Challenging aspects about Home Automation with a Arduino; Group 15

Any program that requires an internet connect on your side, requires a server on the developers side. Your device sends and receives data to and from the server constantly in order for the application to be able to work. This requires the use of server side scripting on the developers side. In our case, we chose to implement this with PHP and MySQL on the desktop side, and MySQL and JavaScript on the backend for the mobile side. For the desktop side, everything was working seamlessly in regards to communication between the server and the database in regards to everything except actually communicating with the devices. We prepared PHP files to issue $_POST and $_GET requests in order to communicate with the Arduino, but we were unable to successfully do this as we were getting errors with our SQL statements for very odd reasons, after unsuccessfully trying to fix this, we decided to implemented a simpler solution. We decided that we would communicate using only $_GET requests between the Arduino and the Server, without interacting with the database at all. In order to exchange information, we employed text files. Depending on what button the user clicked, the server would overwrite that text file with the parameters that the user chose. The Arduino then gets this information through the $_GET request (Arduino performs this request at a very small interval in order to make the device respond quickly) and uses it in its own code which then controls the devices. Similarly, we use output text files (for device status) that the server reads and outputs to the user. In order to implement this for multiple users, we realized there must be multiple copies of these text files, one for each user. So we edited the registration code accordingly to create a new folder on the server where these files are located, and this becomes the root folder of that user. So even though there were challenges, we were able to successfully implement the software that we had set out to do, unfortunately the mobile side of things were a little bit more complicated.

The website was easy to make as there was a simple connection between it to the server on the raspberry pi (website is hosted on the server, which it itself, is the raspberry pi) but the app required data to be sent from the mobile device to the internet, and then to the raspberry pi. This required the use of the android volley library and JavaScript Object Notation or JSON. While these two components are small in the grand library of code, implementation of the volley library and JSON allows for quick and proper data transfer from a server to an application. Normally, every time we want our app to read or send data to and from the server we would have to code a JSON request which would convert the data into a simple readable format but since our project required communicating with the server frequently, we had to figure out a way to efficiently code without having to write requests all the time. This required the creation of a request queue and a singleton class which created a single instance of our request queue. Once this class was created, making network requests was easier as it required much less code. After the code was made, and we started testing, we realized that our requests were not being sent to the server and no data was being exchanged. After cross checking our code with resources online, we still could not figure out how to successfully make a connection. We know there is nothing wrong with the php files that we created because they could be called when we tried communicating with a local server, and the code seems correct. After checking the error logs we realized there were a lot of JSON parsing errors such as “JSON syntax error: Unexpected Token” or “internal server error 500.” We figure the error lies within the JSON code as we are unable to send requests to our server. This was the hardest part of our project and although we were unable to make the application communicate with our server, we were able to successfully implement what we had set out to do, but only on our local server. I hope this account of events gives insight to other groups in the future who want to model a future project in a similar manner.

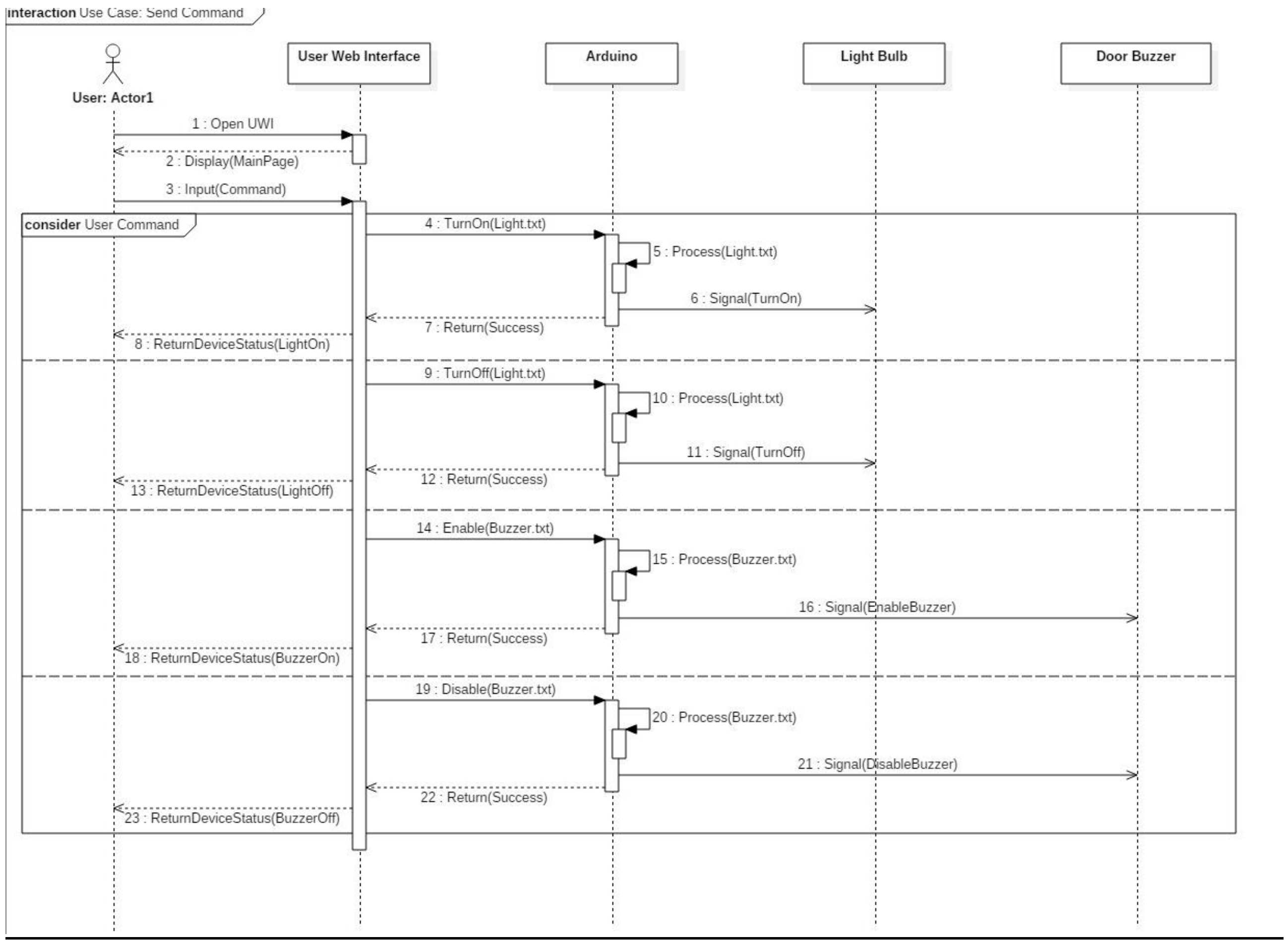
USEFUL LINKS:

https://www.androidtutorialpoint.com/androidwithphp/android-login-and-registration-with-php-mysql/

https://www.androidhive.info/2014/05/android-working-with-volley-library-1/

http://www.instructables.com/id/Raspberry-Pi-Android-App-communication/

ADDITIONAL INTERACTION DIAGRAM:


interaction Use Case: Send Command
User: Actor1
User Web Interface
Arduino
Light Bulb
Door Buzzer
1 : Open UWI
2 : Display(MainPage)
3 : Input(Command)
consider User Command
4 : TurnOn(Light.txt)
5 : Process(Light.txt)
6 : Signal(TurnOn)
7 : Return(Success)
8 : ReturnDeviceStatus(LightOn)
9 : TurnOff(Light.txt)
10 : Process(Light.txt)
11 : Signal(TurnOff)
12 : Return(Success)
13 : ReturnDeviceStatus(LightOff)
14 : Enable(Buzzer.txt)
15 : Process(Buzzer.txt)
16 : Signal(EnableBuzzer)
17 : Return(Success)
18 : ReturnDeviceStatus(BuzzerOn)
19 : Disable(Buzzer.txt)
20 : Process(Buzzer.txt)
21 : Signal(DisableBuzzer)
22 : Return(Success)
23 : ReturnDeviceStatus(BuzzerOff)