My-Flection

Members: Ramzi Hamdalla, Wonho Kang, Yun Lee and Jeremy Ryu

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract* — My-Flection is a smart mirror aiming to help students increase their productivity by helping them start their morning on the right foot through outfit recommendations. We aim to provide a smooth and efficient experience for the user by analyzing their torso and recommending an appropriate outfit based on what they are wearing. By analyzing the color of the clothing, the current weather and availability of clothing in the user's wardrobe, the user can interact with our smart mirror using an app to receive an appropriate outfit recommendation smoothly and quickly.**

***Index Terms*—** Jeston Xavier NX, OpenPose, Arducam, ColorThief, OpenCV

I. INTRODUCTION

College students struggle with many challenges everyday. From the numerous assignments and job search struggles to meeting with friends and going to parties, there are many tasks that keep college students constantly busy. Knowing this we want to help students be more productive by helping them get prepared for their day in the morning. The morning is one of the busiest times of the day and there often are times when people are rushing to leave home to get to class or a meeting. We want to address this challenge by helping students choose their outfits in the morning. People who especially struggle with choosing the "right outfit" for themselves will benefit most from this system as it allows them to smoothly get ready in the morning and help make decisions.

We plan to achieve this by creating a smart mirror that analyzes what the user is wearing on top and recommends appropriate pieces of clothing that go with the top. The user stands in front of the mirror wearing a certain color top and the smart mirror will analyze the torso and detect the top color. Based on the weather, available clothing in the user's wardrobe, and a standard outfit color palette, the smart mirror will recommend an outfit that fits their current clothing.

We accomplish this by implementing computer vision technology to do the analysis and using a user-driven database to provide a more personalized user experience. These two major pieces of technologies will allow the user to comfortably and quickly choose an outfit to their liking.

An example of technology similar to this would be the LG ThinQ Fit mirror, which fits clothing to you by analyzing the body dimensions. Although this technology seemed innovative and was presented at CES 2020, it does not necessarily help with the productivity of the user and moreover makes them feel uncomfortable by measuring body part dimensions. What we aim to do with our smart mirror is the opposite of this as we want to allow the experience to be user-centered so that they feel the most comfortable and efficient when using our product.

II. USE-CASE REQUIREMENTS

Our use-case requirements are driven by providing our users with the most smooth and comfortable experience. We plan to achieve this by focusing on areas the user would be affected the most when interacting with our device. To prioritize a smooth and enjoyable experience, our requirements focus on convenience, time efficiency, and accuracy of the device. We also identified potential areas where our user's experience may suffer in order to decide which metrics are needed to guarantee an enjoyable experience. As a result, our use case requirements are mainly divided into four primary areas which include app response time, outfit recommendation time, detection accuracies, and storage capacity. Each area is essential in creating a smooth experience for the user.

	Components	Requirements	Metrics
Speed	Database	Database latency	< 0.2s
		Database input	< 1s
	Application	On/Off response	< 1s
		LED lighting control	< 1s
		Outfit recommendation	< 5s
Accuracy	Software	Torso detection	> 99%
		Color detection	> 95%
Capacity	Database	Wardrobe storage	> 2GB

Table 1. Use-case requirements

Table 1 is a chart of use-case requirements. In terms of app response time, this includes turning on and off the mirror, how long it takes to input your clothes into the database, and the time to control the LED lights. We believe it is important to create an application that will promptly respond to the user so that they can have a comfortable experience. The quantitative requirements set for this area are database latency under 200ms, On/Off response time of mirror under 1s, LED lighting control under 1s, and database input time under 1s. These are all reasonable delays that occur when using the mirror as delay times on most applications strictly fall under a second not to lose the user's attention and patience.

The second area we focus on is outfit recommendation time. We want the entire process of the outfit recommendation to take less than 5 seconds. This metric was chosen as we thought 5 seconds would be the upper limit for maintaining someone's attention as they are waiting.

Thirdly, in terms of detection accuracy, we wanted to set

specific requirements for torso detection and color detection. First, we want torso detection accuracy to be above 99%. The reason behind this is that our entire project is centered around the device being able to detect the user's torso and analyze the piece of clothing. For color detection, we aim for a success rate of above 95% as we understand that there may be some variables in lighting that can change color shades and present issues with color detection.

Finally, in the area of storage capacity, we want to set a lower limit of at least 2GB of storage as this should be large enough to hold one user's entire wardrobe of clothes.

III. Architecture and/or Principle of Operation

A. Physical System

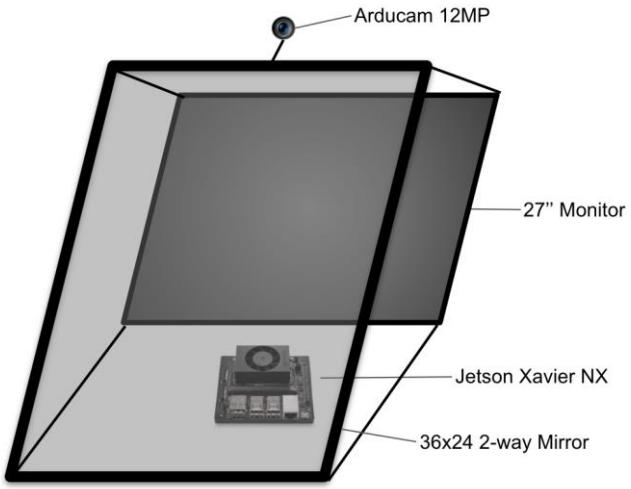

Fig. 1. System drawing of an overall physical system

The diagram of the physical system sketch which entails all hardware components of the project is illustrated in Figure 1. As demonstrated in the diagram, an Arducam will be placed on top of the two-way mirror along with a monitor, and a Jetson Xavier NX placed behind the mirror. Arducam will take a snapshot of a user standing in front of the mirror. The image will be sent to Jetson which is in charge of necessary computations including analyzing the user's torso and top color with relevant software. The analysis results will be fed to the outfit recommendation algorithm which will output recommendations based on a wardrobe database stored in an SD card and real-time weather information collected from the internet. As a final step, the outfit recommendation will be displayed through the monitor that the user can view.

THe Arducam 12MP and Jetson Xavier NX are off the shell from the ECE inventory, and a 27-inch monitor and 36x24 2-way mirror have been purchased. The size of the mirror reflects the expected size of the final product. The monitor has been chosen to be smaller than the mirror but big enough for the user to easily view the recommendation display.

B. Block Diagram - System Architecture

The block diagram in Figure 2 demonstrates the overall system architecture. The project is done mainly in software, and system architecture can be divided into three parts: user interface, computer board, and internet. The user interface is composed of a camera, application, and monitor, which will communicate with the computer board through physical wire connection, wifi connection, and physical wire connection, respectively. While the camera only takes inputs from the user, the monitor outputs display results, and the application will both input and output basic signals and wardrobe information.

The application is used as a general controller of the entire system, which has the most interactions with the user. The main features of the application include 1) turning the mirror on and off, 2) adding and deleting items of the user wardrobe, 3) weighing items based on item availability, and 4) adjusting lightings suitable for torso detection. When adding items to the user wardrobe, users will take a picture of the item and fill in types of clothing(e.g. shorts, jeans, t-shirt, long sleeve) through the itemization form. Then, the app will transmit the picture and form to Jetson, which will detect the item color and put the item data into the user wardrobe database in CV format. Users can delete items from their wardrobe when the items are unavailable and they can also weigh items differently based on their preferences using the application as well.

With the wardrobe database created, users can get outfit recommendations by turning on the mirror and standing in front of it for the mirror to recognize what the user is currently wearing as their top. Based on the user inputs, the camera image, and the wardrobe, Jetson 1) organizes the user wardrobe database, 2) analyzes torso, 3) analyzes top color, 4) collects real-time weather information, 5) recommends outfits, and 6) creates a monitor display for the recommendation.

As mentioned above, the user wardrobe database will be in CV format containing item color, type, availability, and preference weight. Torso and color analysis will be done through open-source APIs OpenPose and Colorthief, respectively. The real-time weather information will be collected through the internet from weatherapi.com which provides weather forecasts based on geolocation information. With the user top color data, weather information, and the user wardrobe, the outfit recommendation algorithm will generate a set of applicable outfits reflecting the weather and user preferences. The basics of the algorithm will be color coordinating charts that provide aesthetic color pairs for outfits. The charts will be organized in CV format grouping colors that match with one another and stored in an SD card along with the user wardrobe.

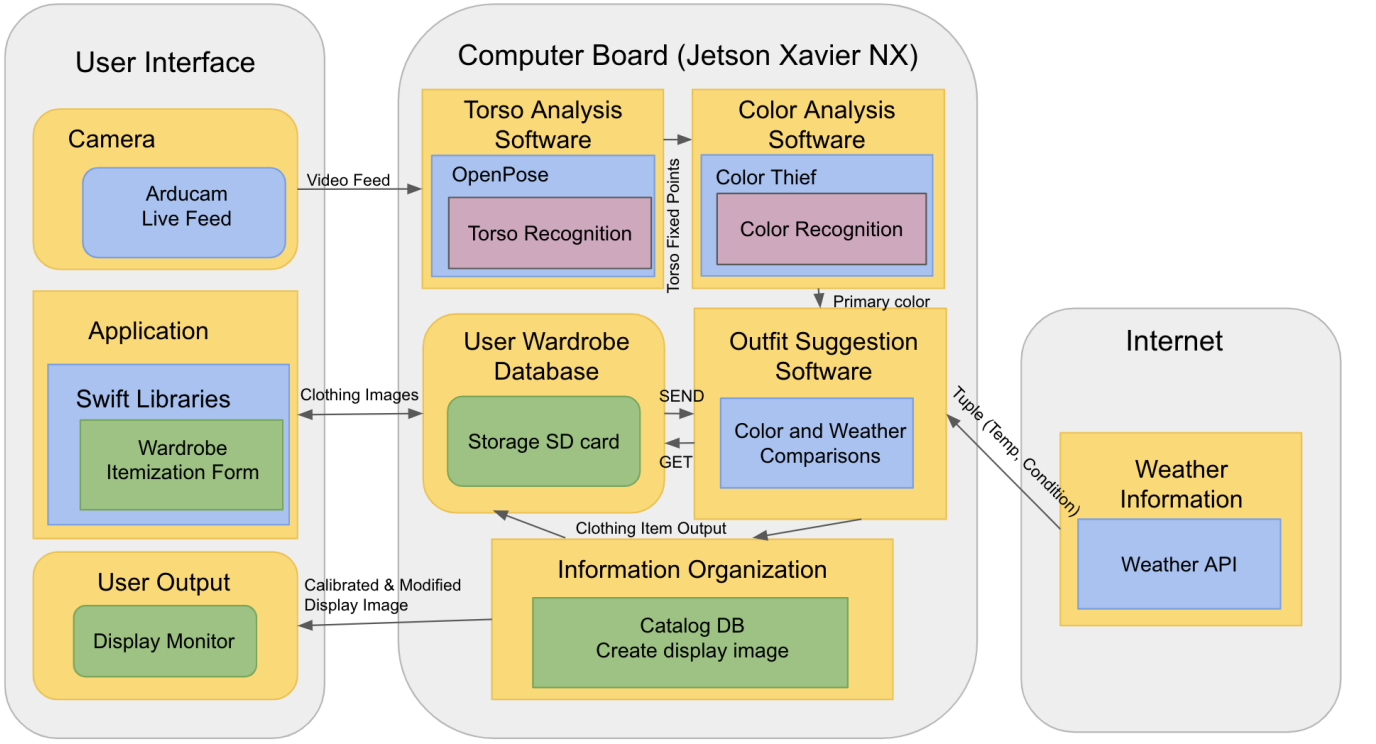

Fig. 2. Overall system block diagram

A set of outfit recommendations will then be displayed to the monitor. During this process, Jetson will pull out corresponding clothing images and information stored in the database. These will be calibrated and modified suitable for the display. The user can rate the recommendations and mark whether they accept or reject the recommendations which response will be used to update weightings of clothing for the upcoming uses.

IV. Design Requirements

The use case requirements, as demonstrated in Table 1, can be categorized into three main components: speed, accuracy, and storage capacity.

A. Speed

Speed will all be tested as a timestamp upon a start of an initial process subtracted to a timestamp upon completion of the final process

- Database latency: The metrics will be measured for the retrieval of data from the database in a 16GB SD card with data stored in CV format for outfit recommendation and monitor display image calibration. The following equation will be used:

  (timestamp upon a successful data retrieval response) - (timestamp time upon data retrieval request)

- Database input: The metrics will be measured for modifying the CV format wardrobe database stored in a 16GB SD card. Database input speed will be measured for adding and deleting a piece of clothing from the database and modifying preference weights for an item. The following equation will be used:

  (timestamp upon a successful modification of data reflected in the database) - (timestamp for database input request)

- Application on/off response: The metrics will be measured for the time taken to turn on/off the mirror using the application user interface. Turning on includes the time from pressing an “On” icon to setting environments and displaying an initial mirror interface through the monitor. Turning off includes the time from pressing an “Off” icon to powering off the mirror. The equations are as follows:

  Turning on the mirror:
  (timestamp upon an initial mirror interface display) - (timestamp for pressing an “on” icon)

  Turning off the mirror:
  (timestamp upon mirror power off) - (timestamp for pressing an “off” icon)

- LED lighting control: The metrics will be measured for adjusting lightings using the application interface for detecting user torso and top color. The equation is as follow:

(time that LED lighting changes are reflected) - (timestamp of pressing an "adjust lightings" icon from the application)

- Outfit recommendation: The metrics will be measured for the entire period of the outfit recommendations from starting torso detection to outputting recommendation results to the monitor. The equation is as follow:

(timestamp of recommendation results display) - (timestamp for torso detection start)

- *B. Accuracy*
- Torso detection: The metrics will measure the capability of identifying the user torso standing in front of the mirror in a stable position within a distance range of 1m~2m, assuming the proper lighting. It will use mAP(mean average precision in percentage) for object detection. The equation is as follow:

$$mAP(\%) = \frac{True\ Positive}{True\ Positive + False\ Positive} \cdot 100$$

- Color detection: The metrics will measure the capability of identifying RGB values of a piece of outfit for user top detection and adding an item to the wardrobe. Assuming the use of the same LED lighting environment, the RGB value of a dominant color of a piece of clothing will be measured. The equation is as follow:

$$Color\ accuracy\ for\ a\ value = \left(1 - \frac{actual - detected}{actual}\right)$$

$$Color\ detection\ accuracy(\%) = \frac{1}{3} \cdot (R\ accuracy + G\ accuracy + B\ accuracy) \cdot 100$$

- *C. Capacity*
- Wardrobe storage: The metrics will measure a storage capacity solely dedicated for the wardrobe out of a 16GB SD card storage. A photo taken from an iPhone is on average 3.5MB; by using this as a base for our estimate, the wardrobe should be able to store approximately 570 items.

V. Design Trade Studies

Design tradeoffs were considered for most of our hardware components of our smart mirror and the user interface. Most of our choices, if not all, were driven by our use-case and the use-case requirements.

A. Computation Hardware - Jetson Xavier NX

The most important aspect of our project is to provide a fulfilling and smooth user experience with our smart mirror. In order to achieve this, it is crucial that the single-board computer we use is able to process live video feeds efficiently. The two options we were considering were the Jetson series and the Raspberry Pi. The Jetson series, specifically the Jetson Nano and Jetson Xavier NX, were designed for AI/ML applications because of their incredible GPU capabilities that significantly outperform the Raspberry Pi. However, the price tag attached to the Jetsons were on the pricier end in comparison to the Raspberry Pi. On the other hand, the Raspberry Pi has been around for a longer time than the Jetson Nano and Xavier which could potentially benefit us more in terms of better documentation and demo projects. After weighing the tradeoffs between using the Jetson Nano/Xavier vs. Raspberry Pi, it was apparent that using the Jetson Xavier would be the best design choice. One of the stretch goals of our project is to apply AI/ML models to our smart mirror which would allow it to analyze, recognize, and learn user's outfits without the need to create key tags in the wardrobe database. Also, our torso detection software, OpenPose, requires NVIDIA GPU compatibility for a smooth experience. Therefore, the NVIDIA Jetson Xavier NX would be the best choice for achieving all our use-case and design requirements.

B. Communication - Wifi

Though the Jetson Xavier NX may boast its incredible computation capabilities, our project will be ineffective if the method of communication to and from the single-board computer is buggy. There were three main options we took under consideration when deciding upon a method of communication between the components of our project and the Jetson Xavier NX: bluetooth, wifi, and wires. The option of using wires was crossed out early because although it is the easiest to set up, our project places a huge emphasis on user experience, so the potential trouble of dealing with messy wires behind or around the smart mirror is highly undesirable. As shown in the block diagram in Section III. Architecture, there are several components of our project that require communication with the Jetson Xavier. Most importantly, the National Weather Service API exists on the Internet which can only be accessed via a wifi compatible chip on the computer board. Another thing to note is that bluetooth is only effective at close distances. If a user were to have a separate wardrobe in another room and the mirror in his/her bedroom, then it would be an annoyance to bring all the clothes to the bedroom and then take photos of them to upload to the database. Therefore, using a wifi chip would significantly decrease this trouble as wifi allows for communication over large distances. Finally, we realized that setting up bluetooth or wifi communication would not be difficult for either as the Jetson Xavier NX we received from the 18-500 inventory already had a WIFI/BT card that came with it.

C. User Interaction - Mobile App

Three alternatives were discussed when working on our design choices for how the user would interact with the smart mirror. This interaction includes turning on and off the mirror, controlling the LED lighting around the mirror, and inputting

the user's wardrobe into the database. Initially, we wanted our smart mirror to be touch-compatible. This seemed like the most intuitive choice as "smart" in "smart-mirror" indicates the presence of a touch screen. However, this idea was quickly scrapped when we realized that buying an IR sensor for the entire smart mirror display was too expensive and unnecessary as the user wouldn't need to touch all parts of the smart mirror and swiping motions on the screen were not a part of our user interaction. Understanding that the smart mirror would only need touch compatibility on specific points on the display, we then considered adding capacitive touch sensors to these specific parts of the smart mirror display. However, this would add more wires and circuitry to the smart mirror that may end up becoming a mess. Furthermore, having the user constantly walk up to the mirror, touch the mirror screen, and then walk back into the camera's field of view would be far too inconvenient and defeats the main goal of our smart mirror's smooth user experience. Therefore, after weighing out all these tradeoffs, it made the most sense to create a very simple mobile app that would allow the user to perform all interactions with the mirror in one collective place. This app would allow the user to turn on and off the mirror as well as control the LED lighting at any distance which prevents the issue of having to walk back and forth to and away from the mirror. Better yet, inputting the user's wardrobe through an app is the simplest method. The user takes a photo on his/her phone, uploads the photos to the app, and the app sends this data to the backend database via API calls.

D. Database - Wardrobe Itemization Form

Two possibilities were brought up when our team was discussing how exactly the computer would obtain the correct outfit from the wardrobe database and what information the database would contain. Our team's initial approach was to implement or use existing image analysis algorithms/models that would analyze the pictures of the clothing that the user inputted and match it with the outfit recommendation output. One potential problem with this design approach is that it requires even more computation which would increase the time it takes for the user to receive an outfit recommendation output on the smart mirror display. We found that this increase in output time would be more undesirable in comparison to the user spending a little more time when inputting his/her clothes into the database. With this in mind, our team decided to place our efforts on creating a wardrobe itemization form that the user would essentially fill out. This form would prompt the user to provide information about the type of the clothing, the color, a photo of the piece of clothing, and a weight from 1-10 indicating how much the user likes the piece of clothing. Hence, when the outfit recommendation says it wants, for example, a black shirt, then it would query the database by applying the filters, for example, color = black and type = shirt. This entirely gets rid of the hassle when it comes to heavy computations and the error-prone nature of solely analyzing the picture of the clothing.

E. OpenCV - C++

OpenCV is used to analyze the snapshot of the user's torso in real-time. It is already well-known that OpenCV is compatible with the Jetson Xavier NX, so the only choice we had to make was what language we would use to implement the image processing and analysis. Our team was more familiar with Python, so we initially decided to stick to a language that we were more comfortable and familiar with. However, after receiving feedback from our proposal presentation, we realized that there could be a severe bottleneck that would obstruct our use-case requirements if we used an interpreted language (Python) over a compiled language like C++. C++ is well-known to be very fast and is used as the primary language in most trading firms due to its speed. Therefore, we decided to use a C++ implementation over a Python one because it is much more important to reduce the risk of long computation times as this would worsen the user's experience.

F. Camera - Arducam

The option of choosing a camera that would provide a live video feed for torso analysis was dependent on whether we valued exceptional quality or ease of setup. The former would have been a professional photography camera. The reason we chose to go with the Arducam is because it was already in the 18-500 inventory and its quality was good enough for image processing and analysis. Also, as mentioned earlier, it is relatively easier to set up with the Jetson Xavier NX.

G. Torso Detection - OpenPose

No trade offs were considered for the torso detection software we would use. The reason for this is because previous projects in past semesters had used OpenPose for torso detection and it proved to be very successful with high detection accuracy. Moreover, it accomplishes our design requirements of being able to recognize and analyze fixed points on a user's torso which allows for smoother and well-rounded testing as well.

H. Color Detection - ColorThief

No trade offs were considered for color detection. We stuck with our original design of using an open-source API called ColorThief. This API is very useful as it outputs the dominant color palette from an image. The API is simple to use and is trustworthy as the author of the API is currently an engineer lead at Square. As long as the image we provide to the database is consistent, then the output from ColorThief will also be consistent. Obviously, this will be thoroughly tested.

VI. System Implementation

- Software Specifications

A. *OpenPose*

OpenPose is a real-time multi-person system to jointly detect human body, hand, facial, and foot keypoints (in total 135 keypoints) on single images. The main feature of OpenPose our project uses is its ability to take in a webcam as an input and a basic image with keypoint displays as a PNG output. This can be achieved either through the command-line demo provided in the repository or using the API. We will connect the software to our hardware (Jetson Xavier NX) and use Windows for our OS because there seems to be a couple issues with MacOS and Linux.

B. *ColorThief*

ColorThief works in Node and can easily be imported and used. The API will be used by calling getColor(image) which returns the dominant color [int, int, int] from an image. The color is returned as an array of three integers representing the red, green, and blue values. The image that is passed in as a parameter can be a PNG which works with the PNG file that is returned from OpenPose. Figure 3. shows an example of an input and output response on the web version of ColorThief for the purpose of a more visual understanding.

Fig. 3. ColorThief dominant color output

C. *WeatherAPI.com*

We plan to use the current live weather condition as well as the temperature to differentiate between an outfit recommendation of short pants vs. long pants and later on shirt vs. jacket. We want to receive live, correct data on what the current weather conditions are like in the user's current location. WeatherAPI.com is a powerful fully managed free weather and geolocation API provider that provides extensive APIs that range from the real time and weather forecast, historical weather, Air Quality Data, IP lookup, and astronomy through to sports, time zone, and geolocation. We have created a unique API key that will be used to make GET requests. An example of a response body is shown in Figure 4. We will extract the temperature, precipitation, and condition fields in the response body.


```
"current": {
    "last_updated_epoch": 1646011800,
    "last_updated": "2022-02-27 20:30",
    "temp_c": 2.2,
    "temp_f": 36.0,
    "is_day": 0,
    "condition": {
        "text": "Partly cloudy",
        "icon": "//cdn.weatherapi.com/weather/64x64/night/116.png",
        "code": 1003
    },
    "wind_mph": 11.9,
    "wind_kph": 19.1,
    "wind_degree": 310,
    "wind_dir": "NW",
    "pressure_mb": 1019.0,
    "pressure_in": 30.08,
    "precip_mm": 0.0,
    "precip_in": 0.0,
    "humidity": 54,
    "cloud": 25,
    "feelslike_c": -2.3,
    "feelslike_f": 27.8,
```


Fig. 4. Part of response body with HTTP request http://api.weatherapi.com/v1/current.json?key=68227f9c42464760a6813145222802&q=15213&aqi=yes

D. *Mobile Application*

We will use React Native to create our mobile application that will allow the user to interact with the smart mirror. The mobile application will be connected to a Node backend which will be used to make get and post requests via an API to retrieve and update the database which will most likely be a simple CSV file. A simple block diagram is shown in Figure 5.

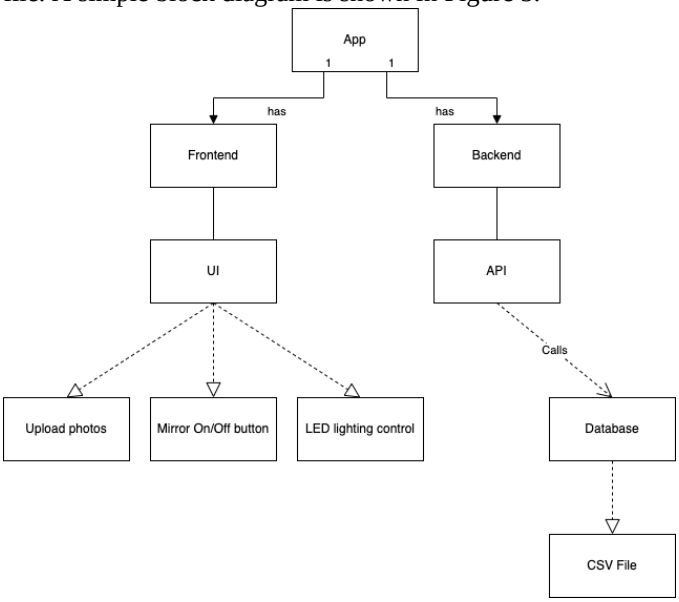

Fig. 5. Simple UML diagram that shows the overall structure of the mobile application.

- Hardware Specifications

E. *NVIDIA Jetson Xavier NX*

As mentioned in our design requirements and design tradeoffs, we are using the NVIDIA Jetson Xavier NX as our single board computer that will handle all the processing power

necessary for live video capture from our Arducam which will then be fed into OpenPose as well as computation of outfit recommendations. The 16GB NVIDIA Jetson Xavier NX module we have has a 384-core NVIDIA Volta™ GPU with 48 Tensor Cores, 6-core NVIDIA Carmel ARM®v8.2 64-bit CPU, 6 GB 128-bit LPDDR4x, and a WIFI/Bluetooth chip.

F. Arducam B0250

We are using the Arcudam B0250 for the smart mirror camera that will take in a live video feed of the user standing in front of the mirror. This video feed will be used for analysis with OpenPose. The frame rate of the camera is 1920×1080 @ 60fps 4032×3040 @ 30fps. Resolution is 4056(H) x 3040(V) 12.3MP. A field of view (FOV) of 65° (H). The camera comes with a customized metal enclosure, a tripod stand, and a set extension adapter for the Jetson Xaver NX.

G. Two-way mirror

We are using a two-way mirror for our smart mirror because it allows for a monitor to be placed behind the mirror and show a display to the user at the same time the user is checking himself/herself in the mirror. The mirror is 36 x 24 inches.

H. Monitor Display

Behind the mirror is a monitor display that will show the outfit recommendation photo to the user. This monitor is an HD monitor with dimensions of 8.33 x 20.05 x 24.41 inches. It weighs 11.49 pounds and has a resolution of 1920 x 1080. The dimensions of the monitor were specifically chosen so that it would have the same width as the mirror. This allows for the monitor to cover more than the top half of the mirror.

VII. Test, Verification and Validation

We have the following metrics that we wish to test and track as design specifications for our project:

- Torso Detection
- Color Detection
- Outfit Recommendation Time
- Mirror On/Off Response Time
- LED lighting On/Off Time
- Database Input Time

Our design specifications were picked with the users comfort and experience in mind. Ideally we want to improve all of these metrics to a point where we believe the user will be reasonably satisfied.

We have similar reasoning for testing our detection design specifications, as well as similar tests. Additionally, we have similar reasoning for testing various time metrics, and similar tests for how to track and record the time it takes for those various design specifications, resulting in repetitively similar tests. Most of our detection system tests include inputting varied inputs into our algorithms, and manually sorting the successful tests from the failed tests. Our timed tests will mostly rely on using computer code to get accurate and precise results that we can use for definitive results.

A. Tests for Torso Detection

We will be testing our torso detection by applying multiple images into our torso analyzing algorithm, and manually examining the success rate of our algorithm in spotting the torso correctly. The torsos provided to the algorithm would vary in positioning and posture in order to not only thoroughly test our algorithm, but to also find the best position and distance for our user. Our goal is to reach 99% accuracy, as we believe that waiting for the mirror to provide an option, only for it to be a failure due to poor torso detection would be a very negative experience for the user.

B. Tests for Color Detection

We will be testing our color detection by applying multiple images into our color analyzing algorithm, and manually examining the success rate of our algorithm in choosing the color correctly. We will be testing many colors within a given range in different lighting conditions to make sure that our algorithm functions under most circumstances, even with changing variables. Our goal is to reach 99% accuracy, as we believe that waiting for the mirror to provide an option, only for it to be a failure due to poor color detection would be a very negative experience for the user.

C. Tests for Outfit Recommendation Time

We will be testing our outfit recommendation time by setting up code that will track how long our algorithms take to run from start to finish and provide a recommendation over multiple trials. We are aiming at performing a total of 10 trials, and achieving an outfit recommendation time of under 5 seconds. We believe waiting excessively long for an outfit recommendation would negatively impact the user experience, and that 5 seconds is the maximum time we can take before the user may begin to get uncomfortable.

D. Tests for Mirror On/Off Response Time

We will be testing our mirror on/off response time by setting up code that will track how long our smart mirror takes to boot up from start to finish over multiple trials. We are aiming at performing a total of 10 trials, and achieving a mirror on/off response time of under 1 second. We believe that longer than this would result in a noticeable latency that may negatively impact the user experience.

E. Tests for LED Lighting On/Off Time

We will be testing our LED lighting on/off response time by setting up code that will track how long our smart mirror's LEDs take to boot up from start to finish over multiple trials.

We are aiming at performing a total of 10 trials, and achieving an LED lighting on/off response time of under 1 second. We believe that longer than this would result in a noticeable latency that may negatively impact the user experience.

F. Tests for Database Input Time

We will be testing our database input time by setting up code that will track how long our application takes to analyze and store the user's wardrobe inside the smart mirror over multiple trials. We are aiming at performing a total of 10 trials, and achieving a database input time of under 1 second. We believe that longer than this would result in a noticeable latency that may negatively impact the user experience.

VIII. Project Management

A. Schedule

As shown in Figure 6 in the appendix, we have provided a fully detailed planned schedule split by days and weeks, with the team responsibilities and our various plans organized at the bottom of the figure.

B. Team Member Responsibilities

As a team, we have divided responsibilities depending on the specialties of each member. Jeremey is specialized in software, and as such his main responsibilities include configuring our outfit analysis algorithms, our color analysis algorithms, and our torso analysis algorithms.

Ramzi is specialized more in hardware, as such his main responsibilities will consist of configuring the hardware implementation of the smart mirror, and the user interface.

Wonho is specialized in software with additional experience in hardware, and his main responsibilities will consist of working on the user interface and the hardware implementation, as well as the outfit analysis, and combining these differing aspects of the project.

Yun is also specialized in software, and as such her main responsibilities additionally include configuring our outfit analysis algorithms, our color analysis algorithms, and our torso analysis algorithms.

C. Bill of Materials and Budget

Our Bill of Materials is stated below in the appendix in Table 2.

D. Risk Mitigation Plans

Our biggest concern is the risks surrounding our torso detection algorithm. It is a field that we have little experience in, and where we are largely relying on readily available libraries in order to be able to put together a method of detecting the user's torso.

As a method of mitigating this risk, we are willing to put together guidelines that will inform the user on what conditions the torso detection will work best, i.e. providing the user with a specific distance which they should stand at, and what angle is also best. We have also considered potentially adding an outline that will provide the user with the proper positioning for the torso detection system to work best.

We are also concerned about the risk of inconsistent lighting in the user's environment. This is an aspect which we have little control over, so we came up with ways to influence this factor. Firstly, we have worked LED lighting into our design which the user can turn on and off to provide themselves with better lighting. If necessary, we can also recommend the user to a well lit environment as part of our guidelines, and analyze their environment to see if the lighting is appropriate.

IX. Related Work

Our project was initially inspired by DIY smart mirrors that are made using Raspberry Pis. However, they only displayed simple information such as weather and your daily schedule and did not have much new content showing on the screen. Also, these types of smart mirrors do not do that much computing within the mirror itself or actively interact with the user. There are other smart mirrors such as the LG ThinQ smart mirror that tailors the outfit to your body as well as the ECE capstone project from spring 2021 that presents a smart mirror that warps the clothing to fit the user. However, there is no smart mirror on the market that necessarily gives outfit recommendations based on what you are currently wearing.

The biggest difference between common DIY smart mirrors and our project is the interaction between the user and mirror as well as the computing process that goes in the mirror. Our goal is to allow the user to easily and efficiently access this process information, in the form of outfit recommendations, and provide the smoothest experience for the user.

X. Summary

Our smart mirror is a device aimed at improving the mornings of college students, who often lack the time or mental power to make careful decisions about their outfit in the early morning. By absorbing the user's wardrobe, and analyzing a little data on what they want to wear for the day, we can provide them with a full recommendation on how to complete their outfit in little to no time at all.

Although it may be slightly pricey, the mirror has a huge impact on how the student starts their morning, and helps them stay fashionable without needing to divert attention away from their studies and how they are spending their time.

We foresee several challenges on the horizon ahead of us. Primarily among them, we will need to make sure that our algorithms don't just work well, but that they work in a timely manner. There are many different parts of our project that rely heavily on code to compute an output based on the given input, and it is a cause of concern that once we combine all these parts, they may result in a large run time for our overall program.

Additionally, our project is heavily constrained by the time constraints placed on each of us individually, and as a group, as we keep up with regular presentations, and external factors.

Because of this, we chose the use-case requirements that we believed were the most relevant to track as important metrics, and set design requirements that we believed to be necessary, yet feasible.

With all of these factors, we hope to be able to provide a viable smart mirror that will display our concept properly, and satisfy its users with its functionality as a quality of life, time saving home improvement for your average college student.

Glossary of Acronyms

LED - Light emitting diode

References

[1] CNET, *CNET: David Priest*, Accessed on Mar 3, 2022, [Online]. Available: https://www.cnet.com/home/smart-home/smart-mirrors-just-make-you-hate-yourself/
[2] https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html
[3] https://lokeshdhakar.com/projects/color-thief/
[4] https://www.weatherapi.com/

APPENDIX

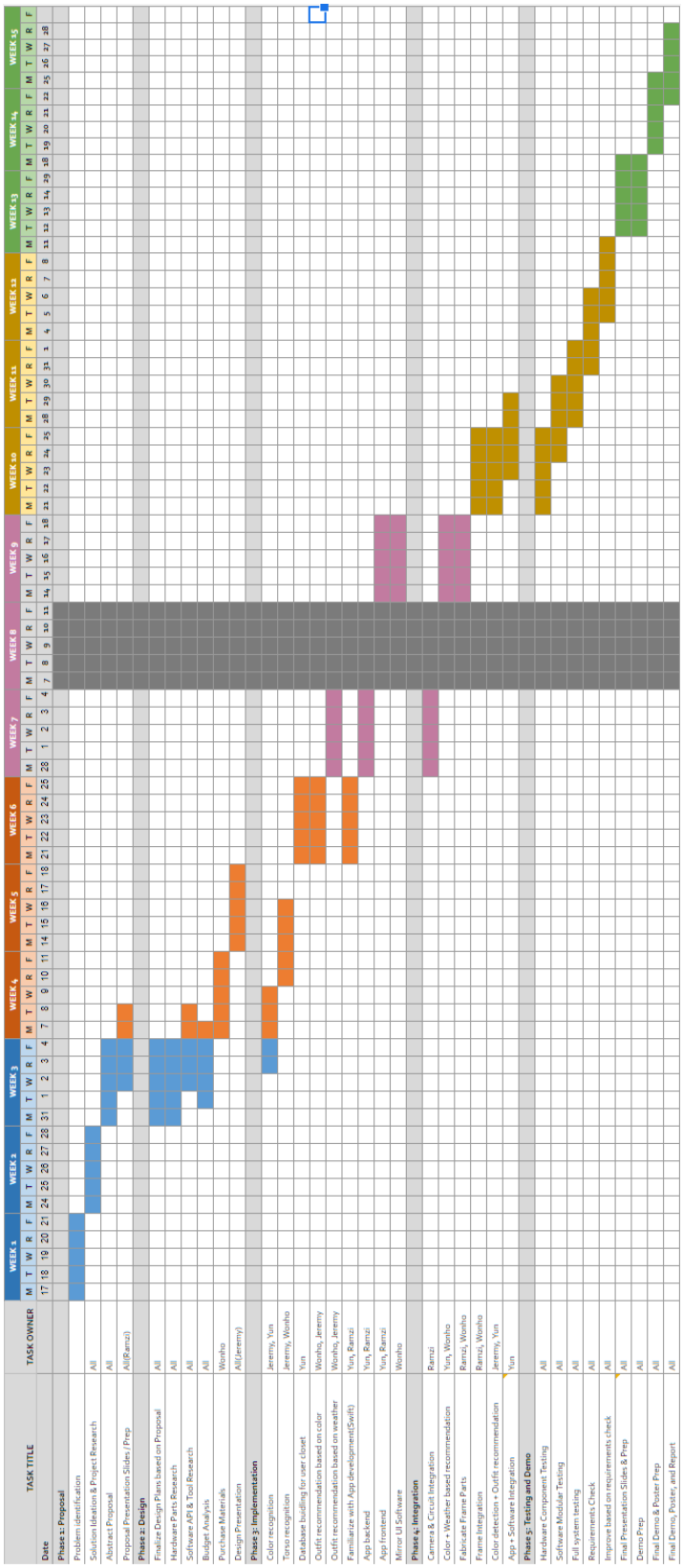

Fig. 6. Schedule with milestones and team responsibilities

Item	Manufacturer	Model Number	Source	Cost
Nvidia Jetson Xavier nx			ECE Inventory	$0
Arducam B0250			ECE Inventory	$0
36x24 Two-Way Mirror	SPEEDYORDERS		Amazon	$192
27-inch HP Monitor	Hewlett Packard	M27ha	Amazon	$174.08
16 GB micro-SD card	Sandisk	SDSQUAR-016G-GN6MA	Amazon	$8.83

Table 2. Bill of Materials and Budget

Budget	$600
Total Costs	$375
Available Budget	$225

Table 3. Costs and Budget Available