of items other than traditional coats, resulting in the last design requirement necessitating our item stand to withstand 120 pounds of weight. In contrast to coats, items such as backpacks can weigh up to or even exceeding 10 to 15 pounds- the reason we chose a 20 pound support goal on each of our 6 hooks. We want cater to hardware integrity by supporting a large weight.

Overall, our design requirements focus on making the software and hardware optimized enough to serve the customer, without any frustration or difficulty.

5 DESIGN TRADE STUDIES

There were several times the team made specific choices to narrow down the design. For example, the choice of a facial recognition library and its trade offs, or what kind of microcontroller or computer to use to read and control our sensors and actuators.

5.1 Design Specification for Facial Recognition

The first main component is our facial recognition system that will check user in and out. It is crucial that whatever detection and recognition model we used is not only fast enough, but accurate enough to meet our design requirements. We were able to narrow down our options to two toolkits, OpenCV or Dlib. After doing initial testing with manually inputted images as training data, both OpenCV and Dlib had their pros and cons. With Dlib, it had access to a shape predictor with 68 face landmarks, which would provide more accurate face alignment for each image processed (leading to more accurate and consistent recognition). Where Dlib falls short is its inability to easily implement real-time facial recognition through a video stream (which is what we are mainly looking for). For OpenCV, although their shape predictor doesn't have as many landmarks, OpenCV comes with built-in functions to easily run a video stream and start detecting/recognizing from them. Since this test also gave similar result for OpenCV as it did with Dlib, we originally decided to go with OpenCV.

OpenCV itself has two different ways of implementing face detection. One implementation is based on the Haar Cascade Algorithm, while the other implementation is a DNN face detector that is based on Single-Shot MultiBox (SSM) with the ResNet-10 architecture as the backbone. It was clear after a bit of research that the Haar Cascade face detection was very outdated and produced a lot of false positives (which we want to reduce as much as possible). Not only is the SSM face detection much more accurate overall, it has a greatly reduced false positive rate compared to Haar implementation and works well in real-time. So we chose to go with the SSM face detection. For the face landmarks and recognition, the clear choice for us to use is the recognizer created by the OpenFace toolkit. As the name suggests, OpenFace is a toolkit built on top of OpenCV that allows for facial landmark detection. This is the first of its kind, and it is perfect for what we need based on our requirements.

After implementing our facial recognition system using OpenCV, we conducted several tests during end to end testing. We wanted to see how our 2 implementations of facial recognition (one using a Gaussian kernel Support Vector Machine (SVM) classifier and one only using Euclidean Distance comparisons) compared with each other in terms of recognition accuracy, the results of which can be seen below.

Although we were able to hit the 95% design requirement with the SVM classifier, we did not consistently recognize faces in real-time with a video stream, so we decided to look for alternatives. We found a library called *face_recognition*, which provided an easy-to-use API for facial recognition. After running the same tests on an implementation using this library, we found the face recognition accuracy to far exceed that of the other two solutions, with an accuracy of about 99% (shown in Figure 10), while also working well in real-time with a video stream. Based on these tests, we decided to pivot to using the *face_recognition* Python library, which used Dlib on the back end but provided improved facial recognition accuracy.

5.2 Design Specification for Hardware Item Stand

The second main subsystem is our hardware stand. Because hardware is a substantial part of our project, a lot of potential approaches and ideas were discarded and revised.

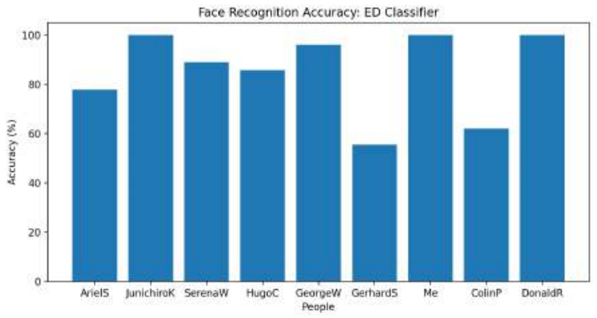


Figure 4: Bar Graph showing Face Recognition Accuracy for the Euclidean Distance Classifier. **Overall Accuracy: 86%**

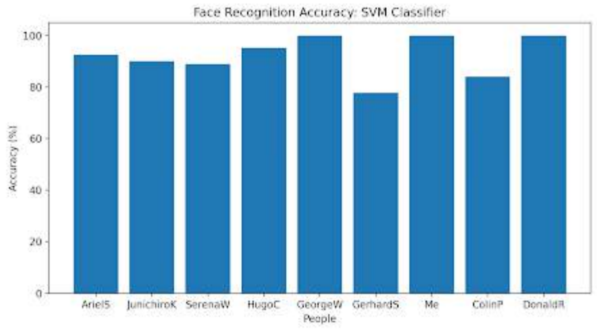


Figure 5: Bar Graph showing Face Recognition Accuracy for the SVM Classifier. **SVM Overall Accuracy: 95%**

5.2.1 Item Stand Design

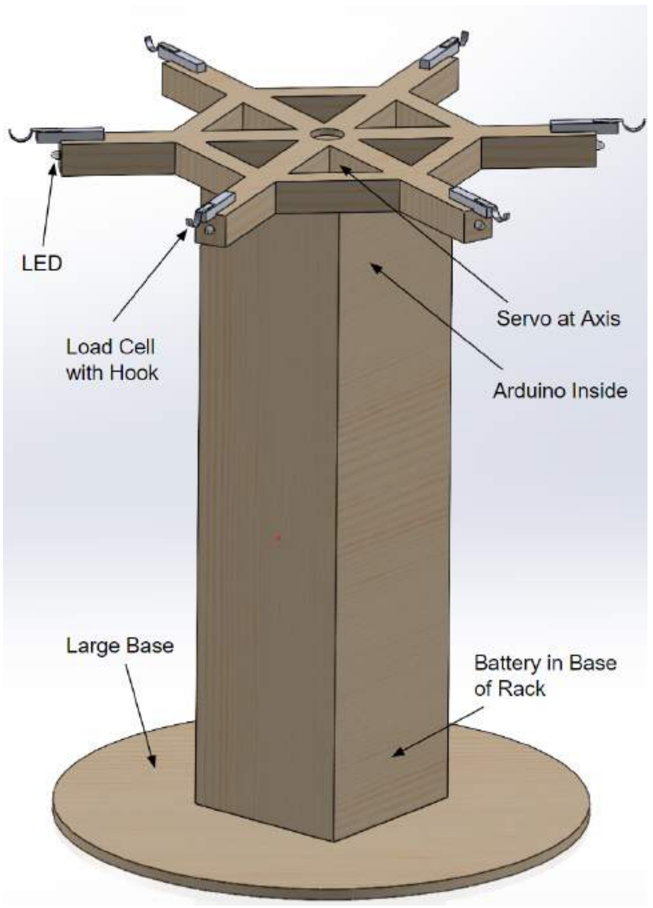


Figure 6: Original item stand design from the proposal presentation

The original item stand design is shown in Figure 6. It features a large, circular base, a smaller and thinner rectangular prism for the main shaft, and a large, hexagonal-shaped upper rack with 6 prongs that house hooks to store user items. The upper rack contained load cells, LEDs, hooks, and an Arduino to process sensor data and control the servo. A large battery in the base of the stand provided weight stability and portable power. After a qualitative assessment of the design and feedback from peers, we made shifted the design to abandon some original components.

The most glaring design flaw was the single axle support of the stand. With the possibility of 120 pounds on the upper rack in accordance with our design specifications, a single axle combined with the fact of no guarantee of even weight distribution could have been disastrous. Secondly, because there are no other connection points, there is no way to control the servo with the Arduino that is on the upper rack. The same reason applies to the inability of the battery at the base of the item stand to power the upper rack.

The second design flaw was the size of hexagonal upper rack. Due to the 6 upper prongs being much longer than the width of the mid rack, weight imbalance and tipping can happen easily, even if the single axle described earlier was reinforced. This ratio was originally chosen to allow for generous clearance of items that are hooked- up to 10 inches. For example, an very thick item could be hung on a hook without it brushing up against the middle main structure of the stand during rotation. But, after examining several common items brought to events, such as backpacks, purses, bags, and coats, we determined that this clearance does not have to be nearly has large. Just 3-4 inches is enough.

In our final item stand design which we will give an overview of later, we introduced a wider midsection and more robust rotation elements, as well as an upper housing which contained many of our electronic components.

5.2.2 Servos, Stepper Motors, and Motor Upscaling

The project required various components to support the design requirements. For the item stand to rotate upon the user check-in or check-out, we required an actuator in the form of a servo or a stepper motor. Originally, the team planned on acquiring and using a large servo, as shown in Figure 6, but there were some problems with this. Firstly, most servos can only rotate 180 degrees, which would not work for our use case of rotate the item stand from any position to face the user. Secondly, a servo, due to its precision angle-setting, is not as robust as a stepper motor. This is not ideal specifically for our project where the actuator has to potentially rotate 120 pounds. Lastly, a servo costs more than a stepper motor. Though the rotation of the rack is very important, the purchase of a component that could be more than $100 along with the other reasons mentioned resulted in our team choosing a NEMA 17 stepper motor. Peer feedback from the proposal presentation reinforced our choice.

After we purchased NEMA 17 stepper motors and installed them into our system, we realized that the torque output of a NEMA 17 was too low for our use case. Because a singular stepper motor was not strong enough to robustly rotate the rack, we decided to install a second backup NEMA 17 stepper motor, resulting in a total of 2 stepper motors rotating one gear in our system. However, this setup was still not strong enough to rotate the rack, even if only some small items were placed on it.

Ultimately, we decided to remove both of the small NEMA 17 stepper motors and purchased a large NEMA 34 stepper motor, with 4.8 newton-meters of torque, perfect for our use cause. This will be expanded upon in our system implementation section.

5.2.3 Load Cell and Load Sensors

Next, to detect item deposit or withdraw, some kind of weight measuring mechanism must be used. There were 2 sensors that could achieve this goal- load cells or load sensors. Load cells are more suited to measure tensile force because it measures different resistances on different sides of a slightly bendable bar. Load sensors measure compression weight, meaning they are the kinds of sensors used in a traditional floor scales. The load sensors could support the measurement of much more weight, while being cheaper. Our team devised various ways of using the compression load sensors to measure tensile weight. For example, if the hook was connected to a ring which was placed on each of the six prongs in Figure 6, then a downward force would compress a potential load sensor placed on top of the prong. Eventually, a load cell was chosen after balancing the extra layer of hardware complexity against the slightly higher cost.

5.2.4 Microcontroller and Computer

With these component choices, we needed to choose a processing unit to read and control them. Two potential choices were gathered- an Arduino Mega, or an Raspberry Pi (or any other small computer). Using a Raspberry Pi could change the solution approach drastically, because a facial recognition model could be built directly into the rack instead of running the model on a personal computer and wirelessly transmitting to an Arduino. With this approach, the system could be entirely self-contained within the rack. This solution route is a viable option, but was abandoned due to time constraints of the project, lack of experience with Raspberry Pi, and difficulties with using a Raspberry Pi as a microcontroller, especially since this project contains a lot of sensors. This means that the extra complexity will not bring any new features to our predetermined design requirements or use case. An Arduino is a easier component to use to control and read from our other components.

5.2.5 Powering Our System

Our original design, as shown in Figure 6, featured a battery to power the a servo and Arduino, allowing portability. But further deliberation and evaluation of our use case resulted in the team abandoning this component. The target customers of event organizers largely have no need of a portable item stand. Firstly, most events have ample power outlet or power transmission infrastructure. Secondly, if the item stand was populated with items up to 120 pounds, it would not be portable anyways. Lastly, a battery as the only source of power meant that the item stand could only be used for a certain period of time, compared to indefinitely when plugged into an outlet.

A large source of power drain in our system is from the stepper motors. Originally, we planned to power our stepper motors using 12 Volt and 2 Amp power bricks, which provided power to the motor drivers. The current going into a stepper motor largely determines the strength of the motor. Obviously, as we switched out our smaller NEMA 17 stepper motors, how we powered our system had to change. We introduced a new 24 Volt and 6 Amp power brick into the system, which had adequate current to power the new NEMA 34 stepper motor.

6 SYSTEM IMPLEMENTATION

6.1 Software Implementation

Though some the Arduinos in this project host some software, the main discussion in this section is about the part of the system that contains the web application and facial recognition code.

For the facial recognition system, we are using the *face_recognition* Python library and its various methods to find faces in a frame from OpenCV, obtain facial embeddings, and compare the facial embeddings to existing facial embeddings in the system. For face detection, the library uses the Histogram of Oriented Gradients (HOG) face detector provided by Dlib, which creates a simple representation of the flow from light to dark pixels in an image to compare with a HOG face pattern generated from many training images. Face landmark generation in the *face_recognition* is also provided by Dlib using their 68-point face estimation model. Finally, the *face_recognition* library uses a Deep Convolutional Neural Network already trained by Dlib to generate face encodings (128-d vectors) for images where similar images have vectors closer together in distance and different images are farther apart. This means that with this one library, we can detect user input (their face being scanned in), obtain their facial landmarks, and represent the user's face as a vector that we can use to compare to other faces in the system.

We use OpenCV to start a video stream and retrieve frames. Upon retrieving a frame, the *face_recognition* library is used to find the bounding boxes of all the faces in the frame. Our algorithm finds the largest face, ensures that the bounding box is big enough (close enough to the camera), and determines if the face is close enough to the center of the screen to avoid partial face capture. Afterwards, a face embedding is generated and compared against existing facial embeddings in the system. If there is a face in the system with a difference threshold of less than 0.4, then the face is recognized and the check out process is initiated. Otherwise, the check in process is initiated.

For the web application, we are using Django as the web framework of choice as it is a framework that uses Python as the back-end language of choice (we have the most experience with Python). Other than the Django framework, we are using the vanilla web stack (HTML, CSS, JavaScript) for the actual construction of the web application as something more complex isn't needed for the requirements of our project. We save the user's face frames into local file storage to allow for easy access and deletion when running face recognition. The purpose of the web application is to provide a simple camera feed so users can see if they aligned their face correctly. It also features a second page which shows the current state of the rack, with 6 hexagon positions. When a hexagon is green, the position is available. When the hexagon is red, the position is occupied and the user's face is displayed. A user can also be manually checked out of the system if facial recognition fails by simply clicking the occupied hexagon. This feature is only available to staff of the event or a manager of the item stand. This feature and the previously mentioned check in and check out processes requires communication with the physical rack. This communication will be created using a wireless transceiver connected to a laptop through an Arduino (more on this later in Section 6.2.3).

6.2 Hardware Implementation

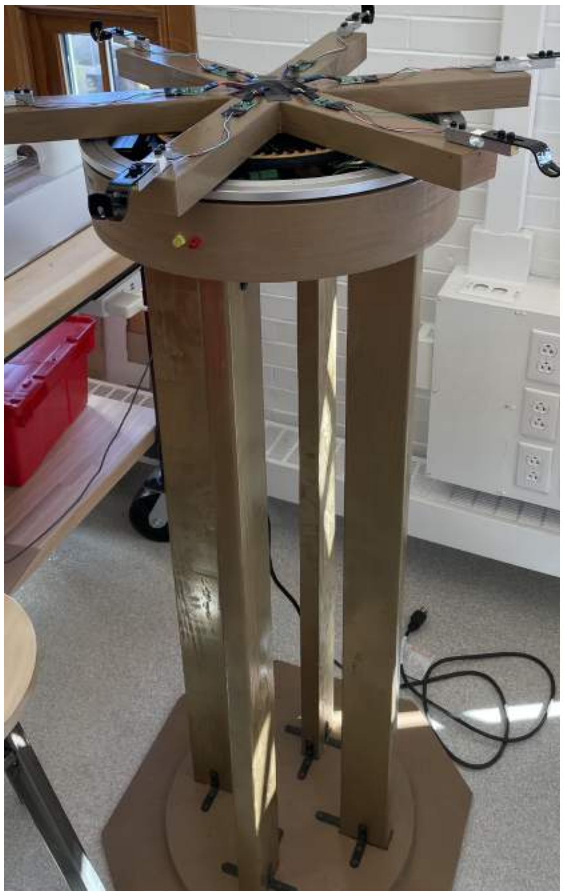


Figure 7: Final Item Stand Design

6.2.1 Item Stand Design

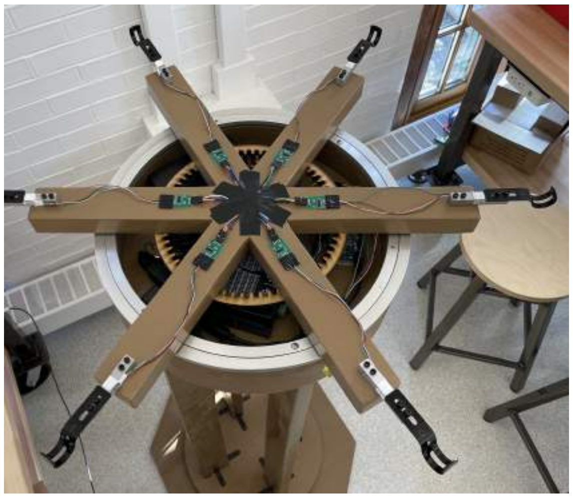


Figure 8: Top View of the Item Stand

The final item stand design is shown in Figure 7, featuring a hexagonal-style, 6-pronged rotating structure as the "upper" rack. This rotating component will be mounted on a turntable bearing, facilitating both support for the items placed and rotational movement to display specific positions for user item placement. Positioned at the endpoints of the hexagonal structure are load cells and large black hooks. On each of the 6 prongs, the load cells wires feed into a HX711 ADC for signal amplification. The wires from the HX711 from each of the 6 sides are fed down through the middle of the rack using a slip ring, allowing wires to withstand the rack rotation.

Below the top hexagonal structure is a circular well area containing the NEMA 34 stepper motor, Arduino Mega, and other components. The circular well also contains an internal gear connected to the upper rack as shown in Figure 7 and 8. The internal gear, with a diameter of 11 inches, will be connected to a smaller 2-inch diameter spur gear. This smaller spur gear will then be connected to the stepper motor, serving as the mechanism to provide rotational power for the rack.

The bottom of this circular area is then connected to 4 planks and a large wooden base which will provide stability for the rack in the case of uneven weight distributions. The entire rack stands at about 4.5 feet tall, allowing users to easily place their items while standing upright.

6.2.2 Electronics

The item stand incorporates several electronic components, including an Arduino Mega, six load cells, six HX711 ADC amplifiers, a NEMA 34 stepper motor with an accompanying DM860I driver, NRF24L01 wireless transceiver, 12 wire slip ring, extension cords, various power bricks, and 2 LEDs.

Due to the small analog signals produced by the load cells, each load cell is paired with an HX711 ADC. As mentioned before, the wires from each HX711 are routed through a central slip ring in the rack to connect with the Arduino Mega. Because 4 wires came from each HX711 (24 total wires) and our slip ring can only support 12 wires, we decided to soldier the power, ground, and clock lines from each of the 6 ADCs together, resulting in only 9 wires going through the slip ring. Each load cell outputs analog values, so we also calibrated each load cell in software using known weights, allowing our code to output a specific human-readable weight value for each scale reading.

Additionally, the Arduino Mega controls the NEMA 34 stepper motor, with 6 amps of current provided by the DM860I motor driver. The Arduino Mega is powered by a 12 volt, 2 amp power brick while the motor driver is powered by a 24 volt, 6 amp power brick to support max holding torque on the NEMA 34. The NEMA 34 is responsible for rotating the item stand to a specific position upon instruction from the Arduino Mega.

The Arduino Mega also control some other components. It is interfaced with a NRF24L01 wireless transceiver so it can send and receive wireless messages through RF signals to and from the rack. The Arduino also controls 2 LEDs, which are attached to the outside of the circular well. The LEDs serve various functions- the yellow LED blinks when there is a pending item retrieval or deposit, while the red LED blinks if a process has timed out or if there is an unauthorized removal of items from the item stand.

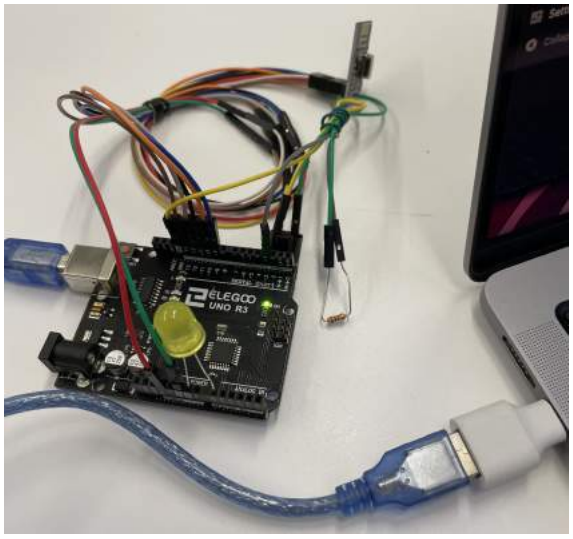


Figure 9: Another Arduino Plugged Into Laptop

Figure 9 shows the other electrical component to our system- Arduino connected to our laptop. This Arduino Uno features a second NRF24L01 to communicate with the wireless transceiver on the item stand (which is connected to the Arduino Mega). It also features a buzzer, which creates noise when the Arduino Uno receives a signal that there may be an unauthorized removal of weight from the rack. Originally, we tried to connect the buzzer to the Arduino Mega on the item stand, so that as much of the electronics as possible is on the item stand. However, when this approach was tested, our system suffered continuous failure for multiple days, possibly due to the high number of components the Arduino Mega already has to interact with. We decided to test out implementing the buzzer on Arduino Uno, and it worked out fine.

6.2.3 Connection to Software

The item stand does not contain any facial recognition or user processing logic, which is done exclusively on a personal laptop. As mentioned, in our system, the Arduino Mega is attached to an NRF24L01 wireless transceiver, which will communicate with an Arduino Uno with the same component connected by serial port to a personal laptop. The laptop is connected to a camera and runs the facial recognition model to ensure items are mapped to faces. Once a user has their face scanned, the software will determine where the a user can deposit or withdraw their item. Then, the software will write to the Arduino Uno through the serial port which will transmit check in instructions or a checkout instruction with a target rack position. The rack can then read if an item has been placed or removed via load cell reading, and transmit this information the same way back.

The Arduino Mega on the item stand also contains software that reads from the wireless transceiver into a custom message type. If the message is a check in, the program uses the balanced-rack algorithm to find the best position for the user to place their item. This algorithm takes into account the weight and positions of existing items on the rack. If the message is a check out instruction, we take the rack position read from the message type. Next, the program rotates to the ideal position then continuously polls on the load cell weight readings until a weight threshold is reached, while flashing a yellow LED to signal a pending transaction. If its a check in process, any weight reading above 0.3 pounds will signal a success that is written back to the other Arduino (the rack position is also passed). If the process is a check out, the load cell will poll until the weight is below 0.3 pounds then send a success signal. The weight threshold of 0.3 pounds was chosen because most objects, such as small water bottles, do not weight that light. We also want to mitigate issues where a load cell might skew or be slightly off of actual weight. Keep in mind that the load cells output minuscule signals where we require an HX711 ADC amplifier to read it, meaning any tiny fluctuations in sensor output can cause a couple of ounces of phantom weight.

If the process does not finish in the time allotted (30 seconds), the transaction will be considered a failure, the red LED flashes, and a failure signal is written back to the other Arduino. In the case that a user attempts to steal items from the rack without being in a process, the rack will detect a weight decrease because it is continuously polling on the weight and will send a signal to the Arduino Uno to trigger the buzzer.

In contrast, the software on the Arduino Uno connected to our laptop is much more simple. It handles reading from serial, interpreting the instruction from the web application, and writing the command so that the Arduino Mega on the rack can process it. This Arduino can then receive several messages back. Success or fail for the current process, or a message to trigger the buzzer because of unauthorized weight changes. If a process ended, the program will write to serial so that our web application can read the results and keep track of user metadata appropriately.

7 TEST & VALIDATION

To verify the integrity and robustness of our system, we conducted various unit and functional tests. This included thorough testing of both the facial recognition software and the physical coat stand for integrity and sturdiness.

7.1 Results for Facial Detection and Recognition

We envision our autonomous coat rack system being utilized in large events characterized by high volumes of attendees. We have designed our system to detect passing faces without attempting to recognize them. Our design requirement is that we only recognize users within a distance of 0.5 meters or less from our camera. If the distance exceeds 0.5 meters, our system should still detect them but refrain from trying to check them in or out.

For testing purposes, we defined four distances between potential system users and our camera: 0.25m, 0.5m, 1m, and 2m. At a distance of half a meter, our system should detect that a person is in the vicinity of our camera and within the specified range distance ([0m, 5m]), indicating that a user is likely attempting to check in or check out a personal item. At distances of 1m and 2m, our system should be able to detect a person's face without attempting to check them in or out, as they are likely to be passersby. To conduct this experiment, we measured the four distances using a measuring tape. One team member was positioned further than 2 meters, and they slowly walked towards the camera, stopping when the system indicated that they were within the recognizable distance. From this point, we measured the distance between the camera and where the team member had stopped. At the 1 meter and 2 meter marks, the system did not attempt to do a check-in. At the 0.5 meter mark, however, the system successfully tried to do a check-in. When the face was positioned at the 0.25m distance, the system was able able to start recognition. We conducted this test for three rounds, with all rounds producing the expected results.

In addition to this, we asked 10 volunteers to check-in or check-out items on the item stand, and when doing so, we made sure to watch how the system behaved with users passing by and walking close to the system. We noticed that for each volunteer, the system would only recognize them within the specified range. In conducting this test, we were able to ensure that the facial recognition system would not attempt to check-in users who were simply passing by, even if they were in the camera frame, satisfying our use case requirement.

Our second design requirement for the facial recognition system was that a user would be recognized within 5 seconds. To test this, we added print statements wrapping our recognition code to time the amount of time the system took to recognize a user's face. For example, we added a statement for when the recognition began and one for when the recognition ended, and used a timer to track the absolute difference between the times of these messages. For the SVM classifier, the average recognition time was about 4.05 seconds. Although these results met our design requirement, we aimed to implement a facial recognition system that was faster to further improve the efficiency of the entire check-in or check-out process. After implementing the facial recognition system using the *face_recognition* library, we again timed the amount of time it took for a user to be recognized. To do this, we asked about 10 volunteers to use the system to check in their items. Once a user was recognized, their frame froze in the web application for a moment, and a signal was sent to the item stand, making it rotate to the user's position. We manually timed the length of time that this took. For all tests, the amount of time it took for the user to be recognized was less than 1 second, meeting our 5 second requirement.

Our final design requirement for facial recognition was to achieve an accuracy of 95%. This ensures that users who check in their items can successfully retrieve them later on. To test the accuracy of the facial recognition, we used 20 faces from online datasets and calculated the percentage of time that our system correctly identified the faces. Using the *face_recognition* library, we achieved an accuracy of 99% (as seen in Figure 10), surpassing our accuracy requirement. However, in real-life scenarios, the achieved accuracy was likely lower due to factors such as the brightness of the testing location and the resolution of the external webcam used for testing.

By considering the distinctions between facial detection and recognition, defining specific criteria for user recognition based on distance, and setting clear performance benchmarks for our facial recognition system, we were able to satisfy all of our design requirements, thus improving the overall effectiveness and reliability of the entire system.

7.2 Results for Item Placement/Removal & Item Position

Upon recognizing a user, we anticipated that the user would move towards the coat rack to either deposit or retrieve their personal item. It is crucial that we accurately discern weight changes on the rack to determine the availability of storage space for additional users. Therefore, we included the requirement to promptly ascertain, within 1 second, when a user picks up or places down their item. This enables us to promptly update our program's data and accommodate the needs of subsequent users who are likely waiting in line.

To test this, we repeatedly placed and removed items on hooks for 4 rounds and monitored whether the system successfully identified when an item was placed or removed. We added timing code to our Python program to print the time that it took for weight updates from the rack to the propagated to the the computer running the web application. For both the check-in process and the check-out process, the time for the weight updates to be propagated was about 609 ms as demonstrated in Table 1.

In a similar vein, we aimed to display items or available slot positions within 7 seconds. To measure this metric, we tested how long it took for the NEMA 34 motor to rotate the empty stand one revolution (equivalent to 4 actual revolutions of the motor shaft due to the 4:1 gear ratio) at a safe speed. We wanted the motor to function optimally even with additional weight on the stand, up to 120 pounds. Additionally, any processing required to check for stolen items or detect weight changes should be almost instantaneous, meaning that the motor should rotate toward any user's position within 7 seconds. This ensures minimal waiting time for users interacting with our system. We conducted testing by checking in various items and timing how long it took for the motor to rotate from its current position to the target position assigned to the user's hook.

For checking in items, we used 6 items of various weights and placed them in the same order for all 4 rounds. This ensured that the weight distribution on the rack remained consistent for all rounds, ensuring that the same positions would be assigned for each item. In the check-out process, users may check out items in a random order. We simulated this by randomizing which items we checked out first in the 4 rounds of testing. Our results show that the item stand can process data very efficiently, as it begins rotating to the user's position instantly, requiring an average of only 4 seconds to do so, with a maximum average of 6.36 seconds across the 4 testing rounds as demonstrated in Table 2.

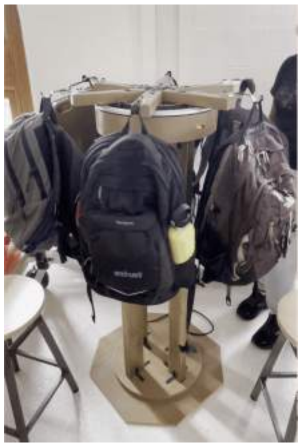


Figure 11: Item Stand Supporting 120 lbs

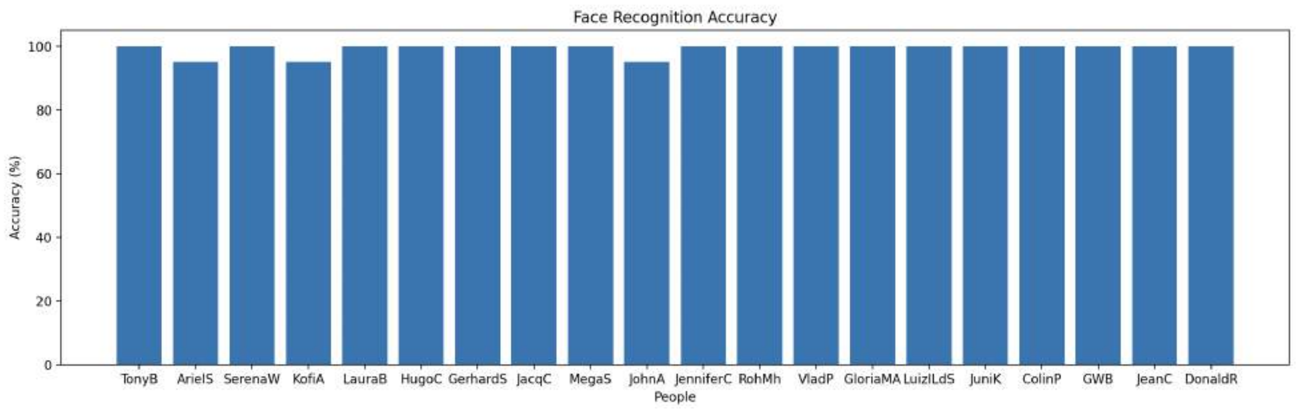


Figure 10: Final Face Recognition Accuracy Graph

Check-in (ms)	Check-out (ms)
610.51	607.40
607.55	608.28
612.48	608.36
606.54	609.30
AVG: 609.27	**AVG: 608.34**

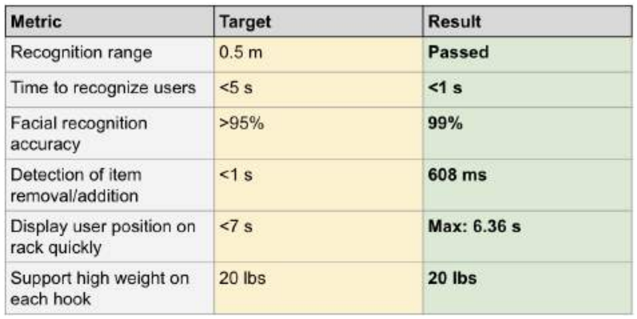
Table 1: Check-in/Check-out System Propagation Time

7.3 Results for Rack Integrity

The rack component of this project consists of six hooks, evenly distributed around the center of the stand. Each hook is designed to withstand weights of up to 20 pounds individually, with a maximum load capacity of 120 pounds for the entire rack. To validate this, we conducted comprehensive testing involving the placement of varying weights on each hook, ranging up to 20 lbs.

To test whether each hook could support 20 lbs, we placed 20 lbs on each hook, one at a time, and rotated the rack 360 degrees. We observed that the wood slightly bent, as expected, but each hook was able to support the weight. To examine whether the rack would remain upright and balanced with an uneven weight distribution, we placed 60 lbs on one side and 0 lbs on the other, checking if the motor could smoothly rotate that weight and if there was any instability. Our results demonstrate that the large base of the rack helps prevent instability even with such an uneven weight distribution. Lastly, to test our design requirement, we placed 20 lbs on each of the hooks to determine if the item stand could support a total weight of 120 lbs. After applying this weight, we rotated the motor to verify if it could handle such weight without skipping steps. Since the NEMA 34 stepper motor has a holding torque of 4.8 Nm, whereas our use case requirements only require 1.98 Nm, this test was successful. An image of the rack support the maximum weight requirement can be found in Figure 11. This extensive testing regimen helped ensure that each hook could reliably support items of different sizes and weights.

Ensuring the durability and stability of the rack is paramount to its functionality. By subjecting it to rigorous testing, including weight distribution and imbalance scenarios, we helped to verify that it withstand real-world usage conditions. This testing approach not only validates the rack's capacity to securely hold items but also enhances user confidence in its reliability.



Metric	Target	Result
Recognition range	0.5 m	**Passed**
Time to recognize users	<5 s	**<1 s**
Facial recognition accuracy	>95%	**99%**
Detection of item removal/addition	<1 s	**608 ms**
Display user position on rack quickly	<7 s	**Max: 6.36 s**
Support high weight on each hook	20 lbs	**20 lbs**

Figure 12: Overall Test Results

8 PROJECT MANAGEMENT

8.1 Schedule

The Gantt Chart presented in Appendix Figure 14 outlines the project timeline for the semester. It includes several significant tasks necessary for completing the project, with each task color-coded to indicate the responsible team member. This schedule differs slightly from the one shown in the design report, particularly in the lack of allocated slack time and its distribution among various project

Action	Round 1	Round 2	Round 3	Round 4	Average
Check-in: Position 0 -> Position 0	<1s	<1s	<1s	<1s	<1s
Check-in: Position 0 -> Position 2	3.89s	3.88s	3.95s	3.67s	3.85s
Check-in: Position 2 -> Position 4	4.27s	3.88s	4.00s	4.13s	4.07s
Check-in: Position 4 -> Position 1	5.06s	5.18s	5.58s	5.57s	5.35s
Check-in: Position 1 -> Position 3	3.88s	3.68s	3.88s	3.82s	3.82s
Check-in: Position 3 -> Position 5	4.40s	3.48s	3.82s	4.07s	3.94s
Check-out: Position 5	2.57s	3.95s	3.97s	5.13s	3.91s
Check-out: Position 4	2.44s	5.00s	4.27s	5.13s	4.21s
Check-out: Position 3	3.76s	3.75s	3.95s	4.33s	3.95s
Check-out: Position 2	5.64s	3.88s	2.59s	5.64s	4.44s
Check-out: Position 1	2.58s	2.45s	2.51s	4.34s	2.97s
Check-out: Position 0	5.25s	7.92s	5.38s	6.89s	6.36s

Table 2: Time to Display User's Position

tasks. Towards the end of the semester, the team encountered some delays, highlighting the importance of the preallocated slack time.

8.2 Team Member Responsibilities

Surafel led the facial recognition aspect of the project. Additionally, he was be responsible for creating a user-friendly web application, which houses the facial recognition algorithm. After the design report, Surafel gained some new responsibilities because of extra features proposed, such as manual user checkout.

Ryan and Doreen took the lead on the hardware components, which involved building the physical rack and integrating electrical components such as load cells and an Arduino Mega. This responsibility also included writing the associated code to program these components and specifying algorithms for how the rack should behave when interacting with users. After the design report and near the end of the semester, Ryan was also responsible for improving facial recognition, while both Ryan and Doreen was responsible for improving the user interface of the web application.

The integration of these sub-systems, end to end testing, and the preparation of presentations and reports, was a collaborative effort among the team. Team members provided support to one another in resolving any issues that arose or components that were unfinished.

8.3 Bill of Materials and Budget

The bill of materials, detailing the materials acquired this semester, can be found in Appendix Table 1. These materials are essential for the successful execution of our project. Additionally, wires, a breadboard, and small circuit components were repurposed from previous class projects.

Several materials were purchased but not used. While performing calculations and testing the integrity of the rack and its rotation ability, we determined that to support the weight requirement of 120 lbs, we would need a motor with a holding torque of 1.98 Nm. However, the NEMA 17 motor only has a holding torque of 0.7 Nm. Consequently, all components relating to the NEMA 17 motor were unused, including the stepper motor itself, the coupling for the stepper motor, and the motor driver for the NEMA 17 (A4988 & L298N). Additionally, the Arduino data cable was unused as it arrived broken.

Overall, the cost for our materials is $634.63. This cost exceeds the $600 threshold set for the class but was approved by our instructor. As described above, the NEMA 17 stepper motor was not powerful enough for our use case requirements, necessitating the purchase of a NEMA 34 motor with a holding torque of 4.8 Nm. In addition to the motor itself, we purchased the DM860I Stepper Driver and NEMA 34 flange coupling.

8.4 Risk Management

The project primarily revolves around two key components: the facial recognition sub-system and the physical hardware rack. The main risks associated with these components pertain to achieving accurate facial recognition and ensuring robust rotation of the rack. To meet user expectations and fulfill use case requirements, it's crucial for users to be accurately matched, and the rack should safely rotate to optimal hook positions, ensuring usability and safety for potential users.

One of the primary challenges encountered during the project was related to the facial recognition system. While this system is integral to our project, the original implementation outlined in the design report fell short of meeting the specified design requirements for recognition accuracy. To address this issue, we sought out an alternative library or solution to replace the original implementation. Our search led us to discover the *face_recognition* Python library, which not only achieved higher accuracy than the original implementation but also performed significantly better during real-time face recognition tasks. Additionally, to mitigate risks associated with facial recognition, we implemented a manual checkout feature (as seen from 3) that allows admins to manually check out a user from the rack without relying on facial recognition. This ensures that checked-in users can always retrieve their items, even if the facial recognition system encounters a malfunction.

Although weight limitations did not pose significant risks, an issue arose concerning the motor used on the rack. While having a rotating rack is essential for the user experience, the initially chosen NEMA 17 motors proved insufficient to smoothly rotate the rack. To address this, we explored various solutions with the available materials, such as adjusting motor placement and considering alternative rotation mechanisms. Ultimately, we opted to invest in the stronger NEMA 34 motor to ensure smooth rotation of the rack. Before purchasing the NEMA 34 motor, we conducted thorough calculations to determine the required torque for rotating the rack and confirmed that the chosen motor would meet these requirements. As safety is a paramount concern when the rack is in motion, we included instructions in the web application advising users to only interact with the rack when the yellow LED is blinking, indicating that it is safe to do so. This measure minimizes risks associated with the rotating rack by clearly signaling to users when it is safe to approach.

9 ETHICAL ISSUES

While our product solution is not primarily aimed at enhancing public health, security, or welfare, there are aspects of our project that positively contribute to this domain. For instance, eliminating the need for a ticket system and simplifying the process of checking in personal items can alleviate stress for individuals attending events. Moreover, with the addition of an alarm system that triggers when unauthorized item removal is detected, our product offers customers peace of mind by ensuring that their items are securely tracked and organized on our item stand.

Furthermore, the adoption of our system by organizations can lead to significant cost savings by reducing the need for event staffing. Additionally, our product eliminates the requirement for external identifiers such as tickets, further lowering event costs for organizations or event creators.

Our facial recognition algorithm utilizes tools like OpenCV to characterize and distinguish different faces, ensuring that factors related to a person's social group, whether cultural, political, or economic, are not considered when checking in or checking out a user. Furthermore, we ensure that the biometric data we gather and store using these tools are deleted when a user checks out of the system, alleviating concerns about the use of personal biometric data.

Additionally, our product can be used in a variety of social contexts and within all types of social gatherings that require attendees to set aside particular personal items when attending such events. Without strict rules being set on the items that can be placed on the rack, other than the assumption that it will mostly be used for coats, users will not be limited in the items they can set aside, but rather in the weights of those items. Consequently, users can utilize our product for their required social gatherings without limitations.

10 RELATED WORK

To initiate our preliminary research, we examined a project completed by a team in Spring 2021, which developed Smart Wardrobe [11]. This project integrated a rotating stand and a clothing recognition model to suggest outfits to users. Drawing inspiration from this project, we designed our rotating rack system, enhancing it with additional components and specifications such as load cells to measure the weight of items on our rack and LEDs to notify users where to place their items.

A team from Spring 2019 [1] developed facial recognition algorithms for their project and we took inspiration from their timing goals and application of facial recognition in every day life.

11 SUMMARY

Overall, we believe our system meets the design requirements. Through our testing, we achieved a high facial recognition accuracy, fast sensor data propagation, and ensured item stand robustness. During our public demonstration, users were able to check in and out their items using facial recognition. One user inadvertently left their personal tote bag in the system and wandered off to explore other projects for more than 20 minutes. They successfully retrieved their item upon returning.

11.1 Limitations and Improvements

There are certainly some limitations to the system's performance. The primary issue we encountered during the public demo was the inaccuracies of the facial recognition system. We suspect that this may have been due to sub-optimal lighting or background conditions in the area. Additionally, we used a relatively inexpensive webcam, and the lower quality of the camera could have significantly impacted facial recognition accuracy by making it harder to generate precise facial embeddings. Moreover, the camera lacked sufficient exposure, resulting in noticeably darker video feed captures.

Another factor contributing to lower facial recognition accuracy is how the system captures user faces. Since the system relies on a single frame to categorize users, it initiates a check-in or check-out process immediately when a user is close enough to the screen. As a result, the user's face is often in motion (moving forward to be detected), which introduces blur. While our testing involved large datasets of still user facial images under ideal lighting conditions, this setup may have contributed to high accuracy during testing but lower accuracy during the live demo.

If we had more time to refine the project and conduct further live testing, there are several improvements we would consider. Firstly, we would invest in a higher quality camera with higher resolution and better exposure to ensure that captured frames are brighter and contain more detail. Additionally, we could attach a light next to the camera to illuminate the user's face during scanning, reducing the impact of external lighting conditions on face clarity. Secondly, we would adjust our algorithm to allow for more time to capture the user's face, ideally when the user is standing still and not moving their face toward the camera. This could be achieved by introducing a short delay before recognition after detecting a user's face close enough to the camera.

11.2 Future work

There are several potential ideas for future improvements. One possibility is to address the issues related to facial recognition accuracy. Testing can be done in conditions where the lighting and camera resolution are subpar, to ensure that the facial recognition algorithm works even in sub-optimal conditions. Additionally, expanding the system could be considered. For instance, we could explore the creation of a larger rack or enable multiple racks to interact with a single web application. Currently, our solution is limited to assisting six customers at a time as a proof of concept.

11.3 Lessons Learned

One lesson we learned was the importance of ensuring that components align with your project's requirements before purchasing them. We invested significant time experimenting with NEMA 17 stepper motors before ultimately transitioning to the NEMA 34 stepper motor, resulting in wasted time and resources on components that were not used in the final implementation.

Testing proved to be another crucial aspect of our project. We meticulously tested each component upon receipt to ensure its reliability and suitability for integration into the system. This approach not only verified the functionality of individual components but also provided valuable insights into their operation, facilitating smoother integration during the later stages of the project. We highly recommend adopting a similar testing regimen for future projects.

Effective time management emerged as a vital skill during the project's execution. Given the open-ended nature of the assignment and minimal external oversight, maintaining a structured approach to task management and avoiding procrastination was essential. By adhering closely to our schedule and allocating time efficiently, we avoided last-minute rushes and were well-prepared for final demos. This experience provided valuable insights into project management and self-discipline.

Lastly, working on this project significantly expanded our knowledge across various domains, including hardware, software, and construction. Many of the technologies employed were unfamiliar to team members, such as load cells and facial recognition systems. We gained practical experience in woodworking techniques and the use of handheld tools, along with insights into integrating hardware and software components seamlessly. Overall, this project not only enhanced our technical skills but also deepened our understanding of fundamental engineering principles.

Glossary of Acronyms

- NEMA -National Electrical Manufacturers Association
- LED - Light-Emitting Diode
- ADC - Analog to Digital Converter
- HOG - Histogram of Oriented Gradients
- SSM - Single-Shot Multi-box
- SVM - Support Vector Machine

References

[1] Team A3. *Door ID.* 2019. URL: `https://course.ece.cmu.edu/~ece500/projects/f19-teama3/`.

[2] Ansh2919. *Serial Communication between Python and Arduino.* 2020. URL: `https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756`.

[3] Arduino. *Stepper Motors.* 2022. URL: `https://docs.arduino.cc/learn/electronics/stepper-motors/`.

[4] Ardumotive. *How to Use a Buzzer (Piezo Speaker) - Arduino Tutorial.* URL: `https://www.ardumotive.com/how-to-use-a-buzzer-en.html`.

[5] *Django CMS.* URL: `https://buttercms.com/django-cms/`.

[6] Last Minute Engineers. *Stepper Motor Control using L298N Motor Driver & Arduino.* URL: `https://lastminuteengineers.com/stepper-motor-l298n-arduino-tutorial/`.

[7] GreatScott! *Electronic Basics 33: Strain Gauge/Load Cell and how to use them to measure weight.* YouTube. 2017. URL: `https://www.youtube.com/watch?v=lWFiKMSB_4M&t=390s`.

[8] Maker Guides. *DRV8825 Stepper Motor Driver Arduino Tutorial.* 2019. URL: `https://www.makerguides.com/drv8825-stepper-motor-driver-arduino-tutorial/#Differences_between_DRV8825_and_A4988`.

[9] *Interactive CSS Grid Generator.* URL: `https://codepen.io/AdamDipinto/pen/eYOaGvY`.

[10] Microcontrollers Lab. *NEMA 34 Stepper Motor: Pinout, Features, Wiring & Interfacing with Arduino*. URL: `https://microcontrollerslab.com/nema-34-stepper-motor-pinout-features-wiring-interfacing-with-arduino/`.

[11] Fred Lee, Henry Lin, and Sung Hyun Back. *Team B2: Smart-Wardrobe*. 2021. URL: `https://course.ece.cmu.edu/~ece500/projects/s21-teamb2/`.

[12] How To Mechatronics. *Arduino Wireless Communication with NRF24L01 – Tutorial*. 2022. URL: `https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/`.

[13] Random Nerd Tutorials. *Arduino Load Cell HX711*. URL: `https://randomnerdtutorials.com/arduino-load-cell-hx711/`.

APPENDIX

Table 3: Bill of materials (Used)

Description	Model #	Manufacturer	Quantity	Unit Cost	Total
Arduino Mega 2560 Rev3	Arduino Mega 2560 Rev3	Arduino	1	$48.40	$48.40
10kg Load Cell + Amplifier 2 piece set	S22X2+S19X2	ShangHJ	4	$11.99	$47.96
Dailydanny Heavy Duty Lazy Susan	AluLS	Dailydanny	1	$35.99	$35.99
4pcs NRF24L01+ Wireless Transceiver Module	3-01-0416	HiLetgo	3	$7.89	$23.67
2x8 Feet 1/2 Inch Sande Plywood	-	-	3	$45.55	$136.65
2x4 Inch, 8 Feet Prime Stud Wood	-	-	3	$3.25	$9.75
Power Supply AC Adapter	4336304932	smooth-elec	1	$9.99	$9.99
10ft Extension Cable	B08Q1YLH8B	BindMaster	1	$7.99	$7.99
Curtain Rod Brackets	B0BVQWTT71	Boxdljh	1	$15.99	$15.99
Right Angle Brackets	JM-40-8P	Jetmore	2	$7.99	$15.98
M4 Screw Assortment	B0BZ6XG8PT	Kadrick	1	$9.99	$9.99
Screws for Lazy Susan	DBT-527	SG TZH	1	$10.88	$10.88
Satin Spray Paint 2X	249070	Rust-Oleum	1	$12.38	$12.38
Shellac Primer	408	Rust-Oleum	1	$11.27	$11.27
Wood Filler	112124	The Gorilla Glue Company	1	$8.68	$8.68
Slip Ring 12 Wire	ZSR022-12D	Taida	1	$17.68	$17.68
Washers for Lazy Susan	DGOL-AT170-SZ08	DGOL	1	$8.55	$8.55
Webcam	TW-05	Tewiky	1	$18.00	$18.00
24V 6A Power Adapter	B0CMZQ3B7Q	HUI	1	$14.99	$14.99
Nema 34 Stepper with Driver	B0CS2RSY59	STEPPERONLINE	1	$75.00	$75.00
Nema 34 Flange Coupling	B0C16XR5QZ	daier	1	$9.99	$9.99
				Grand Total	$563.73

Table 4: Unused Materials

Description	Model #	Manufacturer	Quantity	Unit Cost	Total
Nema 17 Stepper Motor	17HS19-2004S	OSM Technology Co.,Ltd.	2	$13.99	$27.98
3M Arduino UNO USB Data Sync Cable	B08RCJXY1Z	Dafalip	1	$7.99	$7.99
Nema 17 Coupling	B07L1FMBBC	Hamineler	1	$9.49	$9.49
L298N Motor DC Dual H-Bridge Motor Driver	3-01-0032-4PCS	HiLetgo	1	$11.49	$11.49
Stepper Motor Driver Carrier	A4988	Pololu Corporation	2	$13.95	$27.90
				Grand Total	$70.90

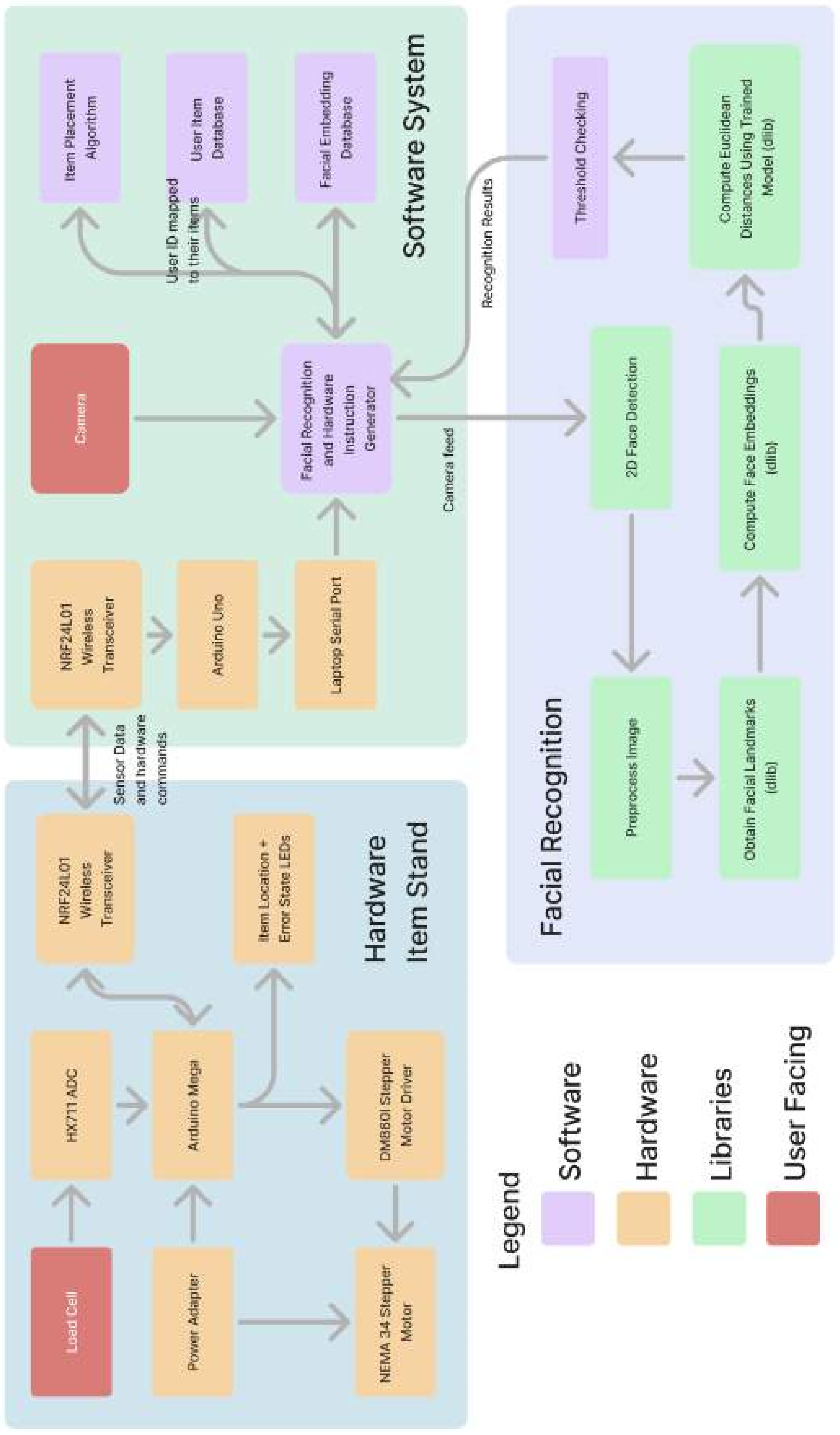

Software System
Item Placement Algorithm
User Item Database
Facial Embedding Database
User ID mapped to their items
Camera
Facial Recognition and Hardware Instruction Generator
NRF24L01 Wireless Transceiver
Arduino Uno
Laptop Serial Port
Camera feed
Recognition Results
Facial Recognition
Threshold Checking
Compute Euclidean Distances Using Trained Model (dlib)
2D Face Detection
Compute Face Embeddings (dlib)
Preprocess Image
Obtain Facial Landmarks (dlib)
Sensor Data and hardware commands
Hardware Item Stand
NRF24L01 Wireless Transceiver
Item Location + Error State LEDs
HX711 ADC
Arduino Mega
DM860I Stepper Motor Driver
Load Cell
Power Adapter
NEMA 34 Stepper Motor
Legend
Software
Hardware
Libraries
User Facing

Figure 13: System Block Diagram

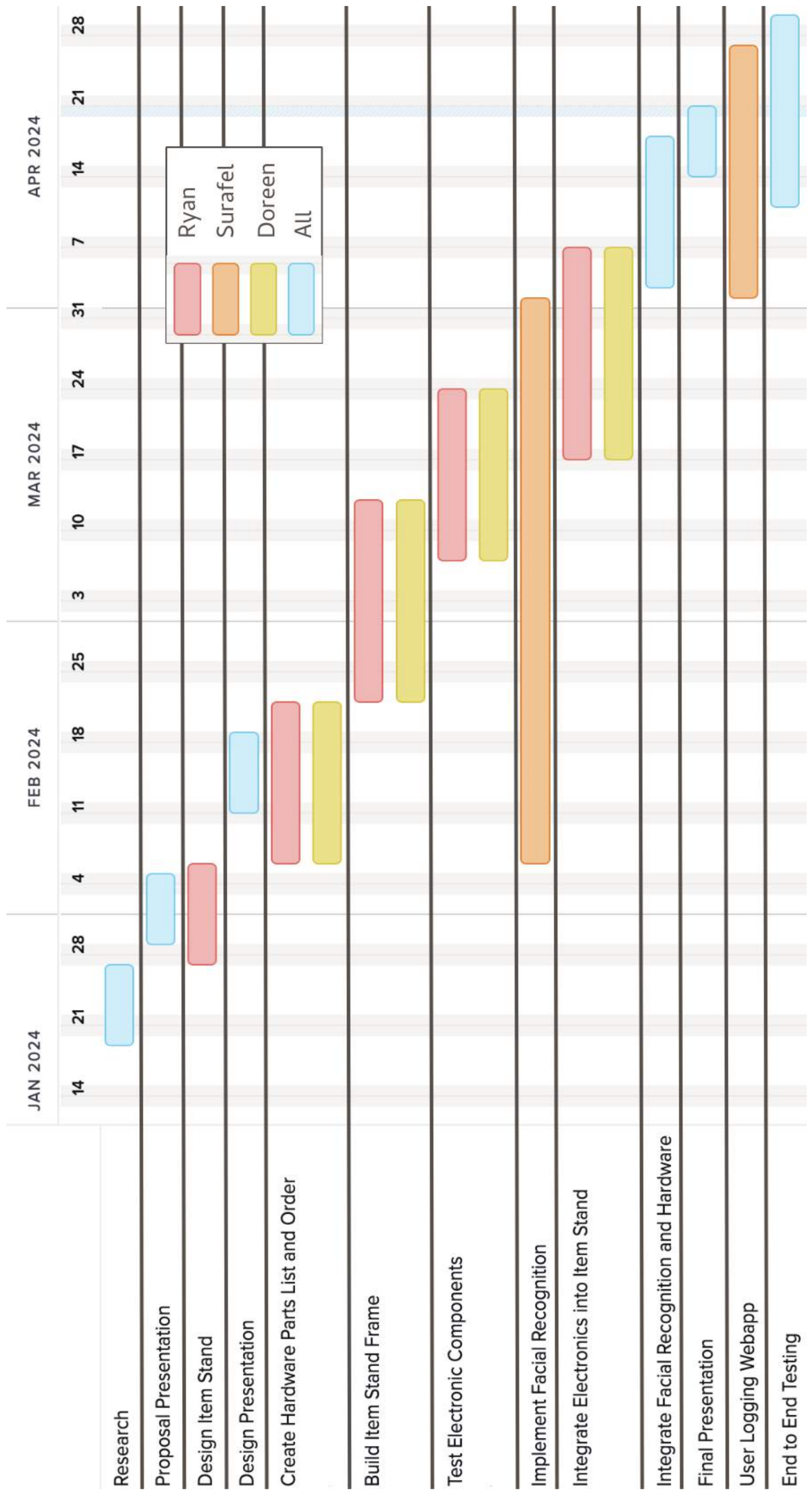


Figure 14: Gantt Chart