1.Data visualisation and occupancy display app:

1.1. Need:

To display the attendance details of the lab members and visualise the data collected from the sensors deployed in Kresit building.

1.1. Files:

1.1.1. MainActivity
1.1.2. Master
1.1.3 UserDetails
1.1.4 Adapters:
1.1.4.1. CustomAdapter
1.1.4.2. CustomExpandableListAdapter
1.1.4.3. DropDownListAdapter
1.1.4.4. DropDownListAdapter2
1.1.4.5. DropDownValuesToPlotAdapter
1.1.4. Fragments(All these files have a corresponding xml file in layout directory which defines how the UI will look.)
1.1.4.1. ApplianceControl
1.1.4.2. Dashboard
1.1.4.3. FloorWiseVis
1.1.4.4. GraphSettingsDialogFragment
1.1.4.5. KresitMqttGraph
1.1.4.6. KresitVis
1.1.4.7. MembersInLab
1.1.4.8. MQTTRaw
1.1.4.9. NotInLab
1.1.4.10. SmartMeterComparison
1.1.4.11. SmartMeterLiveComparissonWebVIew
1.1.4.12. SmartMeterVisWebView
1.1.4.13. Temperature
1.1.4.14. Users
1.1.4.15. WingWiseVis
1.1.5. Utilities
1.1.5.1. HourAxisValueFormatter
1.1.5.2. Util

1.2. Dependencies:

1.2.1.APP - The app is dependent on two libraries. The first one is Paho Android Service(click on it to know more about it and how to use it), which allows the app to subscribe and publish messages to MQTT. The second one is MPAndroidChart(link to github). The documentation of this library is not good, if you want to plot multiple datasets for

live data(each line on the graph is a different dataset). Adding this library is easy and is clearly mentioned in the readme file on its github page.
This app works only when connected to IIT Wireless or any other network that can reach the Beast server which locates at 10.129.23.100. When you run the app, it sets the server's IP in the app to make it work.

1.2.2. Server - The app connects to the server to fetch and show data(except for the visualisation part). The server uses data from three different tables namely occupancy_activity, present_occupancy and temperature which are present in mysql database at 10.129.23.41. The first two tables are used to show the lab members who are present in the lab, their average working hours in a week . And to attach the attendance data in the mail (this option comes is in the user detail page when you click on 'attach' button.

1.3. Architecture:

This app has two parts, first shows data about the occupancy of the SEIL and the second part shows the visualization of the mqtt data that comes from various smart meters and temperature sensors that are deployed in the Kresit building.

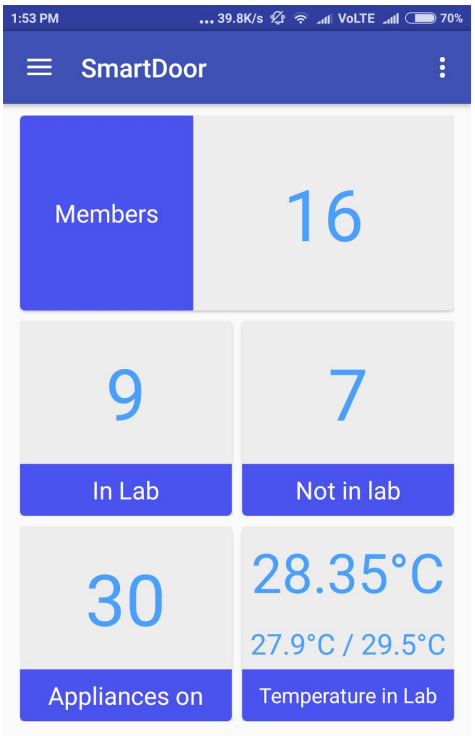

Fig 1.1. Home Screen

In the home screen(Fig 1.1) of the app, it shows the members of the lab that are currently present in the lab, average and min/max lab temperature . Tapping on any of these takes

you to a new screen which shows in detail about that item. These items can also be accessed through the navigation drawer.

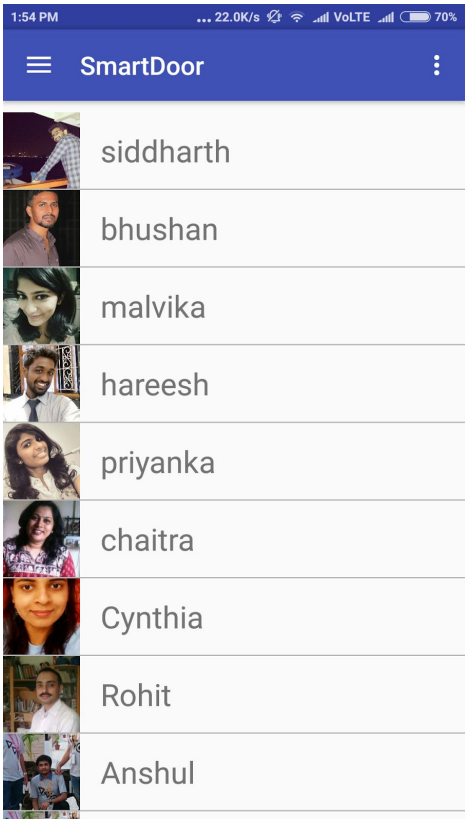

Fig 1.2. Users Page

This page(Fig 1.2) displays the list of all the users(Users.java). Clicking on any of the user will take you to the user details page.

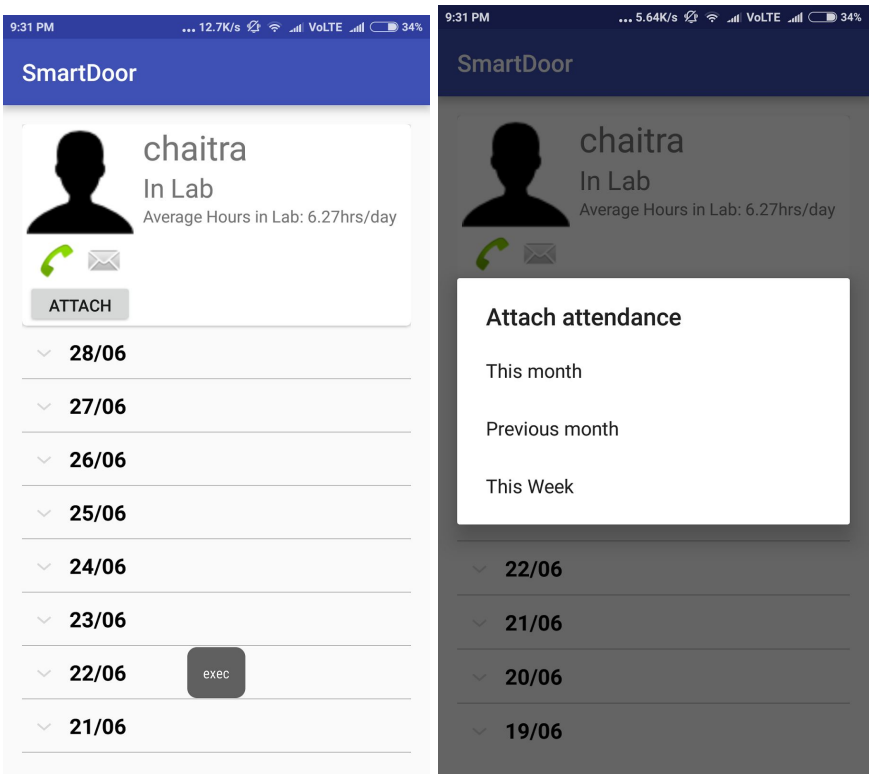

Fig 1.3. User Details

The user details page(Fig 1.3) include the average working hours of lab members every week, attendance of the user for the past seven days, options to call/mail them and attach their attendance while sending a mail. The function to attach attendance gives you three different time periods of which you can attach attendance as shown in the figure(Fig 1.3). The future work in this page can be to display the user's photo instead of the placeholder image.

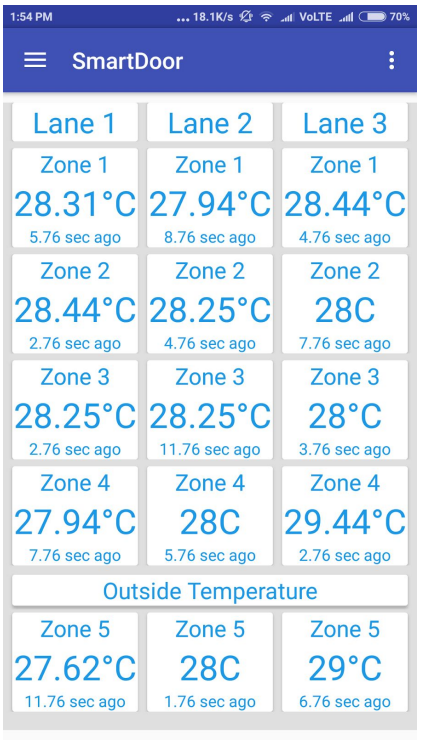

Fig 1.4. Temperature of SEIL

SEIL is divided into lanes and zones for collecting temperature data. There are 3 lanes and each lane is divided into 5 zones. Each of these zones has a temperature and humidity sensor. Zone 5 of each lane is outside the lab. This page(Fig 1.4) displays the temperature details of each of these zones and the time when this was updated into the database. It uses the data from ‘temperature’ table in mysql database located at 10.129.23.41.
This page was build to make sure that all the temperature sensors are logging data properly. The time between which each of these zones should log the temperature should not be more than few seconds.

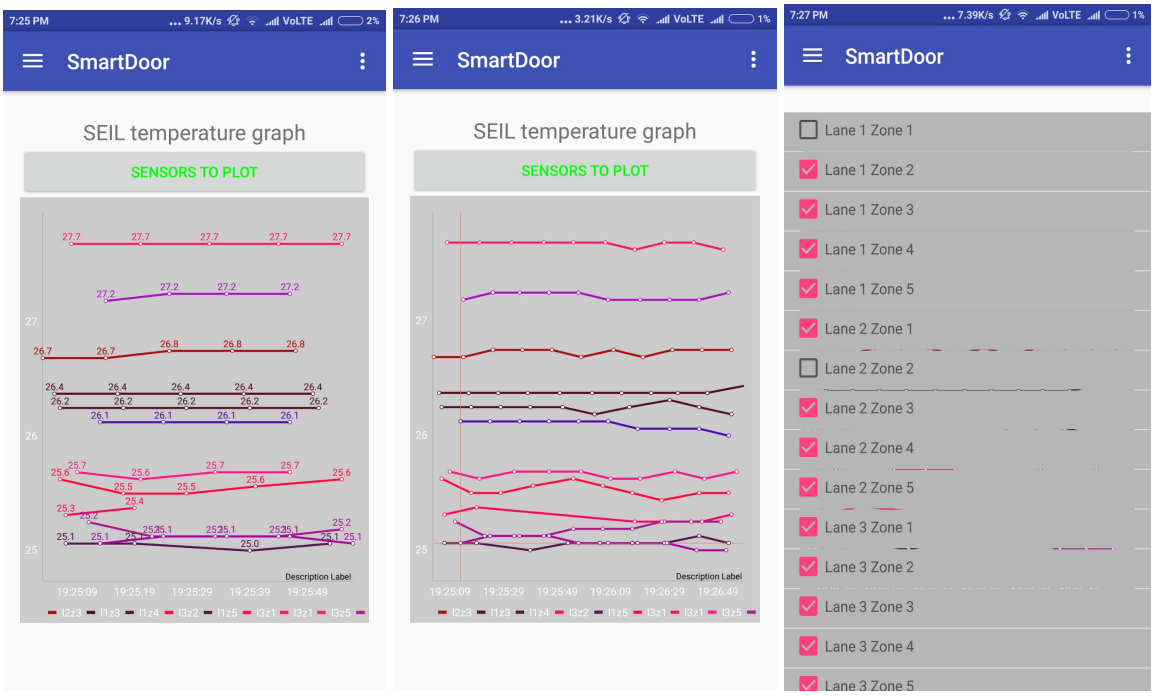

Fig 1.5. SEIL Temperature Graph

This page(Fig 1.5) shows the graph of all the temperature sensors in SEIL. The X-axis displays the time and the Y-axis displays the temperature in degree celsius. This graph does not depend on the database and displays the live data as it is comes from these sensors using MQTT. Plotting this graph requires two different libraries. The first one is Paho MQTT which is used to get data from MQTT server and the second one is MPandroidChart which is used to plot the data. You can select which all temperature sensors data you want to plot. At the bottom of the graph there are rectangles of different color, where each color corresponds to the different lines in the graph. The text next to these rectangles show which temperature sensor that color represents. When there are few data points it displays the temperature values on top of it but when there are many data points it only shows the line to avoid cluttering. You can zoom in/out in any area of the graph.

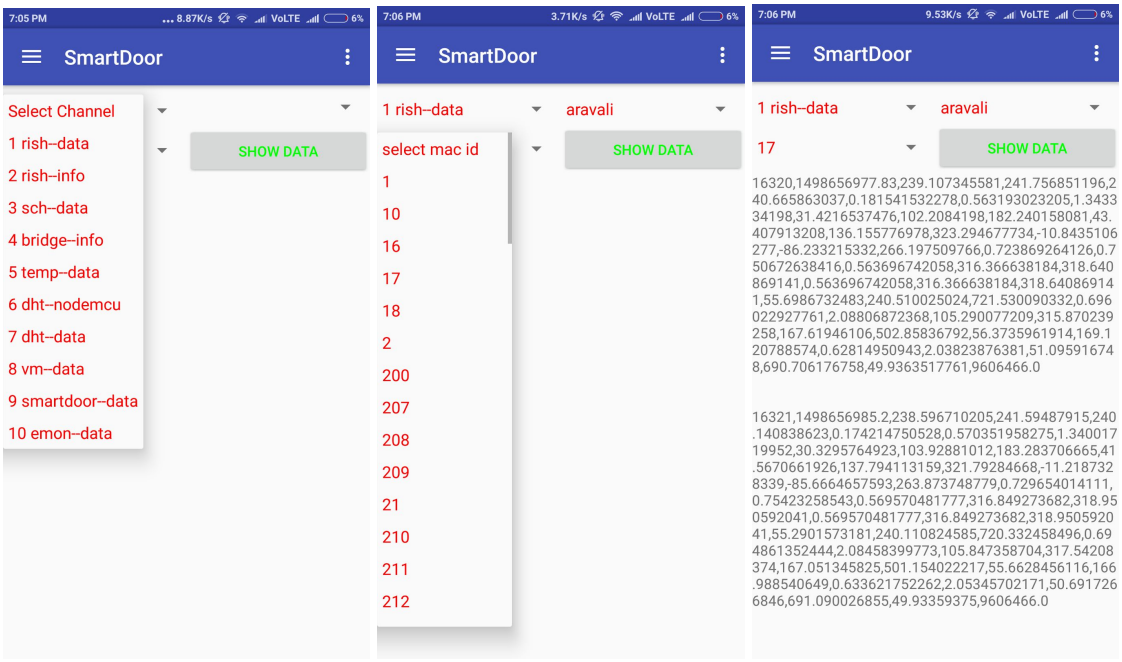

Fig 1.7. Raw MQTT data

This page(Fig 1.7) displays raw MQTT data from all the MQTT channels. It loads all the channel names from the server located at http://10.129.23.41:8080/meta/channels/ when it is opened, so if a new channel get added on the server we can see it directly here without making changes to the app. This page was built for debugging purposes. We can see from this page which all MQTT channels are working properly.

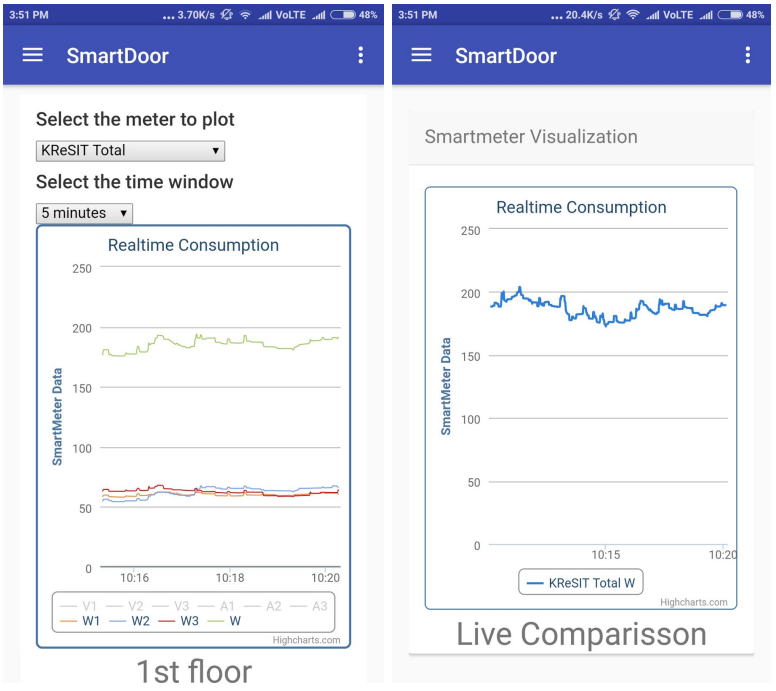

Fig 1.8. Smart meter visualisation

These two pages display the smart meter visualisation system that is at http://www.cse.iitb.ac.in/smartmeter/.

1.4. Known Bugs

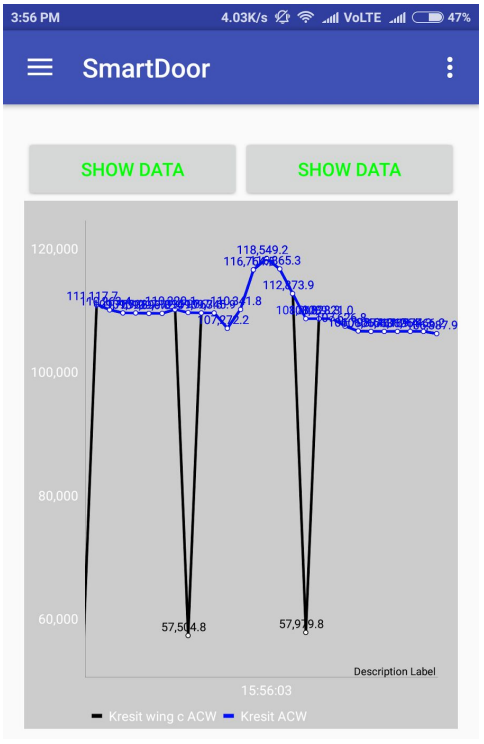

Fig 1.9. Bug in plotting MQTT data of KRESIT smart meters

There is a bug in the graph plotting MQTT data for kRESIT smart meters. As you can see in the graph above the top and the bottom points should form two separate lines but they are getting combined into one. To solve this bug put breakpoints in the ‘createSet’ and ‘addEntry’ functions in KresitMqttGraph.java and debug the code.