Simon Memory App

Bobby Eshleman
Chris Clark
12/7/18

In our childhood we both played a game called Simon which was a 4 button memory game that was a fully integrated hardware game with playability in a local environment. What we proposed to do with our project was to make a game that could be played by utilizing the love of mobile apps while still using the hardware that would bring back that nostalgia factor that we loved in the past.

We would propose to use an application that would be located on the users phone and be used by the user. We would have hardware that would have 4 separate large and illuminated buttons, it would not be in the same configuration or look the same as the original Simon but this was due to our potential stretch goals. We would then recreate the game Simon using the phone and this piece of hardware. The app would store user information (separate users) with high score information for each user to a server that could be accessed outside of the app. This server would also have the ability to show high scores for all of the users.

We identified multiple stretch goals that would go beyond and add to the wonderful experience that we had in our younger days. The first would be that the app would look for local app users that also had the app and if a game was initialized it would notify local app users to see if they wanted to join. The next would be that users could select a hard mode that would allocate six colors instead of just 4 and would be given a multiplier to their score instead of having separate categories on the server. Another stretch goal would be to utilize uncommon security measures to secure data of the users. Granted there is no personal information in the game (this was intended) but securing the data is part of any good embedded system. The final stretch goal would be multiple users using the app/hardware at the same time in a head-to-head manner instead of just single user possibilities.

Within our project we used firmware, mobile applications, and a web framework to implement all details of the project. We did this so that we would get experience utilizing all aspects of software side aspects.

The web app that we created was written in Django with a SQLite3 implementation for the database of users. We did this because there was a plethora of documentation on Django implementations of user databases. Within our SQLite3 system we utilized Django ORM which is an object oriented abstraction layer which works in the layer above the database.

The webapp would serve up the information to the Android app using CSS/HTML/Javascript. This database will host the general manipulations such as create, delete, update, etc for each of the entries within using the RESTful API that was implemented within the Django implementation.

This system would be deployed using a web hosting system called “pythonanywhere” and would serve the data up in JSON format. We ended up changing this within our system because it was easier to read just plain text but we dealt within this in the firmware outlined below.

The user authentications that we implemented using the Django authentication system and supports protections against Cross-Site Request Forgery (using an authentication token) and other primitive types of attacks. The user authentications would allow users to not have to re-login each time they played the game and would also protect the system upon restart or deletion of the game/application.

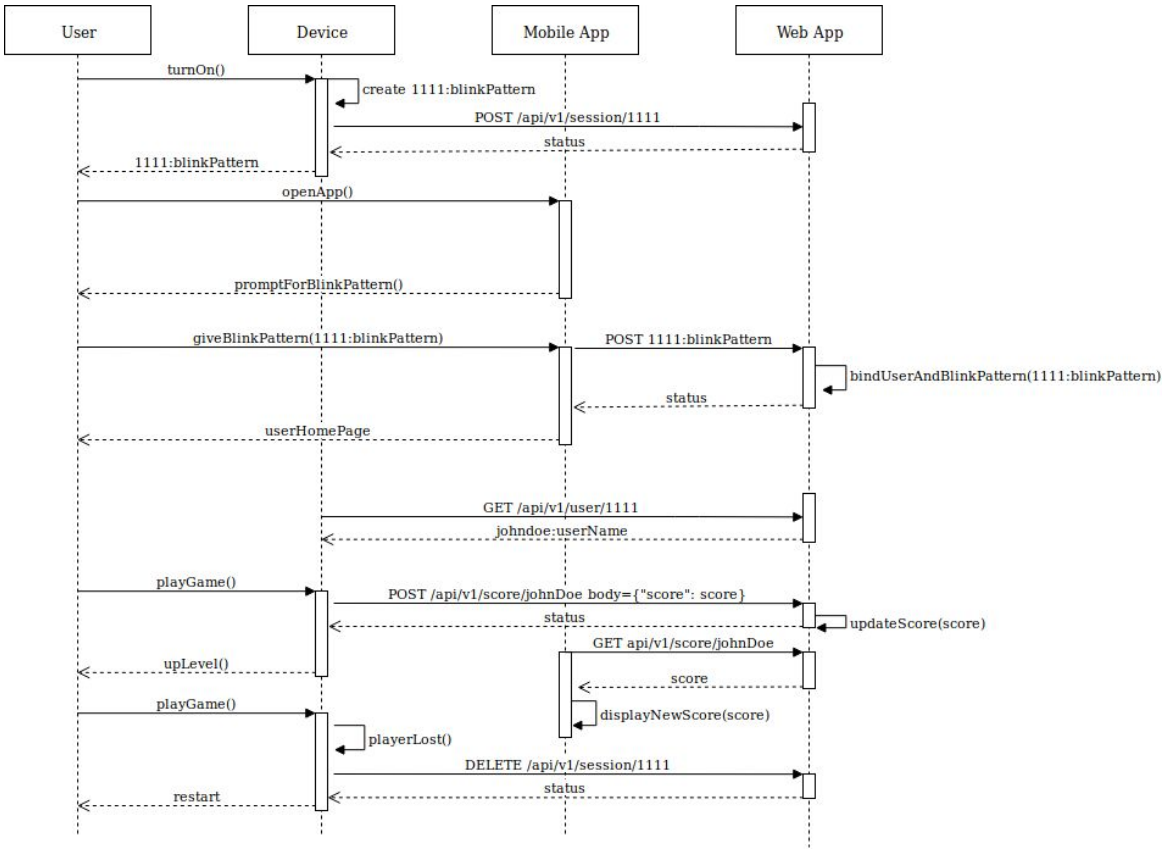
The mobile app that we created was a webview implementation within the Android biosphere which implemented the buttons for username and password entry as well as the buttons for playing the game from the app. This aspect of the project was the simplest as that is what we really focused on throughout this term and specifically in this class. This implementation was created using Java/Android but was getting information served from the web app

For the hardware we started with an ESP32 development board and then started adding buttons with LED’s in the button. We chose the ESP32 development platform over a Raspberry Pi 3 due to the low overhead for development and the ease of use. The ESP32 allowed us to use multiple buttons, connect to our server, and to the web app with startup code that was incredibly modular and quick to develop but hard to master.

The ESP32 also allowed us to development the hardware within the Arduino IDE development environment which is very useful for quick testing and fast design iteration on how we were developing.

The hardware would connect to the app and the server with a randomized server identifier that the user would then see and be able to enter in to the application for connection. Once the ESP32 got back the connection details with user information it would start the game. The hardware would then randomize the memory sequence for the user to enter in on the app. Once the user failed the sequence the hardware would then send a post to the server with the high score for that particular user to be displayed and saved within.

Program Flow Chart of both Software and Hardware:



Upon completion of this project there were a few things that we were proud of and a few things that didn't go so great.

One of the positives in this project was how much we could do with open source software and hardware which is one of the driving forces behind both members of this project group. We were able to put together a hardware and software solution to a game that we both enjoyed.

Within this project we were also both able to work with software and hardware pieces that we had not worked with before. This was incredibly enjoyable to the both of us as we worked through this project. On the software side neither of us had worked with Javascript which gave us some issues but they were overcome. We were also very proud of the server side implementation of the webapp using the CSRF implementation for protection of the sign in data being presented by the user.

Based on our description of the project that we set out to accomplish we achieved these goals. We got an application running on a users phone and got it to connect to a piece of hardware running the Simon game that we designed. We also got the hardware to connect to our custom

server side implementation which contained high score information for each user. So not only was it holding this but also a user database that could be referenced by future versions of this particular implementation.

Some of the problems that we found as we were working through this project is that the hardware that we had chosen was using an interrupt system that would not handle the amount of interrupts (in hardware) that we wanted. This made the hardware button implementation impossible without a new hardware platform to work with. On the software side of the application we had difficulty deploying our webapp to the cloud due to writing custom software for that part of the project. It was also difficult because Javascript was something we did not know very well at the beginning of this project but we quickly regained our legs and worked through our difficulties.

We also had some issues with our implementation of CSRF within the application but in using the tokens that we created it protected the user information from any potential prying eyes.

The setup of the software and hardware is as easy as connecting the hardware to a wifi system that can access the web. You will also need to install the application on to your preferred android phone system even though this would also be able to work with an iOS phone with proper development.

Once you get the hardware setup and the application on to your phone you will need to get your own instance of python anywhere and put up the web application source onto that source. You would then need to change the “siteaddress” and “root” variables in the hardware code to point to your instance of python anywhere.

After you have your hardware setup with the firmware installed and the application installed with the firmware, mobile app, and server side setup everything should connect automatically with the entire system. You should be up and playing Simon within moments after installation.

The distribution of labor within this project was initially pretty divided but then got a little more blended as we went on with the ending being that we both worked on aspects of the others focus.

But for the big part of the project the distribution was:

Bobby:

- Mobile application development
- Server Side main development
- API implementation
- Hardware testing

Chris

- Firmware development
- Hardware build and test
- API implementation
- Server Side assistance