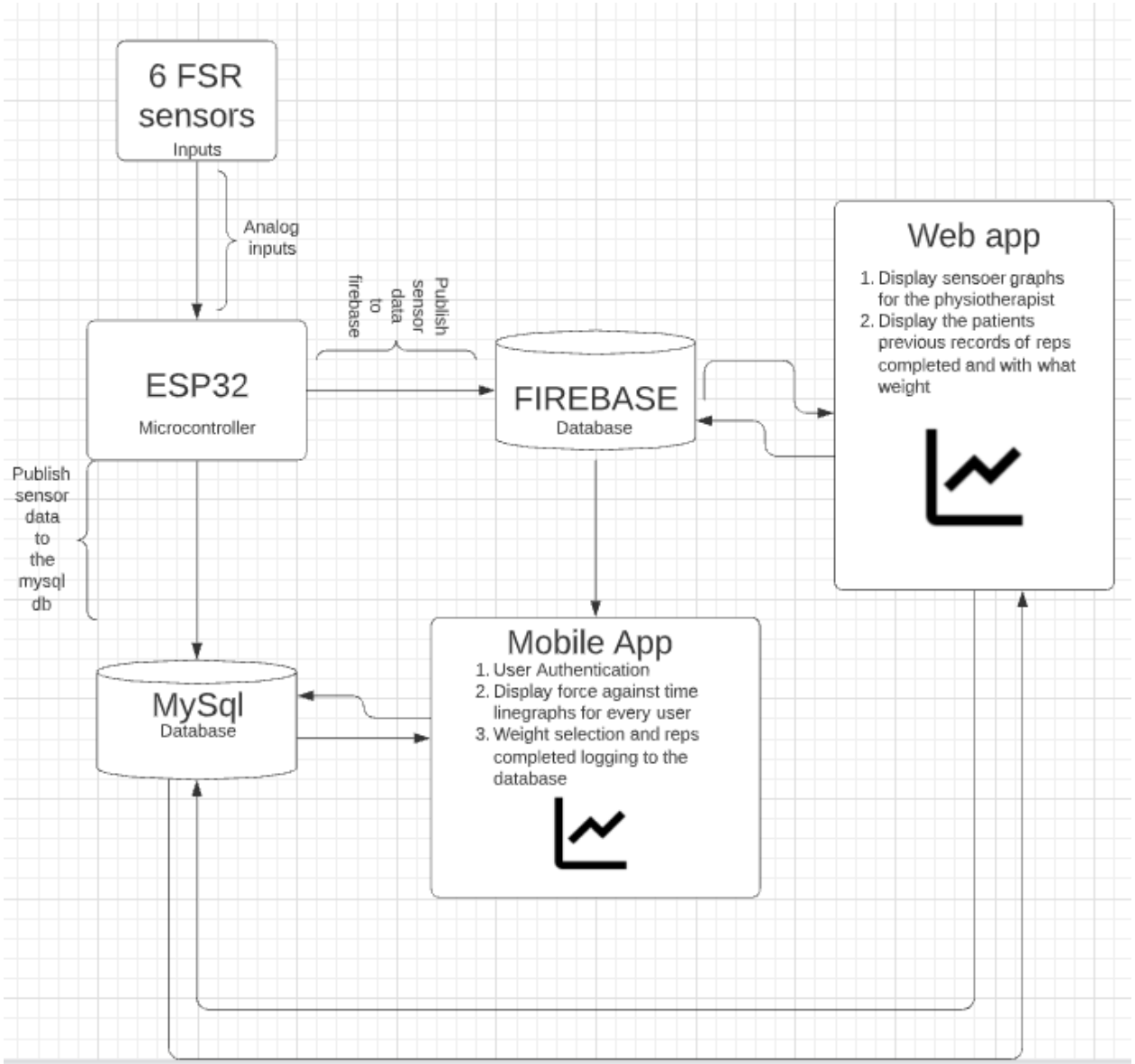
FSRESP32 system Documentation

System Architecture

The system has four major components :

1. The microcontroller and sensor unit
2. The databases
3. The Mobile app
4. The webapp



Methodology

1. Hardware and sensor unit

This is the unit responsible for logging the force applied by the patients as they do their reps. It is composed of the ESP32 microcontroller which is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. The board is responsible for reading the values of the six FSR402 force sensors that the patient uses and logging this data to both our databases one the firebase realtime database and the mysql database all which is done via wifi.

*insert hardware picture here

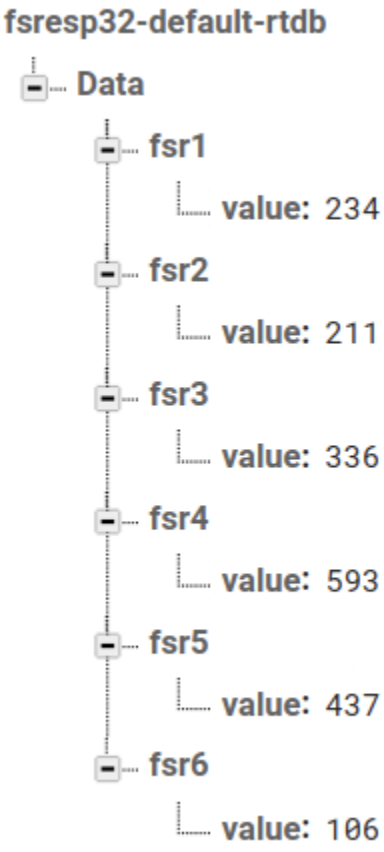
2. Databases

The system utilizes 2 databases as stated above

1. Firebase RealTime Database - This database stores realtime force sensor data from the hardware unit to json collections. The data is used to plot the real time graphs in the mobile app.

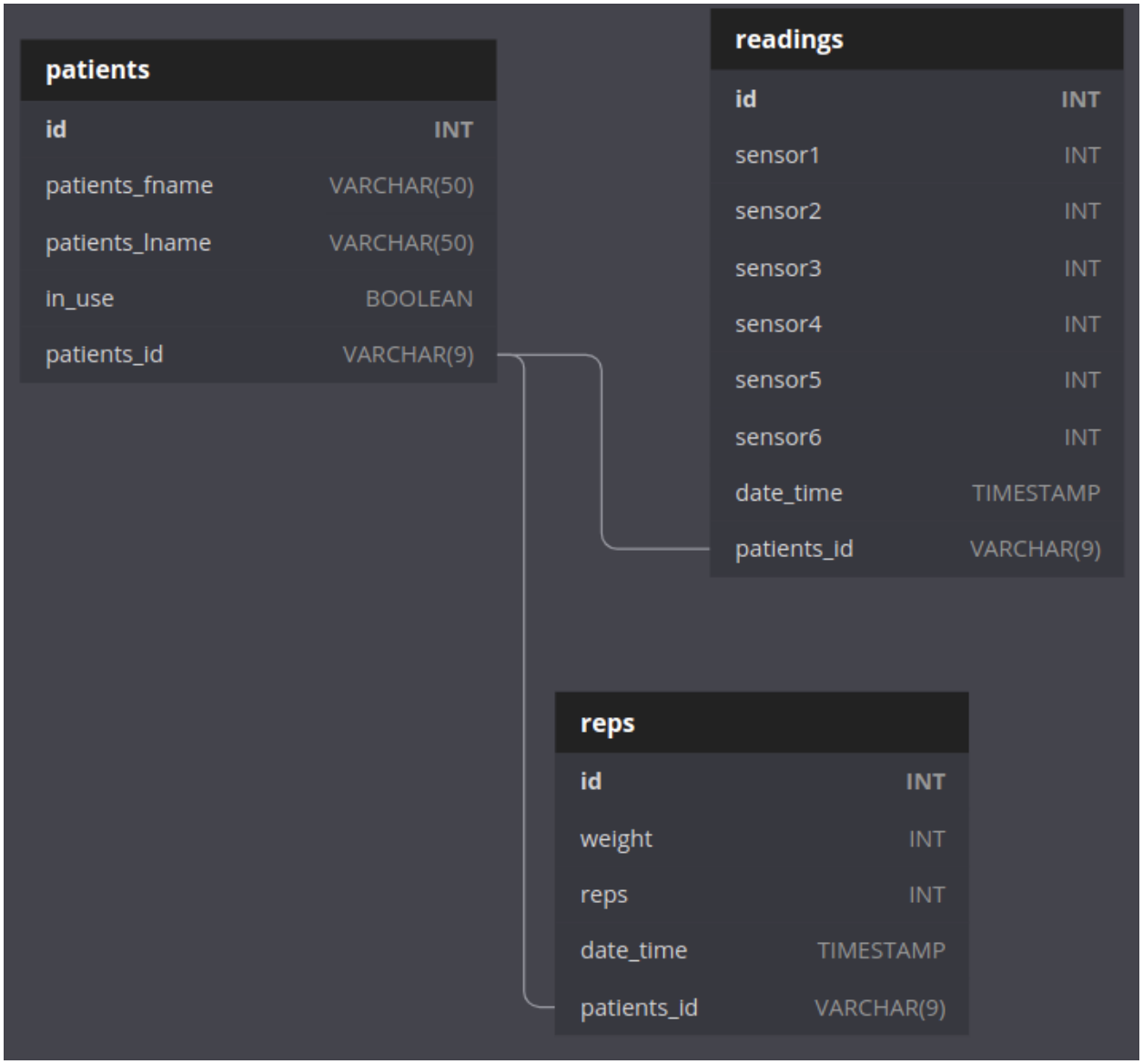
2. Mysql DataBase - This database consists of 3 tables: Patients, Readings and Reps. The Patients table stores persistent records of registered patients that is used in authentication on the mobile app, the reading table stores the force sensor payload from the hardware device and the reps table stores the rep data from the app and the weight selected for every patient

Schemas

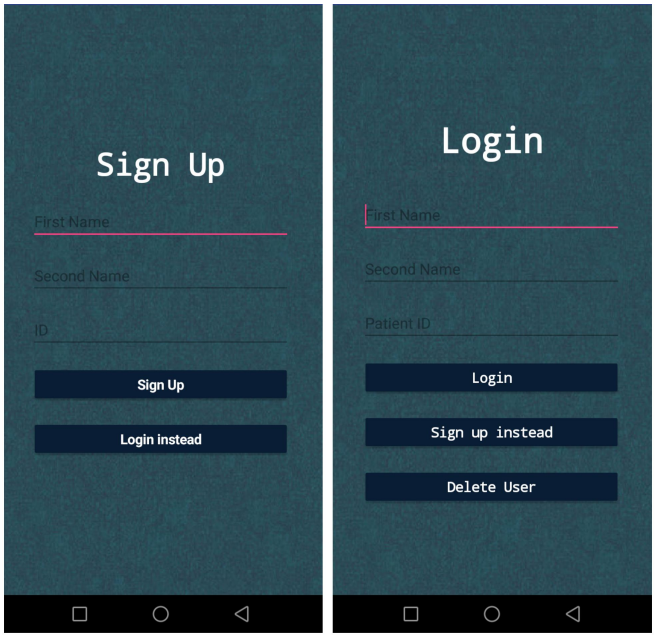


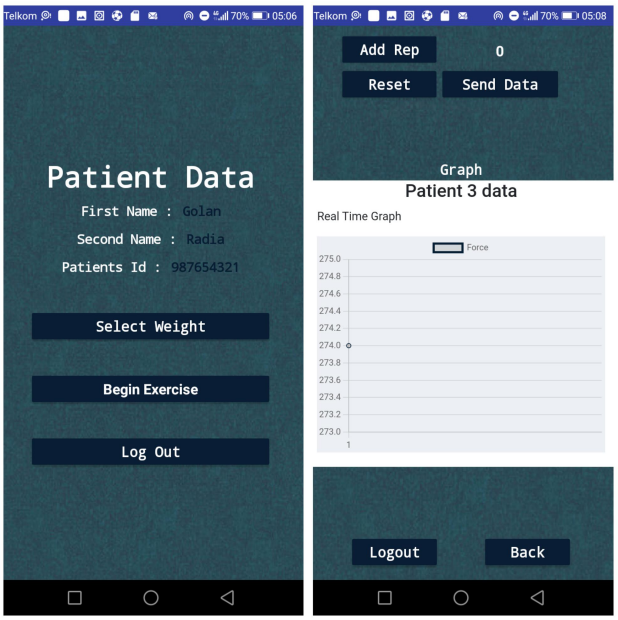
1. Firebase RTDB

2. MySQL database

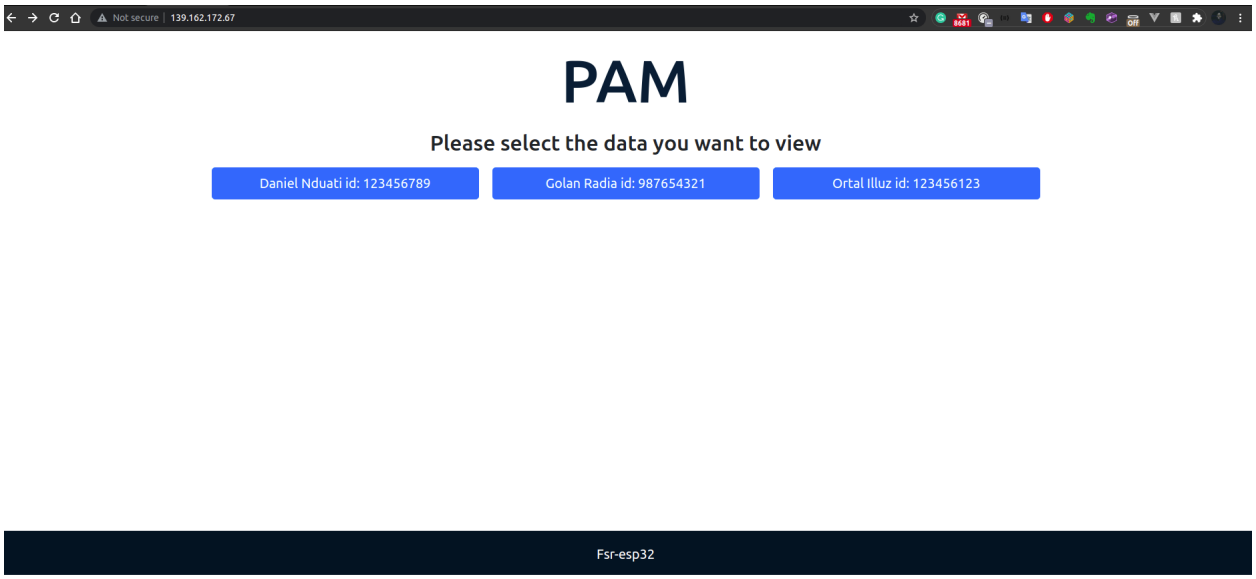


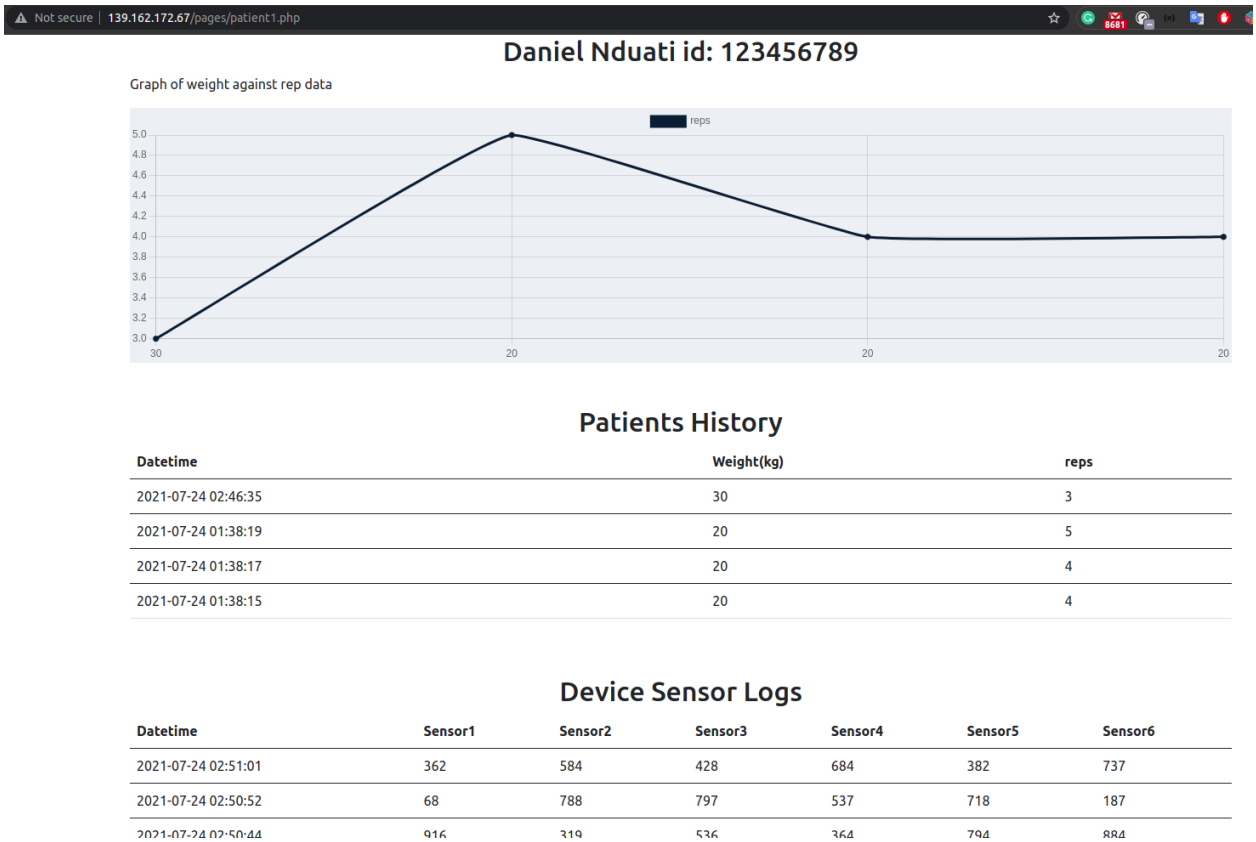
3. Mobile App


Sign Up
First Name
Second Name
ID
Sign Up
Login instead
Login
First Name
Second Name
Patient ID
Login
Sign up instead
Delete User


Telkom 70% 05:06
Patient Data
First Name : Golan
Second Name : Radia
Patients Id : 987654321
Select Weight
Begin Exercise
Log Out
Telkom 70% 05:08
Add Rep
0
Reset
Send Data
Graph
Patient 3 data
Real Time Graph
Force
275.0
274.8
274.6
274.4
274.2
274.0
273.8
273.6
273.4
273.2
273.0
1
Logout
Back

4. Web App


Not secure | 139.162.172.67
PAM
Please select the data you want to view
Daniel Nduati id: 123456789
Golan Radia id: 987654321
Ortal Illuz id: 123456123
Fsr-esp32


Not secure | 139.162.172.67/pages/patient1.php
Daniel Nduati id: 123456789
Graph of weight against rep data
reps
5.0
4.8
4.6
4.4
4.2
4.0
3.8
3.6
3.4
3.2
3.0
30
20
20
20
Patients History
Datetime | Weight(kg) | reps
2021-07-24 02:46:35 | 30 | 3
2021-07-24 01:38:19 | 20 | 5
2021-07-24 01:38:17 | 20 | 4
2021-07-24 01:38:15 | 20 | 4
Device Sensor Logs
Datetime | Sensor1 | Sensor2 | Sensor3 | Sensor4 | Sensor5 | Sensor6
2021-07-24 02:51:01 | 362 | 584 | 428 | 684 | 382 | 737
2021-07-24 02:50:52 | 68 | 788 | 797 | 537 | 718 | 187
2021-07-24 02:50:44 | 916 | 319 | 536 | 364 | 794 | 884

Appendix

Firmware Explanation

1. Line 1-6

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <FirebaseESP32.h>
#include <Arduino_JSON.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
```

Here we include the external arduino libraries used in the firmware this include the wifi library for connection of the esp32 to a wifi network the httpclient for making http requests to our server endpoints, Firebase esp32 for sending data to firebase real time database and the addons to help with firebase authentication

2. Line 9-10

```
const char* ssid  = "dan";
const char* password = "dandandandan";
```

This are your wifi creds

3. Line 14-18

```
const char* getidsUrl =
"http://139.162.172.67/externalcontrollers/getpatientids.php";
//add sensor data endpoint
const char* postDataUrl =
"http://139.162.172.67/externalcontrollers/addboardData.php?";
//patient id in use endpoint
const char* inUseUrl =
"http://139.162.172.67/externalcontrollers/getInUse.php";
```

This are our server endpoint that facilitate the interaction of the hardware and the server mysql database via php programming language

4. Line 22-37

```
const char* api_key = "AIzaSyBDCrjZasI5oLo5yxnuAwQdMdKU4qSc6bE";
const char* database_url = "https://fsresp32-default-rtdb.firebaseio.com";

String data_header = "sensorData";
```

```
// create a firebase data object
FirebaseData fbdo;
// Firebase Authentication Object
FirebaseAuth auth;
// Firebase configuration Object
FirebaseConfig config;
//database path
String database_path = "";
//firebase unique identifier
String fuid = "";
//device auth flag
bool isAuthenticated = false;
```

These lines of code are all related to firebase, here we declare the api key and url essential for authentication and also to create collections in our realtime database that will store our sensor values

5. Line 39-43

```
const int fsr_num = 6;
String fsr_sensors[fsr_num] = {"fsr1", "fsr2", "fsr3", "fsr4", "fsr5",
"fsr6"};
int fsrPins[fsr_num] = {36, 39, 34, 35, 32, 33}; //array of fsr sensors
int forceValues[fsr_num] = {0, 0, 0, 0, 0, 0}; //array of fsr sensor values
```

Line 39 declares the number of fsr sensors in use by the device, line 40 is an array of the names that will be used to create firebase collections for every sensor for value storage on the realtime database. Line 42 is another array of the analog pins of the esp32 microcontroller board where the analog output of the sensors are connected to

6. Line 46-47

```
String patient_id_in_use;//variabe to store the patient id currently using
the device(logged in the mobile app)
String patient_ids[6];//dynamic array to store patient ids obtained from
the server
```

These variables are used to store the patient_id that is currently using the app to dynamically determine whose data is entered into the database

7. Line 81-93

```
void wifiInit() {
  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
```

```
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();

}
```

This lines connect to the wifi

8. Line 95 -117

```
void getInUse() {
  patient_id_in_use = "";
  Serial.println("Getting the patient id currently in use");
  if (WiFi.status() == WL_CONNECTED) {
    String inUsePayload = httpGETRequest(inUseUrl);
    JSONVar inuseObject = JSON.parse(inUsePayload);
    if (JSON.typeof(inuseObject) == "undefined") {
      Serial.println("Parsing input failed!");
      return;
    }
    Serial.print("JSON response = ");
    Serial.println(inuseObject);
    if ((inuseObject[0]) != null) {
      Serial.print("The id in use is: ");
      Serial.println(inuseObject[0]);
      patient_id_in_use = inuseObject[0];
    }
    else {
      Serial.println("No user is currently logged into the app");
    }

  }
}
```

This function gets the patient_id that is currently logged into the mobile app

9. Line 120-141

```
//function to get all the patients currently registered in the server
void getPatientIds() {

  Serial.println("Getting patient ids from the server");
  if (WiFi.status() == WL_CONNECTED) {
```

```
    String idpayload = httpGETRequest(getidsUrl);
    //Serial.println(idpayload);
    JSONVar myObject = JSON.parse(idpayload);
    if (JSON.typeof(myObject) == "undefined") {
      Serial.println("Parsing input failed!");
      return;
    }

    Serial.print("JSON object = ");
    Serial.println(myObject);

    //get every id and store to an array
    for (int i = 0; i < myObject.length(); i++) {
      //Serial.println(myObject[i]);
      patient_ids[i] = myObject[i];
    }
  }
}
```

This function gets all the patient_ids from the server and is used to determine whether an id is valid before pushing data to the mysql database

10. Line 156-181

```
String httpGETRequest(const char* serverName) {
  WiFiClient client;
  HTTPClient http;

  // Your Domain name with URL path or IP address with path
  http.begin(client, serverName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "{}";

  if (httpResponseCode > 0) {
    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);
    payload = http.getString();
  }
  else {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
  }
```

```
  // Free resources
  http.end();

  return payload;
}
```

This function is used to make http get request to server endpoint when fetching in use patient id and all patient ids

11. Line 183-217

```
void getConfigs(void *ptr) {
  const char * firebase_url =
"https://fsresp32-default-rtdb.firebaseio.com/";
  const char * firebase_api_key =
"AIzaSyBDCrjZasI5oLo5yxnuAwQdMdKU4qSc6bE";
  const char * firebase_token = "pDsxpJsu33UWMM5gkxe3hQjacAAbvPvt3fDofYRa";
}

void firebaseInit() {
  // configure firebase API Key
  config.api_key = api_key;
  // configure firebase realtime database url
  config.database_url = database_url;
  // Enable WiFi reconnection
  Firebase.reconnectWiFi(true);
  Serial.println("------------------------------------");
  Serial.println("Sign up new user...");
  // Sign in to firebase Anonymously
  if (Firebase.signUp(&config, &auth, "", ""))
  {
    Serial.println("Success");
    isAuthenticated = true;
    // Set the database path where updates will be loaded for this device
    database_path = "/" + data_header;
    fuid = auth.token.uid.c_str();
  }
  else
  {
    Serial.printf("Failed, %s\n",
config.signer.signupError.message.c_str());
    isAuthenticated = false;
  }
  // Assign the callback function for the long running token generation
```

```
task, see addons/TokenHelper.h
  config.token_status_callback = tokenStatusCallback;
  // Initialise the firebase library
  Firebase.begin(&config, &auth);

}
```

The first function is used to store the firebase config to our realtime database, the second function is responsible for the authentication in order to add force sensor data to their respective firebase collections

12. Line 220-236

```
void sendToServer(String patient_id, int sensor1val, int sensor2val, int
sensor3val, int sensor4val, int sensor5val, int sensor6val) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(postDataUrl);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");
    //prepare the post data
    String httpRequestData = "patientid=" + patient_id + "&sensor1=" +
String(sensor1val) + "&sensor2=" + String(sensor2val) + "&sensor3=" +
String(sensor3val) + "&sensor4=" + String(sensor4val) + "&sensor5=" +
String(sensor5val) + "&sensor6=" + String(sensor6val) + "";
    //post data to the server
    int httpResponseCode = http.POST(httpRequestData);
    Serial.println(httpResponseCode);
    // Free http resource
    http.end();
  }
  else {
    Serial.println("WIFI disconnected!!");
  }
}
```

This is the function responsible for sending the force sensor data to the mysql database that resides in the server. It does this by making a post request to the addBoard data endpoint with the sensor payload and the patient_id in use

Line 237-270

```
void sendToFirebase() {
  // Check that the interval has elapsed before, device is authenticated
```

```
and the firebase service is ready.
  Serial.println("-----------------------------------");
  Serial.println("Sending Sensor data to firebase");
  // Specify the key value for our data and append it to our path
  String root_node = database_path;

  for (int i = 0; i < fsr_num; i++) {
    String sensor_node = root_node + "/" + fsr_sensors[i];
    String node = sensor_node + "/value";
    //get random force data
    int fsr = genRandomData();
    // Send the value our count to the firebase realtime database
    if (Firebase.set(fbdo, node.c_str(), fsr))
    {
      // Print firebase server response
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
      Serial.println("ETag: " + fbdo.ETag());
      Serial.print("VALUE: ");
      printResult(fbdo); //see addons/RTDBHelper.h
      Serial.println("-----------------------------------");
      Serial.println();
    }
    else
    {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
      Serial.println("-----------------------------------");
      Serial.println();
    }
  }
}
```

This function creates unique collection of each sensor in our real time database and adds their respective data under the created collection for use in plotting a real time graph on the mobile app