Environment:

The experiments were carried out in a 34 feet long x 32 feet wide indoor home environment, the layout has been shown in Fig.1.

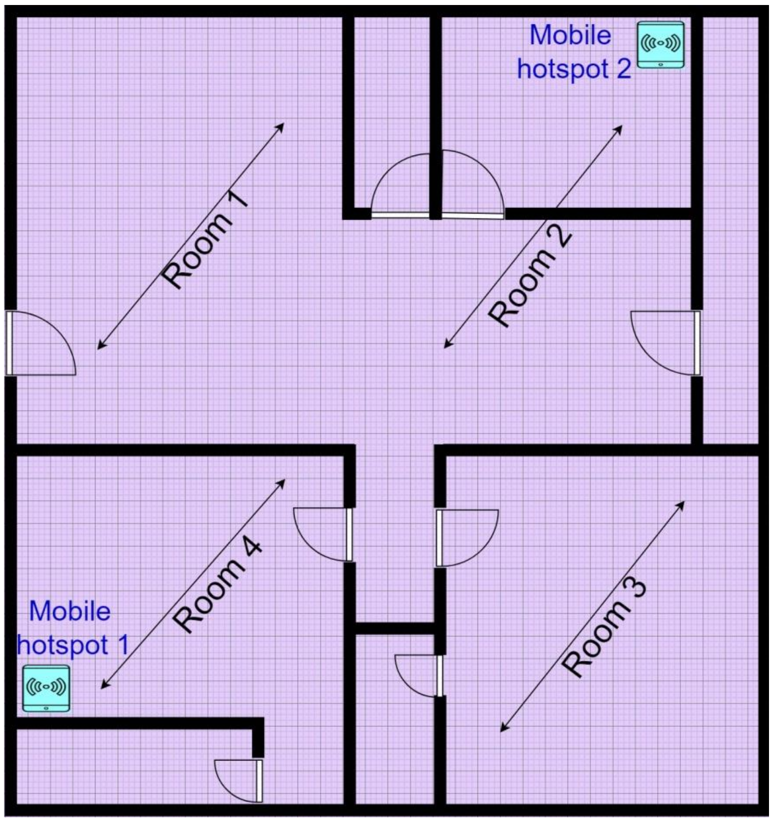


Fig.1. Layout of indoor environment.

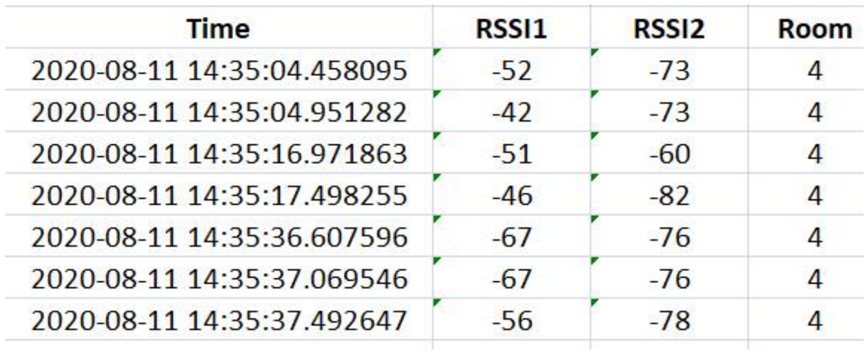
Two mobile hotspots have been used: 1.Motorola G5 mobile and 2.Realme C2. Another mobile Realme2pro has been used for navigation during both training and testing phases. Throughout the experiments either in training phase or testing phase, the mobile hotspots 1 and 2 were kept at the same positions as shown in Fig.1 and their positions have not been disturbed.

The entire environment is classified into 4 target locations Rooms 1 to 4 as shown in Fig.1.

Data collection and Training Phase:

Prior to training, data collection needs to be done Data has been collected individually from each room.

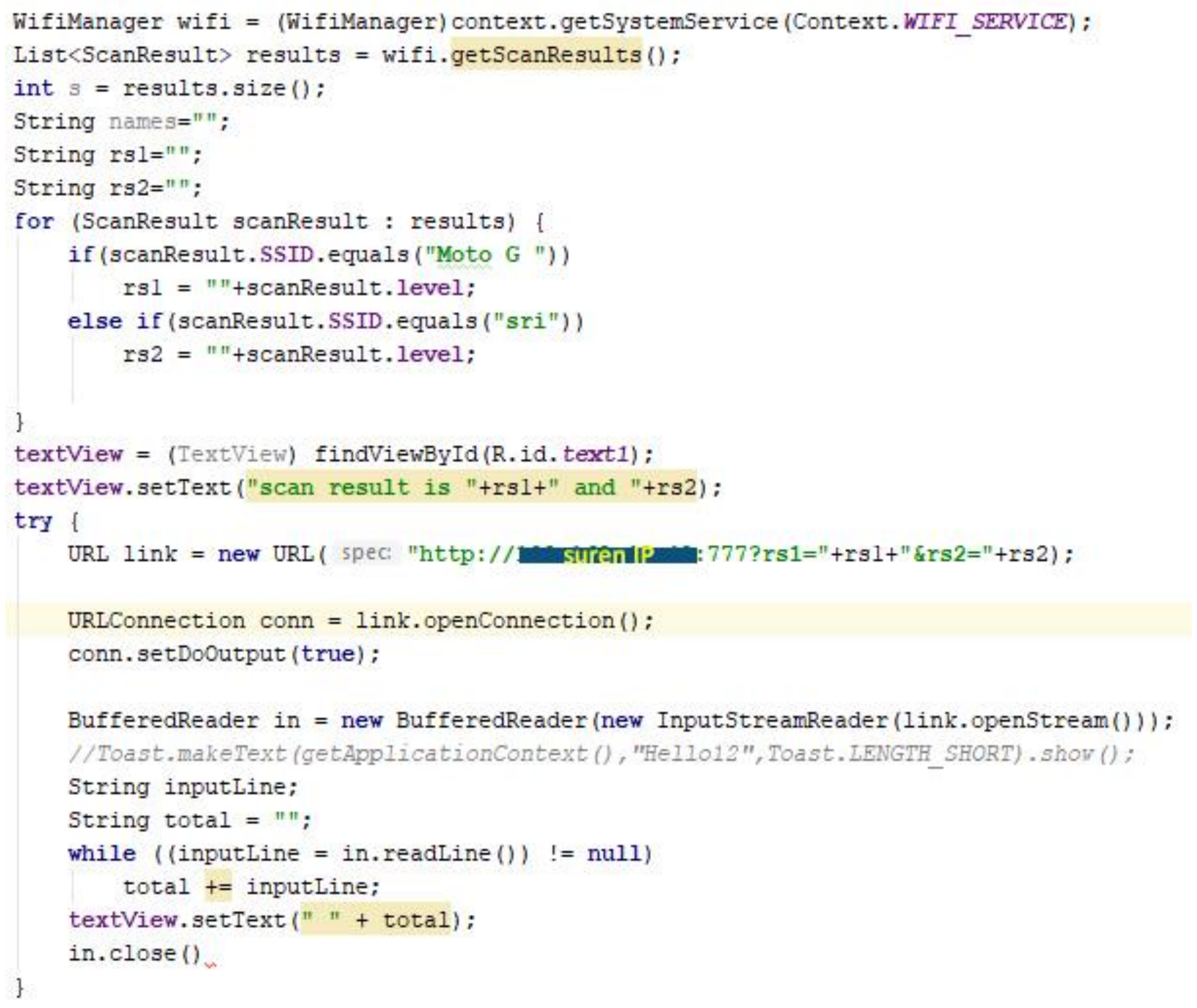
If the data is collected from a particular Room, let’s say Room 4, for this the mobile is carried by a moving person in Room 4 only. The person tries to navigate the mobile through all the space possible in Room 4. The designed android app continuously sends RSSI(Received Signal Strength Indicator) values read from the two mobile hotspots to the server where the data set is maintained in the form of an excel sheet.



Time	RSSI1	RSSI2	Room
2020-08-11 14:35:04.458095	-52	-73	4
2020-08-11 14:35:04.951282	-42	-73	4
2020-08-11 14:35:16.971863	-51	-60	4
2020-08-11 14:35:17.498255	-46	-82	4
2020-08-11 14:35:36.607596	-67	-76	4
2020-08-11 14:35:37.069546	-67	-76	4
2020-08-11 14:35:37.492647	-56	-78	4

Fig.2. Sample of Data collected from Room 4

A sample collection of data points at various places within room 4 has been shown in Fig.2. RSSI1 and RSSI2 represent the RSSI values of mobile hotspots 1 and 2 respectively. Similarly data points were collected from all the rooms.



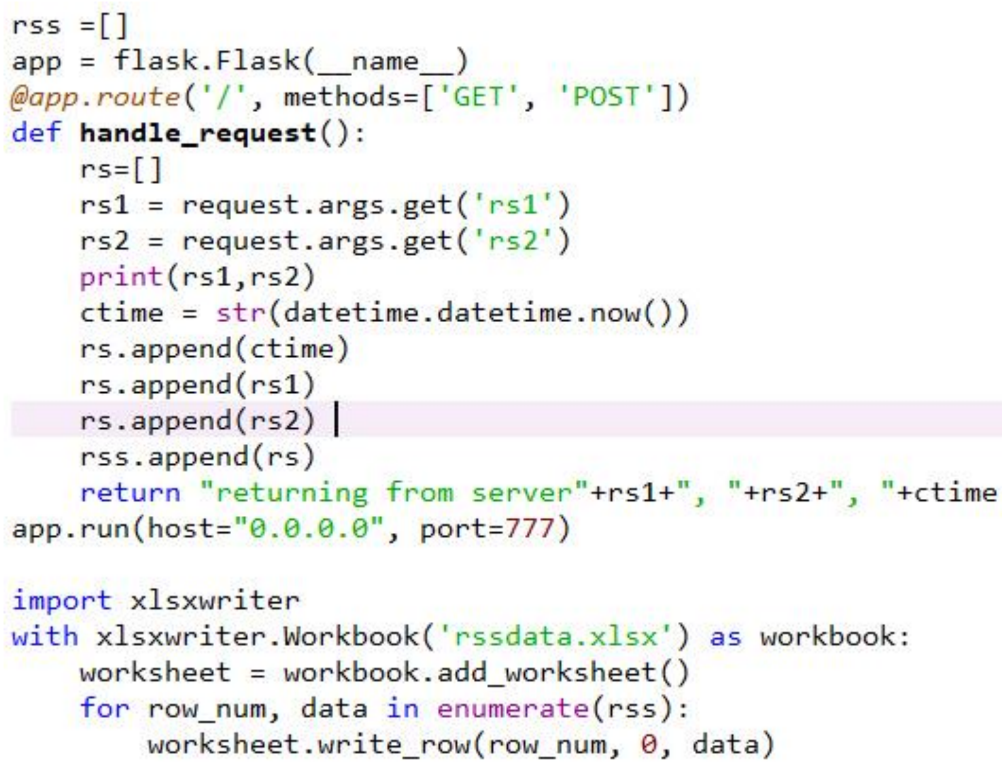
```
WifiManager wifi = (WifiManager)context.getSystemService(Context.WIFI_SERVICE);
List<ScanResult> results = wifi.getScanResults();
int s = results.size();
String names="";
String rs1="";
String rs2="";
for (ScanResult scanResult : results) {
    if(scanResult.SSID.equals("Moto G "))
        rs1 = ""+scanResult.level;
    else if(scanResult.SSID.equals("sri"))
        rs2 = ""+scanResult.level;

}
textView = (TextView) findViewById(R.id.text1);
textView.setText("scan result is "+rs1+" and "+rs2);
try {
    URL link = new URL( spec: "http://[redacted]:777?rs1="+rs1+"&rs2="+rs2);

    URLConnection conn = link.openConnection();
    conn.setDoOutput(true);

    BufferedReader in = new BufferedReader(new InputStreamReader(link.openStream()));
    //Toast.makeText(getApplicationContext(),"Hello12",Toast.LENGTH_SHORT).show();
    String inputLine;
    String total = "";
    while ((inputLine = in.readLine()) != null)
        total += inputLine;
    textView.setText(" " + total);
    in.close()
}
```

Fig.3. code snippet from Android Studio

The code snippet from Android Studio is shown in Fig.3 which does the following tasks:

1. Scans the RSSI values of all WiFi's available.
2. Stores only the required values from the 2 mobile hotspots.
3. Sends the two RSSI's to the server's IP using URL mentioned in code.
4. Receives response from the server if any and displays it.

The above code has been repeated for every 500 milliseconds so that fresh scan results are sent to the server.

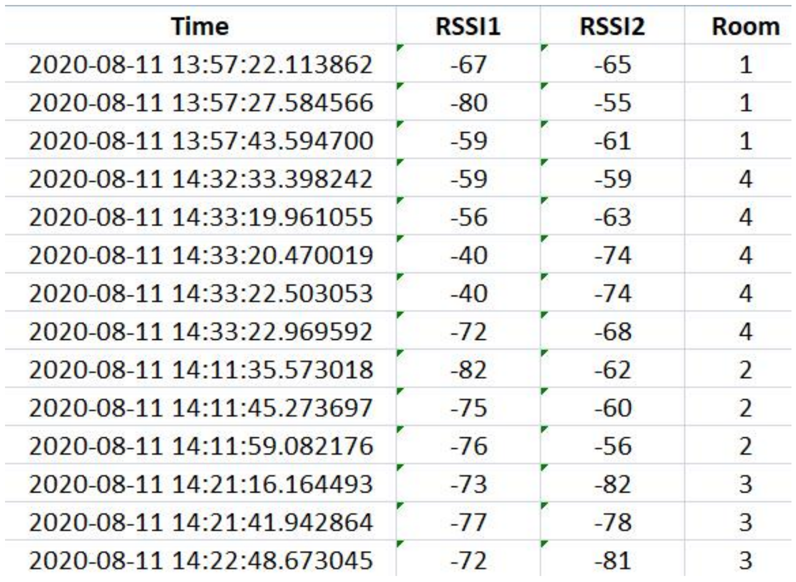


```
rss =[]
app = flask.Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def handle_request():
    rs=[]
    rs1 = request.args.get('rs1')
    rs2 = request.args.get('rs2')
    print(rs1,rs2)
    ctime = str(datetime.datetime.now())
    rs.append(ctime)
    rs.append(rs1)
    rs.append(rs2)
    rss.append(rs)
    return "returning from server"+rs1+", "+rs2+", "+ctime
app.run(host="0.0.0.0", port=777)

import xlsxwriter
with xlsxwriter.Workbook('rssdata.xlsx') as workbook:
    worksheet = workbook.add_worksheet()
    for row_num, data in enumerate(rss):
        worksheet.write_row(row_num, 0, data)
```

Fig.4. Code snippet of python which uses flask server.

The python code snippet shown in Fig.4 does the following tasks.

1. Initializes flask server.
2. Receives requests which contain RSSI values.
3. Responds back with the same RSSI values for confirming acknowledgement.
4. Writes all the RSSI values into excel sheet along with the current time.

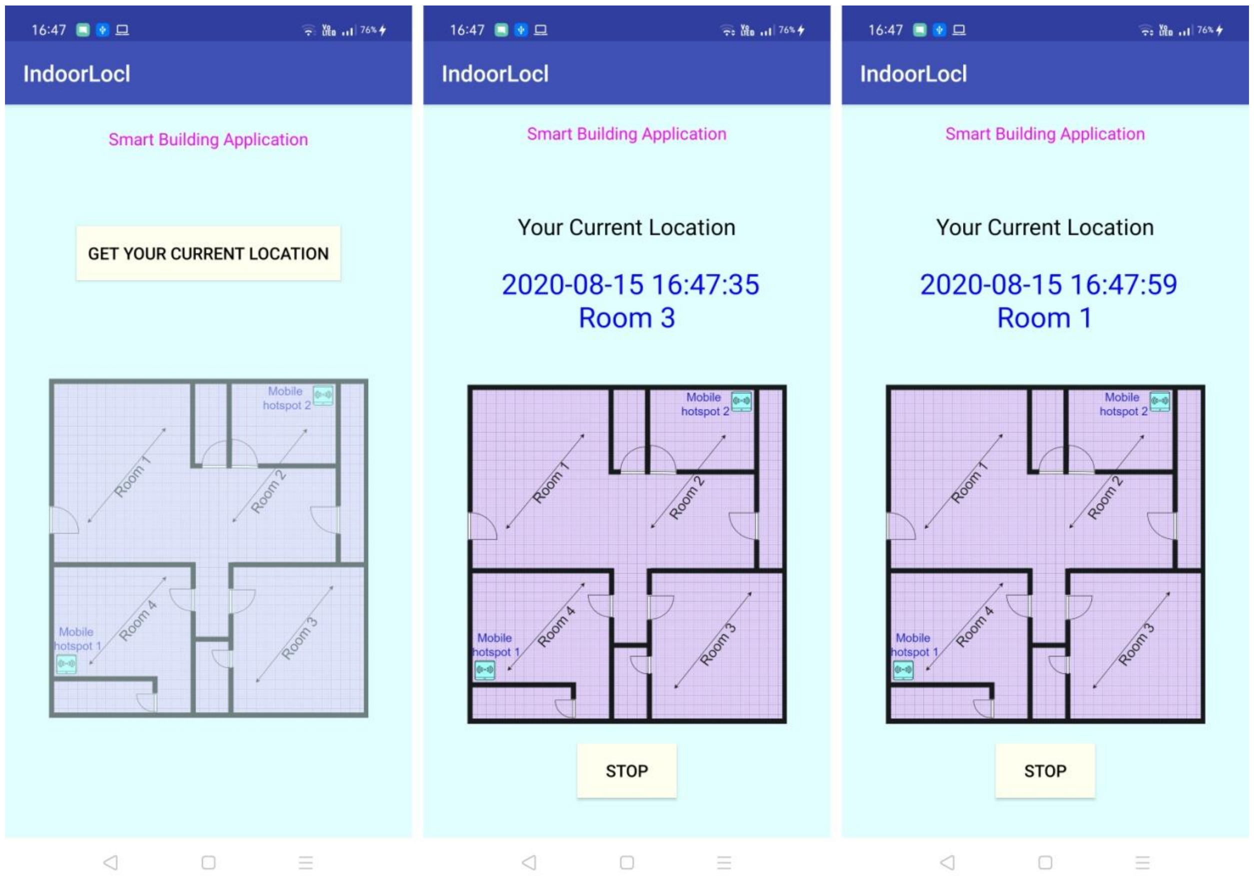


Time	RSSI1	RSSI2	Room
2020-08-11 13:57:22.113862	-67	-65	1
2020-08-11 13:57:27.584566	-80	-55	1
2020-08-11 13:57:43.594700	-59	-61	1
2020-08-11 14:32:33.398242	-59	-59	4
2020-08-11 14:33:19.961055	-56	-63	4
2020-08-11 14:33:20.470019	-40	-74	4
2020-08-11 14:33:22.503053	-40	-74	4
2020-08-11 14:33:22.969592	-72	-68	4
2020-08-11 14:11:35.573018	-82	-62	2
2020-08-11 14:11:45.273697	-75	-60	2
2020-08-11 14:11:59.082176	-76	-56	2
2020-08-11 14:21:16.164493	-73	-82	3
2020-08-11 14:21:41.942864	-77	-78	3
2020-08-11 14:22:48.673045	-72	-81	3

Fig.5. Sample of Data collected from all the rooms.

A sample collection of data points from all the 4 rooms at different instances is shown in Fig.5. In this way a data set of total 3495 data points is collected from all the 4 rooms combined.

Testing Phase:



Your Current Location
2020-08-15 16:47:35
Room 3

Your Current Location
2020-08-15 16:47:59
Room 1

Fig.6. Screenshots of android application

Three screenshots of android application were shown in Fig.6. The first screenshot from the left presents the home page of the application, which contains the layout image with less opacity and a button to get the current location.

Whenever user clicks on this button, then the current location of the mobile along with the time returned from server is displayed as shown in the middle and right screenshots. The location of the navigating mobile gets updated each second.

Apart from real time implementation, testing is done using python for calculating accuracy over a large data set and the summary of results is as follows.

Summary of Testing Results	
Total Data Set	3495 data points
Training Data Set	2621 data points
Testing Data Set	874 data points
Predicted locations matched	788 data points
Predicted locations unmatched	86 data points
Accuracy	90.16%

Table 1. Summary of Testing Results

As said earlier that 3495 data points were collected, this data set was split into training data set with 2621 data points and testing data set with 874 data points. The kNN classifier is trained with the training data set and then it was tested. During testing, the kNN classifier could predict 788 locations correctly.