A Smartphone Application For Home-based Hand Rehabilitation

Kody Fitch, Brandon Shepard, Martin Soto, Joseph Yang
Supervisor: Professor Rahman

CS595 Spring 2019

1 Abstract

Our goal is to design and prototype a smartphone based, hand rehabilitation application. Deliverable at the end of the project will be a fully functional smartphone application that guides and measures rehabilitation through the use of a wearable robotic glove.

2 Functionality

The smartphone application will communicate wirelessly with the robotic glove. The robotic glove will receive exercise direction from the application while communicating glove positional data back to the application. The application will process, display and communicate this glove data to the patient while simultaneously uploading hand data to the cloud. With the data stored in the cloud, the rehab professional can login to the application, access patient data and prescribe exercises remotely.

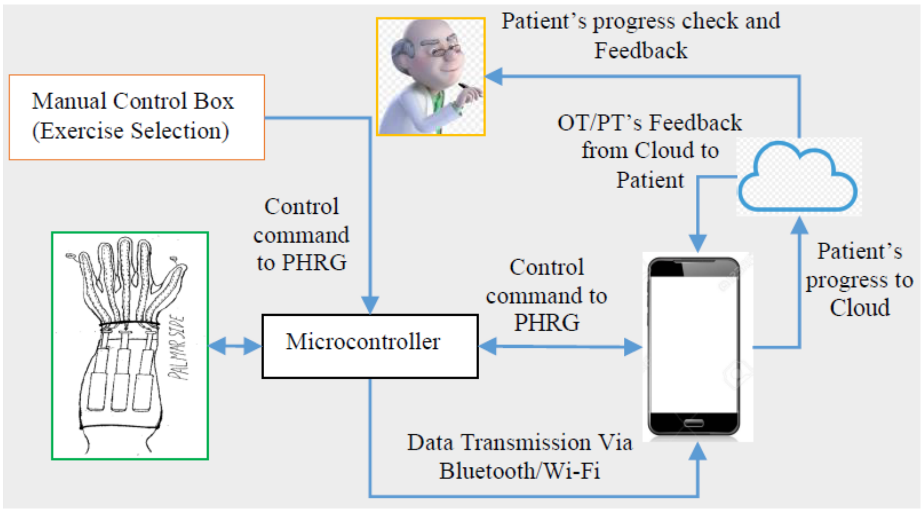

Patient's progress check and Feedback
Manual Control Box (Exercise Selection)
OT/PT's Feedback from Cloud to Patient
Control command to PHRG
Control command to PHRG
Patient's progress to Cloud
PALMAR SIDE
Microcontroller
Data Transmission Via Bluetooth/Wi-Fi

Figure 1: Rehabilitation System Schematic

The long-term goals of the project is to have multiple physical augmentations so that different body parts can be rehabilitated through this application. In addition, in order to reduce workload from physicians and doctors, machine learning can be implemented to analyze the data generated in order to give advice to patients.

3 Semester Scope

Given the time and resource constraints for the capstone, the scope of the project has been scaled down as shown in figure 2. The main change is the removal of

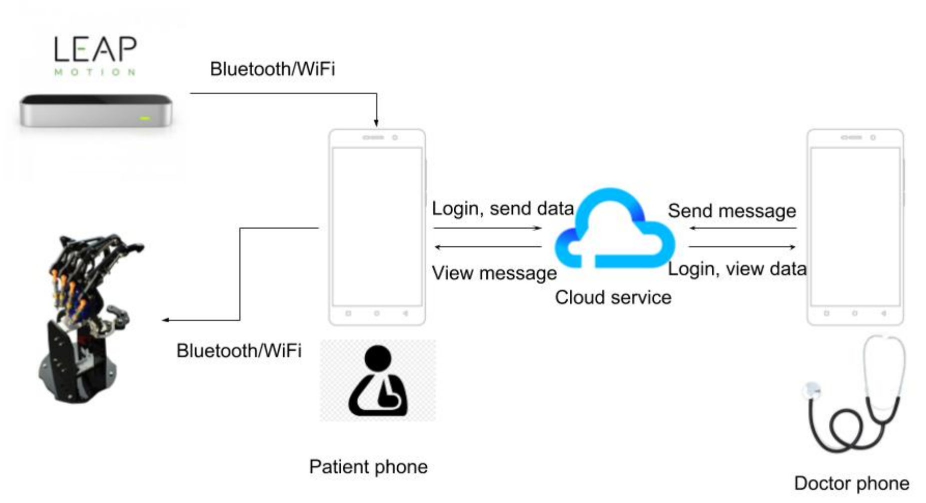


Figure 2: Updated Project Scope

the robotic glove. The glove has not been produced yet, so the project will use a Leap Motion device and robotic hand in its place. The Leap Motion device will reflect the motion tracking functionality of the glove and the robotic hand will reflect the assistive motion of the glove. By doind this, an API and communication protocol can be developed in lieu of the robotic glove. The glove can then be produced with the protocol and API in mind, allowing for straightforward replacement of the Leap Motion device and robotic hand when completed.

4 Technologies

4.1 Smartphone platform

We have chosen to use Ionic. Ionic is a framework based off the popular Angular platform created by Google. Ionic allows for the creation of platform agnosic, hybrid applications. A hybrid application supports both web applications and native applications, making it a strong choice for this project. Additionally, the ionic website appears to be well documentated and the platform comes with the majority of the components needed to complete the project.

4.2 Robotic glove simulation

A robotic glove prototype is being developed by Prof. Rahman's team at the UWM Biorobotics lab. In the mean time we will be simulating the glove with the use of Leap Motion sensor and a bionic robotic hand. Both items are being provided by the UWM Biorobtics lab.

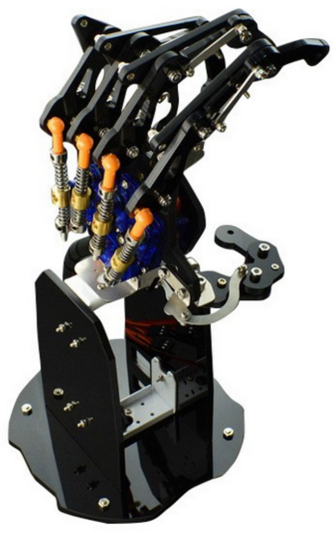


Figure 3: Bionic Robotic Hand

4.3 Cloud service

Amazon AWS will be used for our cloud service. The Ionic team and the Amazon AWS team seem to have a good relationship with each other and have collaborated in the past to make ionic applications easier to integrate with Amazon AWS. Furthermore since Amazon AWS is one of the most popular cloud platforms there is a lot of good documentation for it.

4.4 Justification for & alternative technologies

Project requirements and resource restrictions influence the platform to develop the phone application on. Native apps have better performance and more direct API to the phone's hardware, but require a seperate codebase for iOS and Android. Hybrid apps are great for rapid app development, but come with slower performance. The requirements do not require high performance such as 3D rendering and the app needd to be completed in a short amount of time, so the team has decided to use a hybrid platform.

As with most web dev projects, there are countless alternatives. Some popular alternatives to Ionic include React Native (React), Kendo UI (JQuery) or Quasar (Vue). The platform decision was based on three criterion:

- group skillset
- quality of documentation
- platform coverage

With quality of documentation and platform coverage being sufficient across the various platforms, the differentiator was group skillset. Overall, our group had the most experience with Angular platforms. As such, Ionic was decided upon.

The robotic glove is currently in development, and there does not appear to be a marketed alternative. There appear to be alternatives to leap motion sensors, but since Prof. Rahman has provided and recommended the simulation hardware. As a result, we did not extensively search for alternatives.

In regards to the bionic hand. Generally speaking, as robotic hand precision increases so does cost. The main purpose of the hand is to confirm communication and sending locomotion instructions. It is a temporary device to be used until the glove is ready to replace it, so high precision is not required. Again, Prof. Rahman has lead the decision process for robotic hand selection as he balances budgetary constraints with precision constraints. The product that was ordered is a $219 hand from robotshop.com that is compatible with Arduino.

5 Design

5.1 Leap Motion to Robotic Hand

Since the Leap Motion device is a temporary device for prototyping, the goal is to convert the information from the Leap Motion into a format that is more universal. Two issues to solve with the Leap Motion device is that it produces data in their own structures and that the rate is too fast for the Arduino at 200 per second.

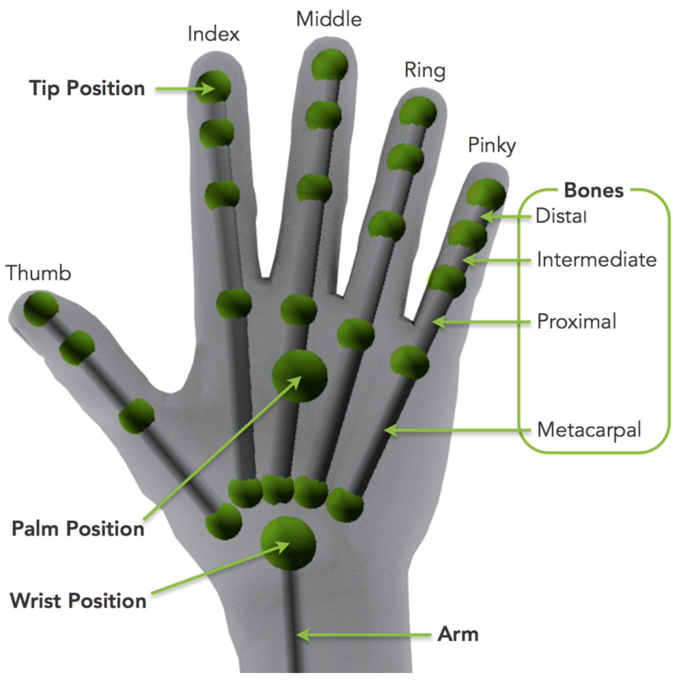


Figure 4: Leap Motion Frame Data

Figure 4 shows that data generated by the Leap Motion device while it tracks the hand. The information needed for the project is the angles of the fingers with respect to the palm. To get this information, the vector of the distal bone for each finger is gathered along with the vector for the normal of the palm. By doing a dot product between the distal vector and the palm vector, the angle between the two can be determined.

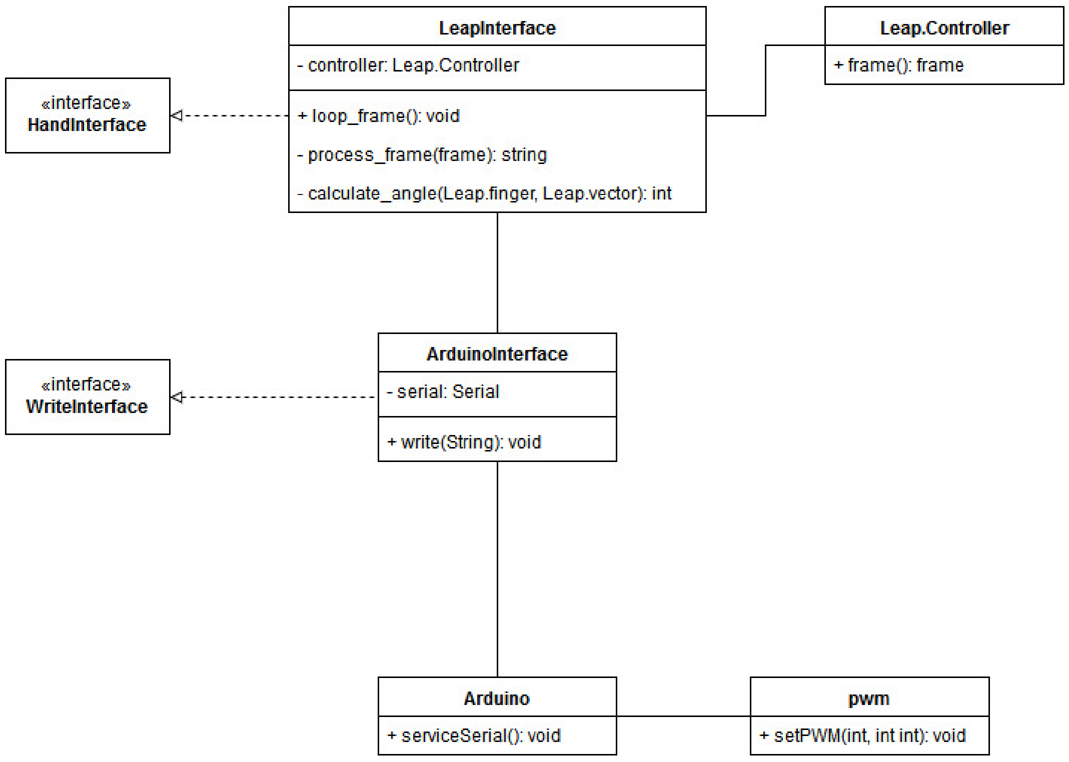


Figure 5: Leap Motion to Arduino

The above figure shows the general design of how data is extracted from the Leap Motion device and process by the Arduino. The data shown in Figure 4 comes in the form of frames via the LeapController. To deal with the 200 frames per second issue mention earlier, LeapInterface runs a a method, loop_frame, that manually polls the LeapController at a reduced rate of 20 frames per second. The frames are then processed as described earlier and passed to the ArduinoInterface.The final string passed to the ArduinoInterface is in the form of a letter deliminated string, with each succeeding letter indicating the next finger. For example, if all the fingers were at 180 degrees, the string generated would be in the form "180a180b180c180d180e".

The ArduinoInterface opens a serial connection to the Arduino and writes the string given to it. In the Arduino, while the serial buffer has information, serviceSerial builds an integer object and sends out the built integer as soon as it encounters an alphabetical character in the buffer. The Arduino is wired to the robotic hand's servos, with one servo for each finger. The servos take instructions in the form of pulse-width modulation (pwm), so another conversion needs to be made. Each servo has different tolerances, so minimum and maximum pwm values are calibrated for each finger. For example, the thumb may have a minimum pwm of 200 and a maximum pwm of 400. serviceSerial maps the instruction, ”180a”, to those values and sends the value ”400” to the pwm library for the thumb.

5.2 Bluetooth Integration

Purely reflecting the tracked movement doesn’t provide enough granularity in assistance for exercises. This is resolved by integrating the ionic application to the process via Bluetooth.

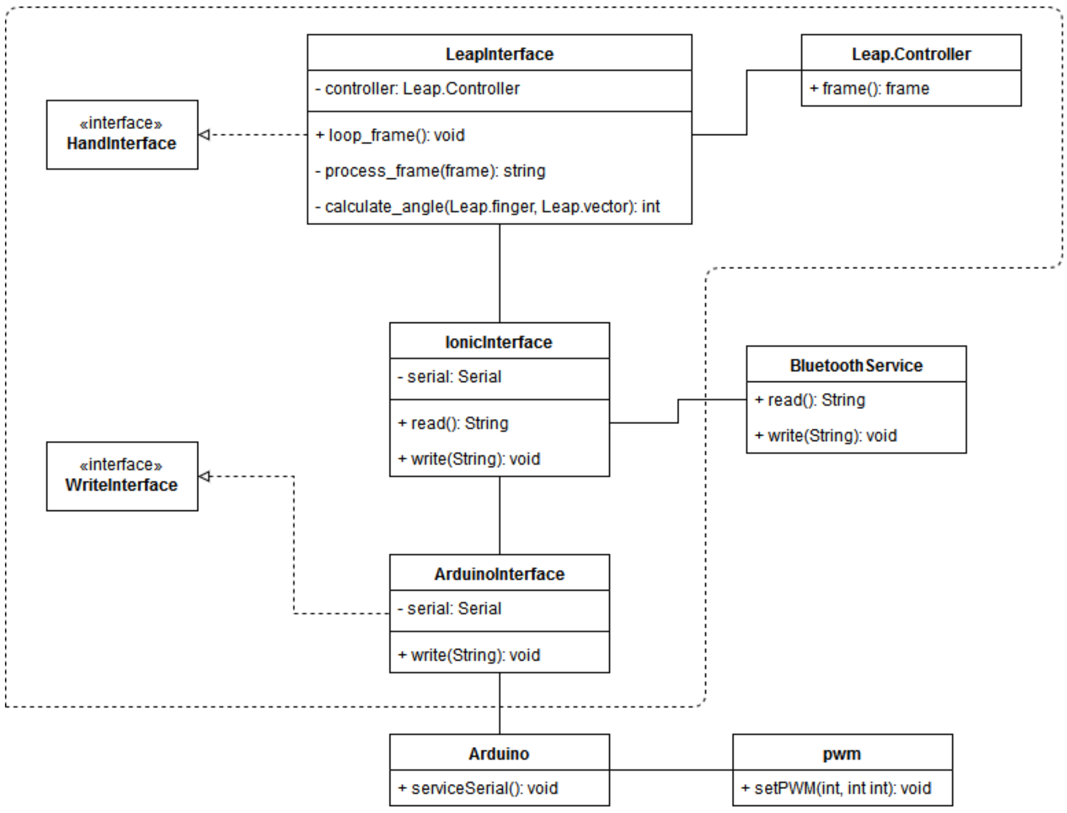


Figure 6: Bluetooth Integration

The figure above shows the updated process with Bluetooth integration. Instead of directly forwarding finger positions to the Arduino, the LeapInterface sends it to the ionic application. This gives the ionic application the opportunity to calculate instructions for the Arduino based on the prescription provided by the doctor. After the calculation, the ionic application sends the updated instructions back to the LeapInterface. Finally, the LeapInterface sends these instructions to the Arduino. The reason the application doesn’t send the instructions directly to the Arduino is to reduce the needed bluetooth connections to one, and because the Leap Motion sensor and the Arduino mechanical hand are prototyping devices for a singular assistive glove device.

For the Leap Motion program, the python library pybluez is used and the ionic application uses the Ionic Native BluetoothSerial library.

5.3 Ionic Application

The ionic application will be the main user interface for both the doctor and patient screens. The screens are dynamically rendered based on server metadata by using NgModules. Since the screens are dynamically rendered, the doctor and patient screens utilize the same UI components.

The general flow of events within the ionic application begins with the user login screen.

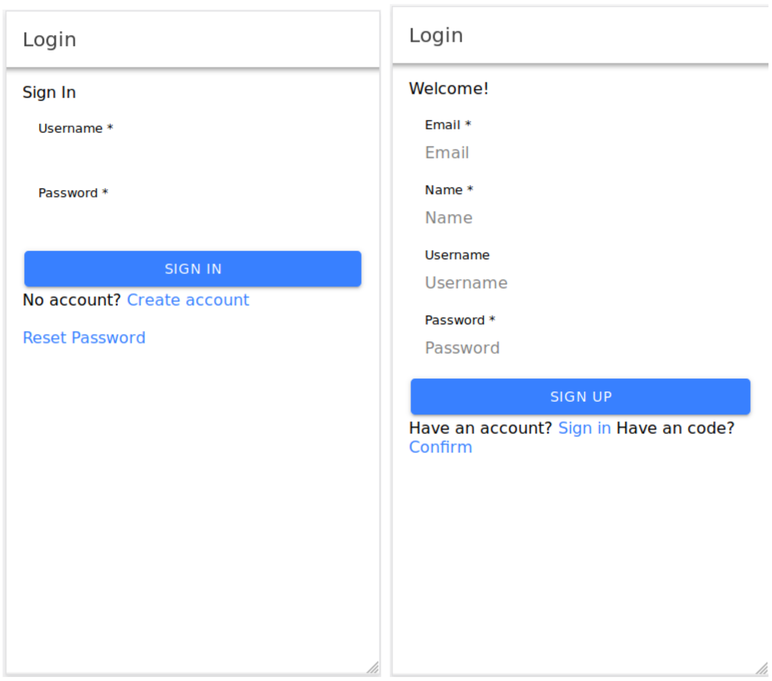


Figure 7: login and registration screens *(AWS cognito)*

This screen will prompt the user for their login and password. If an account needs to be created the user can click the register button to create an account. After authentication, the user will enter the menu screen.

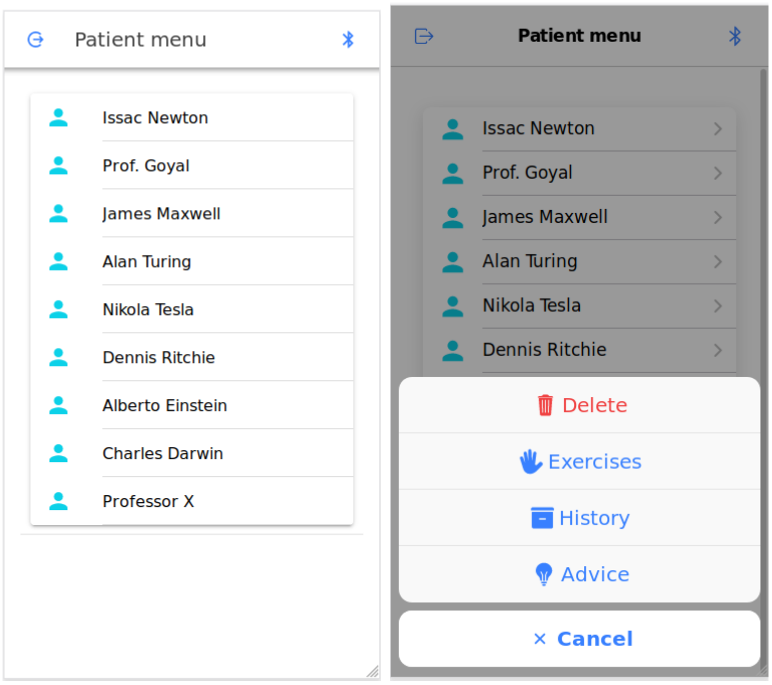
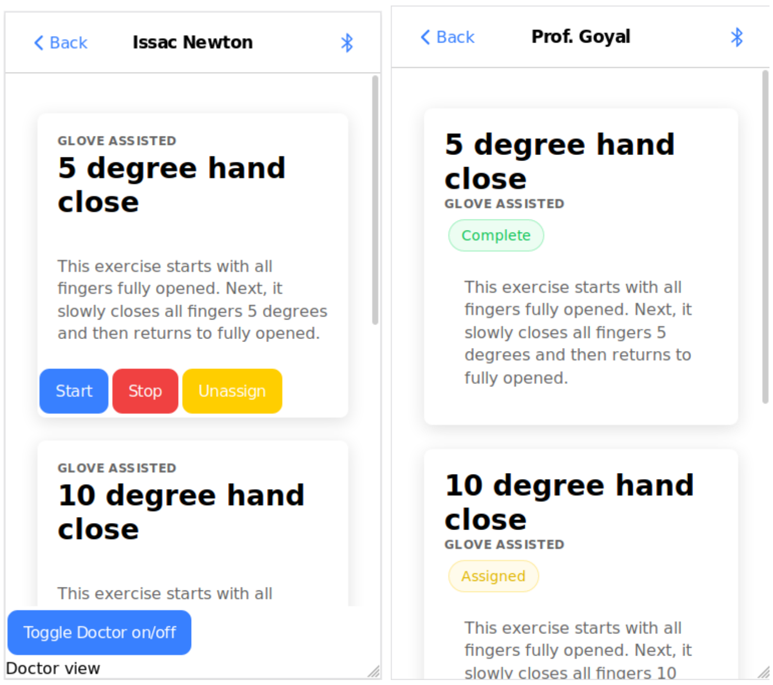


Figure 8: Doctor view of the main menu screen and action sheet. *Note: patient view would only see 1 entry*

If there is only a single patient, then only the patient's name will show up as a menu option. If the user is a doctor then a list of all patients will populate on the menu screen (as shown). After selecting a patient name from the menu screen, the user will be prompted with an action sheet menu. The menu allows the user to delete a user or navigate to either exercise, history or advice screen.



GLOVE ASSISTED
5 degree hand close

This exercise starts with all fingers fully opened. Next, it slowly closes all fingers 5 degrees and then returns to fully opened.

GLOVE ASSISTED
10 degree hand close

This exercise starts with all

5 degree hand close
GLOVE ASSISTED
Complete

This exercise starts with all fingers fully opened. Next, it slowly closes all fingers 5 degrees and then returns to fully opened.

10 degree hand close
GLOVE ASSISTED
Assigned

This exercise starts with all fingers fully opened. Next, it slowly closes all fingers 10

Figure 9: Doctor view of exercise screen (left) and history screen (right) *Note: Patient view of exercise screen would only show assigned items and omit have the 'assign/unassign button'*

The exercise screen is how the doctor or patient can prescribe or execute rehab glove exercises. The history screen is where previous activity is displayed, allowing the user to gauge progress. The advice screen is currently undefined, but it is expected to either display advice from the doctor and/or basic troubleshooting help.

Additional design decision include using an authguard service to protect the logged in screens and a single app module. The AuthGuard functions in conjunction with an authentication service, which manages the storage of our session token. We are using the built in ionic api server to handle our api calls along with the AWS export file. The app is held together with a single app module.

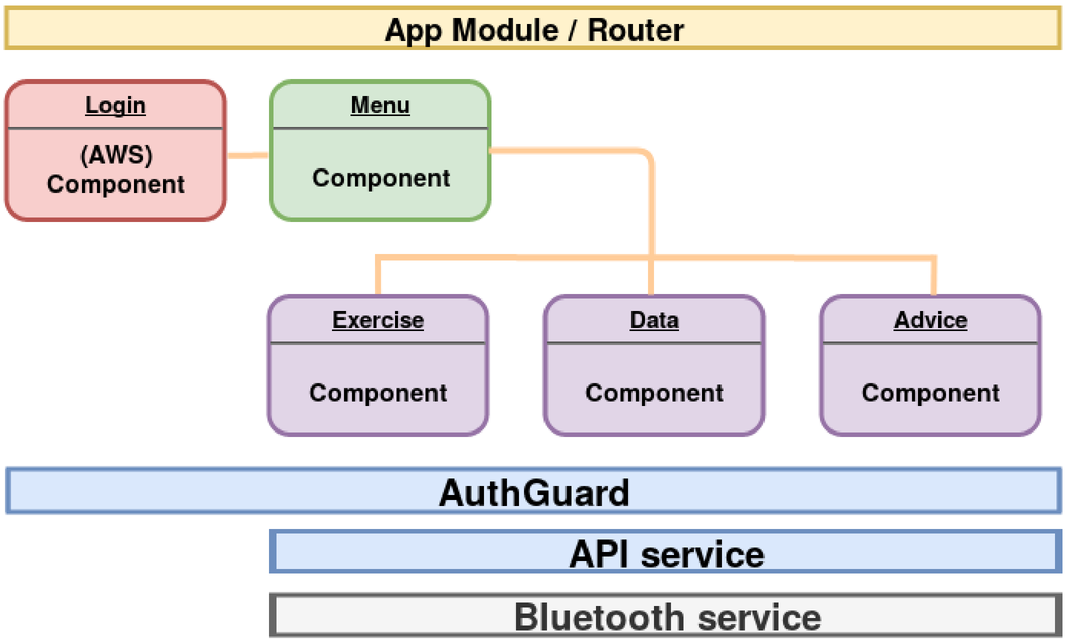


Figure 10: Initial ionic application diagram

5.4 Cloud Service

We used Amazon Cognito to take care of all of our user sign up, sign in, authentication, and forgot password services. All of the code is handled on the AWS side, we just simply display a module object on our html page. We changed the built in UI to match the colors better with the rest of our app.

For our back-end implementation we used AWS API Gateway, AWS Lambda, and AWS RDS. AWS Lambda is a service that allows us to run code without needing a server. Amazon API Gateway is a service that allows us to make API's that can talk with our backend services. We created our own AWS API which was composed of multiple API functions. Each of these functions was composed of lambda functions that either made GET or POST requests to the database. All of this was manually created on the AWS interface and was then used in our code by simply using the API name with AWS Amplify library functions.

To create our tables for our AWS database we used MySQLWorkbench. With MySQLWorkbench we could connect to our AWS database and create, edit or delete tables. With it we created the tables and edited the rows and columns for them. With MySQLWorkbench it was also very simple to add foreign keys to the tables.

The design goal for the database was to avoid data redundancy and structure for simple lookups. In our design we will be using 5 tables: PATIENT, DOCTOR, EXERCISE, PRESCRIPTION, and RECORD. The RECORD table will be used primarily for the history screen. The PRESCRIPTION table will be the main stay for the exercise screen. We may need to add a connection between the DOCTOR and PATIENT tables to help support the main menu screen, but this is undecided currently.

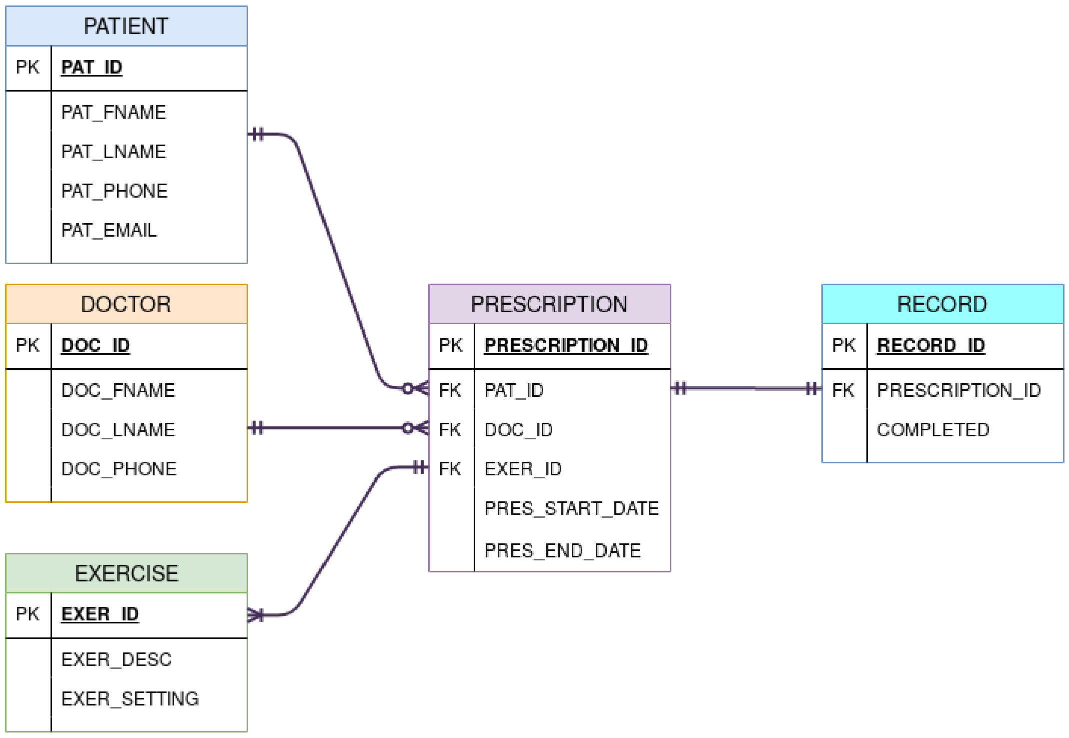


Figure 11: Database ERD Diagram

6 Implementation

Each member has been assigned to a part of the project to be the resident expert in. The assignment is as follows:

- Kody Fitch: Leap Motion
- Brandon Shepard: Ionic Application
- Martin Soto: AWS Cloud
- Joseph Yang: Mechanical Hand

Progress is tracked through milestones in implementation. The first milestone was to establish a serial connection from the Leap Motion device to the mechanical hand and to control the hand through data gathered from the Leap Motion device. The following milestone was to include the Ionic application through a bluetooth connection. The Ionic application would act as a router for data between the Leap Motion device and the mechanical hand. The final milestone was to connect the AWS cloud to the application.

When the process flow is complete with all platforms connected, features will be added as time allows. These features include messaging between doctors and patients, voice control, and unique interfaces for doctors and patients.

7 Conclusion

In conclusion, the project is to design and produce a prototype for a smart-phone based, hand rehabilitation application. This project has many different platforms that need to be connected to each other including the Leap Motion device, Arduino, mechanical hand, Ionic, and AWS. Since the long term goal is to allow for different types of rehabilitation devices to be used, care has been taken for the design of the interfaces between the platforms.