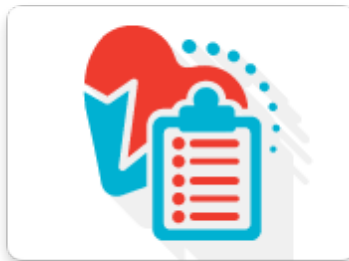


National College of Ireland
BSc in Computing
2016/2017

Name: Fintan Murphy
x12107395

x12107395@student.ncirl.ie

Degree: BSc (Hons) in Computing
Specialisation: Software Systems



Glucohealth

Technical Report



Table of Contents

1	Executive Summary	5
2	Introduction.....	6
2.1	Background.....	6
2.2	Aims	7
2.3	Technologies	8
3	System.....	10
3.1	Requirements.....	10
3.1.1	Functional requirements	10
3.1.2	Data requirements.....	11
3.1.3	User requirements.....	11
3.1.4	Environmental requirements.....	12
3.1.5	Usability requirements	12
3.1.6	Security Requirements	13
3.2	Design and Architecture.....	13
3.2.1	UML	14
3.3	Implementation	18
3.3.1	Bluetooth / Glucometer Device implementation	18
3.3.2	Authentication, firebase implementation	26
3.3.3	Alarm Reminder Implementation.....	28
3.3.4	Pedometer Implementation	29
3.3.5	Database Implementation	29
3.4	Graphical User Interface (GUI) Layout	33
3.5	Testing.....	36
3.6	Customer testing.....	37
3.7	Evaluation.....	38
4	Conclusions.....	39
5	Further development or research	40
6	References.....	41

7	Appendix	42
7.1	Project Proposal.....	42
7.1.1	Objectives	42
7.1.2	Technical Approach	44
7.1.3	Special resources required	46
7.1.4	Project Plan (original)	46
7.1.5	Project Plan (revised).....	47
7.1.6	Technical Details.....	48
7.1.7	Evaluation	49
7.2	Requirement Specification	50
7.2.1	Purpose.....	50
7.2.2	Project Scope	50
7.2.3	User Requirements Definition	51
7.2.4	Functional requirements	52
7.2.5	Requirement 1 <Representation of Correct Data>.....	55
7.2.6	Requirement 2 <Pedometer Functionality>	58
7.2.7	Requirement 3 <Alarm/Reminder Functionality>	61
7.2.8	Requirement 4 < Backup Storage functionality >	63
7.3	Non-Functional Requirements	66
7.3.1	Performance/Response time requirement.....	66
7.3.2	Availability requirement	66
7.3.3	Robust requirement	66
7.3.4	Usability requirement.....	67
7.3.5	Reliability requirement.....	67
7.3.6	Security requirement.....	67
7.3.7	Scalability/Concurrency requirement.....	68
7.3.8	Data Requirement	68
7.3.9	Interface requirements.....	68
7.4	Mockups	69
7.5	Application Programming Interfaces (API).....	73
7.6	System Architecture.....	73

7.7	System Evolution	73
7.8	Monthly Journal.....	74
7.9	Other Materials and Resources Used	86

1 Executive Summary

According to the International Diabetes Atlas (2013), approximately 225,840 people in Ireland are living with diabetes. It is estimated that by 2030 there would be 278,850 people living with the disease it is quite clear that this is a major problem. From a technical standpoint this project sets out to help those people living with the disease to manage it best they can with the aid of an android application and glucometer device.

The application has been developed in Android Studio, the application works alongside a Bluetooth glucometer device for recording users blood sugar readings. The main aim is to move away from the traditional pen to paper method that most diabetics use when recording their blood levels and to utilize all of the great advancements in technologies which have been available to us in recent years.

The user has the be ability to record their blood sugar readings as they normally would however utilizing Bluetooth the glucometer device that wirelessly transmits the data and stores it in the android application which utilises a real-time database for online backup. I believe that by combining the device that is used by all diabetics in conjunction with the application adds great value to the project and makes it differ widely from other similar applications available from the play store.

In addition to this the application the user with a reminder alarm functionality for reminders for medication or anything they desire. A pedometer which is also within application will ensure that the diabetic user will be getting enough exercise per day to ensure good blood circulation.

The main aim of this report is to fully document the process that was involved in the development of this project listing all of the necessary details including requirements, technologies, design, testing, evaluation and further developments.

2 Introduction

Glucohealth is a final year project to attain a bachelor's degree in computing it enables people living with diabetes to help manage their condition with the aid of user friendly application alongside a glucometer device in which all diabetics need to use in order to check their blood sugar levels.

After spending 4 years studying at NCI I have gained expertise and skills in many areas of computing and I have believed that this project was always fully achievable.

2.1 Background

According to Dr. James Gavin III who is a highly praised medical expert noted as leader in the diabetic field playing vital roles in certain studies and findings. According to a report, *"The importance of Monitoring Blood Glucose"* (2007) written by the above mentioned explains that the estimated number of people who are effected globally with diabetes is estimated to be more than 195 million people worldwide and he goes on to estimate that this figure will expect to reach 330 million by 2030.

He explains how the body automatically controls and monitors blood glucose for people with normal glucose tolerance. People that have an impeded glucose tolerance (i.e. a diabetic person) their body will have very little control of their blood sugar levels.

He then goes on to explain that proper self-observing of blood sugar levels can help patients and GPs or medical professionals to more easily adjust to certain therapies and assess how the therapy is monitored. The main benefits of monitoring blood means that patients can assess immediately the impact of a reading and can allow them take quick and adequate interventions to counter the high or low blood sugar levels that might occur. He recommends people living with diabetes to perform blood glucose monitoring at least 3 times per day.

After reading doctor Gavin's report and from firsthand experience myself living with a diabetic patient has given me the ambition to try and solve a real world problem or at least try and make it easier even if for just one patient to help them manage their disease. I have been thinking of this idea for a while and have always had the ambition to build an application that incorporated some kind of device that serves a real world important purpose which I believe this medical device offers this.

Currently on the play store there is a variety of diabetic applications many of the existing applications provide a journal like interface where the user can manually enter their blood glucose readings, food intake as well as providing useful information about the correct diet a diabetic person should be following. There are also many applications that provides static information about certain foods to avoid and best foods to eat to control diabetes.

While some of the applications are very useful and well put together there is very few that have the compatibility of having a glucometer device communicating seamlessly with the application to provide an effortless way of recording and saving the blood sugar readings. Also there was none to be found on the play store that implemented a pedometer for monitoring the step count and daily activity.

2.2 Aims

The aim of this project is to implement and create an android application for diabetics that will work alongside a Bluetooth glucose meter device. The diabetic user can provide a blood sample as they normally would on the glucometer the device. The glucometer device will be able to communicate with the android application and send the important data readings to the android application over a Bluetooth network. The application should look attractive and serve a valuable purpose to the diabetic user.

The important readings will include the blood sugar reading levels that is measured in mmol/L (millimoles per litre), the date of reading and the time of the reading. All of these readings will be sent by the glucometer device and retrieved, saved and stored in the application of the users' mobile device (phone/tablet device).

The application will also include a pedometer that will track the users' movements so that they can ensure they are walking/moving for the recommended 30 minutes per day which is very important for blood circulation especially for people with diabetes.

The application will also provide a reminder alarm so that the user can set reminders for taking blood sugar readings, diet, insulin, medication and whatever else may be necessary.

Overall in addition to all of the above mentioned the application should adhere to the Android developer coding guidelines and standards as explained in the Android Developer Documentation provided by Google.

2.3 Technologies

Android Studio SDK

Android programming was done using the Java programming language. This is framework and environment done all within Android Studio to provide all the functionalities and styling for the GUIs in the project. The android documentation provided by Google online is very extensive and well put together to help developers in the development stages across all areas as well as sample code to help get started all of which helped me greatly throughout the development lifecycle.

SQLite

For saving the data locally SQLite was used. This was used to store user glucose readings, step count, alarms, reminders, insulin readings, and heart rate readings.

Firebase

Firebase has been incorporated into the project at a late stage and had I known about this technology earlier I would have used many other features it has to offer, that being said I have listed some of them extra features in the further development section.

However, I did implement user authentication with the Google sign in API this gave me the framework for storing user details securely and in particular their glucose readings. This is done using Firebase Real-time Database which stores and synchronizes app data in milliseconds using JSON tree structures to store the data.

Firebase was used as an alternative to my original plan to build a RESTful web service and online MySQL database for online backup. Firebase is relatively new and is very popular amongst developers particularly mobile development.

Bluetooth SPP commands and glucometer API

The glucometer device uses SPP (Serial Port Profile) Bluetooth (v2.1) for the communication. This enables the device to communicate wirelessly with the application using the glucometer API and SPP commands.

3 System

3.1 Requirements

The requirements specification is one of the first activities to be carried out in the software development lifecycle - SDLC. For gathering the requirements I researched what would be needed to provide the users' with all of the functionalities and responsiveness required to develop a user friendly Android Application that served fit for the purposes that were proposed in the proposal. By analysing existing applications similar to that of the diabetic health application described here I was able to get inspiration for the various aspects and requirements needed for my application.

3.1.1 Functional requirements

This section identifies the functional requirements of the application and how each shall work. Each requirement is ranked in importance in terms of priority

1. **Application Connectivity with glucometer** – the glucometer and android application will be fully integrated together using the provided API that the glucometer provides.
2. **Representation of Correct Data** – The android application will present and display the correct glucose readings to the user, sent from the Bluetooth glucometer to the application.
3. **Bluetooth communication for Retrieval of data** - This means that the sending/retrieval of the data will be made possible between the glucometer device and application.
4. **Backup Storage functionality** – The application will provide the functionality for saving data both locally (for offline use) and backed up on an online resource (for when internet connectivity is available)
5. **Pedometer functionality** – The application shall provide the functionality for the use of the pedometer to track users' movements

6. **Alarm/reminder functionality** – the application shall provide a GUI for the user so that they can set/add alarms and reminders
7. **User authentication** – the application will provide firebase to enable users to sign up to the application

3.1.2 Data requirements

This section will describe the data requirements which were an essential part to implementing the functionalities mentioned above.

- **SQLite:** the application uses SQLite for locally storing the data on the device this will be created on application installation and will store the users' information such as blood readings, date/time, steps taken per day and any reminders the user will create
- **Firebase:** the application uses firebase for online backup using a real-time database and also used to allow users to sign up and sign in

3.1.3 User requirements

This section will describe the user requirements. These are the requirements that the user must have in order to utilize all of the applications functionalities.

- **Android Device:** the user must have access to an android capable device as this is the main operating system that the application is designed for
- **Bluetooth Device:** the mobile device that the application resides on must have Bluetooth enabled to use all of the functionalities
- **Glucometer Device:** the user must also have an entrahealth glucometer device in order to use the automatic transmission of their blood readings over Bluetooth
- **API:** the user must have an android device with an API level of at least 19 Android 4.4 (KitKat) and previous to this will not be supported however it should cover nearly 80% of users' devices

- **Internet Access:** Internet access used to perform backup to the firebase database and user authentication.

3.1.4 Environmental requirements

In this section the sufficient environmental requirements that were needed throughout the development of the application.

- **Laptop:** This is required to run Android Studio for the development and various tools
- **Android Device:** An android mobile device was used for the testing and deployment of the application
- **Glucometer Devices:** this was needed for testing the application for correct pairing and communication between two devices
- **Control test solutions:** This is used in conjunction with the glucometer device with 3 bottle solutions labelled “high”, “normal”, and “low”. All which mimic 3 blood sugar level readings. This will allow for testing and ensure that the correct data is being sent
- **Android Studio:** This was the main IDE used to code in Java programming language the diabetic application
- **Internet Access:** This was needed for reading various Google Android documentation and to set up and maintain Firebase from the console.
- **Adobe Photoshop:** this was used for designing any logos or graphics that will be used in the application

3.1.5 Usability requirements

This section discusses some of the non-functional usability requirements. These requirements refer mostly to the graphical user interfaces and how the user interacts with the application.

- **Easy to understand:** The application is easy to understand this is especially important for the GUIs and layouts are all user friendly

- **Ease of operation:** This means that all aspects of the application should perform how they should in an easy way for the user and also display any correct error messages and handle them accordingly so that the user is aware of any problems
- **Learnability:** The layout of the GUIs is easily understood as well as the navigation throughout the application so that the user can easily learn how to navigate throughout and perform all of the functionalities mentioned in the functional requirements
- **Attractiveness:** The application looks attractive to the end user so that they can easily comprehend buttons, functions and navigation of the application. Any text is bright and colorful for the user and easily read and understood.

3.1.6 Security Requirements

This section describes the security requirements which are important for the application.

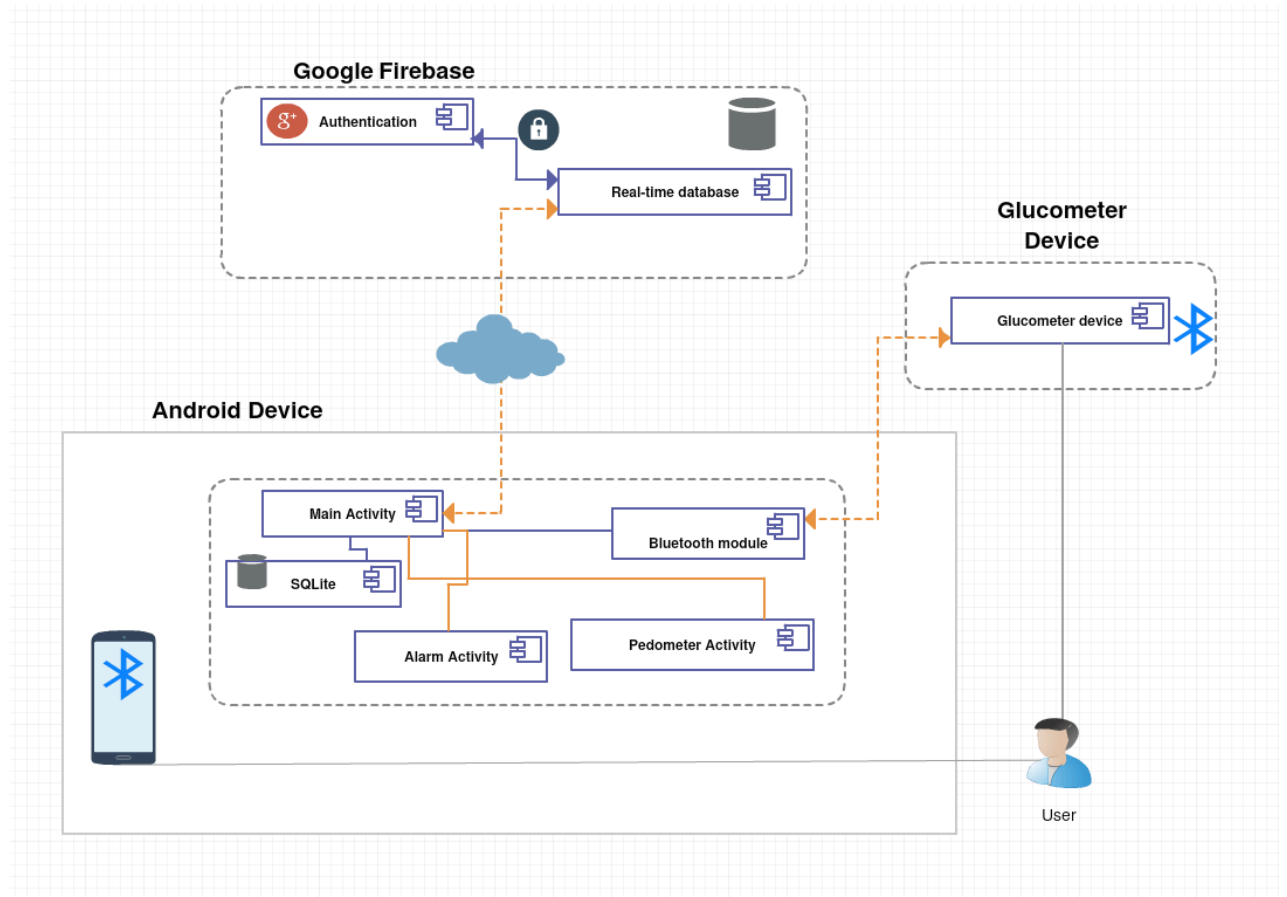
- **Authentication and password encryption:** using firebase which uses encryption for storing passwords.

3.2 Design and Architecture

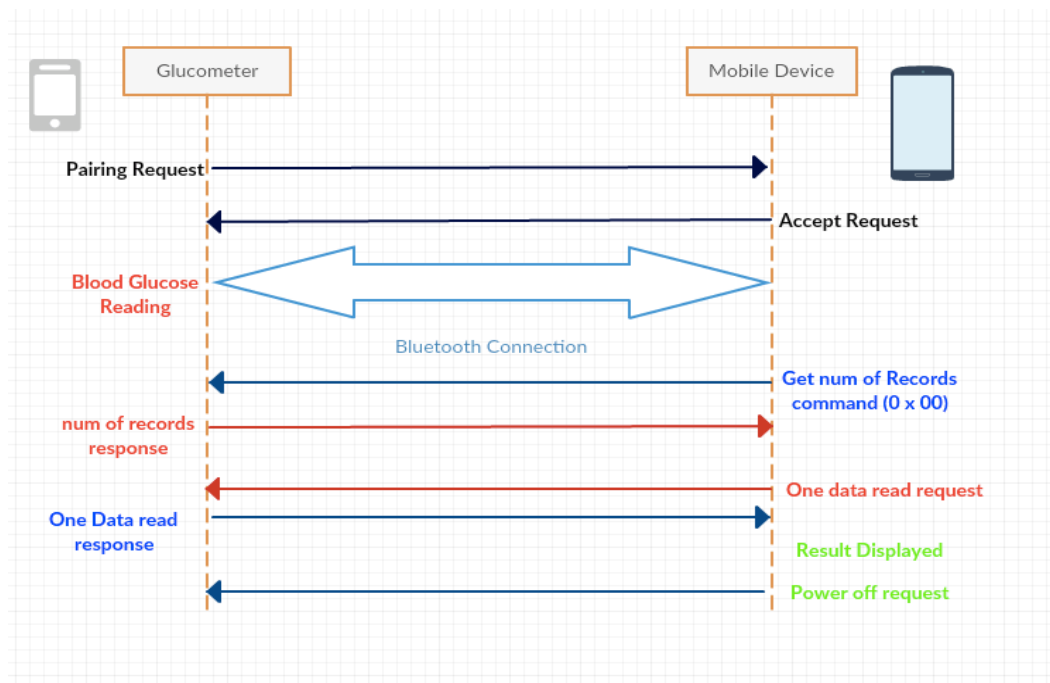
The following is a representation of the system architecture and how all of the components are integrated together. The application built in Android Studio using Java resides on the Android device which has the main activities such as the alarm and pedometer. The application interacts with the Google Firebase in the cloud as well as utilizing backend database SQLite for storage on the device. It also interacts with the Bluetooth Glucometer device all of these interactions are listed here in detail.

3.2.1 UML

System Architecture Diagram



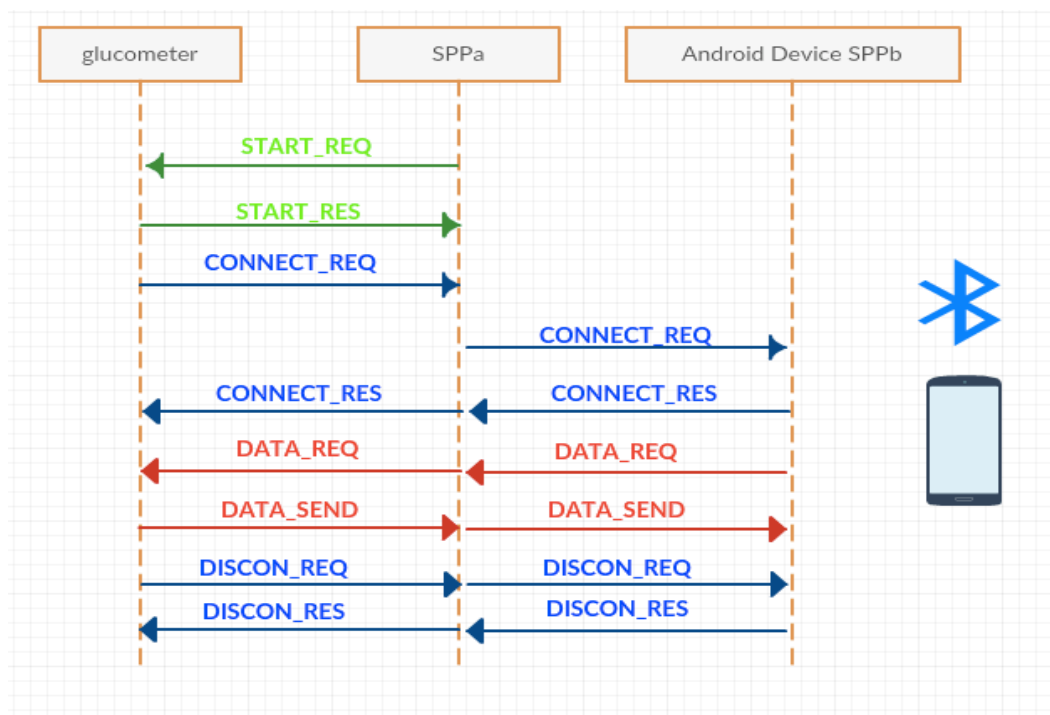
System Sequence Diagram 1 – Sending Glucose readings



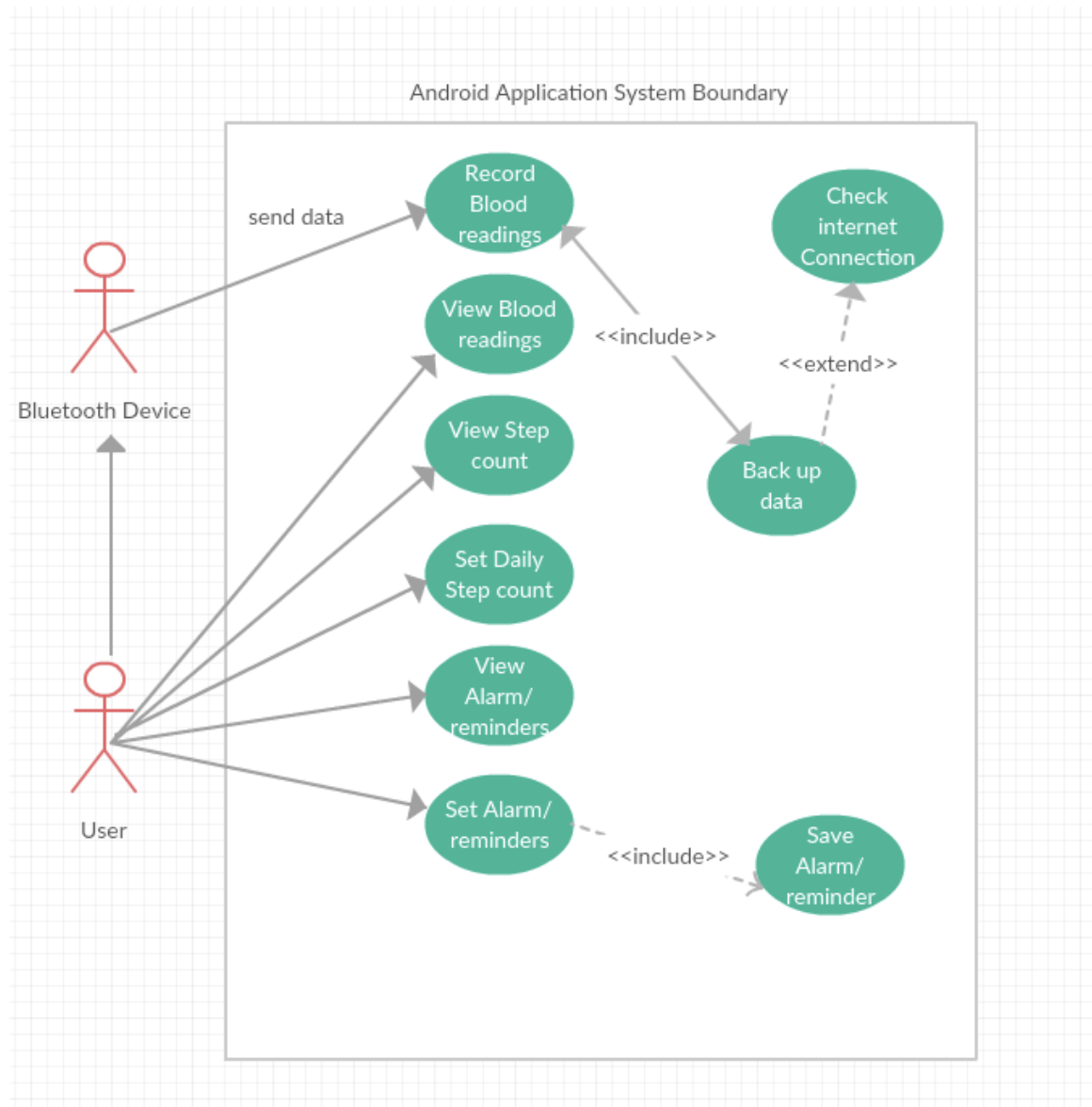
System Sequence Diagram 2 - Sending Glucose readings

SPPa: Initiator,





SPPb = Acceptor



Use Case Diagram



Flow

#	Glucometer		Android Application
1	Power off mode		
2	Insert the test strip into the glucometer		
3	When you place your blood on the test strip, the meter starts to measure the glucose level automatically		
4	The meter displays the result of the blood glucose reading		Execute the Bluetooth program
5	Press "DOWN"(Send)button		
6	Connection request		
7			Accept connection
8		<Bluetooth connection>	
9	Send command		Result display...
10			Disconnection
11	Power off automatically		Terminate the Bluetooth program

3.3 Implementation

In this section I will list code snippets of the implementation of the main algorithms and classes used in the code including databases used within the application.

3.3.1 Bluetooth / Glucometer Device implementation

Activities:

- BluetoothConnectListener (interface)
- BluetoothChatListner
- Record
- MainActivity
- DatabaseHandler

Within the onCreate() method inside the Main Activity (launch activity) an instance of a Bluetooth adapter is obtained the Bluetooth Adapter class is provided by Google in the Android Developer Documentation.

```
import android.bluetooth.BluetoothAdapter;
```

```
// Local Bluetooth adapter  
private BluetoothAdapter mBluetoothAdapter = null;
```

```
// Get Bluetooth adapter  
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();  
  
// If the adapter is null, then Bluetooth is not supported  
if (mBluetoothAdapter == null) {  
    Toast.makeText(this, "Bluetooth is not available", Toast.LENGTH_LONG).show();  
    finish();  
    return;  
}
```

The following code also resides in the MainActivity and is launched on start up the onStart() method checks if Bluetooth is turned on in the Android device if not it requests the user on the device if they would like to turn it on. Otherwise the setupChat() method is called.

```
@Override
public void onStart() {
    super.onStart();
    if(D) Log.e(TAG, "++ ON START ++");
    // If BT is not on, request that it be enabled.
    // setupChat() will then be called
    if (!BluetoothAdapter.isEnabled()) {
        Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
        // Otherwise, setup the chat session
    } else {
        if (mChatService == null) setupChat();
    }
}
```

From the setupChat() method the BluetoothChatService class is initialized to handle and perform the Bluetooth connections all messages from here are handled in the mHandler which gets information back from the BluetoothChatService Activity.

```
// Setting up Bluetooth Chat
private void setupChat() {
    Log.d(TAG, "setupChat()");

    // Initialize the array adapter for the conversation thread
    mConversationArrayAdapter = new ArrayAdapter<String>(this, R.layout.message);
    mConversationView = (ListView) findViewById(R.id.in);
    mConversationView.setAdapter(mConversationArrayAdapter);

    // Initialize the BluetoothChatService to perform bluetooth connections
    mChatService = new BluetoothChatService(this, mHandler);
}
```

Handler() Method for returning information here I am also storing the data to the firebase and local database in the MESSAGE_RESULT and MESSAGE_RESULT1 cases.

```
private final Handler mHandler = handleMessage(msg) -> {
    switch (msg.what) {
        case MESSAGE_STATE_CHANGE:
            if(D) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
            switch (msg.arg1) {
                case BluetoothChatService.STATE_CONNECTED:
                    mConversationArrayAdapter.clear();
                    break;
                case BluetoothChatService.STATE_LISTEN:
                case BluetoothChatService.STATE_NONE:
                    break;
            }
            break;
        case MESSAGE_WRITE:
            byte[] writeBuf = (byte[]) msg.obj;
            // construct a string from the buffer
            String writeMessage = new String(writeBuf);
            mConversationArrayAdapter.add("Me: " + writeMessage);
            break;
        case MESSAGE_READ:
            byte[] readBuf = (byte[]) msg.obj;
            // construct a string from the valid bytes in the buffer
            String readMessage = new String(readBuf, 0, msg.arg1);
            mConversationArrayAdapter.add(mConnectedDeviceName+": " + readMessage);
            break;
        case MESSAGE_DEVICE_NAME:
            // save the connected device's name
            Toast.makeText(getApplicationContext(), "Connected to Glucometer Device", Toast.LENGTH_SHORT).show();
            break;
        case MESSAGE_RESULT:
            //Get Glucose reading from device and save to SQLite Database
            String sugar_level_value = msg.getData().getString("result");
            final DatabaseHandler db = new DatabaseHandler(
                getApplicationContext());
            db.addSugar(sugar_level_value);
            break;
        case MESSAGE_RESULT1:
            // Get Glucose reading from device and get current user in Firebase and save to firebase realtime database
            mFirebaseUser = mFirebaseAuth.getCurrentUser();
            String sugar_level_value1 = msg.getData().getString("result1");
            SugarValue item = new SugarValue(sugar_level_value1);
            mUserId = mFirebaseUser.getId();
            mFirebaseDatabaseReference.child("users").child(mUserId).child("readings").push().setValue(item);
            Toast.makeText(getApplicationContext(), "Success!" + "\n" + msg.getData().getString("result1"), Toast.LENGTH_LONG).show();
            break;
        case MESSAGE_NOT_CONNECTED:
            Toast.makeText(getApplicationContext(), "Device not connected!", Toast.LENGTH_SHORT).show();
            break;
        case MESSAGE_TOAST:
            Toast.makeText(getApplicationContext(), msg.getData().getString(Toast),
                Toast.LENGTH_SHORT).show();
            break;
    }
}
```

The BluetoothChatService class does the heavy lifting for the Bluetooth requirements in setting up connections as well as managing them with the devices. It utilizes 3 threads for the incoming connections, connecting of devices and data transmissions for connected devices. The following code snippet is the thread used in the application for listening for the connections of devices.

```
private class AcceptThread extends Thread {
    // The local server socket
    private final BluetoothServerSocket mmServerSocket;
    private String mSocketType;

    public AcceptThread(boolean secure) {
        BluetoothServerSocket tmp = null;
        mSocketType = secure ? "Secure":"Insecure";

        // Create a new listening server socket
        try {
            if (secure) {
                tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE,
                    MY_UUID_SECURE);
            } else {
                tmp = mAdapter.listenUsingInsecureRfcommWithServiceRecord(
                    NAME_INSECURE, MY_UUID_INSECURE);
            }
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "listen() failed", e);
        }
        mmServerSocket = tmp;
    }

    public void run() {
        if (D) Log.d(TAG, "Socket Type: " + mSocketType +
            "BEGIN mAcceptThread" + this);
        setName("AcceptThread" + mSocketType);

        BluetoothSocket socket = null;

        // Listen to the server socket if we're not connected
        while (mState != STATE_CONNECTED) {
            try {
                // This is a blocking call and will only return on a
                // successful connection or an exception
                socket = mmServerSocket.accept();
            } catch (IOException e) {
                Log.e(TAG, "Socket Type: " + mSocketType + "accept() failed", e);
                break;
            }

            // If a connection was accepted
            if (socket != null) {
                synchronized (BluetoothChatService.this) {
                    switch (mState) {
                        case STATE_LISTEN:
                        case STATE_CONNECTING:
                            // Situation normal. Start the connected thread.
                            Log.d(TAG, socket.getRemoteDevice().getName()+" Socket Type: "+mSocketType);
                    }
                }
            }
        }
    }
}
```

```

        Log.d(TAG, socket.getRemoteDevice().getName() + " Socket Type: " + mSocketType);
        connected(socket, socket.getRemoteDevice(),
            mSocketType);
        break;
    case STATE_NONE:
    case STATE_CONNECTED:
        // Either not ready or already connected. Terminate new socket.
        try {
            socket.close();
        } catch (IOException e) {
            Log.e(TAG, "Could not close unwanted socket", e);
        }
        break;
    }
}
}
}
if (D) Log.i(TAG, "END mAcceptThread, socket Type: " + mSocketType);
}

public void cancel() {
    if (D) Log.d(TAG, "Socket Type: " + mSocketType + " :cancel " + this);
    try {
        mmServerSocket.close();
    } catch (IOException e) {
        Log.e(TAG, "Socket Type: " + mSocketType + " :close() of server failed", e);
    }
}
}
}

```

Reading the Data from the Glucometer Device Algorithms

When connected to application via Bluetooth by pressing the down button on the device the user proceeds to select ok to get results from the device this calls the GetGlucodata() method which opens a thread for getting the readings and calls the readDataFromMeter() method. The readDataFromMeter () method was the biggest challenge as this is what does the actual readings from the device the data is sent in bytes in a packet frame using SPP Protocols which was all new to me so was a huge learning curve. It uses checksum algorithms to check the packets sent. When I first implemented the code it kept on reading all of the records and getting them at once after reading the devices documentation for some time I managed to utilize the proper command message to only retrieve one record. Once the record is read I pass the message to the Handler in the Main Activity mentioned above. I use a Bundle to pass the messages using methods created in a class called Record this does some minor adjustments to readings and getters and setters are called here. However, since the glucometer device is from USA where they use milligrams per deciliter and here in Europe

we measure blood readings as milli-moles per litre so when result is sent its sent as mg/dl so I use this small formula to convert it to mmol/L. This is done before the reading is displayed to the user or written to a database.

Converting from mg/dl to mmol/l

```
@Override
public String toString() {

    double roundOff = result/18.02;
    DecimalFormat df = new DecimalFormat("#.##");
    String date = day + "/" + mon + "/" + year;
    String time = hour + ":" + min;

    return "Sugar Reading: " + df.format(roundOff) + " mmol/L " + "\nDate: " + date + "\nTime: " + time;
}

public String value(){
```

Packet Frame used by (Glucometer device)

Name	STX	SIZE	~SIZE	COMMAND	CKL	CKH
Bytes	1	1	1	1	1	1
Value	0x80	N+1	~(N+1)			
	start	command	Command +DATA		Check- sum(low)	Check- sum(high)

STX: Start of transmission

SIZE: Command + DATA Size

~SIZE: "not" bit-operation for size confirmation.

COMMAND: 0x01 (READ ONE DATA MEASUREMENT)

DATA: 0x01 glucose data, RESULT [10bit], YEAR [7bit], MONTH [4bit], DAY [5bit], HOUR [5bit], MINUTES [6bit]

CKL: checksum low $\sim(\text{sc_sendBuf1}[0] \wedge \text{sc_sendBuf1}[2])$

CKH: checksum high $\sim(\text{sc_sendBuf1}[1] \wedge \text{sc_sendBuf1}[3])$

Example Reading:

Glucose Result: 124mg/dL

DATA	BIT	DECIMAL	BINARY
RESULT	10	124	0000101

Code implementation for mentioned above reading from glucometer

```
// Reading data from Glucometer
public boolean readDataFromMeter() throws Exception {
    Message msg;
    Message msg1;
    Bundle bundle;
    Bundle bundle1;
    Record record;
    Record lastRecord = null;

    int recordsCount = 0;

    byte sc_sendBuf1[] = new byte[6];
    byte sc_receiveBuf1[] = new byte[7];

    // get the number of records
    sc_sendBuf1[0] = STX; //stx
    sc_sendBuf1[1] = (byte)0x01; //size
    sc_sendBuf1[2] = (byte) ~ ( sc_sendBuf1[1] ); //~size
    sc_sendBuf1[3] = CMD_NUM_RECORDS; //command
    sc_sendBuf1[4] = (byte) ~(sc_sendBuf1[0] ^ sc_sendBuf1[2]); //checksum L
    sc_sendBuf1[5] = (byte) ~(sc_sendBuf1[1] ^ sc_sendBuf1[3]); // checksum H
    try {
        sc_receiveBuf1 = dialogWithMeter(sc_sendBuf1, 7);
    }
    catch (Exception e)
    {
        closeConnection();
        //Logger.printException(e);
        return false;
    }
    num_records = sc_receiveBuf1[4];
    Log.d(TAG, "NUM_REC = " + num_records);
    bundle = new Bundle();
    bundle.putInt("progressMax", num_records);

    byte sc_sendBuf2[] = new byte[7];
    byte sc_receiveBuf2[] = new byte[13];

    /*
    * one record request, 0x01
    */
    sc_sendBuf2[0] = STX; //stx
    sc_sendBuf2[1] = (byte)0x02; //size
    sc_sendBuf2[2] = (byte) ~ ( sc_sendBuf2[1] ); //~size
    sc_sendBuf2[3] = CMD_RECORD; //command
    sc_sendBuf2[4] = (byte) recordsCount; //data
    sc_sendBuf2[5] = (byte) ~(sc_sendBuf2[0] ^ sc_sendBuf2[2] ^ sc_sendBuf2[4]); //checksum L
    sc_sendBuf2[6] = (byte) ~(sc_sendBuf2[1] ^ sc_sendBuf2[3]); // checksum H
```



```

try{
    sc_receiveBuf2 = dialogWithMeter(sc_sendBuf2, 13);
}
catch(Exception e)
{
    closeConnection();
    return false;
}

record = new Record(sc_receiveBuf2, 5);

if ( record.getResult() == 0 ) {
    if ( recordsCount == 0 ) {
        Log.d(TAG,"Error, meter records not found!");
    }
    /* No data in meter */
} else {
    if ( record.equals(lastRecord) ) {
        //lastRecord = record; /* the same record, try again */
    } else {
        Log.d(TAG,"Record " + Integer.toString(recordsCount));
        // Send the reading results to the UI Activity
        //SQLite
        msg = mHandler.obtainMessage(MainActivity.MESSAGE_RESULT);
        //Firebase
        msg1 = mHandler.obtainMessage(MainActivity.MESSAGE_RESULT1);
        //SQLite
        bundle = new Bundle();
        //Firebase
        bundle1 = new Bundle();
        //SQLite
        bundle.putString("result", record.value());
        //Firebase
        bundle1.putString("result1", record.toString());
        //SQLite
        msg.setData(bundle);
        //Firebase
        msg1.setData(bundle1);
        //SQLite
        mHandler.sendMessage(msg);
        //Firebase
        mHandler.sendMessage(msg1);
        recordsCount++;
    }
}

Thread.sleep(100);

```

After reading call closeConnection(); (switches device into off state)

```

        recordsCount++;
    }
    Thread.sleep(100);
}

closeConnection(); // close connection when device is no longer connected
if ( recordsCount == 0 ) {
    return false;
}
Log.d(TAG,"Read " + recordsCount + " records");
return true;
}

```

3.3.2 Authentication, firebase implementation

Activities:

- GlucoseListFirebase
- DatabaseHandler
- LoginActivity
- SignUpActivity
- ResetPasswordActivity
- MainActivity

The following is a small snippet of code that shows the use of the Google sign up API. This activity is called by the MainActivity if the user is not signed in imports and flow of code can be seen below.

```
import com.google.android.gms.auth.api.Auth;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.auth.api.signin.GoogleSignInResult;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.GoogleApiClient;
```

obtain an instance of the sign in api options and GoogleApiClient

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
mGoogleApiClient = new GoogleApiClient.Builder(this)
    .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
    .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
    .build();
```

Obtaining Firebase instance

```
// Initialize FirebaseAuth
mFirebaseAuth = FirebaseAuth.getInstance();
```

The following code checks if google sign in button has been pressed if so the sign in method is called and a google sign in dialog gets passed to the intent for the user to sign in also the onActivityResult is called. In the onActivityResult method if sign in is success I hide the buttons and edit text boxes and display a progress bar. This makes the sign in look and feel nicer for the user.

```

@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.sign_in_button:
            signIn();
            break;
        default:
            return;
    }
}

private void signIn() {
    Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        if (result.isSuccess()) {
            View a = findViewById(R.id.btn_reset_password);
            a.setVisibility(View.GONE);
            View b = findViewById(R.id.btn_signup);
            b.setVisibility(View.GONE);
            View c = findViewById(R.id.sign_in_button);
            c.setVisibility(View.GONE);
            View d = findViewById(R.id.btn_login);
            d.setVisibility(View.GONE);
            View e = findViewById(R.id.password);
            e.setVisibility(View.GONE);
            EditText f = (EditText) findViewById(R.id.email);
            f.setVisibility(View.GONE);
            EditText g = (EditText) findViewById(R.id.password);
            g.setVisibility(View.GONE);
            progressBar.setVisibility(View.VISIBLE);
            // Google Sign In was successful, authenticate with Firebase
            GoogleSignInAccount account = result.getSignInAccount();
            firebaseAuthWithGoogle(account);
        } else {
            // Google Sign In failed
            Log.e(TAG, "Google Sign In failed.");
        }
    }
}

```

Another thing that should be mentioned about Firebase is that once the user is signed in the user id is unique meaning that any glucose readings will be stored in the firebase database and retrieved upon signing in so that each users' glucose readings are backed up in case of failure or uninstallation of application.

3.3.3 Alarm Reminder Implementation

Activities:

- AlarmReminder Activity
- AlarmReminder Service
- AlarmReceiver
- AlarmAdapter
- AlarmSetter
- CreateAlert
- CreateNote
- MainActivity

The implementation of the Alarm Reminder feature was done referring to the Google Developer site which provides the proper documentation how to implement various components.

One challenge that was specific to diabetics was the ability to reset the time for hourly, daily, weekly and monthly. The following snippet of code shows how I overcame this in the onReceive() method inside the AlarmReceiver class which extends WakefulBroadcastReceiver as referred in the Android Developer documentation. Here I use a cursor to get the Reoccurrence string and check if the string matches HOURLY, DAILY, WEEKLY, MONTHLY. I then add the appropriate time using the java.util.Calendar package depending on the condition then update the time in the database.

```
// database instance
DatabaseHandler database = new DatabaseHandler(context);
// Cursor to getItem id
Cursor cursor = database.getItem(id);
// Move to first
cursor.moveToFirst();

// get the recurrence which is set on create in CreateAlert method HOURLY, DAILY
int occurrence = cursor.getInt(cursor.getColumnIndex(DatabaseHandler.KEY_RECURRENCE));
Calendar time = Calendar.getInstance();
time.setTimeInMillis(cursor.getLong(cursor.getColumnIndex(DatabaseHandler.KEY_TIME)));

if (occurrence > 0) {
    if (occurrence == HOURLY) {
        time.add(Calendar.HOUR, 1);
    } else if (occurrence == DAILY) {
        time.add(Calendar.DATE, 1);
    } else if (occurrence == WEEKLY) {
        time.add(Calendar.DATE, 7);
    } else if (occurrence == MONTHLY) {
        time.add(Calendar.MONTH, 1);
    }
}
database.updateTime(id, time.getTimeInMillis());
Intent setAlarm = new Intent(context, AlarmReminderService.class);
setAlarm.putExtra("id", id);
setAlarm.setAction(AlarmReminderService.CREATE);
context.startService(setAlarm);
}
```

3.3.4 Pedometer Implementation

The Pedometer functionality code was not written by myself it was an open source library found online that can be implemented into any android device that has a hardware step sensor. I have managed to successfully implement it and it is fully integrated and functional within my application. Had given more time I would have liked to adopt this and make it more my own and modify certain aspects.

3.3.5 Database Implementation

The following is the SQLite database implementation one problem I encountered when saving the Glucose readings time as the current time was it kept setting an hour ahead for some reason to overcome this I used `KEY_DATE + " DATETIME DEFAULT (datetime('now','localtime'))"`. This can be seen in snippet below note that I created tables for pulse, heartrate, and insulin also.

Creating the necessary tables

```
// Creating Tables
@Override
public void onCreate(final SQLiteDatabase db) {
    final String CREATE_PULSE_TABLE = "CREATE TABLE " + TABLE_PULSE + "("
        + KEY_PULSE_ID + " INTEGER PRIMARY KEY," + KEY_DATE + " DATETIME DEFAULT CURRENT_TIMESTAMP," + KEY_RATE + " TEXT"
        + ")";
    db.execSQL(CREATE_PULSE_TABLE);

    final String CREATE_WEIGHT_TABLE = "CREATE TABLE " + TABLE_WEIGHT + "("
        + KEY_WEIGHT_ID + " INTEGER PRIMARY KEY," + KEY_DATE + " DATETIME DEFAULT CURRENT_TIMESTAMP," + KEY_WEIGHT + " TEXT"
        + ")";
    db.execSQL(CREATE_WEIGHT_TABLE);

    final String CREATE_SUGAR_TABLE = "CREATE TABLE " + TABLE_SUGAR + "("
        + KEY_SUGAR_ID + " INTEGER PRIMARY KEY," + KEY_DATE + " DATETIME DEFAULT (datetime('now','localtime'))," + KEY_SUGAR + " TEXT"
        + ")";
    db.execSQL(CREATE_SUGAR_TABLE);

    final String CREATE_INSULIN_TABLE = "CREATE TABLE " + TABLE_INSULIN + "("
        + KEY_INSULIN_ID + " INTEGER PRIMARY KEY," + KEY_DATE + " DATETIME DEFAULT CURRENT_TIMESTAMP," + KEY_INSULIN + " TEXT"
        + ")";
    db.execSQL(CREATE_INSULIN_TABLE);

    final String CREATE_ALARM_TABLE = "CREATE TABLE " + TABLE_ALARM + "("
        + KEY_ALARM_ID + " INTEGER PRIMARY KEY, " + KEY_DESCRIPTION + " TEXT, " + KEY_NAME + " TEXT, " + KEY_INFORMATION + " TEXT, "
        + KEY_RECURRENCE + " TEXT, " + KEY_TIME + " LONG"
        + ")";
    db.execSQL(CREATE_ALARM_TABLE);

    db.execSQL("CREATE TABLE " + TABLE_STEPS + " (date INTEGER, steps INTEGER)");
}
```

Upgrading the necessary tables

```
// Upgrading database
@Override
public void onUpgrade(final SQLiteDatabase db, final int oldVersion, final int newVersion) {

    if (oldVersion == 1) {
        // drop PRIMARY KEY constraint
        db.execSQL("CREATE TABLE " + TABLE_STEPS + "2 (date INTEGER, steps INTEGER)");
        db.execSQL("INSERT INTO " + TABLE_STEPS + "2 (date, steps) SELECT date, steps FROM " +
            TABLE_STEPS);
        db.execSQL("DROP TABLE " + TABLE_STEPS);
        db.execSQL("ALTER TABLE " + TABLE_STEPS + "2 RENAME TO " + TABLE_STEPS + "");
    }

    // Drop older table if existed
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_PULSE);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_WEIGHT);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_SUGAR);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_INSULIN);
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_ALARM);

    // Create tables again
    onCreate(db);
}
```

Storing the sugar glucose readings locally in SQLite I replicated this for storing and recording insulin, heartrate and weight

```
/**
 * Storing sugar details in database
 */
public void addSugar(final String sugar) {
    final SQLiteDatabase db = this.getWritableDatabase();

    final ContentValues values = new ContentValues();
    values.put(KEY_SUGAR, sugar); //sugar

    // Inserting Row
    db.insert(TABLE_SUGAR, null, values);
    db.close(); // Closing database connection
}
```

Storing and updating alarms

```
/**
 * Storing and updating details in Alarms
 * */
public long addAlarm(String name, String information, long time, int recurrence) {
    SQLiteDatabase db = getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_DESCRIPTION, "reminder");
    values.put(KEY_NAME, name);
    values.put(KEY_INFORMATION, information);
    values.put(KEY_RECURRENCE, recurrence);
    values.put(KEY_TIME, time);
    return db.insert(TABLE_ALARM, null, values);
}

public boolean updateAlarm(Integer id, String name, String information, long time, int recurrence) {
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(KEY_NAME, name);
    values.put(KEY_INFORMATION, information);
    values.put(KEY_RECURRENCE, recurrence);
    values.put(KEY_TIME, time);
    db.update(TABLE_ALARM, values, KEY_ALARM_ID + " = ? ",
        new String[]{Integer.toString(id)});
    return true;
}
```

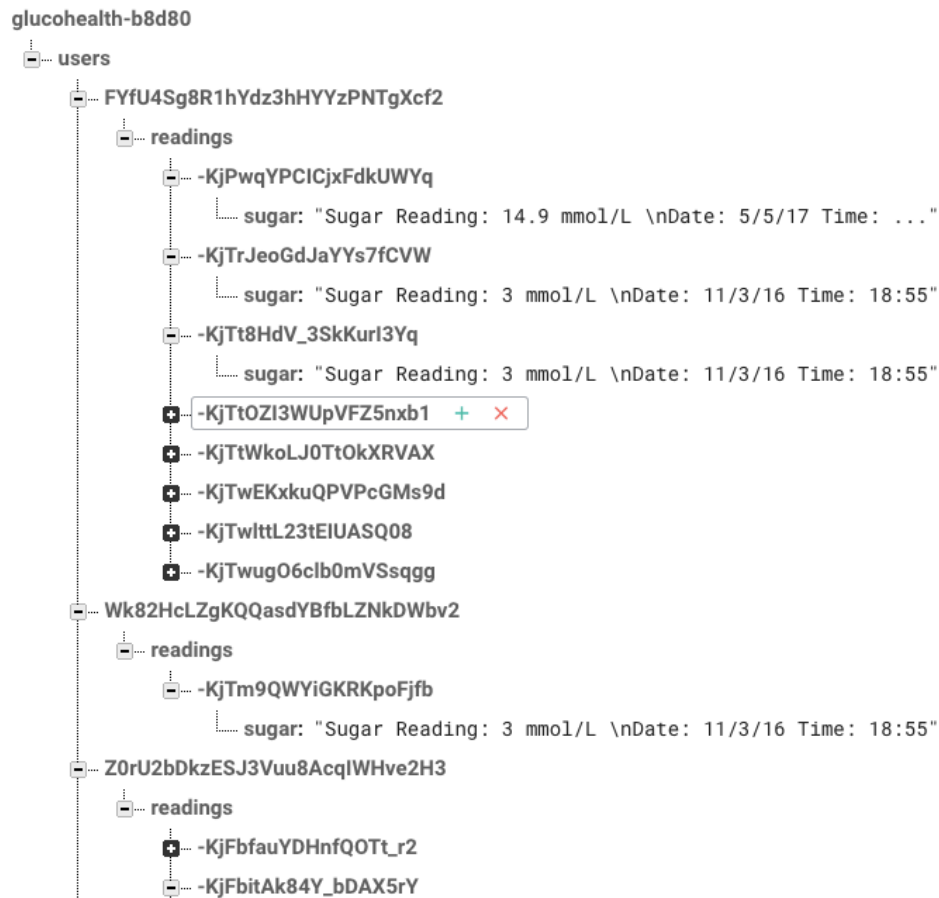
Firestore Implementation

The firestore implementation is done by registering to firebase and dropping the necessary JSON file structure and configuration files in the Build.gradle within the android application. Once this was done I could enable the authentication in the console and set my rules for the JSON tree structures for saving the users and data.

Registered Authenticated users

Authentication					WEB SETUP ?
USERS					SIGN-IN METHOD EMAIL TEMPLATES
<input type="text" value="Search by email address or user UID"/>					<button>ADD USER</button>
Email	Providers	Created	Signed In	User UID ↑	
fmurphynci2016@gmail.com		May 6, 2017	May 6, 2017	2WRAkqeWrCZtLISD46wNrlqD4jt2	
finono.1@hotmail.com		May 6, 2017	May 6, 2017	FYfU4Sg8R1hYdz3hHYyzPNTgXcf2	
scoobymurphy14@gmail.com		May 6, 2017	May 6, 2017	Wk82HclZgKQqasdyBfbLZNkDWb..	
fmurphynci2017@gmail.com		May 3, 2017	May 7, 2017	Z0rU2bDkzESJ3Vuu8AcqlWHve2H3	
fintanmurphy10@gmail.com		May 2, 2017	May 7, 2017	oh34Pg0suldnIjimWdkSdHGrlf2	
olive.murphy@aviva.com		May 6, 2017	May 6, 2017	zWZqYc1jpyPdIAYCjuiMURXK2PF2	
Rows per page: 50					1-6 of 6 < >

Data being stored about users' glucose readings in JSON trees

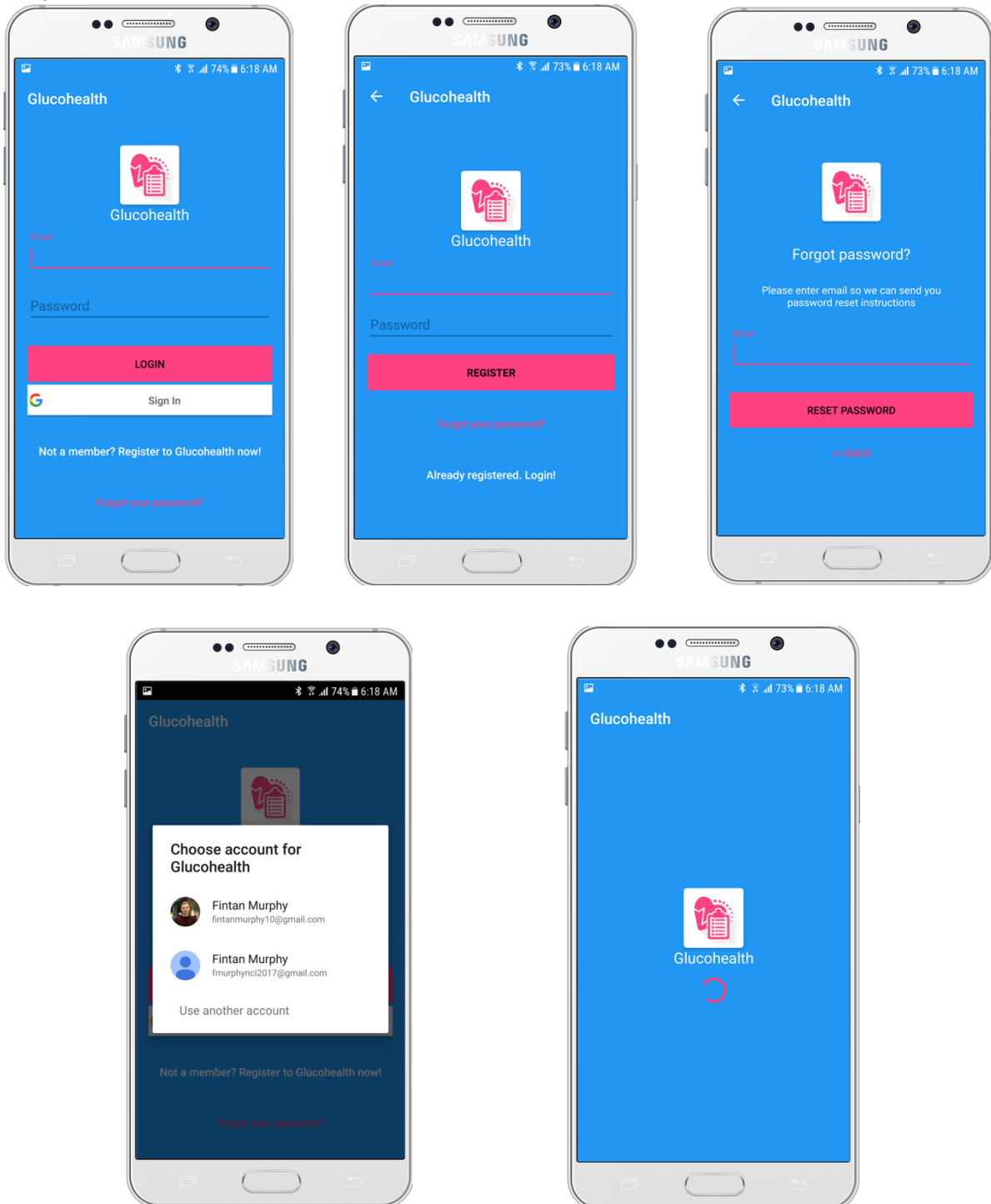


Rules set so user id assigned to readings

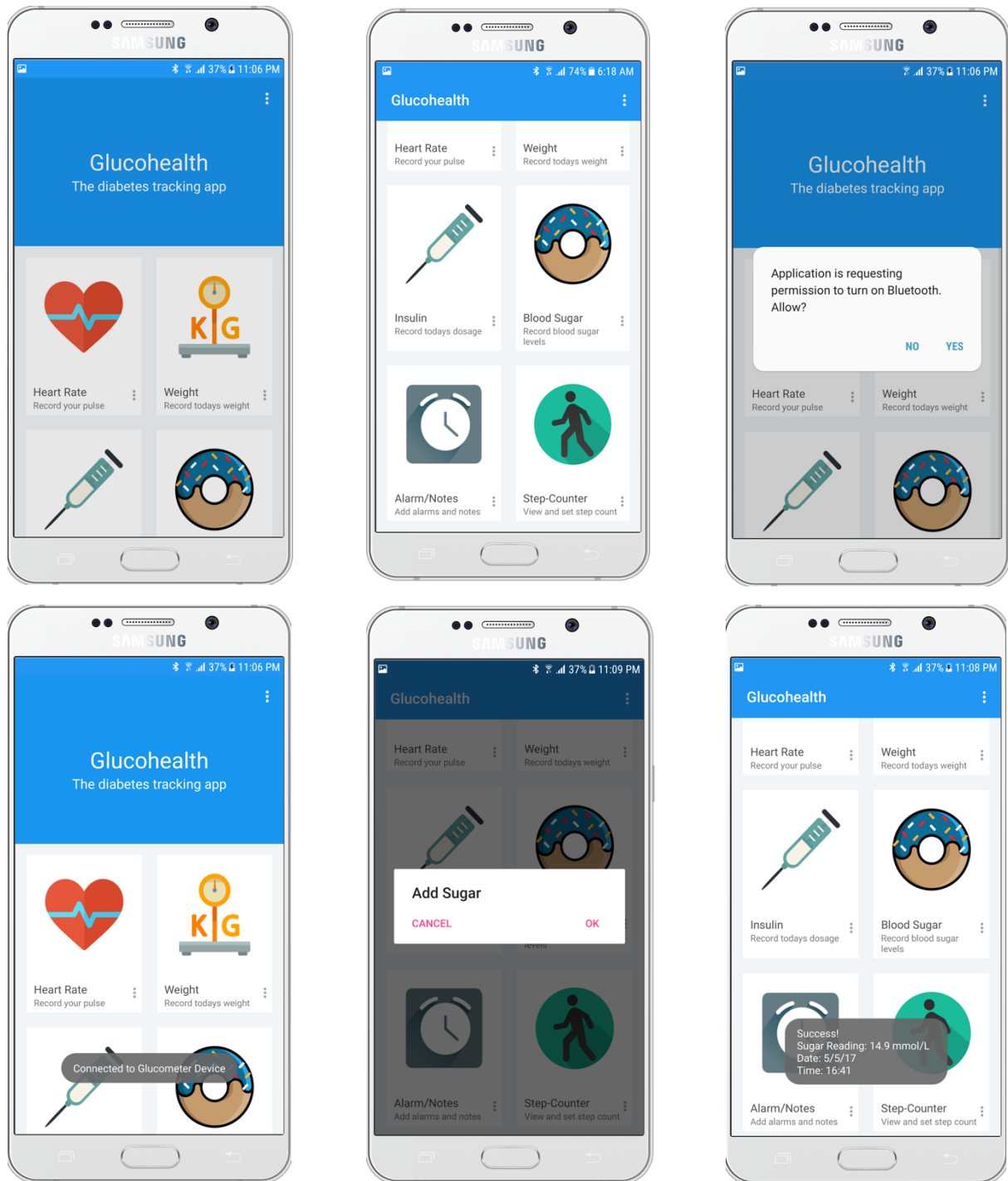
```
1 {
  "rules": {
    "users": {
      "$uid": {
        ".read": "auth != null && auth.uid == $uid",
        ".write": "auth != null && auth.uid == $uid",
        "readings": {
          "$reading_id": {
            "sugar": {
              ".validate": "newData.isString() && newData.val().length > 0"
            }
          }
        }
      }
    }
  }
}
```


3.4 Graphical User Interface (GUI) Layout

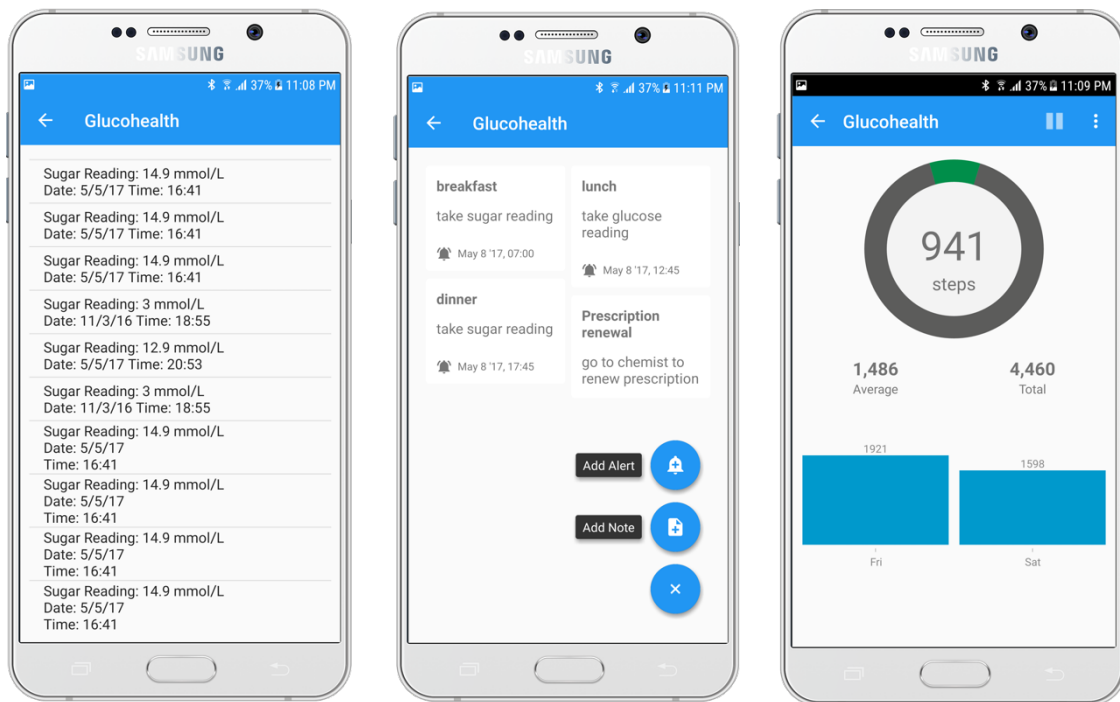
The following GUIs are of the login/register/reset activity and the Google API sign in process.



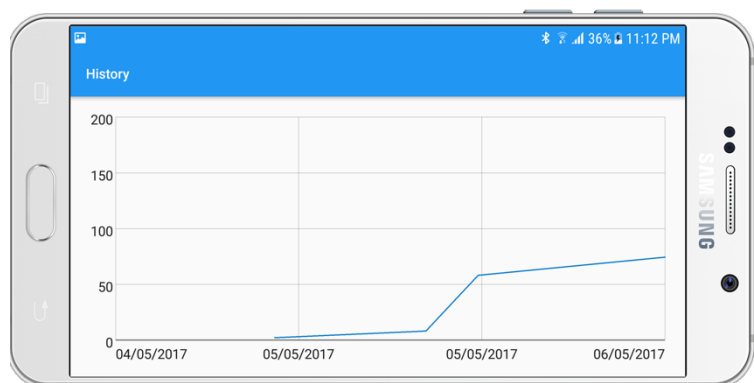
The following screenshots are of the main activity home GUI using RecyclerView Cards with draw-able images to enhance the appearance and feel for the user. The message for requesting Bluetooth can be seen and the success toast message when connected to device.



The following GUIs are of the Glucose readings list view, Graph view, Alarm/reminder and Pedometer.



The readings are located under the sugar reading card in the recycler view when the user selects view history. Similarly, the Alarm and Pedometer are located under their specific card and as can be seen in the alarm GUI 2 buttons appear when adding to select either to add an alert or note. The pedometer displays in a pie chart the steps taken in green what has been completed and the grey to be taken. Bar charts are used to display the previous days with step count. From the menu options in the pedometer you can select target step count and step size. The graph GUI to the right is for sugar, weight, insulin, and heartrate. It plots the date on the x axis and the readings on the Y axis this still requires some fine tuning.



3.5 Testing

Device Component Testing

Testing conducted for this application was particularly important because of the nature of it in utilizing a medical device. The device itself displays readings which are 99.999% accurate so this was a good starting point to use to verify that the readings being sent were correct. In order to test the system with the device I ordered three bottles of control solution that mimics three levels of blood sugar readings high, low and normal. The devices were tested first to ensure that all of the components were working correctly. Once I was sure that the results from the solutions and all of components within the glucometer device were up to the standard I was then able to incrementally build out the android application using Java programming language and testing in a continuous process.

With each new layout done in XML or new piece of functionality written in Java tests were performed many times over to ensure that all was acting as it should do.

Espresso and Junit

I managed to use Androids espresso and Junit testing features to test the LoginActivity entering in various combinations of Strings and characters to test the logic of valid email and passwords upon registration and Signing in. I used Firebase Test Lab to test the application running on various devices also.

Bluetooth testing

After my GUIs including the login were incrementally tested I knew then I could start to implement the code logic for the Bluetooth. In regards to testing this firstly the Bluetooth connectivity was tested by pairing to ensure that a secure proper connection was being established. After many tests between establishing a connection with the glucometer device and Android Application I moved onto the next stage in implementing the code logic to receive the correct readings to the application.

Data testing

This stage in testing was to test the data results from the device being sent to the application this was relatively easy as once the corresponding reading displayed I knew it was properly working. I used the test control solutions to mimic various different real life scenarios however the results were being passed in mg/dl. This needed to be fixed so to overcome this I needed to write a small calculation to convert from mg/dl to mmol/l at first my calculations and code logic was wrong which made the results appear slightly off. After some correction to the code and many more testing with the solutions the readings were displaying accurately at all times. I verified this by downloading a table conversion of mg/dl to mmol/l.

3.6 Customer testing

Once I was happy with the results from the main testing I then moved on to user or customer testing. Two participants were chosen to evaluate and test the application with the glucometer device I was confident that the results would be positive. The users were approached and I explained what I would like for them to do they agreed to help. I allowed them to use the device and application for a 24-hour period and asked them to write a small paragraph of what their findings and feedback.

User 1: Engineer, Ciaran Murphy (control solutions), 37: *“The Glucohealth application performs particularly well creating a user account was simple and effortless using my Google account. The layout of the application is simple and to the point easy to follow and use. The application unlike many applications available on the play store solves an important problem. Although not diabetic myself I can see the idea behind the device and how beneficial it could be for such a user. I did make use of the alarm to remind me to test the devices and blood solutions and it worked pretty nicely. I also made use of them pedometer which I found useful however it would be nice if it used GPS. I would give the application a rating of 5/5 as I believe the idea is excellent and great real potential.”* Ciaran Murphy

User 2: Business Analyst, Kelly Doyle (Diabetic), 26: *“I found the Glucohealth application useful for keeping a diary of my blood sugar levels. The levels are sent very quickly this is useful as sometimes I am in a hurry at lunchtime and forget to take note of the reading. I would personally like to see the application on the app store but I think it could display the results more visually pleasing for the user although this has no effect on the purpose of the application. If it provided more efficient graphs about the levels, I would have given a rating of 5/5 but because of that I will give 4/5. It definitely is an innovative idea and has real market place potential in my opinion.”* Kelly Doyle

3.7 Evaluation

As well as the mentioned testing Junit and Espresso and customer feedback and reviewing the application was evaluated for performance. Glucohealth was evaluated through the use of the monitor provided in Android Studio this runs when the application is running in debug mode. It measures CPU and memory performance this was evaluated on a Samsung S6 and J5. It uses very little memory and CPU on the S6 the memory (RAM) remained consistent at 30-33mb. This is very little for a device with 3 GB. Also spikes in network use only upon login and recording the readings.



4 Conclusions

The aim was to develop a fully functional mobile application using android and Java that is fully integrated with a device that serves a real world problem so to speak. I believe that I have achieved this goal and I have enjoyed the journey in developing the application it has helped me understand more than I could ever imagined from day one in terms of technologies and programming skills that I have achieved from this project.

Although there is great room for improvements and opportunities to enhance the Glucohealth application to help improve all aspects of it. I feel that it could serve 1000s of diabetic users effectively and with a great purpose. There are many diabetic management apps and glucose tracker applications available but in my opinion I think mine differs and stands out from the rest utilizing the meter and other features such as step counter and user authentication which many of the applications do not offer.

The advantages of Glucohealth is that it offers a simple interface easy to use and easily understandable to the user and it works effortlessly with the glucometer device to store readings both locally and online in the cloud with minimal effort.

The disadvantages of Glucohealth is that some aspects of the application could be more finely tuned and implement additional features or touches to add even more value to it. This might include more refined scrollable and zoom-able graphs with a simple calendar to select dates to plot the various readings. Other disadvantages such as not implementing some logic to alert a user if they are within a dangerous reading or are on track to having a deficient or higher blood sugar reading. However, all of this is manageable and doable and I am excited to further develop it further in the future to enhance and perfect it even more.

All in all, I am very satisfied with the Glucohealth project and feel I have met all of the requirements if not fully in every case but definitely to some degree that is acceptable given the time frame that was available.

5 Further development or research

After working and developing this application and investing many hours I am eager to move to the next phase. I will get feedback from entra-health the providers of the glucometer they have already shown a keen interest in the project and have asked me to update them at the end of the development as they would be interested in reviewing my application. I will continue after this year (2017) to research and tweak the application to the best of my ability and potentially add in more features or finish some of the features that were not completed to my level of satisfaction such as the graphs and alerts mentioned previously. Given more time for development and research I feel that I can refine the application and it could have real potential for the market place.

Although I am optimistic that entra health will take a keen interest in my finished product If they decide that they are not interested in working with me with my application or take no real interest in it I would like to find a better solution. Since I believe that there is a real marketplace for the combination of a glucometer device and application to manage diabetes and solving a real world problem, I would be keen to try and find a price competitive supplier who can supply me with my own glucometer device that includes a Software Development Kit and all of the relevant firmware documentation.

This way I could refine my Glucohealth application and make it more attractive for users and release it on the play store and potentially in future develop the application in objective c so that it can be available for iPhone users too. Additionally to this if I managed to find a supplier I could sell the product as a package selling the compatible device and application as one using Ruby on Rails to build an attractive web application to host and display details.

6 References

[1]

Diabetes. 2017. *Diabetes Prevalence*. [ONLINE] Available at:
<https://www.diabetes.ie/about-us/diabetes-in-ireland/>. [Accessed 21 October 2016].

[2]

R Gavin III, J, 2007. Us Endocrine Disease 2007. *The Importance of Monitoring Blood Glucose*, [Online]. 2007, 1-6. Available at:
http://diabetes.ascensia.cz/static/documents/Bayer_reprint_proof1.pdf [Accessed 25 October 2016].

[3]

Forbes, Lee Mathews. 2017. *This Is How Google Keeps 1.4B Android Devices Safe*. [ONLINE] Available at: <https://www.forbes.com/sites/leemathews/2017/02/20/this-is-how-google-keeps-1-6-billion-android-devices-safe/>. [Accessed 5 May 2017].

7 Appendix

7.1 Project Proposal

Glucohealth

Fintan Murphy, x12107395

Email, x12107395@student.ncirl.ie

BSc (Hons) in Computing Year 4

October 21st, 2016

7.1.1 Objectives

The aim of this project is to implement and create an android application for diabetics that will work alongside a Bluetooth glucose meter device. The user will be able to provide a sample of their blood on the glucometer as any diabetic normally would. The glucometer device will be able to communicate with the android application and send the important data readings to the android application over a Bluetooth network. The application should look attractive and serve a valuable purpose to the diabetic user.

The important readings will include the blood sugar reading levels which is measured in mg/dl (Milligrams per decilitre), the date of reading and the time of the reading. All of these readings will be sent by the glucometer device and retrieved, saved and stored in the application of the users' mobile device (phone/tablet device).

The application will also include a pedometer that will track the users' movements so that they can ensure they are walking/moving for the recommended 30 minutes per day which is very important for blood circulation especially for people with diabetes.

The application will also provide a reminder alarm so that the user can set reminders for taking blood sugar readings, diet, insulin, medication and whatever else may be necessary.

In order to complete this project many objectives must be completed. The application will be built using Java in Android Studio and data will be stored locally using SQL-lite for offline use. For backup to an online resource the application will use a network connection (when available) to store and retrieve data using a web-based service. The glucometer device itself comes with an API (Application programming interface) that can be interfaced with Android, and Java enabled mobile devices perfect for the needs of this project. The glucometer utilizes the standard SPP (Serial Port Profile) over Bluetooth (v2.1) for communication dialog.

All this means is that once the device API is integrated with the Android application the blood glucose records, date and time can be retrieved from the device to the application where it can then be saved locally and online.

Once the glucometer device has been integrated with the application the reminder/alarm clock will need to be coded so that a user can set reminders as mentioned above.

The application will include a pedometer as mentioned above, most android-powered devices have built in motion sensors. The pedometer will count steps by using the sensors in the phone. To build the pedometer android have provided all the required documentation on how to implement this using their provided classes and methods.

Main Objectives to be completed:

- Build android interface
- Build SQL-lite database for local storage for offline use
- Integrate glucometer device API with android application
- Integrate pedometer to android application
- Register 365 domain name for online resource and build MySQL DB to perform HTTP requests from application
- Build MySQL database to store and retrieve data

7.1.2 Technical Approach

Obviously the technical approach may change a little along the way but once the requirements are written up there will be a much clearer picture on the exact way how this project will be completed.

The technical approach that will be used to complete this project is to first start with the development of the application itself this will be done using Java in the Android Studio. The application will be built incrementally testing as its being developed and once the application is looking and feeling how it should with all the appropriate GUIs that should be included then the next steps can be undertaken.

The next step will be to refer to the requirements document and how the SQLite database should be designed. This will be designed and implemented into the application so that the data can be stored logically and correctly within the application as will be specified in the requirements. Once the application and local storage has been successfully implemented and tested using some test data to store and read from the SQLite database the next step can be tackled.

The next step will be to implement the glucometer device into the application using the provided API that comes with the device. This will involve some of Bluetooth commands and proper Bluetooth coding conventions as mentioned by google. So the protocols for retrieving blood sugar records, retrieving date and

time will need to be fully understood in order to get this functionality working fully. Even though it'll be a challenge in getting this part working it should be an interesting topic to learn about Bluetooth commands and the whole integration of an external device to the application itself. Once this is successfully done it will be tested incrementally to ensure the correct data and retrieval of data is how it should be.

The next stage will be to incrementally build the application and add functionality for the reminder/alarm and pedometer. This will take some time to build and test with the probability to run into some problems and challenges time will need to be managed accordingly so that everything is done on time.

The next step will be to register a domain name on register365 so that a MySQL database can be built to support the application for online storage using HTTP commands to send and retrieve data from the online MySQL database. This may be a challenge to ensure the correct practices are taken to ensure backup and security.

The next stage will be to get the application to communicate with the online database using HTTP commands to send and retrieve the data.

At this stage the blueprints for the project will be done and it will be about adding in the final touches and testing. The main aim is to build the application with all the mentioned functionality that works as it should for the user with no incorrect readings or hiccups then if there is time at this stage the design look and feel can be tidied up and made look more professional but main aim is to have all the functionality.

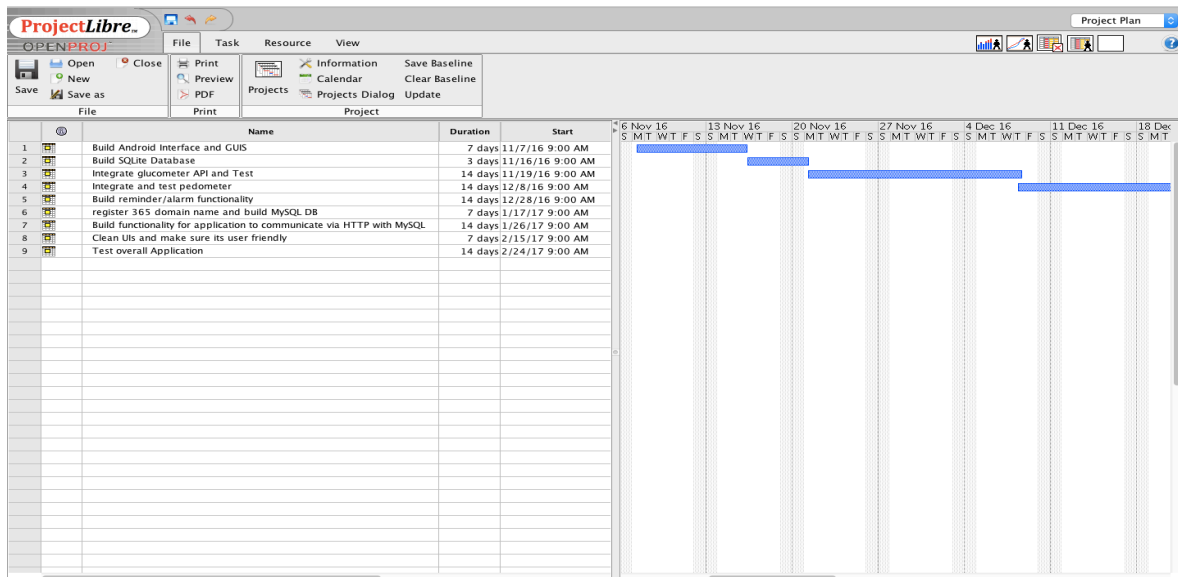
When the requirements are done this technical approach may change for various reasons however what has been mentioned here should be a rough estimate how it will be incrementally built and tested along the way.

7.1.3 Special resources required



























The following resources are required to build this project

- Android Device for demo and testing (phone and/or tablet)
- Bluetooth Glucometer Devices
- Automatic coding test strips
- Glucose high/normal/low control solutions
-

7.1.4 Project Plan (original)



7.1.5 Project Plan (revised)

		Name	Duration	Start	Finish
1		Project Proposal and Project Plan document	19 days?	27/09/16 08:00	21/10/16 17:00
2		Device selection and Purchase	5 days?	03/10/16 08:00	07/10/16 17:00
3		Septemeber Reflective Journal	1 day	03/10/16 08:00	03/10/16 17:00
4		Requirement Specification functional and non-functional	20 days?	24/10/16 08:00	18/11/16 17:00
5		October Reflective Journal	1 day?	07/11/16 09:00	08/11/16 09:00
6		Prototype and mid-poin technical report	15.875 d...	18/11/16 09:00	09/12/16 17:00
7		Novemeber Reflective Journal	1.875 days?	02/12/16 09:00	05/12/16 17:00
8		Tutorials and android development research	14.875 d...	02/12/16 09:00	22/12/16 17:00
9		December Reflective Journal	1 day?	02/01/17 09:00	03/01/17 09:00
10		Exam study and preparation	8.875 days?	28/12/16 09:00	09/01/17 17:00
11		January Reflective Journal	1 day?	06/02/17 09:00	07/02/17 09:00
12		Android Development of Bluetooth for device	10.875 d...	07/02/17 09:00	21/02/17 17:00
13		Development of application GUIs	11.875 d...	23/02/17 09:00	10/03/17 17:00
14		February Reflective Journal	1 day?	06/03/17 08:00	06/03/17 17:00
15		SQLite Database Creation	5.875 days?	10/03/17 09:00	17/03/17 17:00
16		Integrate and test Application with Glucometer	5.875 days?	17/03/17 09:00	24/03/17 17:00
17		Fix issues with Application and Device	11 days?	24/03/17 08:00	07/04/17 17:00
18		March reflective Journal	1 day?	06/04/17 08:00	06/04/17 17:00
19		Make readings appear correct and 1 per reading	6 days?	10/04/17 08:00	17/04/17 17:00
20		Develop Alarm/reminder functionality	9 days?	17/04/17 08:00	27/04/17 17:00
21		Develop List view and graph for insulin, sugar, heart rate and weight	4 days?	25/04/17 08:00	28/04/17 17:00
22		Integrate Firebase and user authentication	3 days?	26/04/17 08:00	28/04/17 17:00
23		Implement pedometer API	3 days?	01/05/17 08:00	03/05/17 17:00
24		Test over all Application and tidy UIs	2 days?	04/05/17 08:00	05/05/17 17:00
25		Complete documentation - technical report	3 days?	05/05/17 08:00	09/05/17 17:00

****note****

the full project plan was designed on a mac using Project Libre. To see chart and full plan visit here to download and/or view the PDF.

<https://drive.google.com/open?id=0Bz4CtT677y47cDBHaEtFRFdOenc>

7.1.6 Technical Details

The main implementation language that will be used in the Android application itself will be Java in Android Studio. Android Studio which is the official integrated development environment for Android development. It is freely available and very well documented online.

The Bluetooth Glucometer device provides an API this can be integrated with a variety of platforms but for the purpose of this project it'll be implemented in Android studio as it supports java enabled mobile devices.

Android Studio provides many ways to store data and SQL-Lite is one of them this will be used to store data from the application locally and for offline use.

Register365 will be used to create an online MySQL database so that the android application device can fetch and send data using HTTP commands.

The pedometer will be implemented in android studio which provides all the documentation of the various classes and methods inbuilt in the android java packages and libraries.

Main languages and libraries to use:

- Java (Android Studio)
- SQLite
- MySQL
- HTTP Commands
- Glucometer API
- Motion sensors documentation on android development found in link below

https://developer.android.com/guide/topics/sensors/sensors_motion.html

7.1.7 Evaluation

The way in which this project application will be evaluated will be first and foremost incrementally built and tested with every new piece of functionality that is added. Once the application is built the device API will need to be integrated and tested for the correct blood glucose readings.

The device comes with a bottle of test control solutions of glucose for low/normal and high glucose measurements. This will allow the testing to imitate real life scenarios of the user using their own blood for correct readings. The device should send the readings in a timely manner so that the user can quickly have the readings stored without waiting long periods of time for the data to be sent so this will need to be tested also.

Once the reliability of the device tests is completed the pedometer will be implemented and tested. The way in which the pedometer can be accurately tested from a user's perspective is to physically move and walk about to ensure the correct step counts are accurate and precise.

Test will be created for the reminder/alarms functionality also as its being built.

7.2 Requirement Specification

7.2.1 Purpose

The purpose of this document is to set out the requirements for the development of an Android application that will be developed in Android studio with the primary language being Java. The android application is a diabetic health application that works alongside a Bluetooth glucometer device for storing glucose blood sugar readings, the readings will be stored locally on the mobile device using SQLite. In addition to storing the data locally it will also be backed up on an online resource, Register365 will be used to create an online MySQL database so that the android application device can fetch and send data using HTTP commands.

This document will also illustrate how the Android application will work and the complete declaration of the overall development of the application as well as any additional features that will be included such as the pedometer and alarm reminder. It will also illustrate the various technologies that are used and how they all integrate into the application.

The intended customers will be anyone who is living with diabetes type 1 or type 2. It is an all in one health application aimed specifically for people with diabetes and that have an android phone or tablet device.

7.2.2 Project Scope

The scope of the project is to develop an all in one Android health application/tool for people living with diabetes which they can use to help manage their daily routines and store important data such as their blood sugar readings.

The android application shall work alongside a glucometer device that will allow the user to take their blood sugar readings. Using Bluetooth the glucometer will send the readings wirelessly to the android application where it will be stored locally on the mobile device using SQLite and also backed up online as mentioned in the project purpose.

In addition to this the application shall also include a pedometer that will track the users' movements.

The application will also provide a reminder alarm so that the user can set reminders for taking blood sugar readings, diet, insulin, medication and whatever else may be necessary for them.

Definitions, Acronyms, and Abbreviations

API - Application Program Interface

MG/DL - Milligrams per deciliter

SPP – Serial Port Profile

IDE – Integrated Development Environment

Glucometer – Glucose Meter

GUI – Graphical User Interface

HTTP - Hypertext Transfer Protocol

ANR – Application Not Responding

7.2.3 User Requirements Definition

Android is the most used mobile device operating system out there, the application will be able to be used both offline and online making the application a very useful tool to aid people living with diabetes. The following are the requirements that the system will have for the users using the diabetic glucometer health application:

1. User should be able to record their readings to the android application using the glucometer

The Glucometers provided API should be fully integrated with the Android application so that users can send/retrieve their readings data from glucometer to the application

2. User should be able to see their blood readings displayed on the Android application

The Application should provide the functionality to display the users blood sugar readings on the users' mobile devices after retrieval from the Bluetooth glucometer device

3. The user should be able to see the status of their blood readings (low, normal or high)

The application should have the functionality to check the reading determine and display whether the blood reading is low, normal or high and notify the user in the GUI application using color indicators

4. The user should be able to see their step count displayed within the application

The application should implement a fully functional pedometer that measures the step count of the user and notify them if they reach their daily targets

5. The user should be able to set alarms/reminders

The application should provide the functionality for the user to select times and dates for them to set alarms and reminders and to provide some message to display such as “take medication”, “record blood sugar readings”

7.2.4 Functional requirements

- **Representation of Correct Data** – The android application shall present and display the correct data readings to the user, sent from the Bluetooth glucometer to the application.

This is the highest ranked functional requirement mainly because it is imperative that the readings are correct especially when dealing with something so sensitive such as a users' health. An incorrect blood sugar reading could potentially cause some stress towards the user so this

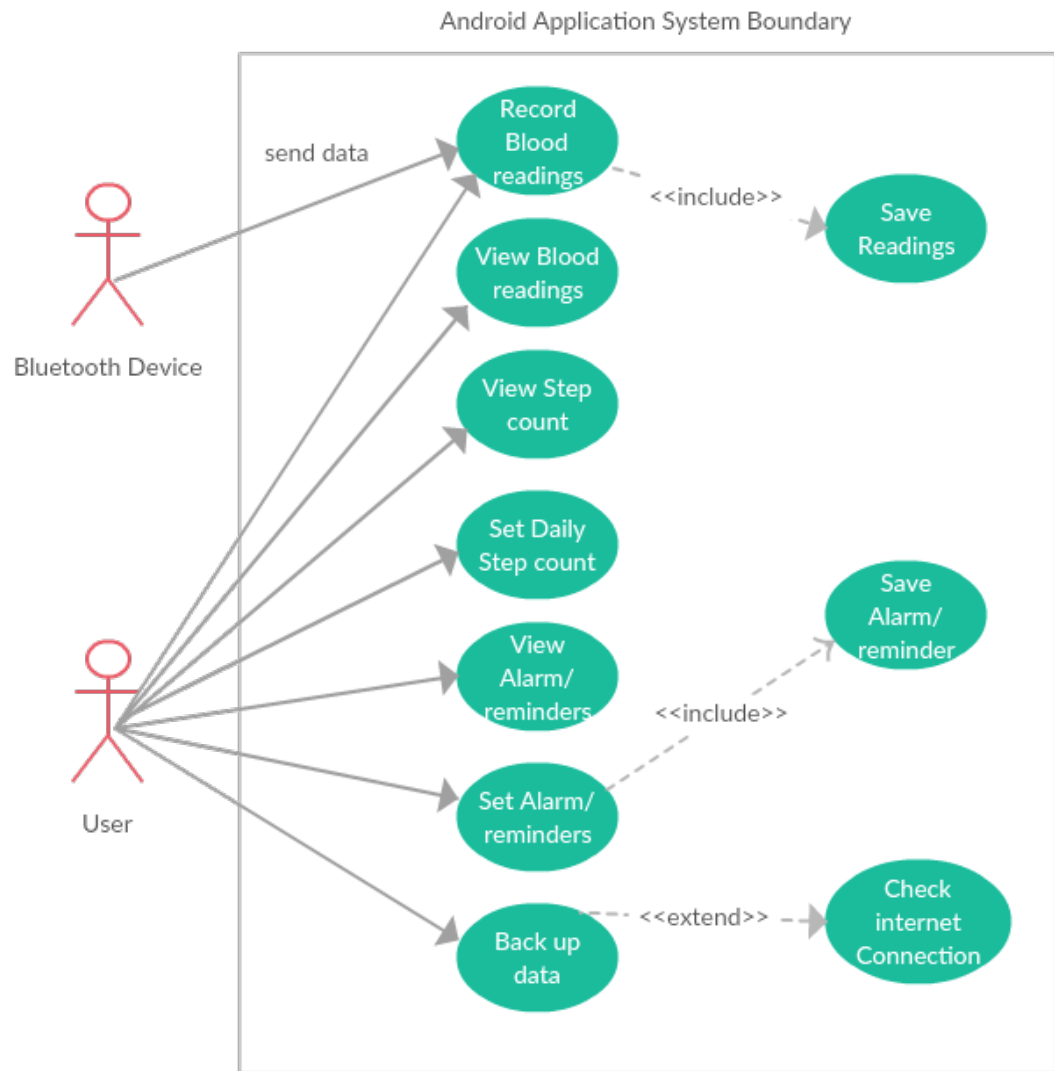
cannot be stressed enough how this functional requirement is extremely important!

- **Application Connectivity with glucometer** – the glucometer and android application shall be fully integrated together using the provided API that the glucometer provides
- **Bluetooth communication for Retrieval of data** - This means that the sending/retrieval of the data shall be made possible between the glucometer device and application.
- **Backup Storage functionality** – The application shall provide the functionality for saving data both locally (for offline use) and backed up on an online resource (for when internet connectivity is available)
- **Pedometer functionality** – The application shall provide the functionality for the use of the pedometer to track users' movements
- **Alarm/reminder functionality** – the application shall provide a GUI for the user so that they can set/add alarms and reminders

Use Case Diagram

The following Use Case Diagram provides an overview of all the functional requirements that the system will provide. Each requirement is identified uniquely with a sequence number for a clearer interpretation of how the system works.

The Use Case Diagram provides an overview of all functional requirements.



7.2.5 Requirement 1 <Representation of Correct Data>

Description & Priority

This requirement describes the process of the Bluetooth Glucometer Device sending the correct data to the Android device so that the user can view the blood readings. The importance of this requirement is essential since the representation of the diabetic users' blood level readings must be 100% accurate and correctly displayed.

Use Case

Record Blood Readings and View Blood Readings

Actors

User & Bluetooth Glucometer Device

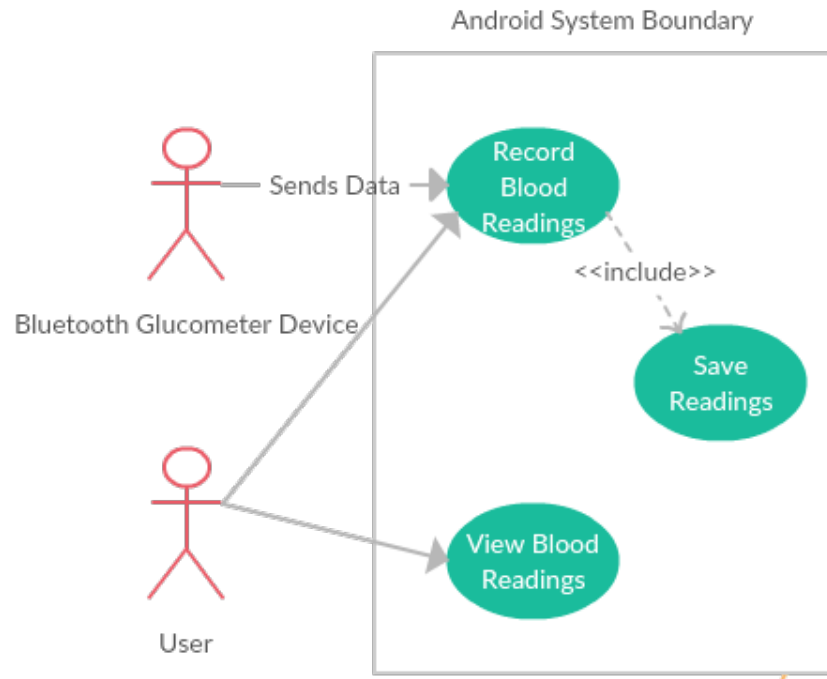
Scope

The scope of this use case is so that the user is provided with the correct data on the Android device that is sent from the Bluetooth Glucometer Device

Description

This use case describes the process of the Bluetooth Glucometer Device sending the correct data to the Android device so that the user can view the blood readings

Use Case Diagram



Flow Description

Precondition

A user must have the Bluetooth Glucometer Device and user must have the Android Application installed on a Bluetooth Android Device so that both the Glucometer and Android device can interact.

Activation

This use case starts when a <user> provides their blood sample on the Glucometer Device and proceeds to send the data across to the Android application

Main flow

1. The <user> opens the Application and establishes a connection with Glucometer Bluetooth Device.
2. The <user> proceeds to the User Interface to send blood readings
3. The <user> proceeds to Glucometer Bluetooth Device and provides blood sample, and proceeds to send data to Android Application
4. The <Bluetooth Glucometer Device> uses SPP protocols to send the data across the Bluetooth Network
5. The System accepts and receives the Blood readings from <Bluetooth Glucometer Device> and saves locally
6. System displays correct representation of data for the <user> to view

Alternate flow

A1: <Connection not established>

1. The application is unable to establish connection to <Bluetooth Glucometer Device>
2. The <User> is prompted with error message indicating that connection was unsuccessful and to check connection and try again.
3. The <user> proceeds to retry, checking both connections are online and available for transfer and continues from main flow point 1.

Exceptional flow

E1: <Transfer aborted>

1. The <User> aborts the transfer before data is sent from <Bluetooth Glucometer Device>
2. The <User> is prompted with error message indicating that transfer has been unsuccessful
3. The use case continues at position 2 of the main flow

Termination

The system presents the correct transferral of the Data to the <user> and displays for <user> to view and accept

Post condition

The system goes into a wait state for the <user> to proceed to a desired functionality of the Application

7.2.6 Requirement 2 <Pedometer Functionality>

Description & Priority

This requirement describes the functionality of the pedometer so that the user can view and set the step count within the Application system. The importance of this requirement is high as it is one of the key functionalities that the application shall provide.

Use Case

View Step Count & Set Daily Step Count

Actors

User

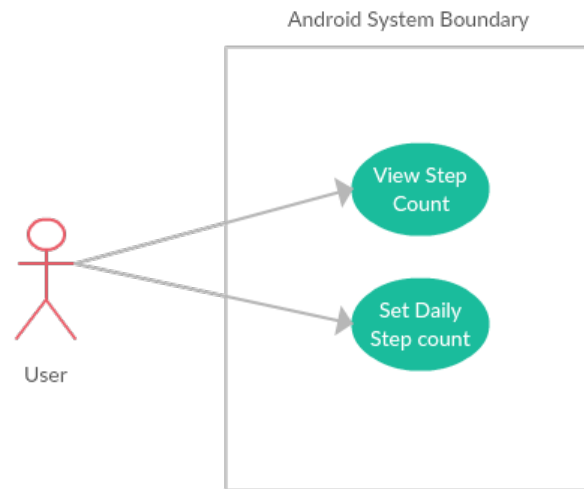
Scope

The scope of this use case is to enable the user to view daily step count as well as set daily step count

Description

This use case describes the process of the pedometer functionality enabling the user to view and set daily step count.

Use Case Diagram



Flow Description

Precondition

The Android application is installed and open on the Android Device

Activation

This use case starts when a <user> proceeds to the pedometer functionality within the navigation menu

Main flow

1. The <user> opens the Application and proceeds to pedometer option within navigation menu
2. The <user> selects set daily step count button by tapping with his/her finger
3. The system displays an input box to the <user> to enter in desired daily step count
4. The <user> enters in daily step target from range 1,000 - 12,000 steps and taps add button with their finger
5. The System sets the desired step target and saves it

6. The <user> is prompted with a Toast message that the daily step count has been set

Alternate flow

A1: <Incorrect Step range>

1. The <user> enters in step range > 12,000
2. The System displays a message to the <user> warning them that an incorrect step range has been entered and to retry
3. The use case continues at position 4 of the main flow

A2: <Incorrect Step range>

1. The <user> enters in step range < 1,000
2. The System displays a message to the <user> warning them that an incorrect step range has been entered and to retry
3. The use case continues at position 4 of the main flow

Exceptional flow

E1: <user exits >

1. The <user> opens the Application and proceeds to pedometer option within navigation menu
2. The <user> selects set daily step count button by tapping with his/her finger
3. The system displays an input box to the <user> to enter in desired daily step count
4. The <user> enters in daily step target from range 1,000 - 12,000 steps and forgets to tap the add button and exits pedometer
5. The <user> must proceed to continue from main flow point 1

Termination

The system is terminated when the user adds desired step count and exits

Post condition

The system goes into a wait state to record movements from user and also to wait for user to proceed to another functionality within the application

7.2.7 Requirement 3 <Alarm/Reminder Functionality>

Description & Priority

This requirement describes the functionality of the Alarm/reminder so that the user can view and set the alarms/reminders within the Application system. The importance of this requirement is high as it is one of the key functionalities that the application shall provide.

Use Case

View alarms/reminders & Set alarms/reminders

Actors

User

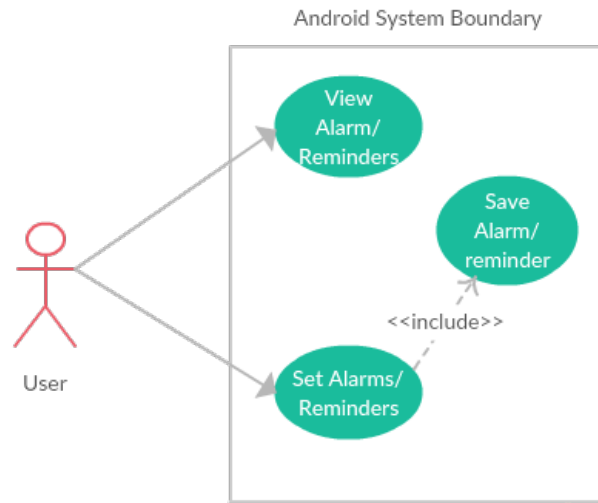
Scope

The scope of this use case is to enable the user to view their alarms/reminders as well as set alarms/reminders

Description

This use case describes the process of the alarms/reminders functionality enabling the user to view and set alarms/reminders.

Use Case Diagram



Flow Description

Precondition

The Android application is installed and open on the Android Device

Activation

This use case starts when a <user> proceeds to the alarms/reminders functionality within the navigation menu

Main flow

1. The <user> opens the Application and proceeds to alarms/reminders option within navigation menu
2. The <user> selects set alarms/reminders button by tapping with his/her finger
3. The system displays an interactive timer for the <user> to select times and days
4. The <user> selects the desired days and times and taps add reminder button with his/her finger
5. The System sets the reminder and saves it

6. The <user> is prompted with a Toast message that the alarm/reminder has been set and saved
7. The <user> can proceed to view reminders or add another

Alternate flow

Exceptional flow

E1: <user exits before adding>

1. The <user> opens the Application and proceeds to alarms/reminders option within navigation menu
2. The <user> selects set alarms/reminders button by tapping with his/her finger
3. The system displays an interactive timer for the <user> to select times and days
4. The <user> selects the desired days and times and exits
5. The <user> must start from main flow point 1

Termination

The system is terminated when the user adds desired alarms/reminders and exits the alarms/reminders functionality

Post condition

The system goes into a wait state to wait for user to proceed to another functionality within the application

7.2.8 Requirement 4 < Backup Storage functionality >

Description & Priority

This requirement describes the functionality of the Backup Storage so that the user can save the data to an online database from the Application. The importance of this requirement is high as it is one of the key functionalities that the application shall provide.

Use Case

Backup Data

Actors

User

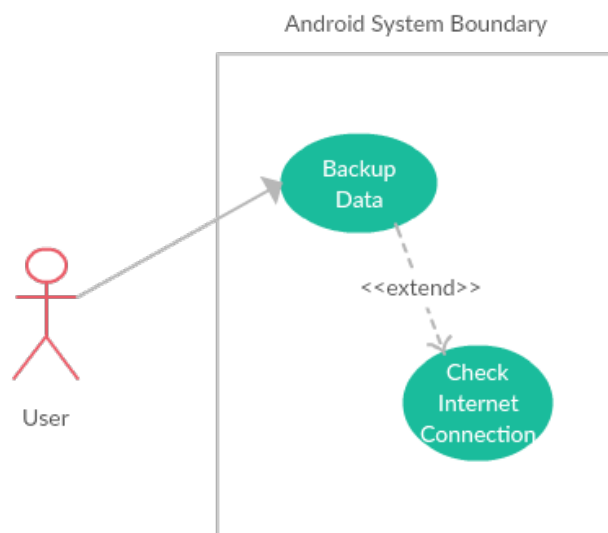
Scope

The scope of this use case is to enable the user to back up the data to an online Database

Description

This use case describes the process of the backup and storage functionality enabling the user to back up data online.

Use Case Diagram



Flow Description

Precondition

The Android application is installed and open on the Android Device and some data is saved and stored locally on device and user has internet connectivity

Activation

This use case starts when a <user> proceeds to the backup option within the navigation menu.

Main flow

1. The <user> opens the Application and proceeds to backup option within navigation menu
2. The <user> selects backup button by tapping with his/her finger on the backup
3. The system displays a connecting message box for the <user> indicating that a connection is being made
4. The system displays a connection successful message box for the <user> indicating that a connection has been made
5. The System provides a progress bar displaying the percentage
6. The <user> is prompted with a Toast message displaying that the backup has been completed

Alternate flow

A1: <No internet connection>

1. The application is unable to establish an internet connection to
2. The <User> is prompted with error message indicating that connection was unsuccessful and to check internet connection and try again.
3. The <user> proceeds to retry, checking internet connection and continues from main flow point 1.

Termination

The system is terminated when the progress bar reaches 100%

Post condition

The system goes into a wait state to wait for user to proceed to another functionality within the application

7.3 Non-Functional Requirements

7.3.1 Performance/Response time requirement

Response time is one of the highest ranked non-functional priorities for this application and it shall be managed properly to ensure that the application is performing smoothly with quick response.

The user should not have to be left waiting any longer than 3-5 seconds. The implementation of a Progress Bar in the UI will be useful so that users won't get frustrated waiting for results to be sent, it will let the user perceive that progress is being made.

7.3.2 Availability requirement

The Diabetic application should have a continuous availability level meaning it should be available to use 24 hours 7 days a week without any outages.

Since this application is dealing with sensitive information it is imperative that the application is available to the user at all times.

7.3.3 Robust requirement

It must be robust enough to operate under stress and deal with invalid inputs. This shall include dealing appropriately with inconsistent readings from the glucometer and able to deal with recording the movements within the pedometer feature that the application will provide for the end user.

7.3.4 Usability requirement

This will relate to how easy the application shall be for the user to use and the layouts should not be cluttered. It must be easy for a non-technical user to use also, the following are the key usability requirements that the application shall have:

- **Interface** – this is how the application GUI should look and feel it should be consistent easy to understand and easy to navigate throughout. The application should not take long for the user to understand how to use it
- **Layout performance** – The use of re-using layouts and optimizing layout hierarchies within the Android application should be used to give the end user the feeling of smooth scrolling interfaces with a minimum memory footprint
- **Scaling/resizing** - the application will support multiple mobile and tablet devices scaling and resizing accordingly depending on the device it is running on
- **Efficiency of use** – the applications goals should be easily met and accomplished by the user with no errors
- **Error handling** – the application should handle any errors accordingly and provide the usage of error messages for the user to easily understand

7.3.5 Reliability requirement

This is similar to availability requirement however the reliability is important for the application to work and deliver correctly at all times under various environmental conditions. The software will be tested for being as robust and reliable as possible so that the users experience with the application is smooth and as error free as possible.

7.3.6 Security requirement

While this application is dealing with raw data such as blood sugar readings, security is not a really important nonfunctional requirement for this application as

there is no sensitive information being handled other than the raw data which is just numbers. However, the data transfer from the glucometer to the android device application uses Bluetooth which can easily be intercepted or corrupted so it is important to use the right procedures to ensure that the protocols used are secured in way that it cannot be intercepted easily or corrupted.

And also while backing up data to the online resource the correct procedures should be used to ensure that data protection is met.

7.3.7 Scalability/Concurrency requirement

The application should have the ability to deal with increasing or decreasing workloads, for example the application should be able to still operate the pedometer feature while backing up data online or receiving data from a glucose meter device. It should be able to handle all operations concurrently and be optimized to do so. It will also have enough capacity between local storage and online backup so that all data is secured and stored with no loss of data.

7.3.8 Data Requirement

This section describes the data requirements that are important for the Android Application to work as it should. To achieve this SQLite will be used for storing blood glucose readings and alarm/reminder data as well as the pedometer users step count locally on the android device. An online MySQL database will also be implemented to secure data and back the data up in case of any potential loss of data on the Android device itself. This will be done by registering a domain name on Register365 which will hold the MySQL database where the data for the blood glucose readings will be stored.

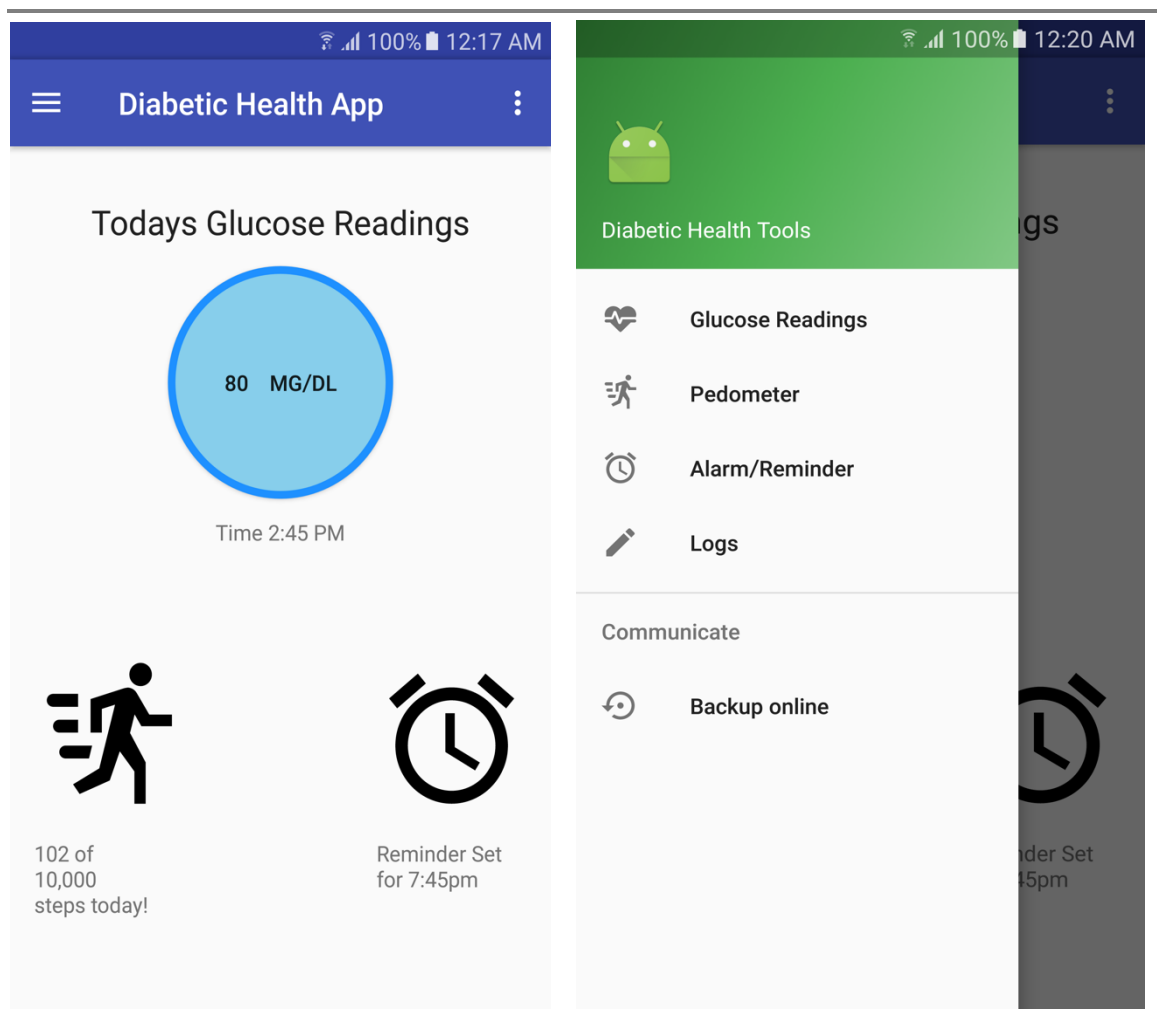
7.3.9 Interface requirements

The Android application will be purely GUI driven since it is a mobile application. To ensure the application looks and feels responsive it will incorporate Material Design.

The application shall be interfaced with the Glucometers API so that communication is made possible for sending data between the two devices.

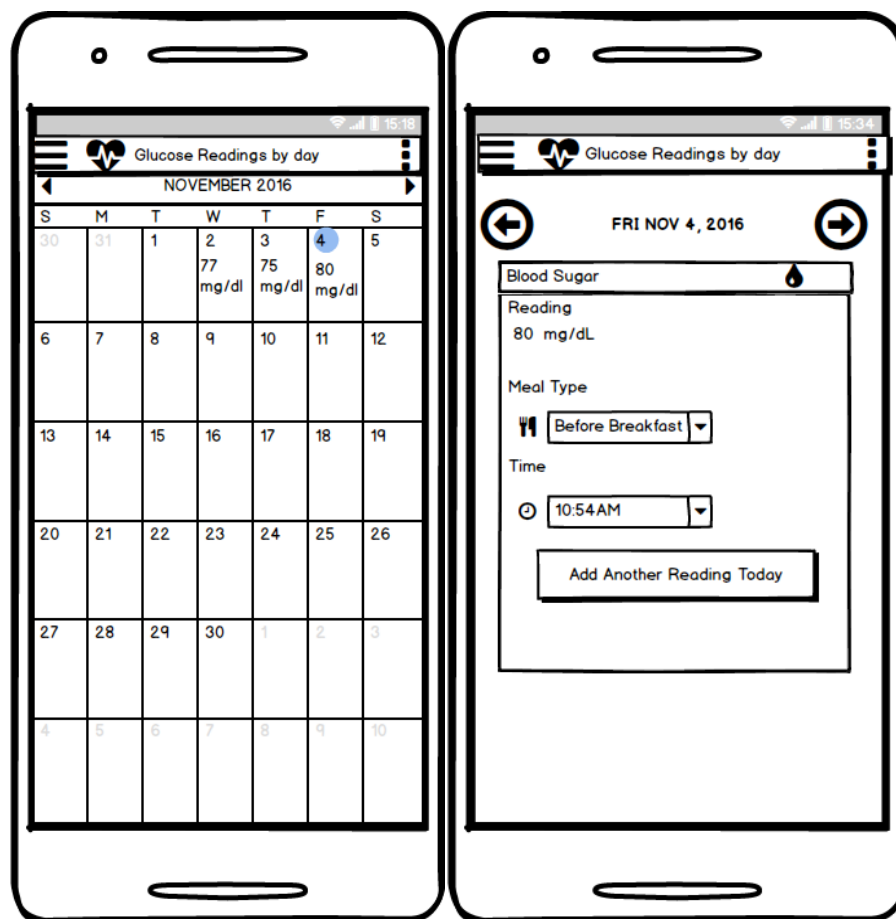
Also there will need to be an online web server hosting a MySQL database to allow for the sending and retrieval of the data. The way in which the android application will communicate with the client-server is through a REST API that will be created and this will allow queries to be made using HTTP methods and parse the results in JSON format.

7.4 Mockups

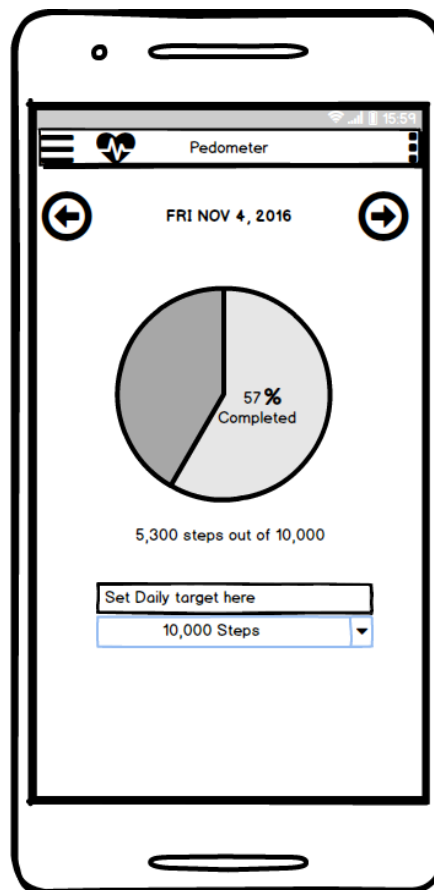


The above screenshots are quick mockups/prototypes made in Android studio of the application the 3 widget like components that can be seen in the first home screen will display to the user their last glucose reading with the time. The second widget like component will display the number of steps taken in that day and the reminder widget will display when the next reminder is due. Obviously these are only mockups and the final production of the application is expected to change throughout the development stages.

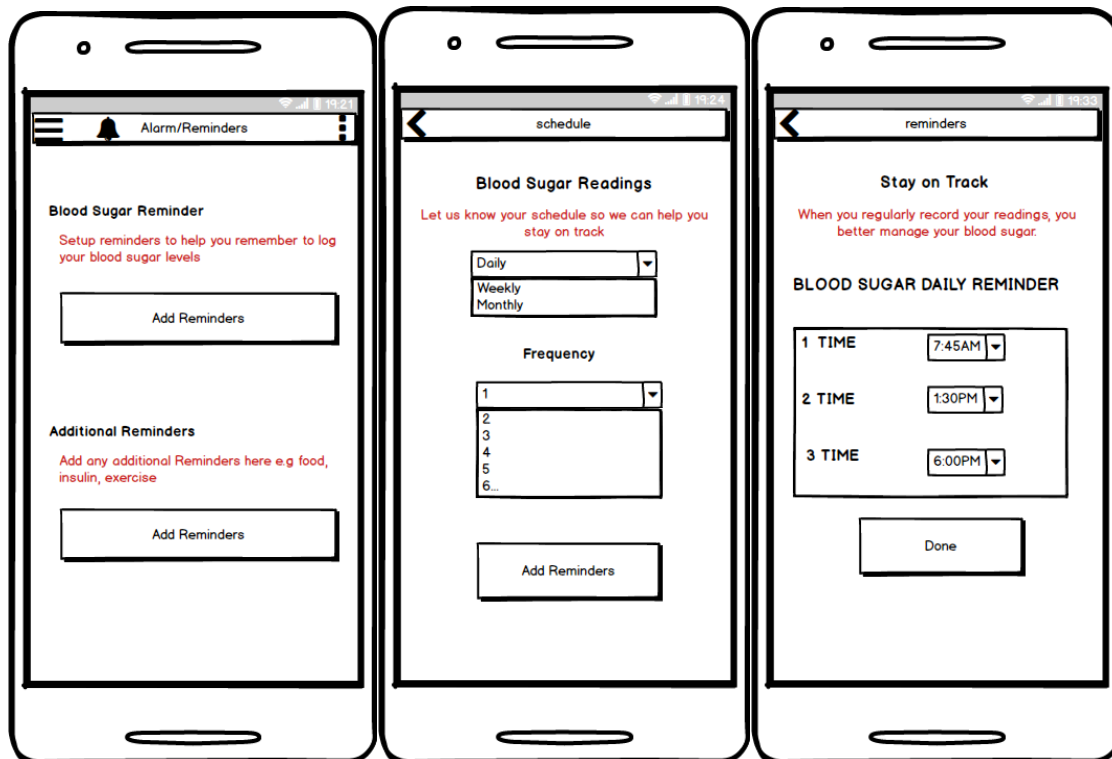
The second screen is something the application will include it is a navigation drawer activity that uses material design this will enable the user to navigate to the various features of the application. The backup online feature will enable the user to backup all of the locally stored glucose readings to the online server. The mock ups for the glucose readings, pedometer, alarm/reminders, and logbook can be seen below.



The above mockups describe the glucose feature which is seen in the above prototype mockups when the user selects the glucose readings from the navigation menu they will be firstly brought to a screen that displays an interactive calendar. The user can choose the day they desire from the calendar to view previous readings and to add other readings manually or automatically by using the glucometer device to communicate.



The above mockup GUI displays what the user sees when he/she clicks into the pedometer option in the navigation menu described in the prototype mockup above. From here the user can set their daily targets and view previous days and view the amount of steps taken.

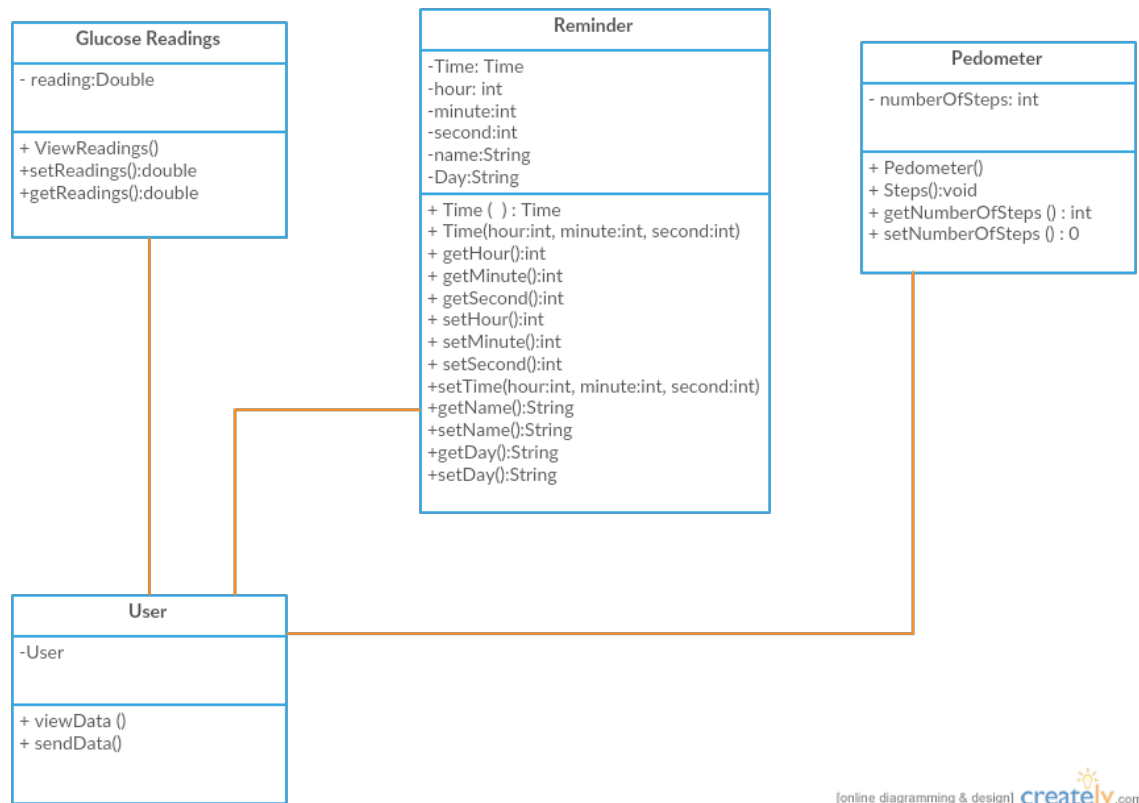


The first mockup screen above shows what will be displayed to the user when they select the alarm/reminder option from the nav bar menu. From here the user selects “Add Reminders” button and it will navigate to the second screen which can be seen in the second mockup above. From here the user selects from daily, weekly, monthly from a drop down menu and the frequency of the reminder per day/week/month. The user then proceeds by selecting the Add reminders button and will be brought to the third screen which is shown in the last mockup above. From here the user selects the times based on the number of frequencies chosen in the previous screen. They can then proceed to enter the times and there will be an option for repeating daily. When the user is happy with their entries they select done and will be notified that the reminders have been set successfully.

7.5 Application Programming Interfaces (API)

This system will use an API provided by the glucometer device to enable the glucometer device to communicate with the Android application to send/retrieve data.

7.6 System Architecture



7.7 System Evolution

The system evolution of the diabetic application will evolve greatly after the first stages of development. The potential for it to go live on the play store will depend on Entrahealth the company who is supplying the glucometer device. They have already shown great interest in it and are awaiting status updates from the application itself so overtime with some improvements it can be evolved and used by many users across the globe.

7.8 Monthly Journal

September

September 21st 2016

Yesterday evening was my first lecture of 4th year given to us by Eamon Nolan. The lecture consisted of the introduction to the software project that is required by all to be completed. We have two weeks to think of an idea we are happy with and then we must present/pitch the idea to a panel of 3 judges who will decide whether the project is worthy of doing or not.

So today the 21st of September I am sitting down and giving some real thought as to what I will do for my final year project. I am not usually good at coming up with ideas it was definitely not my strong point in the previous 3 years for team projects and other projects. However, after brainstorming browsing the play store and looking for some inspiration I managed to come up with 4 ideas that I think may be good enough. I have listed the ideas below.

1. My first idea: a solar powered Wi-Fi weather station using Arduino and electrical components to build and then to build an android application that can communicate and retrieve data from the Weather Station. I was thinking maybe I could show the changes of temperature and humidity on some kind of live graph.
2. Second idea: A parental Control Application done in android for parents to manage and monitor their children/teenagers' usage on their mobile devices tablet/phone
3. Third idea: Android diabetic application for monitoring glucose blood levels, diet, exercise and potentially have reminders to take medication or insulin
4. Fourth idea: A login system, Web Application using facial recognition

Over the next few days I hope to have my idea chosen I will firstly need to research the four ideas a little bit more and see what I can do with them and which one is best. I must be careful that my idea has enough scope and will be good enough to do and also realistic and doable because I am not the strongest programmer.

September 25th 2016

Over the past 2 days I have really researched the four topics/ideas I came up with and I have narrowed it down to the one that I think will be best for me to do and also one that will keep me interested and motivated to do.

So from my last journal entry I had listed 4 ideas one of the ideas listed was an android application for diabetics to use to manage their disease. My father was diabetic and I was his carer for the last few years of his life so I feel like I have a good understanding of the disease and hands on experience in what is required by a person living with diabetes. After looking through the play store of applications that already exist for managing diabetes. I felt like I needed to add something novel to the application that differs it from others so that it isn't just a "glorified alarm" for taking medication and diet tips etc.

After some brainstorming I realized that being a diabetic you must regularly check and take note of blood sugar levels doctors may instruct patients living with the disease to check as many as 4 to 8 times per day. This had me thinking how could I store this data easily for the users.

A glucometer or glucose meter is a medical device that takes a sample of your blood and measures the concentration of glucose in the blood. I thought is there a device out there that I could integrate with the android application that I build. After some internet research I found a glucometer device that has Bluetooth. I emailed the company that supplies this device and I asked them was there an API or SDK kit available so that I could use it in my final year project alongside the android application that I build.

The company Entra-health located in San Diego USA returned my email and answered my questions that I had for them. They explained to me that the device for such a project was ideal. They explained that the glucometer can be implemented into Android, java enabled mobile devices.

I feel now that this device working alongside my android application would be perfect. The idea now is to build a diabetic application that can retrieve the blood sugar level readings from the Bluetooth glucometer and store them with the time and dates, readings of the levels and in a way that can be easily read by the user themselves and to have the data secured and readily available for doctors to view rather than the traditional way of writing the measurements in a notebook. In addition to that I will have a reminder alarm that the user can set for them to manage their diet, medications and whatever else is necessary.

While everyone should be doing 30 mins exercise a day to stay healthy and active it is also important for diabetics to do this to improve circulation of the blood and keep their cardiovascular system in check. So I also will look into the possibility of introducing a pedometer into the application this can monitor the users step count and time spent moving and monitor their 30 mins per day.

Over the next few days I will see what is realistic and research the difficulty of developing this application and see what should be left out or included. I have never developed an android application before so this in itself will be a challenge for me and I think integrating a physical device to communicate with my android application will make it stand out from others.

September 29th 2016

Over the past couple of days, I researched how I can complete my application and found some useful resources to help get me started. I cannot really look too deeply into it right now as far as how it'll be done exactly because next Tuesday the 3rd of October I will be pitching my idea in front of a panel of 3 lecturers. So right now I am just focusing on my pitch and how I can convince the panel that my idea is good enough to complete for my final year project. Once my project is

hopefully accepted I will then research even more and start fleshing out my project proposal which will be due end of October. Hopefully by then I will have a better understanding of what needs to be done in order to get started.

October

October 8th 2016

My project was accepted on the 5th of October now that I know that I am fully committed to the project I can start researching and looking more deeply into what needs to be done and how it will be done. The pitch went well and the panel of 3 seemed to really like my idea however they said in order to achieve fullest marks I must try and include everything that I explained to them that is the pedometer feature, having data stored both offline using SQLite and online resource using a server for backup and security as well as the key feature of using the glucometer device.

Today I will watch some video tutorials on how to create a server and save data to it so that I can backup and fetch data from an online resource within my android application. I will do some tutorials for creating some test data and sending the data to an online server using HTTP methods this should give me an understanding how it will work in my Diabetic Health Application. I will also look into the documentation for the pedometer provided by Android Studio (google) and see what I will need to do to implement it.

October 15th 2016

This week I haven't had very much time to concentrate on the project with other stuff happening in my other modules. Last weekend I found some useful tutorials that I completed which gave me a good insight on what needs to be done to get various components of my application working. I have received my glucometer devices this week from Entra-health so over the weekend I plan to have a look at the documentation for using the glucometer API with my android Application.

October 22nd 2016

After reading the documentation from the API last week I have realised that building and integrating the API with my android device will be a big challenge as well as including the other features. The key to getting everything working is to manage my time accordingly I will set out a new project plan this week so I can get started on a prototype and I plan to start my requirements next week and in reading week which will be the week after. I have been assigned my supervisor Padraig De Burca and I will email him in the coming days to arrange our first meeting.

October 29th 2016

I have made a project plan so I will use that to help me make a good start over the next couple of weeks. I have started on my requirements specification and will be working on this over the next 2 weeks. I made some mock-ups and a basic prototype in android studio to get an idea how I want the application to look and feel I will add this into my requirements. Next week is reading week so I plan on using the week to get a substantial amount of my requirements completed/done. I will also look at developing a further prototype and try get my application communicating with the glucometer device. I only got time to email Padraig today and I am waiting to hear back from him to schedule the first meeting probably won't be until after reading week.

My Achievements

This month, I was able to do some more research on my project and also some tutorials. I was also able to get nearly half of my requirement specifications done and I have a basic home and navigation bar prototype built in Android Studio. The tutorials that I have completed online will be surely of good use to me in the coming weeks

My Reflection

I felt that in my first month of working with my project that I did not progress as I would like to and I will really need to get a lot of work done in the next month. However, I am satisfied with the progress I have made with the requirements which will be due in the second week of November.

November

November 10th 2016

I have been busy with other modules this week but I did manage to do some tutorials in Android Studio that will help in the creation of the Diabetic Application. Because I am new to Android development going through the various tutorials will surely be helpful for me in the coming weeks when I really start developing more.

November 15th 2016

This week I managed to get some of the main GUIs done I now know how many activities I should have in my application. I will continue learning more about development with android over the next few weeks I bought a useful book “Android Programming-The big nerd ranch guide “by Philips, Stewart, hardy and Marciano. The book has some great guides for creating and developing various components in android studio as well as tips that will be useful throughout the development over the coming months.

I have been also working away on my project specification which is nearly completed this has been my main focus and will be over the next couple of days it is due the 18th of November.

November 24th 2016

I didn't get much done this week as there has been a lot of pressure with CAs in other modules. Last week I completed my requirements specification document and successfully uploaded it. After completing the requirements specification it has helped me understand more about how I can tackle various problems in my

Diabetic Health Application. Over the next week I will try and flesh out the prototype more and see what I can come up with in order to present in December.

My Achievements

This month, I was able to do some more research on my project and also some tutorials from the book I purchased. I was also able to get my requirements specifications completed

My Reflection

I feel this month has been hectic with CAs and other things going on in my other modules. But I am satisfied that I managed to complete a good requirements document and has given me insight on what needs to be completed over the next few months.

December

December 10th 2016

This week I have been trying to complete all of the other projects and CAs due for other modules as well as preparing for exams. I have started developing the needed GUIs last month for the diabetic application so that I have some kind of prototype to show for the midpoint presentation. I met with my supervisor again this week to discuss the preparation for the midpoint and what will be expected of me.

December 15th 2016

Now I have all my GUIs done using a material design provided by google. I feel that this will be enough for the midpoint and after the exams I can start tackling the main functionalities that is required. I am also thinking of using firebase for my database and maybe implement a login feature using the firebase, this is not

mentioned in my project requirements so maybe if there is enough time to implement this I will.

December 20th 2016

This week has been hectic Christmas is two days away yet I am still rushing to finish other module projects and CAs. I have completed the Mid-point presentation last week and I feel that it went well the two interviewees seemed happy with the idea and progress made however they urged me to get developing and to concentrate on the important sections of the project as soon as the exams are finished.

December 21st 2016

Today I got my marks for midpoint presentation I got 67% which I am quite happy with I feel that I may have got closer to 70+ given more time as I was having trouble with other modules but 67% is good considering.

My Achievements

This month, I was able to complete my GUIs and Mid-point which I feel went good marks 67% and I am quite pleased with that

My Reflection

I feel this month has been hectic with CAs and other things going on in my other modules and getting everything in on time. But I am satisfied that I managed to complete requirements document attend my midpoint and got a decent grade considering all of the pressure with other things going on. Given more time I would have liked to strive for that 70+ mark going forward my time will be managed more accordingly.

January

January 20th 2017

Exams are finished and I done well so a pleased with all results I have enjoyed a week break and now need to get stuck in to the project yet again. This week was the first week back in the college since the exams I have met with Padraig my supervisor and we discussed the plan going forward.

January 26th 2017

This week I started the development for the application even though the first part that I am doing is the easier part I feel that it will help motivate me to really get developing. The part I have started developing is just a simple reminder/alarm functionality this should only take two weeks or so to do and will be a starting block for me.

My Achievements

The start of this month, was occupied with exams and studying, however in the past 2 weeks I have met with my supervisor to discuss the project plan and also I have begun to implement one of the main features into the application.

My Reflection

This month has flown in because of the exams so progress made was really minimal however by making the start with the reminder and alarm functionality is enabling me to visualize the completed project. By doing this simple part first I think it will get me in gear and motivate me to tackle the harder parts later on.

February

February 1ST 2017

This week I hope to finish the main functional parts of the reminder and alarm functionality as described in the requirements. There has been some issues in developing this I am not the strongest coder so each new core functionality requires some time for me to learn and understand. I am learning during the process and becoming better programmer as the weeks pass by slowly but surely.

February 15th 2017

Over the past two weeks I have managed to implement a basic alarm and reminder feature into the application while some core components still need to be developed for this feature, I am pleased with what I have managed to do with it so far. It uses a material design and the alarm can be set to repeat. I am having some issues with it sounding an actual alarm at the moment but hope to resolve this soon.

February 27th 2017

The issue I had has been resolved last week and now the alarm is sounding as it should and displaying the notification. I will add some other additional bits that I mentioned in the requirements document but for now my supervisor how I met this week has recommended that I start the main functionalities as these are the most important. Because of the lack of programming skills that I have it is going to be a huge challenge for me to try and implement the glucometer device with the android application, however it must be tried now so this will be my main focus in the coming weeks and I am sure it will be no easy task!

My Achievements

This month I have achieved actually starting and developing some main features and functionalities that I described in the project requirements document. Most of the alarm and reminder functionality is working as it should there will be some

needed tweaks and added changes to be done but for the coming weeks I really need to try connecting the application with the device I have the Bluetooth modules in place so it is trying to understand how to integrate the two.

My Reflection

This month I am happy with what I managed to do however I may have wasted some days not doing what I should have been. Also I think I may have been procrastinating the main functionality section that must be implemented as this is the core functionality of my application. After talking with my supervisor he has recommended that I start this bit so this is my main focus for the coming weeks ahead.

March

March 3rd 2017

This week I managed to develop the GUIs better looking and I put in place the required SQLite databases.

March 22nd 2017

I managed to pair the application with the Bluetooth glucometer device however the readings are all coming at once. Even so this is a huge step in the development as it is a core part of the application.

My Achievements

This month I have actually achieved a big part of the development so I am really happy with it. However, I really need to try connecting the application with the device more effectively I have the Bluetooth modules in place and integrated but it is reading all the readings from the device this will be the next step.

My Reflection

I managed to get the core functionality working granted not perfectly but it is done and it gives me positivity in the coming weeks this is my last required reflective journal I have cut things short and didn't manage time as best as I could have but I seem to work best under pressure so in the coming weeks I will fix the alarm and get the device reading as it should. I am worried that I won't have time for the online storage and pedometer but I will give it my all next few weeks until the submission date.

7.9 Other Materials and Resources Used

Tutorials, APIs and Libraries used to develop

[1]

Android Developer. 2017. *AlarmClock*. [ONLINE] Available at: <https://developer.android.com/reference/android/provider/AlarmClock.html>. [Accessed 28 February 2017].

[2]

Android Developer. 2017. *Bluetooth*. [ONLINE] Available at: <https://developer.android.com/guide/topics/connectivity/bluetooth.html>. [Accessed 1 March 2017].

[3]

GitHub - Sam. 2014. *An image loading and caching library for Android focused on smooth scrolling*. [ONLINE] Available at: <https://github.com/bumptech/glide>. [Accessed 30 March 2017].

[4]

Gitbhub - Dmytro Tarianyk. 2015. *Android Floating Action Button based on Material Design specification*. [ONLINE] Available at: <https://github.com/Clans/FloatingActionButton>. [Accessed 13 April 2017].

[5]

Firebase Google. 2017. *Firebase Documentation*. [ONLINE] Available at: <https://firebase.google.com/docs/android/setup>. [Accessed 17 April 2017].

[6]

GitHub - Paul Cech. 2014. *An Android chart and graph library*. [ONLINE] Available at: <https://github.com/blackfizz/EazeGraph>. [Accessed 23 April 2017].

[7]

GitHub - Thomas Hoffmann. 2014. *Lightweight pedometer*. [ONLINE] Available at: <https://github.com/j4velin/Pedometer>. [Accessed 23 April 2017].

[8]

GitHub - Sergey Margaritov. 2017. *ColorPickerPreference for android to create colour picker in preferences. Project created as Library*. [ONLINE] Available at: <https://github.com/attenzione/android-ColorPickerPreference>. [Accessed 23 April 2017].

[9]

GitHub - Jonas Gehring. 2017. *Android Graph Library for creating zoomable and scrollable line and bar graphs*. [ONLINE] Available at: <https://github.com/appsthatmatter/GraphView>. [Accessed 25 April 2017].

Other sites used for tutorials, design and reference:

- <http://www.androidhive.info/>
- <http://www.tutorialspoint.com/>
- <http://stackoverflow.com/>
- <https://codelabs.developers.google.com/>
- <http://www.vogella.com/tutorials/AndroidTesting/article.html>
- <https://android-material-icon-generator.bitdroid.de/#section-material-icons>
- <https://mockuphone.com/#ios>
- <http://www.color-hex.com/color/ff1493>
- <https://developer.android.com/reference/java/text/SimpleDateFormat.html>
- <https://www.mathsisfun.com/binary-decimal-hexadecimal-converter.html>
- <http://www.diabeteschart.org/mgmmol.html>