Android Network Cell Analyzer

(Group 5)

Prepared for

Prof. Ali I Hussein

Written by

Ali Rammal
Abdel Rahman El Kouche
Karim Abou Daher
Karim Dib

GitHub Repo:

https://github.com/Ali-Rammal1/NetworkCellAnalyzer

Abstract

This report presents the development of the "Network Cell Analyzer," a cross-layer Android system designed to capture, analyze, and visualize real-time mobile network data. The mobile application, built using Kotlin in Android Studio, gathers cellular metrics including network type, signal strength, SNR/SINR, and frequency bands across 2G, 3G, 4G, and 5G networks. The data is transmitted via HTTP requests to a Flask backend hosted on AWS, which interfaces with a PostgreSQL database via Supabase for persistent storage and statistics computation. This system helps users assess signal quality, monitor operator coverage, and understand frequency band usage trends. Authentication features ensure secure access, and the server dashboard supports administrative control and user log monitoring. Live testing on real devices and emulators confirmed robust performance under varied network conditions.

Introduction

Mobile users often lack insight into the quality and consistency of their cellular connections. Our app, “Network Cell Analyzer” was initiated to bridge this gap by

enabling users to monitor real-time network performance metrics and statistics of previous preformance via. Leveraging Android's Telephony APIs, the app captures granular cellular data and sends it to a backend system built with Flask. This backend processes the data and provides detailed statistics accessible from the app itself. Hosted on an AWS server, our Supabase's PostgreSQL database stores user data and information in a secure and scalable matter. The goal of our project is to empower users and researchers with the necessary information to be able to diagnose weak signal zones and thus potentially fix any underlying issues.

Methodology and Architecture

Overall Stack

- **Frontend**: Android Studio (Kotlin) + HTML/CSS for the server interface
- **Backend**: Flask (Python)
- **Database**: Supabase with PostgreSQL (cloud-hosted on AWS)
- **Communication**: Tokenized RESTful HTTP (JSON format)

The architecture ensures modularity, allowing independent scaling of the mobile and backend components. Secure communication, user authentication, and real-time stats rendering are key elements.

Server Workflow

1. **Start-up**: Flask server launches and connects to Supabase.
2. **Receive Data**: POST requests from the Android client carry JSON payloads with signal data, timestamp, and user ID.
3. **Validate and Store**: Data is validated and stored in the PostgreSQL DB.
4. **Query**: GET requests with user-selected dates return filtered statistics.
5. **Respond**: JSON responses with signal averages, SNR, frequency usage, etc.

Server Database Overview

The database is built on PostgreSQL and includes three main tables:

1. cell_data

Stores all the network measurements sent by users. Fields include:

- id: Unique record ID

- user_id: User ID (or "guest" if anonymous)
- email: User email (or guest email)
- operator: Mobile network operator name
- signal_power: Signal strength (in dBm)
- snr: Signal-to-noise ratio (in dB)
- network_type: 2G, 3G, 4G, 5G
- frequency_band: Frequency band (e.g., 1800 MHz)
- cell_id: ID of the connected cell tower
- client_timestamp: Time at which the device collected the data
- user_ip: IP address of the device
- user_mac: MAC address of the device
- upload_time: Server-side timestamp when data was received
- device_brand: Device manufacturer brand

2. users

Manages registered users. Fields include:

- id: Unique user ID
- name: Full name of the user
- email: Unique email address (used for login)
- password_hash: Hashed password for secure authentication
- created_at: Registration timestamp

3. tokens

Handles authentication tokens for session management. Fields include:

- id: Unique token ID
- user_id: Linked user ID (foreign key)
- token: Secure 64-character authentication token
- created_at: Token creation timestamp
- expires_at: Token expiration timestamp (default 30 days)
- is_revoked: Boolean flag to invalidate tokens if needed

Application Backend Overview

Real-Time Information Retrieval

- Uses TelephonyManager and CellInfo APIs
- Detects:
    - **Signal Strength**: .getSignalStrength().getLevel()
    - **Network Type**: GSM, WCDMA, LTE, NR

    - **Band Mapping**: ARFCN → 2G, UARFCN → 3G, EARFCN → 4G, NRARFCN → 5G
    - **SNR Metrics**:
        - 5G: ssSinr or csiSinr
        - 4G: rssnr
        - 3G: Ec/Io
        - 2G: Estimate using **Bit Error Rate (BER)**
- Timer task collects and sends data every 10 seconds
- Device UUID used for identification

Statistics

- Users select start/end dates
- Fetches:
    - Average signal level per operator
    - Network type distribution
    - Frequency band usage
    - SNR stats across generations
- JSON parsed and displayed in clean text format
- Plots the information into informative charts

User Interface Overview

Login and Authentication

- Users can log in with email and password, register a new account, or continue as guest.
- Password rules enforced during registration (minimum 6 characters, one uppercase, one lowercase, one digit, one special character, no spaces).

Real-Time Information Screen

- Shows Operator**,** Signal Strength (dBm)**,** SNR/Signal Quality (dB)**,** Network Type, and Frequency Band.

Statistics Screen

- Select 6 Hours, 24 Hours, 7 Days, or Custom Date Range.
- Displays:
  - Average Signal Strength
  - Average SNR
  - Total Data Points
  - Network Type Distribution
- Includes a line chart (signal strength over time) and a pie chart (network type distribution).

Server Interface

- Monitor real-time logs and user activity
- Search by email for user-specific logs

Server-Application Communication

API endpoints:

- **POST** /register: register a new user
- **POST** /login: login and get a token
- **POST** /logout: logout and revoke token
- **POST** /upload: upload cellular data (with or without token)
- **POST** /refresh-token: refresh authentication token
- **GET** /validate-token: check if token is valid
- **GET** /api/stats: server statistics for admin dashboard
- **GET** /api/user-stats: fetch user-specific statistics (by email and date range)
- **GET** /api/server-user-stats: detailed user connection history for dashboard
  GET /api/all-users: list all registered users

Testing

- Conducted using both **Android emulators** and **physical devices**
- Server tested via Postman

- DB monitored via Supabase dashboard
- Simulated high-frequency data push to validate scaling
- Verified band and SNR detection across network types

Problems Faced

- Performance issues with Android Studio
- Permission handling complexities
- Technical issues with statistical data accuracy
- Accessing the correct functions from TelephonyManager
- Testing and validating backend queries across Supabaseand Flask

---

Running the Application

- Clone the GitHub repository: https://github.com/Ali-Rammal1/NetworkCellAnalyzer
- Launch the Flask server and ensure your **Supabase credentials** are active.
- In Android Studio, open the project and go to the gradle.properties file.
- Set the value of **API_BASE_URL** to the IP address of the server computer where the backend is running.
- Sync the project with Gradle files (click **"Sync Now"** if prompted).
- Connect your physical device (or use an emulator).
- Ensure that **your phone and server are connected to the same Wi-Fi network**.
- Run the app from Android Studio.
- Login or continue as guest, and start collecting real-time network data!

Conclusion

The Network Cell Analyzer is a complete tool that helps users understand how mobile networks perform. With its easy-to-use app and strong backend, it shows how cellular signals change and how different networks behave. Its modular design, cloud hosting, and support for 2G to 5G networks make it useful for both personal use and research. In the future, we plan to add real-time alerts and machine learning features for even better insights.

Division of Work

- **Ali Rammal**: Android app and Back-end Development
- **Abdel Rahman El Kouche**: Server Setup, Server Interface, and Database setup
- **Karim Abou Daher**: UI Design and PowerPoint.
- **Karim Dib**: Testing, Server Setup and Report.

All team members contributed to debugging, testing, and refining both backend and app features.

References

- **Android Developers** – TelephonyManager Documentation
- **Supabase Documentation** – https://supabase.com/docs
- **Medium** – "How to Make a Client Android App with Flask Backend"
- **Wikipedia** – LTE and 5G Frequency Bands
- **GeeksforGeeks** – How to Build a Simple Android App with Flask Backend