Hunter Culler, Paul Guszak, Gabriel Jones

Senior Design 2

Spring 2021 Project Report

Project Explanation

The project we chose to tackle is a smart mirror adaptation program. We chose a previously working code base to start with and build off of that. This project was worked on by all the same members when the first version of this project was taken on. The only change now is goals and there is one less team member. As we had the hardware knowledge, we wanted to add more software capabilities and learn more from the process. Our approach to this project was that we wanted to incorporate real-world scenarios. That being said, having a working code base is usually something that happens in a real job scenario. One where you need to familiarize yourself with code and functionality first; before just jumping into the coding process. As the latter is usually what happens first in a school environment.

At the starting point of this we had an App that runs APIs taking data from hard coded scripts to be functional, so we chose to make this more dynamic. As a team we wanted to make this code base more user friendly and functional. To make the process of running this program we chose to implement a way to sign in and launch the smart mirror program. We used another web application (based on our own first python implementation) in node.js to make this, making a functional local database for accounts on the running raspberry pi. This simple implementation on the local device makes the process super easy to use and not too confusing for the user. After the simple front end user experience in conjunction with the database, we then have a way to carry over that information to the smart mirror app.

Our first implementation was a python script for a CLI. The user signs in (or creates an account), then in the background the python script runs code to take the just entered user info into a local server. We chose to use a local flask server to lessen the complication of possible connection problems for the user since the base of this project uses a wirelessly connected raspberry pi. This then stores the information we want to display from the user profile to a file in the server that could be read and shown on screen for better user experience. Then when there is a way to take the data from this server (location, name, status, etc.) the final magic mirror can be run. All of this display is to help the user in an informative way.

The final display is to have many modules display a plethora of information to help the user as they use this mirror application. As a final product, this display is fitted under a frame that has a thin 2-way mirror film. This covering allows the information from the monitor's display to show through allowing the user to read and look at the display as a normal screen, while then also being able to look back at themselves as one would in a normal mirror. This dual functionality is meant to give extra ease to the user. From being able to learn and see information in front of them such as weather, news, time, etc. But also being able to use this hardware to function as a mirror too to help with the users self-care routine.

Work Distribution and Roles

Throughout this project there were very distinct roles. As there were only three members on the team there was extra work and ideas that needed to be done. That being said, all of the responsibilities and tasks were all crossed and shared at some point between all the members of the team. Other than those small crossovers, in the beginning and during this project, there were definitely major rolls for everyone. The three people in the team are Hunter Culler, Gabriel Jones and Paul Guszak. All being students who worked on this project in the past we all understand how the first version of the project worked. We chose roles that we could learn from but that were not terribly different so we all could have a starting knowledge base to contribute. Hunter Culler was the team member to take on the front-end design of this project. He decided to create a UI that was simple and easy to use. That would also help with our data that we chose to be important for this type of project. Gabriel Jones' main tasks were more back end oriented. As a team we decided to have him work with libraries to leverage better encryption to best solve how to securely save and handle having user information while running a local server. Paul Guszak was the team lead for the project. During the first part of the project made sure to get familiar with the code base completely to help the team understand how the magic mirror app functioned, as well as worked with the team's supervisor and the team to make sure plans and developments were going as planned. While also making sure to check in and help the team on anything such as decisions or editing code to show outlines of functionality that needed to be done for the project.

Project Components

For this project there are 3 major components to make this project function in the implementation the team had envisioned. We chose to add many components that would be easy to run on a small powered device but simple enough to continue using throughout time.

The first component is the language JavaScript. This is a language we used to create two of our working components that make our project function. We leveraged this versatile language in two major ways. The first was to edit and tweak the code base first. To add in modules and designs to the already working program for a more user-friendly display. To add in functionality to make a more dynamic program with the other components. We also used this language to make a second web application to function as a front end for a better user experience. As Hunter was the team member to mainly work on it, he used JS to run the dynamic side of page rendering. As well as get data from HTML pages and the accompanying forms for user information entry to save to a database. This ability makes the two JS components very versatile and can be interchanged with the python component due to functionality.

Building off of a user-friendly sign in/up page, we also created a python script as another major component. The point of this script is to be the "middleman" that takes the user information from the sign in and sign up. This component is one of the most important as it carries and deals with all the user information, to dealing with security, to also dealing with communicating with the data based used. This component can be used but also in place of the JS implementation. As it uses specific hashing for the inception and safety of the user data. It uses a SHA-256 to hash the password so that the plain text password will not be the security for the user accounts. Once this is done, we have a hash that is virtually impossible to recreate and also a program that can compare and deal with this interface between the database. In addition to

having a way to encrypt our user data, our python script is also responsible for launching the magic mirror application after sign in but also responsible for making the local server. This is a flask server that is a continuous loop created in this script. Till the web app gets the information it needs; this python script gives the needed data over for the magic mirror program to display for enhanced user friendliness and ease of use.

The final major component is one that has already been mentioned. Using MongoDB for our project is the backbone of this project. Getting our different code components (Python and JS) to work with the database is the reason we can have a user base. Our thoughts were that our DB would store the user's info in case our product of a working magic mirror is used by multiple people in the same house. But also, as a secure way to store user information for user safety.

Current State

Over the course of this project there have been many positive strides to complete this project. As of right now every major component is fully functioning. We have a functioning code base (the edited JS) that is the main MagicMirror program. As of now this program can take in the location and use API's to then retrieve and display information. Additionally, it can now display the status of the user as well as the current user who is signed in along with their data. Our front-end web app is launchable through the local host port. This is fully functional and works for the user sign in and signup to communicate with our other programs and the database to store the user information. Our python script is also completely functional and can be a simplified CLI if there are issues launching the web app for the UI. The encryption is also completely functional for better security for the user(s). Alongside the web app front end and python CLI, they can both completely communicate with the database as long as there is an internet connection, as our MongoDB is cloud based. With all of this information there are commands in code to display MagicMirror programs and all chosen in code modules and information. In all, all of the components and major module functionality are working.

Future Implementations

Even though during the span of this project all the main components worked as they were planned on building there were a few small issues that we would like to address in the future that only require more time to address.

The first one is the connection of all these components. Due to the old version of the hardware (raspberry pi) there were issues getting all the components to run concurrently on the same machine. As the libraries used in our front-end web app and python encryption were too new for the old OS to handle and were unable to be properly installed. However, we believe a newer version of a Raspberry Pi can fix this problem.

The only other small problem with the "weaving together" of all these components. The process of figuring out the communication 100% between our various different language components (JS web apps and python script).

Aside from running out of time to implement other desired functionalities, connecting all

our working components of the project, our project was a successful learning opportunity and is something that time will grant the full completion of.

Conclusion

Over the course of the team's final year, this project has definitely shown to be a good challenge for the group. There were always stumbles and hills to climb to get the functionality the team wanted. We chose to implement things that we are confident that we can figure out as a team. This led to a meeting of what we thought was possible with the knowledge we have for the rest of the project. And not to "bite off more than we can chew". So, we decided to make a more user-friendly approach to this project and add more software based functionality rather than hardware additions. As we knew this would best benefit our learning based off of what we already knew going into this. Over the course of this project, we had regular meetings to check in and talk about the progress and if plans needed to change for final implementations and issues that were arising. In the end we were happy with what he had completed and able to solve with regard to this puzzle that presented problems throughout the course of the project. But in the end, there were still implementations the team wanted to solve. However, we are all glad that we took on the project to learn more about various components about a project to learn ourselves about Web apps, security, raspberry pi functionality, and hardware software integration.