Project Experience Report – After Action Review

Team Name

Mirror++

Date of Review

April 8, 2022

When Review Was Completed

After project completion

Participants

Annoor Rahman

Qurrat Ulain

Summary of Project

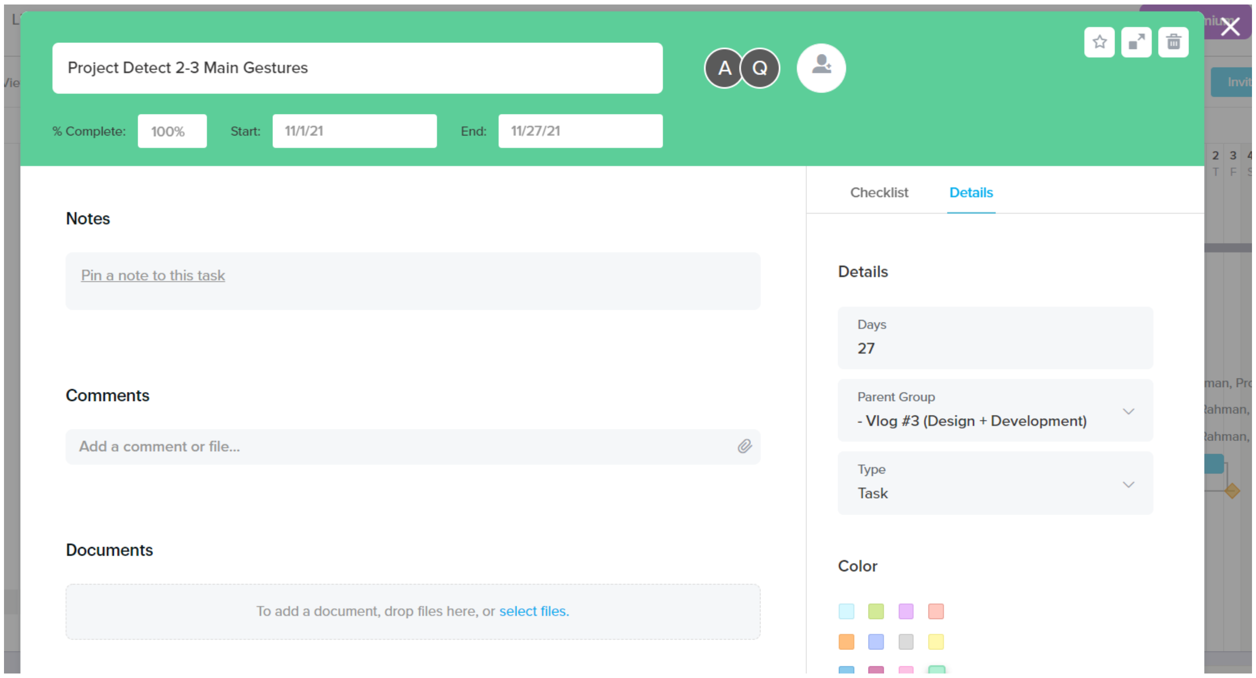
For our final project, we decided to create a smart mirror aimed toward people with different forms and levels of disabilities. It performs operations based on gestures and voice control, depending on user preference. It also extends functions already available on smart devices and makes them more accessible. Mirror++ is also able to automate existing IoT devices like lights and outlets.

What went well and why?

Throughout the process of creating the smart mirror, several steps went well and became important towards the completion of our final product. To begin, the choice of customers for our project went well. We began by creating a product targeted mainly towards people with various hearing and speaking impairments, but with the advice of our mentor, Professor Yow, and an analysis of the market available, we decided to broaden our target customers to also include the rest of the public. This expansion allowed us to introduce voice control into our mirror and working with both gestures and voice has improved the overall functionality of the mirror.

Another decision that went well for us was the choice to switch from using a Microsoft Kinect to a regular web camera. While the decision to use Microsoft Kinect had been a strong one given its tracking ability, we quickly discovered there was not much support available for it online. Given the nature of our project and our lack of experience using Microsoft Kinect, we decided we were better off using a web camera and pairing it with Python's OpenCV and MediaPipe libraries. This allowed us the freedom of exploring camera tracking as it exists in Python without needing to introduce an unfamiliar SDK into the mix.

Our next fruitful decision was to organize our thoughts and timeline using the Gantt Chart. We used a service, TeamGantt (https://www.teamgantt.com/) to create a Gantt chart for our project, which provided an engaging interface that made documenting deadlines and progress much simpler than using another method. A screenshot of the Gantt chart interface is below:


Project Detect 2-3 Main Gestures

% Complete: 100%   Start: 11/1/21   End: 11/27/21

Notes

Pin a note to this task

Comments

Add a comment or file...

Documents

To add a document, drop files here, or select files.

Checklist   Details

Details

Days
27

Parent Group
- Vlog #3 (Design + Development)

Type
Task

Color

Figure 1: Gantt Chart Snippet

By being forced to think of tasks in terms of how long they will take and document how close to completing them we were, we were reminded to stay on track with our goals and loosely follow the Gantt chart to make sure we finish all our tasks on time.

In addition, the division of tasks between Annoor and I worked well. We didn’t know each other prior to the capstone, so going into the project, neither of us was aware of the other’s work ethic. However, with clear communication and honesty, we were able to determine each other’s points of strength and divide the project in that way. Our general structure of dividing the tasks in front-end and back-end tasks also resulted in an efficient work style and we were able to combine our work to produce a functioning final product.

Creating our low-fidelity diagrams was another thing that worked out well for our group. By working together to visualize what we wanted our mirror interface to look like and putting our thoughts on paper, we knew the general design requirements we needed to follow for the final product. Once we had that idea, it was a just a matter of finding the right Kivy methods to produce the results we outlined and making sure they were as visually appealing as we initially thought. A sample of the low fidelity diagrams we began with are below:

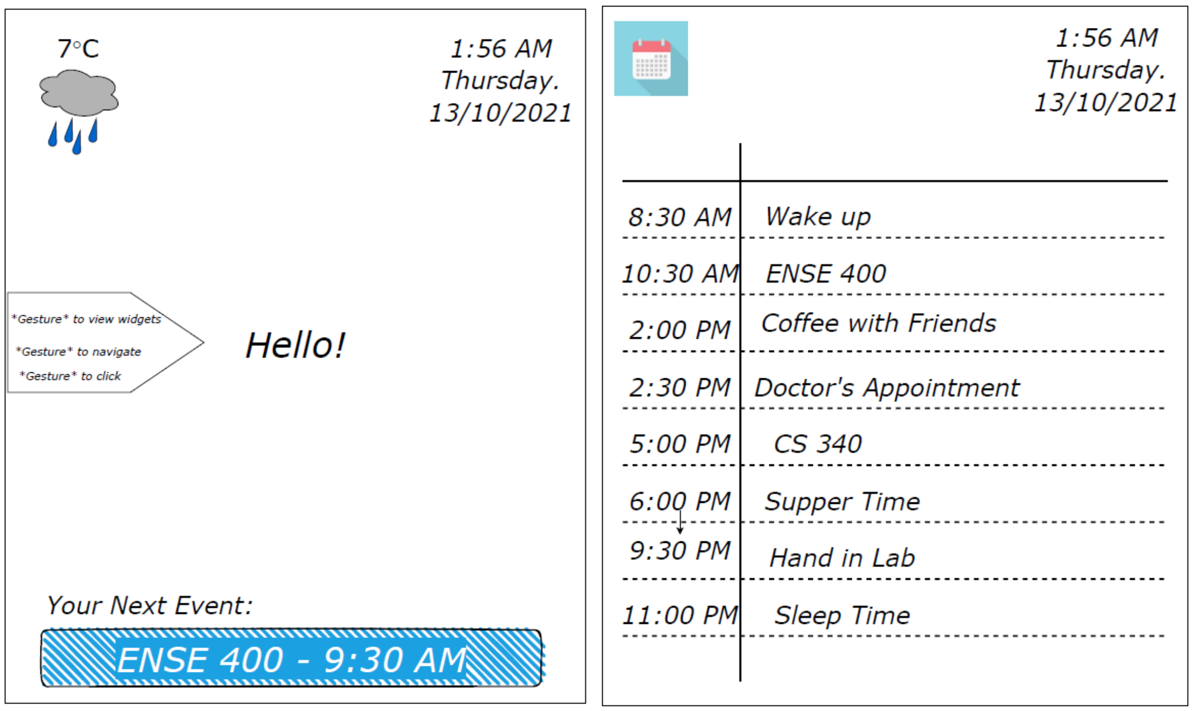


Figure 2: Low-Fidelity Diagram Samples

We are also proud of the gesture tracking that was achieved in the final product. When we began, the gesture tracking was very inconsistent and unreliable, but with continuous modifications and vigorous testing, it is now a reliable and smooth process. The gestures that have been programmed work regardless of the dominant hand that is used, and users can rely on the mirror performing the operation they intend.

Finally, the user testing process worked well to give us lots of different suggestions. Going into the user testing, we had a biased view towards the functionality of our mirror. We knew that the gestures and voice commands worked, and we weren't expecting to get many varied responses to the mirror. However, immediately during set up we realized that the smart mirror text wasn't visible at all in natural light. This flagged us to the importance of user testing and testing in general, so we became a lot more serious in the process. Secondly, we hadn't realized that the gestures were coded to primarily recognize left-handed gestures, as Annoor is left-handed. So, when right-handed people tried to casually use the mirror, it became unpredictable with its outcomes. Lastly, we realized that the microphone we had previously been using was not strong enough to detect audio commands clearly, which flagged us to the need for an external microphone. Through user testing, we found out these key suggestions and were able to improve our final product and get it prepared for project day.

What can be improved and how?

While our project had many strengths, there are also ways in which it can be improved in the future. The most immediate improvement is the use of a stronger processor in the mirror.

Initially, we had decided to use a Raspberry Pi. This would allow the mirror to be a portable machine without external connections as the Raspberry Pi comes embedded with a camera and microphone. We purchased and installed the strongest Raspberry Pi available to use, and at first, the Pi was able to withstand the mirror program and run. However, once we finalized our code and included all the different voice and gesture functionalities, the Raspberry Pi processing speed was too slow to successfully operate the mirror. There was an option to optimize our code to the extent that the Raspberry Pi would be able to handle it, but given the time constraints we had, this option wasn't feasible. As a result, we ended up using a Windows Desktop to run the smart mirror program instead. This worked significantly better as the desktop had higher processing power and was able to withstand the load and operate at higher speeds. We decided to use the desktop for project day just to ensure the functionality of the mirror, but as a product, we would need a stronger processor in place of the desktop. The system design (hardware) is shown below:

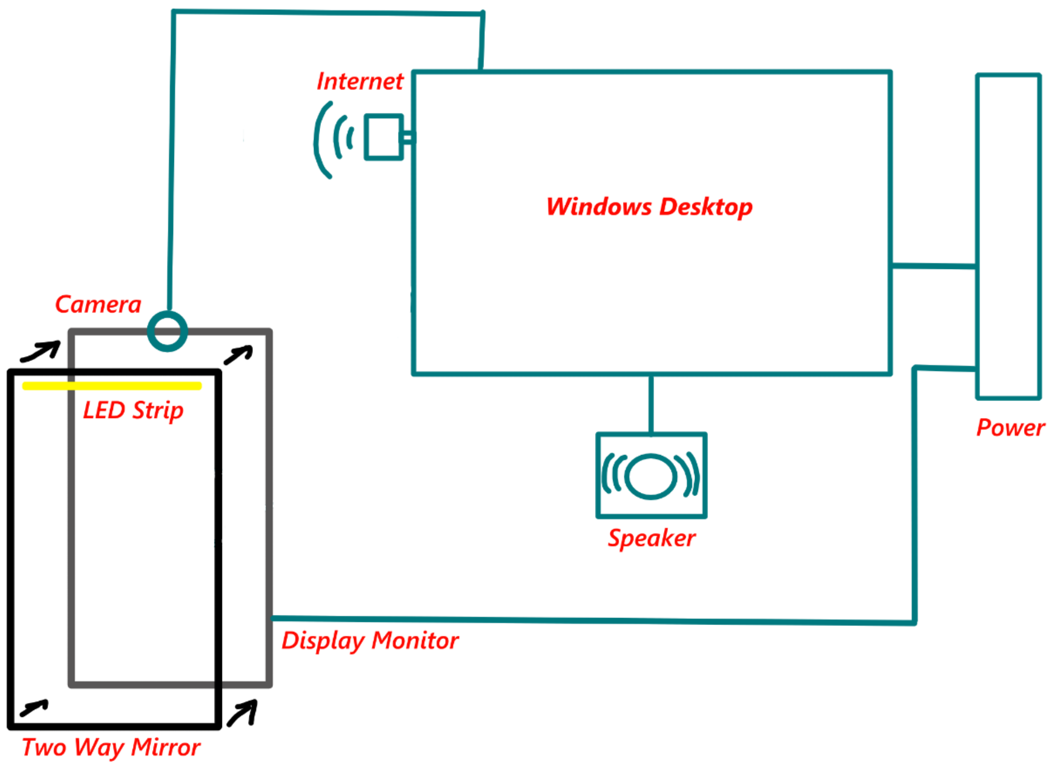


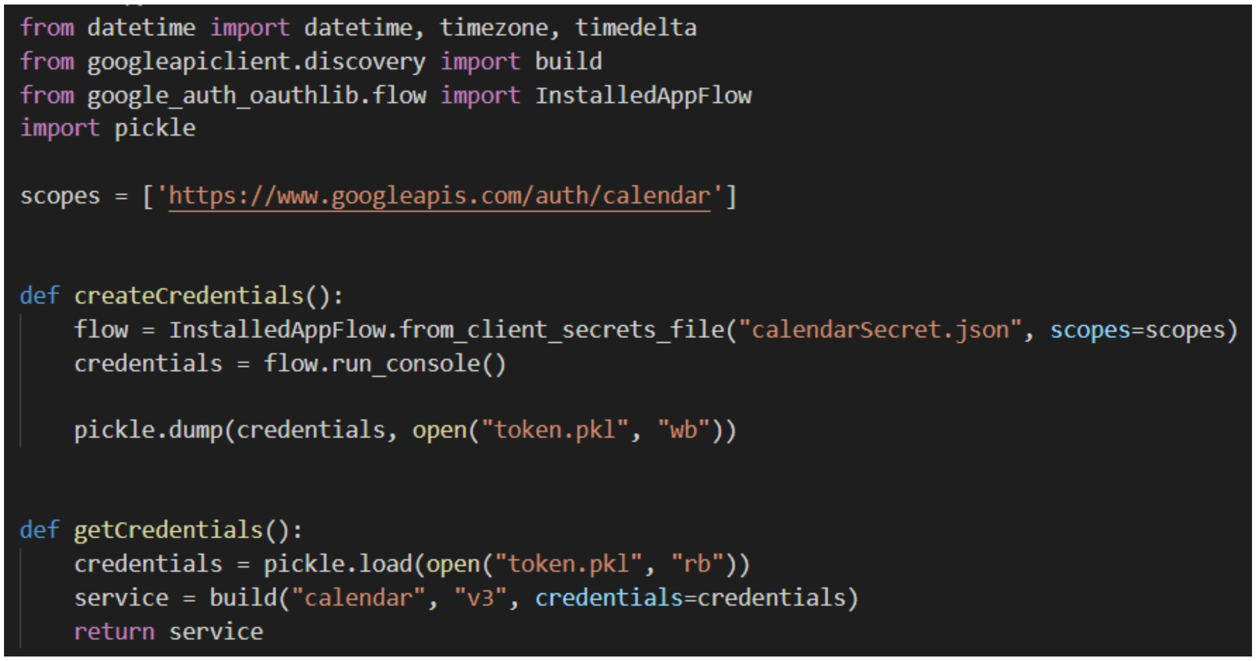
Figure 3: System Design

With the current design, there are multiple external connections from the mirror to the desktop. However, with a strong embedded processor, the mirror would not need any external connections and if it did, the connections would not be visible outside the mirror.

Another improvement would be the integration of a higher quality camera and microphone. With our current camera and microphone, the mirror is able to recognize gestures and voice commands accurately if the user is standing directly in front of the mirror and close to the mic. However, as a product, users would want to operate the mirror from different places in their room and have it continue to recognize gestures and commands. With a better camera, the mirror can capture a wider angle and still perform gesture operations, and with a better microphone, users would be able to say voice commands from far away and the mirror could recognize the words clearly.

In addition to the hardware improvements, we saw that a major improvement the mirror could use would be to have an interface for users to personalize the mirror with their accounts. We have multiple personalized options in the mirror, such as connecting your google calendar, personal IoT devices, etc. Currently, these are hard coded into our program, with our credentials programmed into the mirror. As a product, each mirror would need the user's personal information, and since it is not viable to have user's code anything for the mirror, we would need an application that they can use to input their information.

Further on the previous point, in that situation, we would also need high quality encryption for the user data. Since connecting the Google Calendar API requires the use of the user's Google account, it would be essential to have strong security methods on the mirror that would prevent any data leaks. Currently, the mirror does not include any security encryptions to prevent data leaks, so that would be a necessary improvement.



```
from datetime import datetime, timezone, timedelta
from googleapiclient.discovery import build
from google_auth_oauthlib.flow import InstalledAppFlow
import pickle

scopes = ['https://www.googleapis.com/auth/calendar']


def createCredentials():
    flow = InstalledAppFlow.from_client_secrets_file("calendarSecret.json", scopes=scopes)
    credentials = flow.run_console()

    pickle.dump(credentials, open("token.pkl", "wb"))


def getCredentials():
    credentials = pickle.load(open("token.pkl", "rb"))
    service = build("calendar", "v3", credentials=credentials)
    return service
```

Figure 4: Code Snippet for Google Calendar Integration

As shown in the figure above, the current process for integrating a user's google calendar information includes the use of the tokens and saving information in a json file. This works for our current purposes as we only need to have one static calendar integrated, but in a real world scenario, we would need to improve our program to allow users to add or change calendars on the go without the need to hard code anything.

Lastly, an improvement we would make to the project if we could redo it is to complete the user testing at least 3-4 weeks earlier than we currently did. User testing proved to be an integral part of the project process, and we left with important feedback that was used to better the performance of the mirror considerably. Had we completed this process weeks earlier, we would have been able to act on further improvements. One participant’s feedback that we were not able to take into consideration was the improvement of our user interface. We discovered that in our efforts to create a minimalistic mirror, our resulting user interface was not very aesthetically pleasing. The participant had good advice in terms of asking us to improve the user interface and make it more engaging, but given the time we had left, improving the user interface was beyond our scope. However, had we done the user testing a few weeks prior, we would have enough time to improve the UI and complete a better product overall.

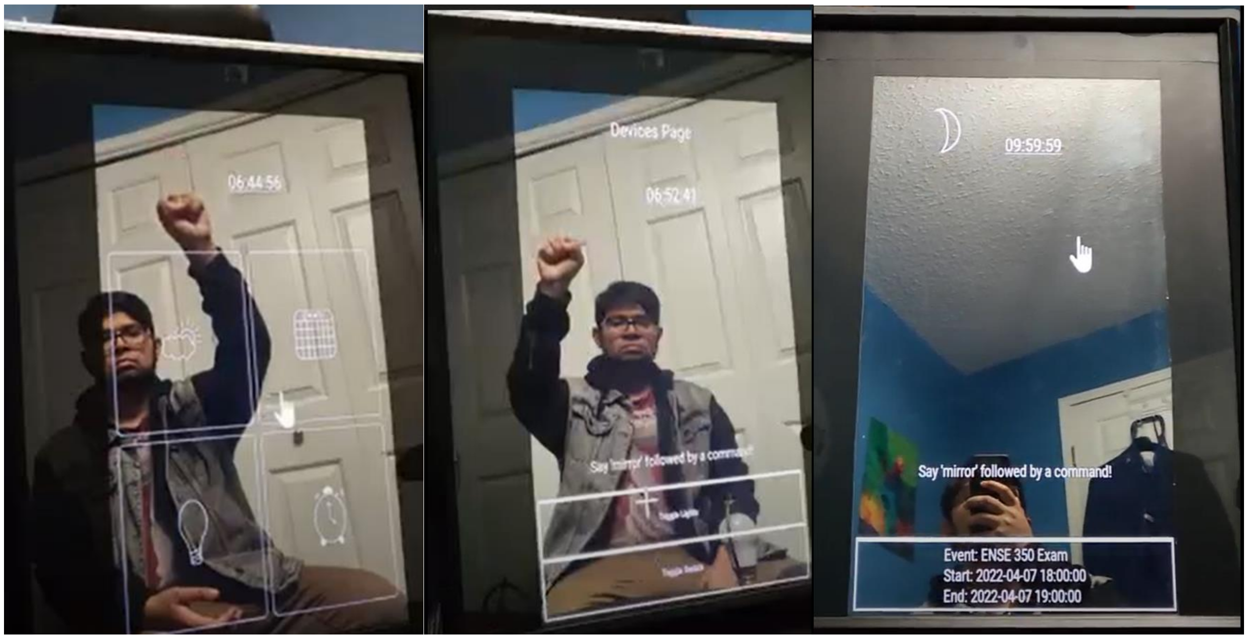

06:44:56
Devices Page
06:52:41
Say 'mirror' followed by a command!
09:59:59
Say 'mirror' followed by a command!
Event: ENSE 350 Exam
Start: 2022-04-07 18:00:00
End: 2022-04-07 19:00:00

Figure 5: User Interface Samples

As shown above, our user interface is on the minimalistic side. We wanted to prioritize that so users can still see themselves in the mirror while using it. However, based on public opinion during user testing, the current design is hard to see especially in a busy atmosphere. Given more time, we would have improved this either with the introduction of another color to contrast the white, or an alternative solution.