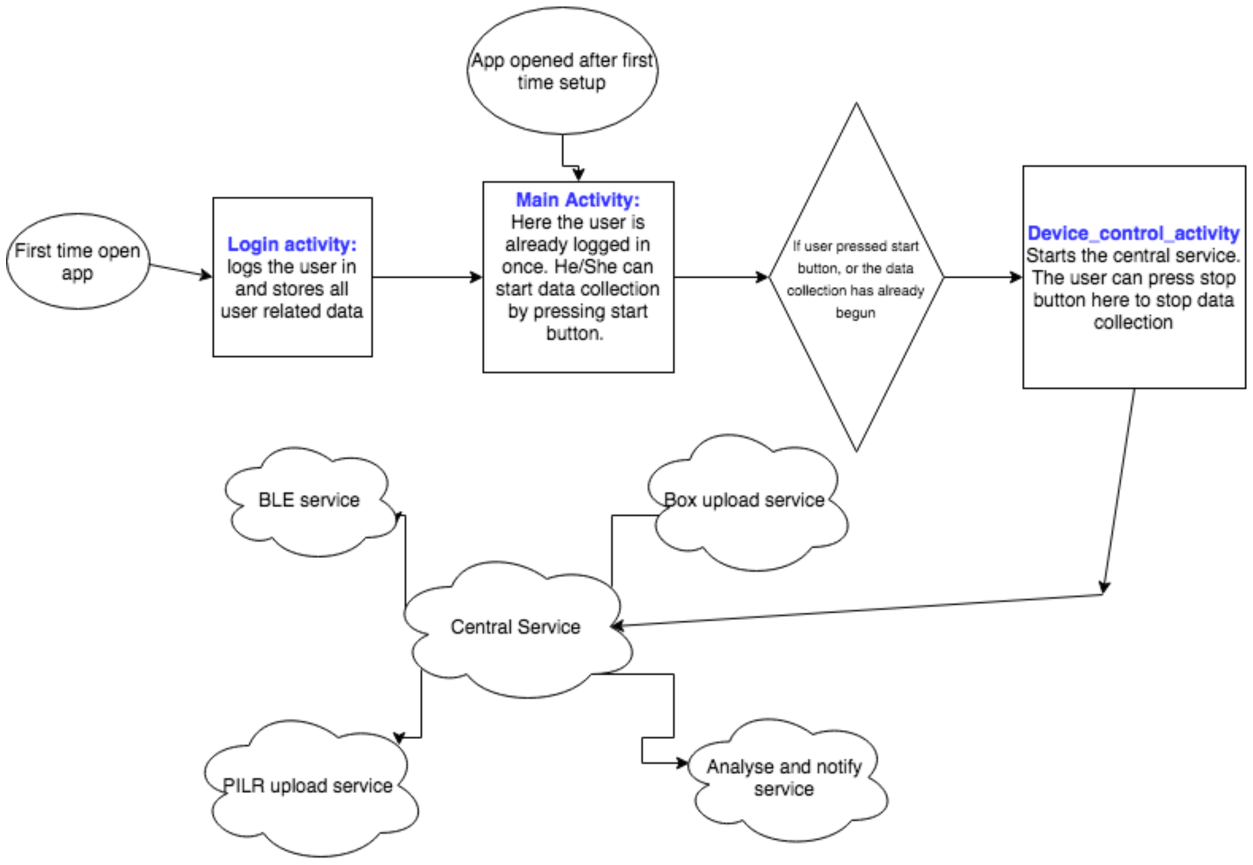
2. *Smartphone android app: Aim-health*

This app is responsible for receiving data from bluetooth, and determining whether it is accelerometer data, or image data. If it is accelerometer data, then it uploads it to PILR and in parallel, passes the data to the linear model and determines whether it is data corresponding to eating, or not eating. If the data is image data, then it stores the pace number and keeps storing these packets of byte as and when they are received in an sql database. Finally when it receives the end or last packet of image, it puts everything into a file, and gives it the same name as the micro controller informed it, and uploads it to box.

It logs a user in, using the user's PILR credentials. The user would be given a box account as well, and the user will have to login to the box account once too.

Once the user logs in and selects the appropriate device from the list of bluetooth devices displayed, he can press start, and a service would start. This service is called the central service, because it always keeps running in the background, and interacts with other services and activities. It keeps listening to bluetooth messages, and reads and responds to them. It is responsible for transferring what is read from bluetooth to other services, to upload to PILR, box , analyse, notify etc.


App opened after first time setup
First time open app
Login activity: logs the user in and stores all user related data
Main Activity: Here the user is already logged in once. He/She can start data collection by pressing start button.
If user pressed start button, or the data collection has already begun
Device_control_activity Starts the central service. The user can press stop button here to stop data collection
BLE service
Box upload service
Central Service
PILR upload service
Analyse and notify service

Each activity, and service what it does is commented in the app.

Its important to note about a few things:

The codeless has a custom service, whose uuid is “866d3b04-e674-40dc-9c05-b7f91bec6e83”

It has 3 characteristics:

Characteristic 1: “914f8fb9-e8cd-411d-b7d1-14594de45425”

Characteristic 2: “3bb535aa-50b2-4fbe-aa09-6b06dc59a404”

Characteristic 3: “e2048b39-d4f9-4a45-9f25-1856c10d5639”

Characteristic 1, is the characteristic that has to be written into by the app, to acknowledge reception.

Characteristic 2, is the characteristic that has to be read by the app, which will contain the data.

Characteristic 3, is the characteristic that has to be registered for notification by the app. When a notification is received on this characteristic, then characteristic 2 has to be read.

Details about each module in the app

Login Activity

Description:

This activity, logs the user in to their PILR account, so that the feature data received over bluetooth can be uploaded to PILR.

Important functions, and details:

i) onCreate: This function is called when the module is called (using startActivity(intent)). Here all we do is set up the email form, so that we can read their content when user updates it. We add some listeners, to call appropriate functions when user clicks submit button, when user enters the userid and password.

ii) attemptLogin(): This is called when login button is pressed. So, it checks whether email is valid or not(whether it has @ sign , etc), an checks whether password is has minimum length and then submits these to the UserLoginTask().

iii) UserLoginTask():

- This is a class, which extends a class called AsyncTask. The class Async task is used for running tasks in a separate thread, separate from the main thread. This is required for running network related operations, because accessing internet from the main thread, is going to cause a lot of lag in the running of the app. This class has the property that, whenever it is instantiated, the doInBackground() method is called, and when the task there is executed, the onPostExecute() is called. If there was an error and the task was cancelled, then the onCancelled() will be called.
- Here in UserLoginTask, what the doInBackground() method sends a GET request to the ***/api/v1/token?activation_key=*** endpoint of PILR website. Activation key here is the combined string of userid and activation key. If correct details are provided, then the website returns the user details as a JSON object, which we will take and handle in the onPostExecute() method.
- So based on whether or not the entered details were correct, we receive a corresponding JSON object. So, in onPostExecute(), we see if we have received the user details in JSON object or that we have received "access denied". If "access denied" is received then we tell the user that entered credentials are invalid. Otherwise, we store all the user details now in SharedPreferences. SharedPreferences is a type of file storage in android. We can store small amounts of data as key value pairs here, and they will be encrypted and stored in the app space, such that user cannot access or understand it.
- Some all the keys like name, project, etc will have participant id suffixed to it, since participant id is the only unique field. Once everything is stored, the MainActivity is called.

Main Activity

Description:

This is the activity that comes up first, whenever the user opens the app. If the user has not logged in, then the login activity is called. If user has logged in, and has started data collection, then device control activity is called, where the user is given an option to stop data collection. If user has logged in, but not started data collection, then the initial screen shows up, which gives the option to start day collection.

Important functions, and details:

i) **onCreate:** It checks whether the data collection service is already running in the background. If yes, it redirects to deviceControlActivity. If not, it checks if user is logged in, if yes, then it fetches the details, else redirects to login activity.

ii) **Login info is stored as:** The login info is stored in a private file which is stored as SharedPreferences, and is available in the Android library. This app is intended to work on one participant per device, so steps are taken accordingly.

iii) **Button Clicked():** If the user clicks the start button on the main screen, then this will take them to DeviceScanActivity, where user can select which bluetooth device to pair with.

Device Scan Activity

Description:

This activity, scans the bluetooth devices that are advertising nearby, and displays it. If you click on a device, then it stops scanning, and attempts to connect to the device, and proceeds to DeviceControlActivity.

Important functions, and details:

i) onListItemClick: This is the only function we have to care about. All other functions are from the default bluetooth example. Here just after connecting, let it know which activity to redirect to.

DeviceControl Activity

Description:

This basically mainly starts the central service, and gives the UI to the user, to stop it. The service keeps collecting data and uploading and analysing, in the background. We can close the app, and the service would still be running. So, now, if we open the app again, then directly DeviceControlActivity shows up, where you have the option of stopping the service. It also has a textbook, where the stuff received from bluetooth keeps showing up. It basically has a BroadCast Receiver, so, whichever service or activity broadcasts to this activity, then that data is put on the text box.

Important functions, and details:

i) **onCreate:** Here, it checks whether the central service already running in the background. If yes, then it will bind to that service, and continue to receive stuff. If not, then it will start the service.

ii) **BroadcastReceiver:** It has registered a BroadcastReceiver, which receives bluetooth connected, disconnected, data available broadcasts from Both BluetoothLeService, and CentralService. Doesn't matter which it receives, both carry the same data. [Sometimes I have noticed that it doesn't receive the broadcast sent by BluetoothLeService, so even in central service, make sure to broadcast].

iii) There is a button for stopping the Service.

Central Service

Description:

This is the most important service, which takes care of everything, and runs in the background, and can be stopped by pressing stop button in the DeviceControlActivity.

Important functions, and details:

i) **OnCreate():** It is called when service is created. It checks the shared preferences for latest connected bye address, and sets the member variable of the class to it. It also starts a notification, that shows up in the mobile, showing that it is running.

ii) **onStartCommand**(): This is called when the service is started by the deviceControlActivity. It receives the bluetooth address over the intent, and connects to it. It also registers a broadcast receiver with the BluetoothLeService, which is a Standard library service giving us several broadcasts like Connected, disconnected, Services Discovered, Data available, etc.

iii) **displayGattServices():** This function is called when central service receives a broadcast that services are discovered. What you have to do here is loop through the services, and go through their characteristics, do what you want with it. Here my Service will have an RX and a TX characteristic. I stored the TX characteristic, to write to it, and I enabled notification on the RX characteristic, so that I get notified whenever data is available.

iv) BroadcastReceiver: The code is pretty self explanatory. You can see that when DATA_AVAILABLE broadcast is received, that data is broadcasted to analyseNotify service (to do exactly that), to PILR upload service (to upload the data with timestamp to PILR), and to deviceControlActivity (To display the data in the textbox).

AnalyseNotify Service

Description:

This is a service which takes care of storing and analysing the data that comes over BLE, and also notifies the user of the analysis.

Important functions, and details:

i) **manage_data**(String data, byte[] bytes)**:** The parameters are the same data that has been given by the device, bytes carries it in raw form, whereas data is the same byte array converted into string. For example, if byte[0]=0x41, then data[0]='A'. So, basically the ascii values. Firstly, this function examines the header and determines whether it is an image or accelerometer data, and decides to take action accordingly.

   I. If it is an image, i.e It has IMG header. Then the data format is: IMGS<name=4 bytes><byte1><byte2>..<byte234>\r\0 for starting packet
   IMG<packet_no><byte1><byte2>….<byte234>\r\0 for other packets and.
   IMGE<byte1><byte2>..<byte234>\r\0 for the end packet.

      a. If it is the start packet, then, do the following:

         1.

   II.

ii) attemptLogin(): This is called when login button is pressed. So, it checks whether email is valid or not(whether it has @ sign , etc), an checks whether password is has minimum length and then submits these to the UserLoginTask().