Noise - SNR Collector Final Report

Version <1.0>

2015719243

Cem ŞANAL

18/12/2017

CONTENT

1. PROJECT OVERVIEW

1.1. Project Name: Noise - SNR Collector

1.2. Introduction

Noise – SNR Collector is a user oriented and mathematical ANDROID application. Mobile network signal levels are understood via finding signal-to-noise ratio. Users could see signal levels via their mobile phones by walking on their destination and they could see signal levels in different areas from the data fetched from the database. SNR data is saved according to its location. Users could see other users SNR data. Signal analysis and network speed assumptions could be performed. Routes could be drawn.

1.3. Project Definition

Briefly, Noise meter for ANDROID Mobile Phones. In this project, the aim is to design and development of signal-to-noise ratio (SNR) measuring software for ANDROID based mobile phones. The volume of sounds using the ANDROID system helps to measure the noise (Noise Meter). An apparatus for comparing sound intensity levels usually in decibels. The signal quality could be easily understood. Mandatory features: Average, min, max and peak signal to noise ratios in dB will be shown with a nice interface. Users would draw routes from the map. Determining and showing the best and the worst path according to signal levels is done by the software. Every created route path is determined according to users SNR data.

1.4. Github link

https://github.com/sanal-cem/Noise-SNR-Collector

1.5. Google Play link

https://play.google.com/store/apps/details?id=com.snrc.noise.noise_snr_collector

1.6. Languages

English and Turkish

2. FUNCTIONAL REQUIREMENTS

1.1. Calculation Requirements

1.1.1. The signal-to-noise ratio (SNR) of the connected mobile network should be calculated. – FREQ01

1.1.2. The peak signal-to-noise ratio (PSNR) of the connected mobile network should be calculated. – FREQ02

1.1.3. The average signal-to-noise ratio of the connected mobile network should be calculated. – FREQ03

1.1.4. The minimum signal-to-noise ratio of the connected mobile network should be calculated. – FREQ04

1.1.5. The maximum signal-to-noise ratio of the connected mobile network should be calculated. – FREQ05

1.1.6. New route from current or clicked location to the destination should be calculated. – FREQ06

1.2. User Interface Requirements

1.1.7. The interface should be user friendly. – FREQ07

1.1.8. The user would see a graph which compares sound intensity levels. – FREQ08

1.1.9. Sound intensity levels would be shown in decibels. – FREQ09

1.1.10. Sound intensity levels would also be shown in the other required formats. – FREQ10

1.1.11. The signal-to-noise ratio (SNR) should be shown in the graph in the main activity. – FREQ11

1.1.12. The peak signal-to-noise ratio (PSNR) should be shown in the graph in the main activity. – FREQ12

1.1.13. The average signal-to-noise ratio should be shown in the graph in the main activity. – FREQ13

1.1.14. The minimum signal-to-noise ratio should be shown in the graph in the main activity. – FREQ14

1.1.15. The maximum signal-to-noise ratio should be shown in the graph in the main activity. – FREQ15

1.1.16. The SNR data would be shown on a map with colors while Noise – SNR Collector is working in a separated activity. – FREQ16

1.1.17. The adjusted route should be shown on the same map while Noise – SNR Collector is working in a separated activity. – FREQ17

1.1.18. The graph should be refreshed in at least 2 seconds interval. – FREQ18

1.3. Data Processing Requirements

1.1.19. Individually saving signal to noise ratio data based on location data and time. – FREQ19

1.1.20. Sound intensity levels should be saved as decibels. – FREQ20

1.1.21. The Signal-to-noise ratio should be calculated and saved on the decibels scale. – FREQ21

1.1.22. The average signal-to-noise ratio should be calculated and saved on the decibels scale. – FREQ22

1.1.23. The minimum signal-to-noise ratio should be calculated and saved on the decibels scale. – FREQ23

1.1.24. The maximum signal-to-noise ratio should be calculated and saved on the decibels scale. – FREQ24

1.1.25. Saved SNR and location data should be sent to the application server regularly. – FREQ25

1.1.26. MYSQL database via Amazon RDS would be used as database server. – FREQ26

1.1.27. Saved SNR data should be fetched regularly by each client. – FREQ27

1.4. Users' Profile Requirements

1.1.28. A user account should have a mail address. – FREQ28

1.1.29. A user account would have an id. – FREQ29

1.1.30. A user account would have a password. – FREQ30

3. NON-FUNCTIONAL REQUIREMENTS

1.1. System

1.1.1. The project should be operated in the ANDROID operating system. – NREQ01

1.1.2. The ANDROID system should be between 4.0 to 8.0 versions. – NREQ02

1.1.3. When a new software update arrives, user should not use the old version. – NREQ03

1.1.4. Software should work as a background service whether or not program will be closed. – NREQ04

1.2. Performance

1.2.1. **(Response Time <= 1sec):** The system should be able to respond to a request of a user in not more than a second. – NREQ05

1.2.2. **Throughput:** Multi-tasking should be enabled to allow multiple users, initially at least 10 simultaneous users should interact with the system without having to wait others to finish their work with the system. – NREQ06

1.3. Scalability

1.3.1. Simultaneous users should be increased gradually after the first release. – NREQ07

1.4. Capacity

1.4.1. Initially at least 10 simultaneous users should interact with the system. – NREQ08

1.5. Maintenance

1.5.1. When a potential problem or a security threat has been recognized inside the system, maintenance might be performed as an update. – NREQ09

1.6. Availability

1.6.1. The system should be available day and night except the maintenance times and when updates are needed. – NREQ10

1.7. Security

1.7.1. The system should be secure for the server side. – NREQ11

1.7.2. The system should be secure both for the client side. – NREQ12

1.8. Standards

1.8.1. Decibels should be used as the standard format. – NREQ13

1.8.2. Other formats should be changed to decibels before saving. – NREQ14

4. DESIGN

1.1. System Components

1.1.1. Users Mobile phone

1.1.2. Cell Tower (LTE, 4.5g, 3g, 2g)

1.1.3. Wi-fi Router

1.1.4. PHP Server

1.1.5. MYSQL Database

1.2. Diagrams

1.2.1. Use Case Diagram

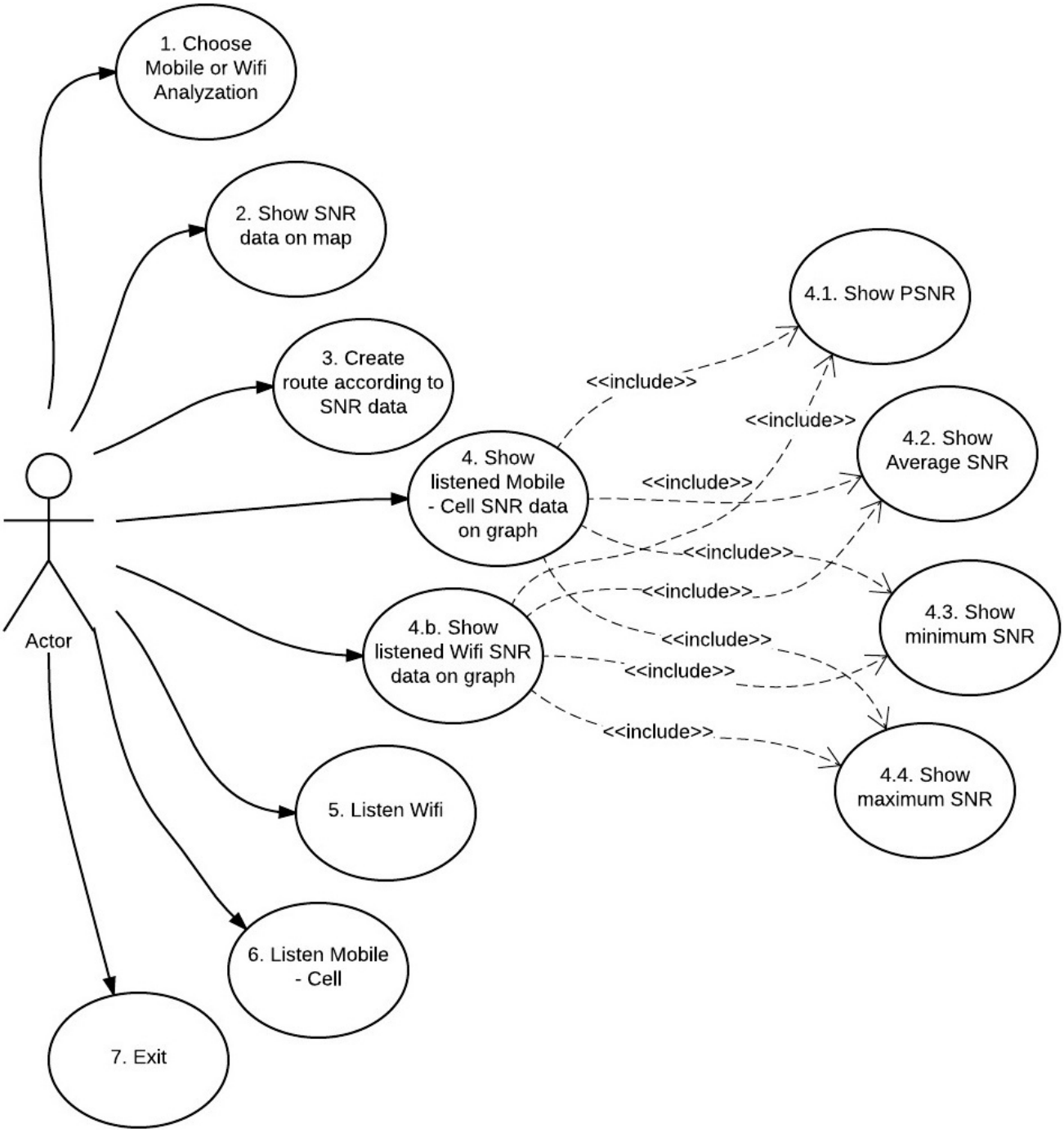


Figure 1: Use Case Diagram

1.2.2. Use Case Descriptions

4.1.1.1. Choosing Mobile or Wifi Analyzation

Condition	Definition
Id:	UC01
Title:	Choosing Mobile or Wifi Analyzation.
Description:	User should enable and choose Wifi or Mobile - Cell for analyzation. If the user did not enable chosen option, "please enable Wifi network or please enable Mobile network" questions will be prompted.
Primary Actor:	User
Preconditions:	Main Activity should be opened.
Postconditions:	Navigation activity of wifi or mobile - cell will be opened or "please enable Wifi network or please enable Mobile network" questions will be prompted.
Main Success Scenario:	Navigation activity of Wifi or Mobile - Cell will be opened.
Frequency of Use:	Once per app execution.
Status:	Finished.
Priority:	3/5

4.1.1.2. Show SNR Data on map

Condition	Definition
Id:	UC02
Title:	Show SNR Data on map.
Description:	SNR data should be shown to the user clearly.
Primary Actor:	User
Preconditions:	Map View Activity should be opened and SNR data should be fetched from the database.
Postconditions:	SNR data will be seen with tapping.
Main Success Scenario:	SNR data is shown properly to the user in a recognizable level.
Frequency of Use:	Once when Map View Activity is opened.
Status:	Finished. - More development could be done.
Priority:	5/5

4.1.1.3. Create route according to SNR data

Condition	Definition
Id:	UC03
Title:	Create route according to SNR data.
Description:	New route according to users target should be created. User should indicate the destination. Application should automatically create a new route according to SNR values in the database.
Primary Actor:	User
Preconditions:	Map View Activity should be opened.
Postconditions:	Created route should be shown to the user or "no route could be created" message should be shown.
Main Success Scenario:	Created route is shown to the user.
Frequency of Use:	As many times as users wish.
Status:	In development.
Priority:	4/5

4.1.1.4. Show listened Mobile - Cell SNR data on graph

Condition	Definition
Id:	UC04
Title:	Show listened Mobile - Cell SNR data on graph.
Description:	Mobile - Cell SNR data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Graph View Activity should be opened.
Postconditions:	SNR data should be sent to the application database (CSNR table) via PHP regularly if SNR data could be read properly.
Main Success Scenario:	SNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.5. Show listened Wifi SNR data on graph

Condition	Definition
Id:	UC04b
Title:	Show listened Wifi SNR data on graph.
Description:	Wifi data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Graph View Activity should be opened and wifi data should be listened.
Postconditions:	SNR data should be sent to the application database (WSNR table) via PHP regularly if SNR data could be read properly.
Main Success Scenario:	SNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.6. Show PSNR

Condition	Definition
Id:	UC041
Title:	Show PSNR data in the Graph View Activity.
Description:	PSNR data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Wifi or Mobile - Cell data should be listened.
Postconditions:	-
Main Success Scenario:	PSNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Not Finished. Only implemented for finding image PSNR. For finding signal PSNR, mathematical calculations are needed.
Priority:	5/5

4.1.1.7. Show Average SNR

Condition	Definition
Id:	UC042
Title:	Show Average SNR data in the Graph View Activity.
Description:	Average SNR data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Wifi or Mobile - Cell data should be listened.
Postconditions:	-
Main Success Scenario:	Average SNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.8. Show Minimum SNR

Condition	Definition
Id:	UC043
Title:	Show Minimum SNR data in the Graph View Activity.
Description:	Minimum SNR data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Wifi or Mobile - Cell data should be listened.
Postconditions:	-
Main Success Scenario:	Minimum SNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.9. Show Maximum SNR

Condition	Definition
Id:	UC044
Title:	Show Maximum SNR data in the Graph View Activity.
Description:	Maximum SNR data should be shown regularly to the user in the Graph View Activity.
Primary Actor:	User
Preconditions:	Wifi or Mobile - Cell data should be listened.
Postconditions:	-
Main Success Scenario:	Maximum SNR data is shown properly to the user.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.10. Listen Wifi

Condition	Definition
Id:	UC05
Title:	Listen Wifi.
Description:	Signal from the Wifi module should be listened.
Primary Actor:	User
Preconditions:	Wifi Graph View Activity should be opened.
Postconditions:	-
Main Success Scenario:	Wifi signal data should be sent to the graph view function.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.11. Listen Mobile – Cell

Condition	Definition
Id:	UC06
Title:	Listen Mobile - Cell.
Description:	Signal from the Mobile - Cell module should be listened.
Primary Actor:	User
Preconditions:	Mobile Graph View Activity should be opened.
Postconditions:	-
Main Success Scenario:	Mobile - Cell signal data should be sent to the graph view function.
Frequency of Use:	Continuously when Graph View Activity is opened.
Status:	Finished.
Priority:	5/5

4.1.1.12. Exit

Condition	Definition
Id:	UC07
Title:	Exit
Description:	Application should be closed when Exit clicked from the Navigation Activity or phones back button pressed many times.
Primary Actor:	User
Preconditions:	-
Postconditions:	-
Main Success Scenario:	Software is closed.
Frequency of Use:	Once when software is opened.
Status:	Finished.
Priority:	5/5

1.2.3. Activity Diagram

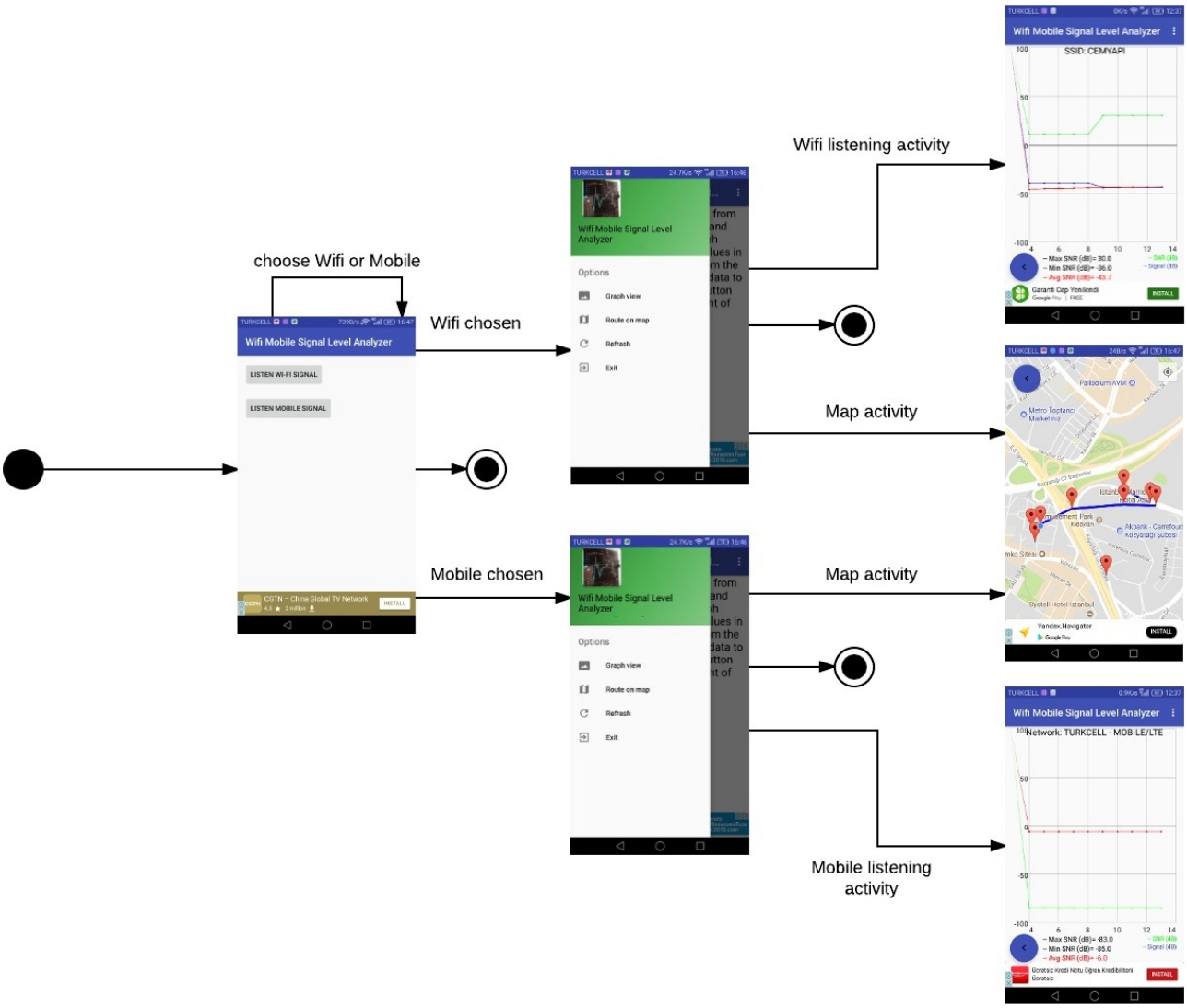


Figure 2: Activity Diagram

1.2.4. Package Diagram

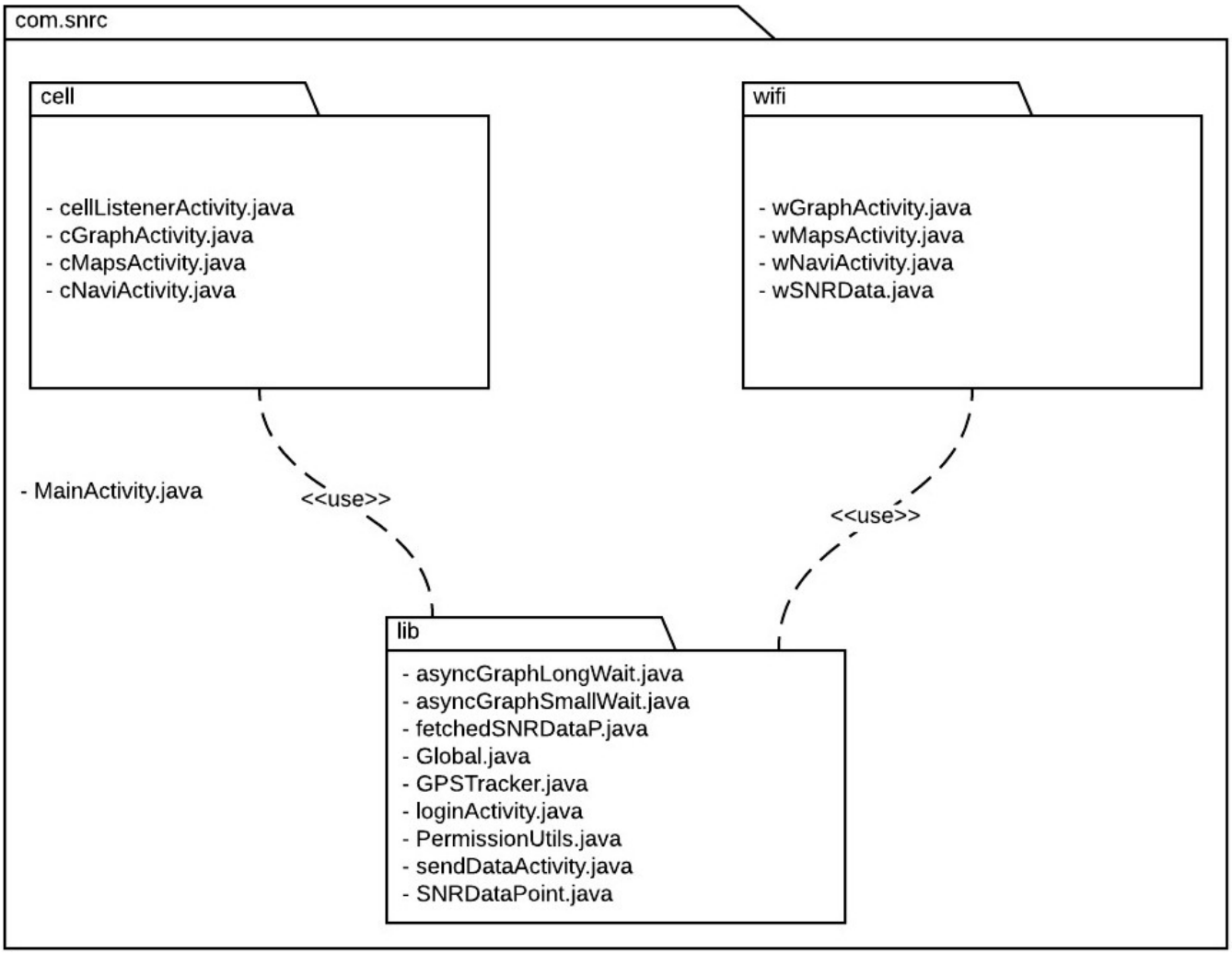


Figure 3: Package Diagram

1.2.5. Data Flow Diagram

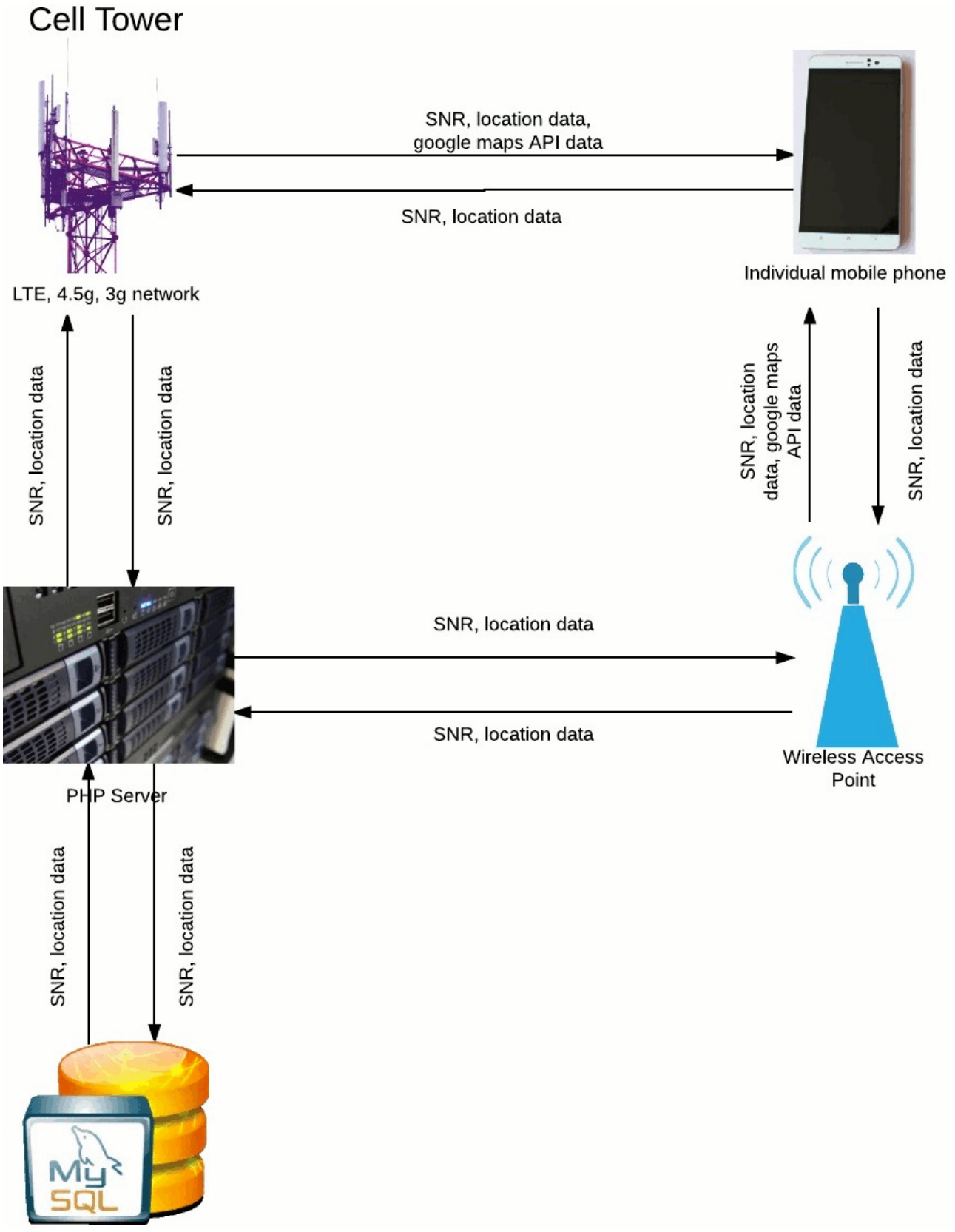


Figure 4: Data Flow Diagram

1.2.6. cell Package Class Diagram

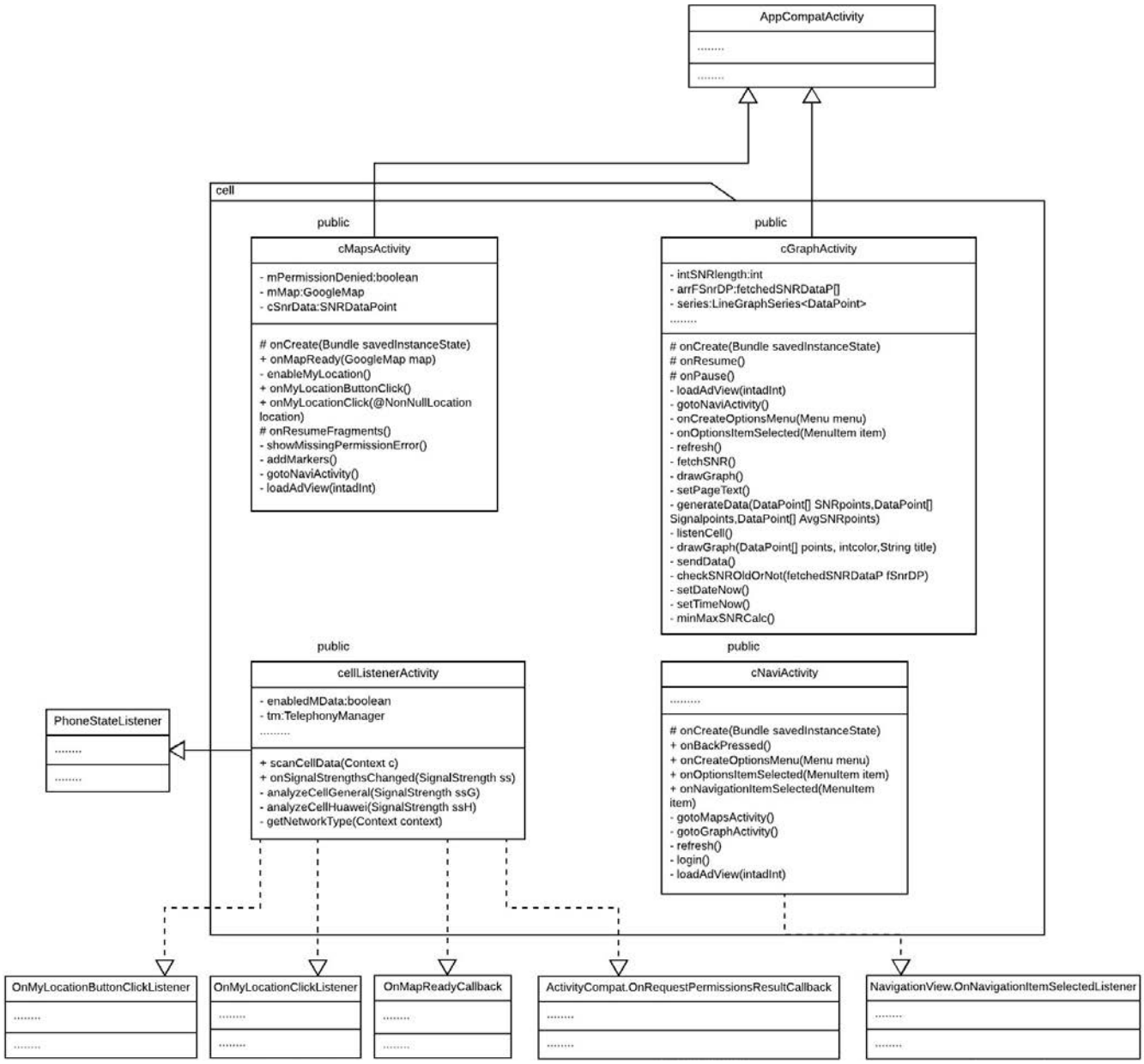


Figure 5: cell Package Class Diagram

1.2.7. lib Package Class Diagram

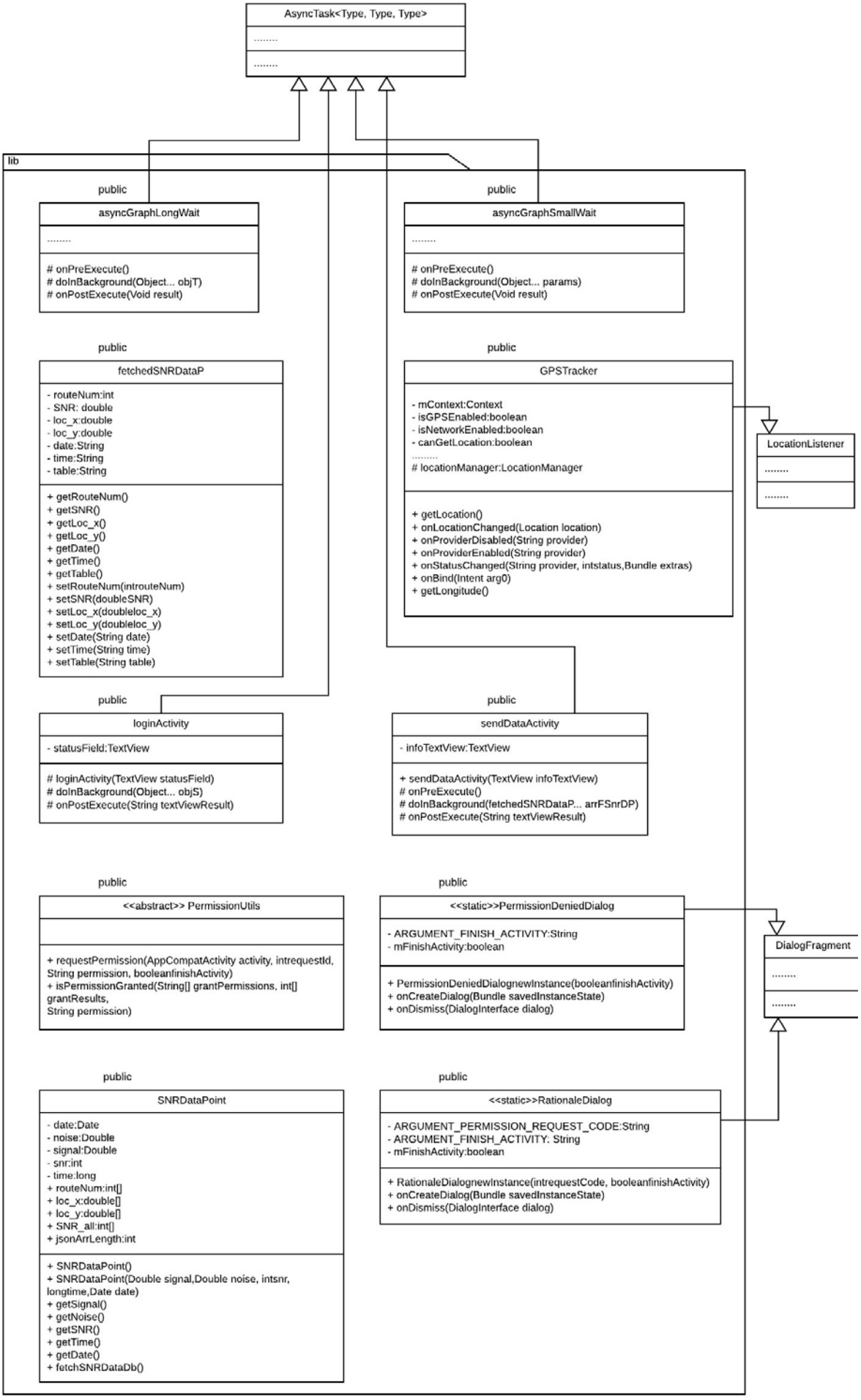


Figure 6: lib Package Class Diagram

1.2.8. Analyzing Wifi Mobile Signal General Sequence Diagram

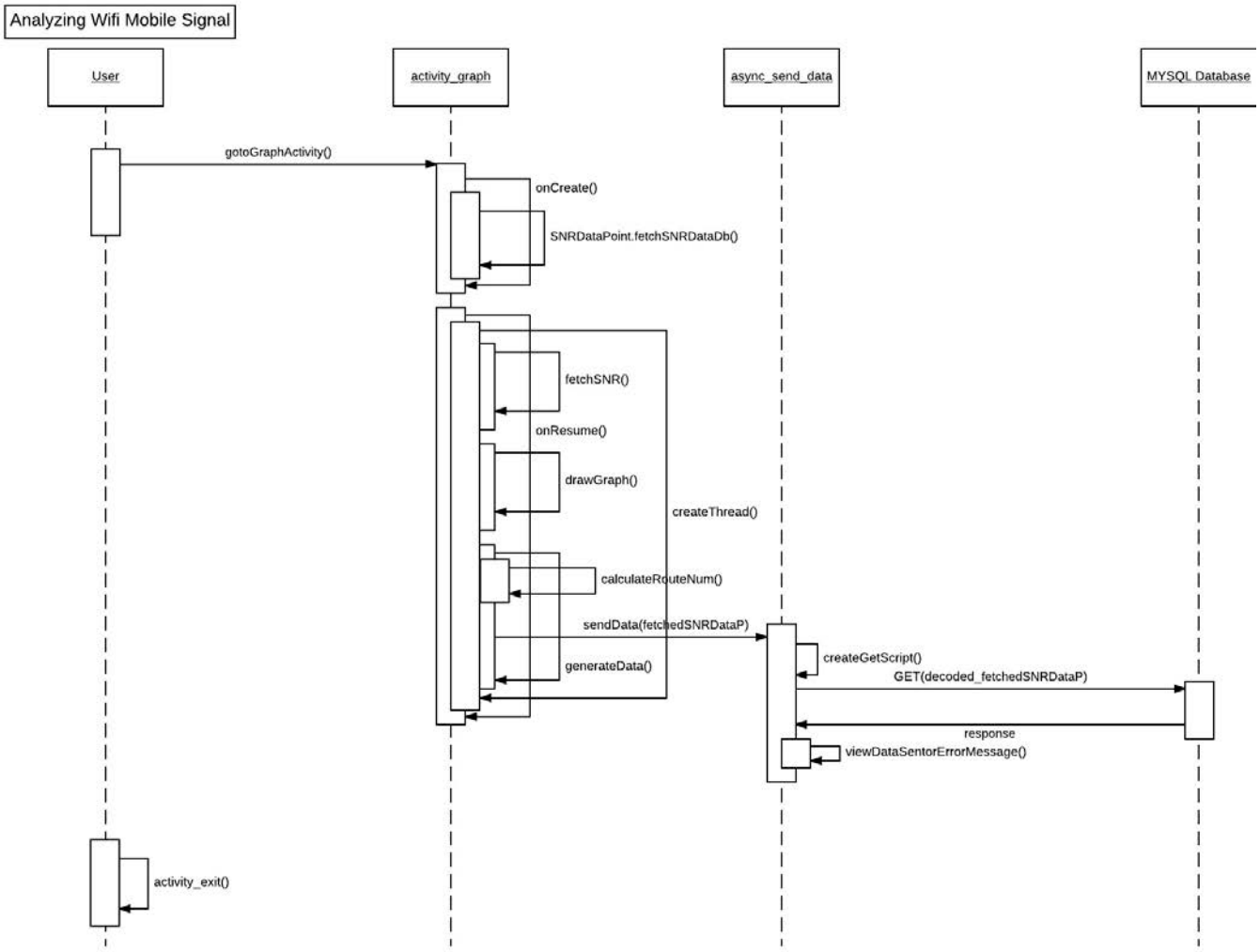


Figure 7: Analyzing Wifi Mobile Signal General Sequence Diagram

1.2.9. Analyzing Mobile - Cell Signal Sequence Diagram

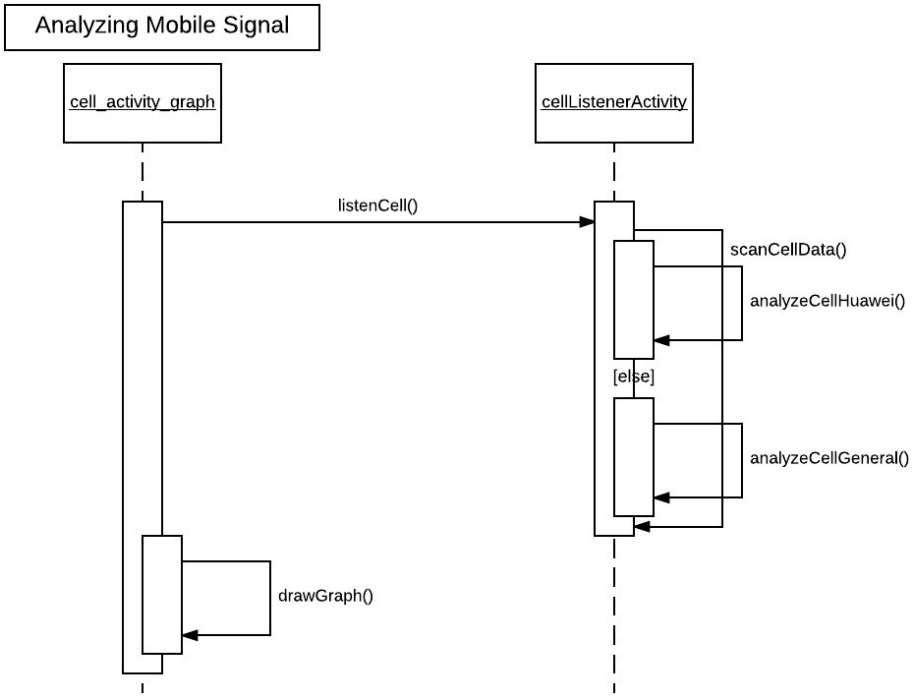
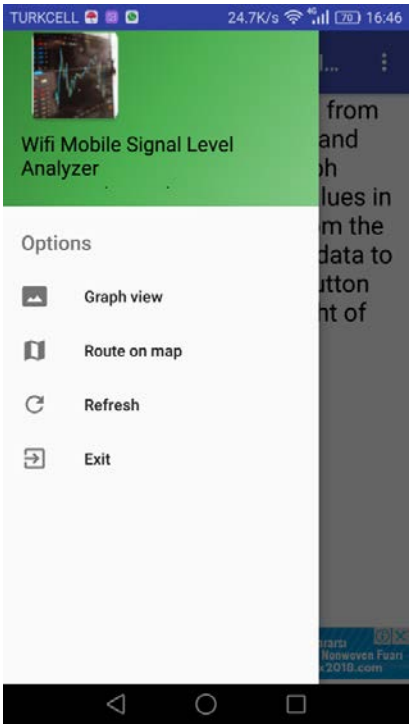


Figure 8:Analyzing Mobile - Cell Signal Sequence Diagram

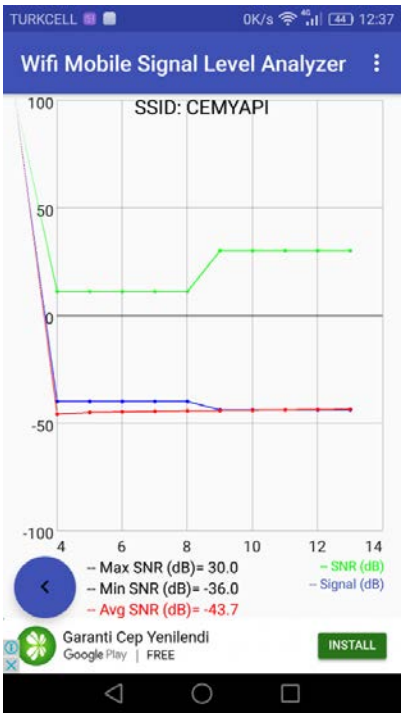
1.3. Screenshots – Manual

1.3.1. Firstly, user should choose Wifi or Mobile for reading the signal.

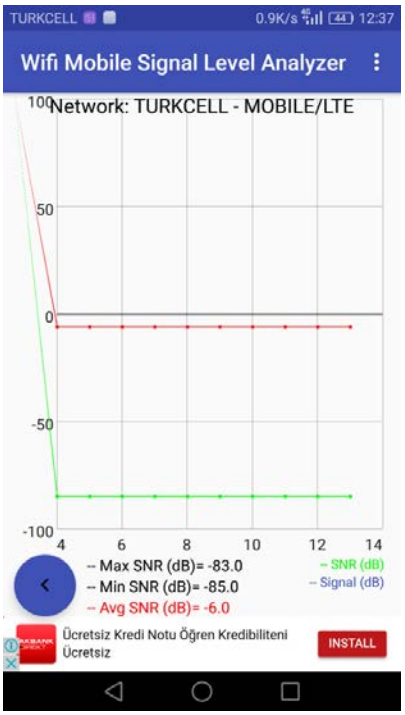
1.3.2. Main activity



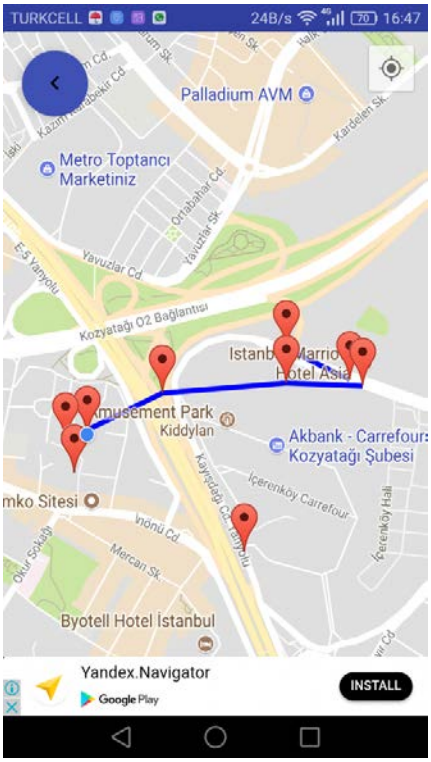
1.3.3. Wifi Signal Reding activity



1.3.4. Mobile Signal Reading activity



1.3.5. Map page: All Wifi or Mobile Signal data are shown in the map.



5. IMPLEMENTATION

1. Used API's

1.3.1. Google Maps API

1.3.2. Google gms API

Google's API for advertisements.

2. Sending and fetching data from PHP server and MYSQL database

Database queries should be done to the PHP server using AsyncTask in Android as below.

Sending SNR data with HTTP GET and AsyncTask:

```
    @Override
    protected String doInBackground(fetchedSNRDataP... arrFSnrDP) {
        String textViewResult = "";
        HttpResponse response;

        response = null;
        try {
            String link = "http://server_string/sendData.php/?username=" + "root" + "&password=" +
"asdcemasd"
                    + "&ROUTENUM=" + arrFSnrDP[0].getRouteNum() + "&SNR=" + arrFSnrDP[0].getSNR()
                    + "&LOC_X=" + arrFSnrDP[0].getLoc_x() + "&LOC_Y=" + arrFSnrDP[0].getLoc_y() +
"&SDATE=" + arrFSnrDP[0].getDate() + "&STIME=" + arrFSnrDP[0].getTime() + "&TABLE=" +
arrFSnrDP[0].getTable();

            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet();
            request.setURI(new URI(link));
            response = client.execute(request);

            textViewResult = "Data Sent";

        } catch (Exception e) {
            e.printStackTrace();
            textViewResult = "error " + response;
        }
        return textViewResult;
    }
```

3. Software Development Issues

1.3.1. Listening Mobile Cell signal

Listening cell signal is very easy in Android. Extending the PhoneStateListener Activity, below function should be used.

```
    @Override
    public void onSignalStrengthsChanged(SignalStrength signalStrength) {
        super.onSignalStrengthsChanged(signalStrength);
    }
```

Normally, SignalStrength object is enough to listen the cell signal. Use below functions inside onSignalStrengthsChanged() function.

Public methods	
int	describeContents() describeContents()
boolean	equals(Object o) Indicates whether some other object is "equal to" this one.
int	getCdmaDbm() Get the CDMA RSSI value in dBm
int	getCdmaEcio() Get the CDMA Ec/Io value in dB*10
int	getEvdoDbm() Get the EVDO RSSI value in dBm
int	getEvdoEcio() Get the EVDO Ec/Io value in dB*10
int	getEvdoSnr() Get the signal to noise ratio.
int	getGsmBitErrorRate() Get the GSM bit error rate (0-7, 99) as defined in TS 27.007 8.5
int	getGsmSignalStrength() Get the GSM Signal Strength, valid values are (0-31, 99) as defined in TS 27.007 8.5
int	getLevel() Retrieve an abstract level value for the overall signal strength.
int	hashCode() Returns a hash code value for the object.
boolean	isGsm()
String	toString() Returns a string representation of the object.
void	writeToParcel(Parcel out, int flags) writeToParcel(Parcel, int)

SNR - signal-to-noise ratio, dbMs, etc. could be easily fetched.

1.3.2. Listening Cell-Mobile signal via SignalStrength

Executing the phoneListener:

```
TelephonyManager tm = (TelephonyManager) c.getSystemService(c.TELEPHONY_SERVICE);
tm.listen(this, PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
```

Listener for the signal strength.

```
@Override
public void onSignalStrengthsChanged(SignalStrength ss) {
    super.onSignalStrengthsChanged(ss);

    //if phone is not Huawei, execute the general analyzing code.
    analyzeCellGeneral(ss);
}
```

```
private void analyzeCellGeneral(SignalStrength ssG) {
    if (ssG.isGsm()) {
        if (ssG.getGsmSignalStrength() != 99)
            Global.myNetwork_Signal = ssG.getGsmSignalStrength() * 2 - 113.0;
        else
            Global.myNetwork_Signal = ssG.getGsmSignalStrength() + 0.0;
    } else {
        Global.myNetwork_Signal = ssG.getCdmaDbm() + 0.0;
    }
    Global.SNR = ssG.getEvdoSnr() + 0.0;
}
```

1.3.3. Listening Mobile-Cell Signal in different phone brands

- To listen cell signal, SignalStrength object is not enough. In Listening Mobile Cell signal link, listening cell signal for many phone brands is told. Some brands like Huawei do not let us to listen signal properties easily. That should be a security precaution. The functions return 0 or -1 when they are called.
  In Huawei, below results returned;

```
signalStrength.getCdmaDbm() = -1
signalStrength.getGsmSignalStrength() = 0
signalStrength.getEvdoDbm() = -1
```

I found an easy way to listen signal from different phone brands.

1. Decode the SignalStrength object.

```
@Override
    public void onSignalStrengthsChanged(SignalStrength signalStrength) {
String ss = signalStrength.toString();
       // That ss string contains 16 different objects in
       // Huawei and these objects change from brand to brand. Object number also changes.
```

2. Seperate the string using the spaces as below:
   `String[] s = ss.split(" ");`
3. Assign as below:
   `networkSignal = Double.parseDouble(parts[1]);`

1.3.4. Analyzing Cell-Mobile signal in different phone brands

Reading signal from some phone brands, SignalStrength functions do not work properly. They return 0 or -1. Checking SignalStrength functions should be done. If these functions return 0 or -1, SignalStrength string could be read and splitted. Splitting SignalStrength and reading signal data from Huawei Honor 6 is below.

```
private void analyzeCellHuawei(SignalStrength ssH) {
        String ssignal = ssH.toString();
        String[] parts = ssignal.split(" ");
        if(ssH.getCdmaDbm() == -1 && ssH.getGsmSignalStrength() == 0 && ssH.getEvdoDbm() == -1  )
{
            if (!parts[1].toString().equals("0") || !parts[2].toString().equals("0") ||
!parts[3].toString().equals("0")) {
                Global.myNetwork_Signal = Double.parseDouble(parts[3]) + 0.0;
                Global.SNR = Double.parseDouble(parts[7]) + 0.0;
            } else {
                Global.myNetwork_Signal = Double.parseDouble(parts[12]) + 0.0;
                Global.SNR = Double.parseDouble(parts[11]) + 0.0;
            }
        }
        else {
            analyzeCellGeneral(ssH);
        }
    }
```

```
    private void analyzeCellGeneral(SignalStrength ssG) {
        if (ssG.isGsm()) {
            if (ssG.getGsmSignalStrength() != 99)
                Global.myNetwork_Signal = ssG.getGsmSignalStrength() * 2 - 113.0;
            else
                Global.myNetwork_Signal = ssG.getGsmSignalStrength() + 0.0;
        } else {
            Global.myNetwork_Signal = ssG.getCdmaDbm() + 0.0;
        }
        Global.SNR = ssG.getEvdoSnr() + 0.0;
    }
```

Below is the SignalStrength string.

```
part[0] = "Signalstrength:"  _ignore this, it's just the title_
parts[1] = GsmSignalStrength
parts[2] = GsmBitErrorRate
parts[3] = CdmaDbm
parts[4] = CdmaEcio
parts[5] = EvdoDbm
parts[6] = EvdoEcio
parts[7] = EvdoSnr
parts[8] = LteSignalStrength
parts[9] = LteRsrp
parts[10] = LteRsrq
parts[11] = LteRssnr
parts[12] = LteCqi
parts[13] = gsm|lte|cdma
```

1.3.5. Creating time interval after every sensor read

Waiting would be done inside a thread not to interrupt the whole activity. fetching data, drawing the graph are done real time in the code below:

```
    @Override
    protected void onResume() {
        super.onResume();
        Global.Moving_Avg_Signal = 0.0;
        try {
            mTimer1 = new Runnable() {
                @Override
                public void run() {
                    if(graph != null)
                        graph.removeAllSeries();
                    if (Global.jsonText != null) {
                        fetchSNR();
                        drawGraph();
                        setPageText();
                    }
                    new asyncGraphSmallWait().execute(mHandler, mTimer1, 40);
                }
            };
            new asyncGraphSmallWait().execute(mHandler, mTimer1, 40);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

1.3.6. AsyncTask execution with sending different variables

First AsyncTask example:
For creating delayed background processes without halting whole application, AsyncTask is needed. Parameter type should be written to the class definition. In the doInBackground declaration, parameter should be written as doInBackground(paramType... params). asyncTask class should be called with different type parameters inside the function call.

```
// Calling the asyncTask method:
new asyncGraphSmallWait().execute(mHandler, mTimer1, 40);
```

```
public class asyncGraphSmallWait extends AsyncTask<Object, Void, Void>
{

    @Override
    protected void onPreExecute() {
        super.onPreExecute();

    }
    @Override
    protected Void doInBackground(Object... params) {
        Handler mHandler = (Handler) params[0];
        Runnable mTimer1 = (Runnable) params[1];
        int time = (int) params[2];
        mHandler.postDelayed(mTimer1, time);
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);

        //this method will be running on UI thread

    }
}
```

6. CONCLUSIONS

1. Future Work

1.3.1. PSNR calculation

PSNR – Peak Signal-to-noise ratio calculation will be mathematically calculated and added to the software. Users should understand PSNR from the graph view activity.

1.3.2. Enhanced Security

Database queries should not be done using HTTP GET method. A secure method should be found and implemented.

1.3.3. More proper location reading

Location is read by the GPS service. However, GPS service does not respond very fast and it could not be used very easily as tracking the route from the real google maps. Android's GPS service should be understood better and the application should be developed accordingly.

1.3.4. Enhanced route creation

1.3.5. Advertising

Usage statistics from November 2017 to December 2017 could be seen. Advertisement for more users is needed.

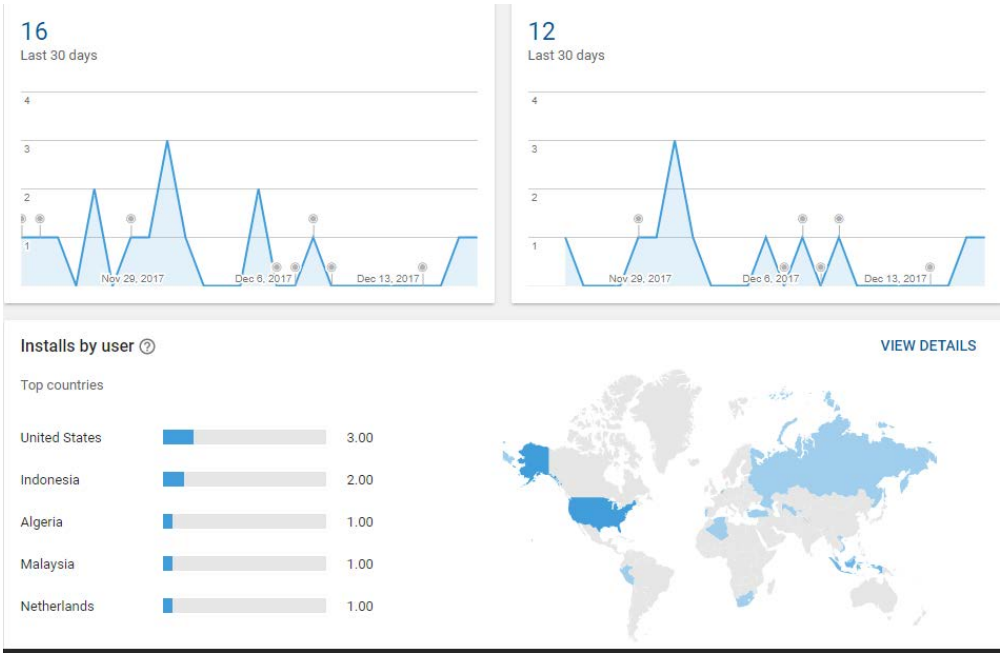


Figure 9: Usage statistics from November 2017 to December 2017

1.3.6. Language

Translating the software into different languages.

7. GLOSSARY

1.1. SNR

Signal-to-noise ratio (abbreviated SNR or S/N) is a measure that compares the level of a desired signal to the level of background noise. SNR ratio which is the ratio of signal power to the noise power, generally shown in decibels. SNR is measured as in the formula below:

$$\mathrm{SNR}_{\mathrm{dB}} = (A_{\mathrm{signal,dB}} - A_{\mathrm{noise,dB}})$$

1.2. PSNR

The psnr function implements the following equation to calculate the Peak Signal-to-Noise Ratio (PSNR):

$$PSNR = 10\log_{10}(peakval^2/MSE)$$

Where peakval is either specified by the user or taken from the range of the image datatype (e.g. for 8-bit image it is 255). MSE is the mean square error, i.e. MSE between A and ref.

1.3. Average SNR

Average SNR will be understood from real tests.

1.4. Minimum SNR

Analyzed minimum SNR value in a time interval.

1.5. Maximum SNR

Analyzed maximum SNR value in a time interval.

8. REFERENCES

1. Finley, Micheal (May 21, 2017). SNR Calculation. Northern Arizona University Dynamic and Active Systems Laboratory.