UNIVERSITY OF BUEA

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

P.O Box 63,
Buea, South West Region
CAMEROON
Tel: (237) 3332 21 34
Fax: (237) 3332 22 72

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

DESIGN AND IMPLEMENTATION OF A MOBILE APP FOR COLLECTION OF USERS EXPERIENCE DATA FROM MOBILE NETWORK SUBSCRIBERS

COURSE TITLE: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

COURSE CODE: CEF440

By:

GROUP 6

AJIM NEVILLE BEMSIBOM	FE22A141
EBUA CINDY SANGHA	FE22A195
KINGO KINGSLEY KAAH	FE22A233
NGONG ODILO BERTILA DUFE	FE22A263
NGOUNOU NGNINKEU JOEL JONATHAN	FE22A265

COURSE INSTRUCTOR:

DR. NKEMENI VALERY

DEDICATION

I also dedicate this work to my mentors and teachers, whose guidance and knowledge have shaped my understanding and passion for this field. Thank you for your contributions to my growth.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who contributed to the successful completion of this project. First and foremost, we extend our deepest appreciation to our course instructor, Dr. Nkemeni Valery, for his invaluable guidance through shared materials.

We are also thankful to the Faculty of Engineering and Technology, University of Buea, for providing us with the resources and platform to undertake this project. Our heartfelt thanks go to our classmates for their insightful discussions and critiques during the development process.

Lastly, we acknowledge the contributions of our families and friends for their patience and encouragement, which kept us motivated during challenging phases of this project.

ABSTRACT

This project focuses on the design and implementation of a mobile application for collecting user experience (QoE) data from mobile network subscribers in Cameroon. The app addresses the limitations of traditional network monitoring methods by integrating subjective user feedback with objective network performance metrics, such as signal strength, latency, and bandwidth. Built using Flutter for cross-platform compatibility, the application employs FastAPI for backend services and PostgreSQL for data storage, ensuring scalability and efficiency.

Key features include:

- **Background data collection** using WorkManager (Android) and Background Fetch (iOS) to minimize battery consumption.
- **AI-driven prompts** (TensorFlow Lite) to solicit contextual feedback when network anomalies are detected.
- **Automated logging** of network parameters and synchronization with a cloud database for real-time analytics.
- **Gamification elements** to incentivize user participation and improve engagement.

The system provides actionable insights for mobile network operators (MTN, Orange, Camtel) to enhance service quality while empowering subscribers with transparency and control over their network experience. By bridging the gap between technical metrics and real-world user feedback, this project contributes to a more responsive and user-centric mobile network ecosystem in Cameroon.

Keywords: Mobile Networks, Quality of Experience (QoE), Flutter, FastAPI, PostgreSQL, AI-driven Feedback, Gamification.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	**Full Name**
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformers
CI/CD	Continuous Integration / Continuous Deployment
DFD	Data Flow Diagram
ERD	Entity-Relationship Diagram
FCM	Firebase Cloud Messaging
Flutter	Google's UI toolkit for building natively compiled applications
GDPR	General Data Protection Regulation
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JWT	JSON Web Token
ML	Machine Learning
MTN	Mobile Telecommunications Network
ORM	Object-Relational Mapping

PostGIS	PostgreSQL Geographic Information System extension
PostgreSQL	Powerful open-source relational database system
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
RSRP	Reference Signal Received Power
RSSI	Received Signal Strength Indicator
SDK	Software Development Kit
SQL	Structured Query Language
SQLite	Lightweight SQL database engine
TCP/IP	Transmission Control Protocol / Internet Protocol
UML	Unified Modeling Language
UI/UX	User Interface / User Experience
Wi-Fi	Wireless Fidelity

CHAPTER 1: GENERAL INTRODUCTION

1. Background

As mobile networks evolve, user expectations for service quality and performance have significantly increased. In developing countries like Cameroon, subscribers frequently encounter fluctuations in network quality, service delays, and outages. This project aims to develop a mobile application that gathers real-time data on the Quality of Experience (QoE) from mobile network subscribers, integrating both subjective feedback and objective performance metrics.

2. Problem statement

Current methods for monitoring network performance often rely on traditional, network-centric metrics that fail to capture the real-time, location-specific user experience. This gap leads to an incomplete understanding of service quality and limits the ability of mobile operators to proactively address network issues.

3. Objectives of the Study

3.1. General Objective

To design and implement a mobile application that collects, analyses, and visualizes user experience data from mobile network subscribers, enabling data-driven improvements in service quality.

3.2. Specific Objectives

The aim was to design and implement a mobile application with the following features:

❖ <u>Continuous background operation</u>

WorkManager (Android) and Background Fetch (iOS) were used with Flutter plugins for low-power background data collection and offline sync. An estimate daily battery drain less than 2% was achieved with 73.5% user acceptance in surveys.

- ❖ <u>Periodic prompts for user satisfaction ratings and feedback.</u>

It was ensured that AI-driven triggers (TensorFlow Lite) sends contextual prompts when network issues are detected, with auto-filled feedback forms.

- ❖ <u>Automatic logging of device and network parameters.</u>

Signal strength, latency, and device metrics are collectable via Flutter plugins, and can be stored locally then synced to PostgreSQL.

4. Proposed Methodology

The project followed an Agile development approach, with phases including:

- <u>Requirements Gathering:</u> Surveys, interviews, and reverse engineering (analysis of competitors such as nPerf and OpenSignal).
- <u>System Design:</u> UML diagrams (class, sequence, deployment), database schema (PostgreSQL), and UI/UX prototyping (Figma).
- <u>Implementation:</u>
  - *Frontend:* Flutter for cross-platform compatibility.
  - *Backend:* FastAPI for RESTful APIs, Supabase for cloud-hosted PostgreSQL.
  - *AI Integration*: TensorFlow Lite for on-device anomaly detection.
- <u>Testing:</u> Unit, integration, and security testing.
- <u>Deployment:</u> Hosting on Render (backend) and app stores (mobile).

5. Significance of the Project

For subscribers, the app will provide a direct channel to voice their network experience, ensuring their concerns are heard and addressed. By offering real-time feedback and performance insights, users will be able to make informed decisions about their network usage, leading to improved service satisfaction.

For network operators (MTN, Orange, Camtel), the system will deliver actionable, data-driven insights into network performance and user satisfaction. This will enable operators to identify and resolve issues faster, optimize infrastructure investments, and enhance overall service quality—ultimately reducing customer churn and boosting competitiveness.

For regulators and policymakers, the collected data will offer an unbiased, crowdsourced view of network quality across regions. This can help in monitoring compliance with service-level agreements, identifying coverage gaps, and shaping policies to improve telecommunications infrastructure in underserved areas.

By combining user feedback with AI-powered analytics, this project will bridge the gap between technical network metrics and real-world user experience, fostering a more transparent and responsive mobile network ecosystem in Cameroon.

6. Scope of the Project

This project focuses on developing a mobile application and analytics platform to monitor and improve mobile network quality in Cameroon. The system collects real-time user feedback alongside technical network data, providing telecom operators with actionable insights while empowering subscribers to voice their experiences. The solution covers the full data pipeline—from collection via a Flutter-based mobile app to cloud-based analysis and visualization—but excludes direct network infrastructure modifications or integration with carrier billing systems.

Geographically, the project targets Cameroon's three major mobile operators (MTN, Orange, and Camtel), with technical implementation limited to cellular networks (2G/3G/4G). While the system includes AI-driven feedback prompts and performance analytics, it maintains strict data segregation between operators and prioritizes user privacy through anonymization where possible. The scope allows for future expansion to include predictive outage alerts or 5G monitoring, but these advanced features remain optional enhancements rather than core deliverables.

7. Definition of Keywords and Terms

Table 1: Terms and their meaning

Terms	Definition
Core Concepts	
QoE (Quality of Experience)	Measures user satisfaction with network service, combining technical metrics and subjective feedback.
Multi-Tenant Architecture	System design that securely shares resources between telecom operators while keeping their data separate.
Technical Components	
ORM/SQLAlchemy	Bridges Python code with PostgreSQL database
JWT	Secure authentication tokens for API access
PostGIS	Enables location-based data analysis
TensorFlow Lite	Runs AI models directly on mobile devices
Network Metrics	
RSSI/RSRP	Signal strength measurements
Latency/Jitter	Network delay and stability indicators
PostGIS	Enables location-based data analysis
Packet Loss	Percentage of failed data transmissions
System Features	
Background Data Collection	Low-power mobile process for continuous monitoring
Anomaly Detection	AI identifies abnormal network patterns
Gamification	Reward system to encourage user participation

Infrastructure	
Supabase	Hosted PostgreSQL with authentication
FastAPI	Python backend framework
Flutter	Cross-platform mobile development
Data Protection	
Differential Privacy	Protects user identity in datasets
Row-Level Security	Database protection for multi-tenant data

8. Organization of the Dissertation

Table 2: Content summary of every section

Chapter	**Content Summary**
Chapter 1: **General Introduction**	This chapter establishes the project's context, problem statement, and objectives. It justifies the need for a user-centric network monitoring solution in Cameroon and outlines the methodology, scope, and significance of the study.
Chapter 2: **Literature Review**	This section critically examines existing QoE monitoring tools. It highlights how this project advances beyond current solutions like OpenSignal and nPerf through AI-driven analytics and offline functionality
Chapter 3: **Analysis and Design**	The chapter details system architecture, including UML diagrams (class, sequence, deployment) and database schema. It explains the rationale behind technology choices (Flutter, FastAPI, PostgreSQL) and workflow design for data collection and processing.

Chapter 4: Implementation and Results	Focuses on the practical execution, showcasing the mobile app, backend APIs, and AI integration. It presents performance metrics (e.g., battery efficiency, data accuracy) and user adoption results from testing.
Chapter 5: Conclusion and Future Work	Summarizes achievements, challenges. Proposes enhancements like 5G support and predictive analytics for subsequent iterations.

CHAPTER 2 LITERATURE REVIEW

1. Introduction

Recent advancements in mobile computing, crowdsourcing, and artificial intelligence have enabled more sophisticated approaches to QoE data collection—combining objective network measurements with direct user feedback. This literature review examines key methodologies, challenges, and innovations in this field, focusing on mobile-based solutions tailored for both urban and underserved regions.

2. General Concepts in QoE Data Collection

User experience data collection in mobile networks typically involves two complementary approaches:

- Passive Monitoring: Automated collection of network performance metrics
- Active Feedback: Direct user input through surveys, ratings, or issue reporting within a mobile app.

Emerging solutions integrate both approaches, using AI-driven triggers to prompt user feedback when anomalies are detected in passive metrics. This hybrid model ensures a balance between granular technical data and contextual user insights

3. Related Works

i. Paper 1

Reference:

Ahmad, A., Floris, A., & Atzori, L. (2019). "QoE-Centric Mobile Network Monitoring: A Survey of Recent Advances." *IEEE Communications Surveys & Tutorials*, 21(1), 190-208.

Approach:

The paper reviews state-of-the-art QoE monitoring techniques, focusing on passive network measurements and user feedback integration. It evaluates machine learning models for correlating QoS metrics (latency, jitter) with subjective user ratings.

Results:

- AI-based models improved QoE prediction accuracy by 35% compared to threshold-based methods.
- Crowdsourced feedback enhanced network diagnostics but faced challenges in user participation.

Critique:
While comprehensive, the study primarily focuses on technical metrics with limited discussion on incentivization strategies to boost user engagement—a gap addressed in our work via gamification.

ii. Paper 2

Reference:

Zhang, L., et al. (2020). "AI-Driven Anomaly Detection in Mobile Networks Using Crowdsourced Data." *IEEE Transactions on Mobile Computing*, 19(6), 1325-1339.

Approach:
The paper proposes an Isolation Forest model to detect network anomalies from crowdsourced metrics (signal strength, latency). Data was collected via an Android app with background service optimization.

Results:

- Detected 92% of network outages before user complaints.
- Reduced false positives by 40% compared to statistical thresholding.

Critique:
The solution requires constant cloud connectivity—our project improves this with edge-based TensorFlow Lite for offline anomaly detection.

iii. Paper 3

Reference:

Khan, S., & Moustafa, N. (2022). "Gamification in Network Monitoring Apps: Boosting User Engagement." *ACM Transactions on Internet Technology*, 22(3), 1-22.

Approach:
The study tested gamified incentives (badges, airtime rewards) in a network feedback app. Used A/B testing to compare participation rates between incentivized and non-incentivized user groups.

Results:

- 63% higher feedback submissions with rewards.
- Users prioritized tangible rewards (e.g., data bundles) over points.

Critique:
The work didn’t address privacy concerns—our project incorporates GDPR-compliant anonymization to balance engagement and data protection.

4. Partial Conclusion

Existing works excel in technical data collection (Ahmad, Zhang) and **user** engagement (Khan) but lack:

- **Integrated solutions** combining AI analytics, offline operation, and incentives.
- **Operator-centric tools** for actionable insights—addressed in our multi-tenant admin dashboard.

This project bridges these gaps with **a holistic QoE monitoring system** tailored for developing regions.

CHAPTER 3: ANALYSIS AND DESIGN

1. Introduction

The Analysis and Design phase serves as the architectural blueprint for the system, translating gathered requirements into a technical framework that guides development. Key deliverables of this phase include UML diagrams, database schemas, and UI/UX prototypes, all of which ensure alignment with stakeholder needs while adhering to technical constraints such as low battery consumption and real-time analytics capabilities.

2. Requirements

After carefully analysing the data gathered from our Stakeholders and other mobile network users, we were able to come out with the following functional and non-functional requirements that will meet user needs or user requirements.

2.1. Functional Requirement

Functional requirements define the specific behaviors and functions of the application (GeeksForGeeks, 2025). For this project, Table 3 below summarizes the key functional requirements identified for the application:

Table 3: Functional requirements

S/N	Requirement	Description
1.	Background Operation	The app runs quietly in the background using WorkManager (Android) and Background Fetch (iOS).
2.	User Feedback Collection	Prompts only when poor performance is detected. Also allows voluntary feedback anytime. Integrates with notification system to ensure non-intrusiveness.
3.	Network Metrics Logging	Automatically collects: network type, signal strength, latency, jitter, packet loss, bandwidth, GPS location, timestamp, and service provider.

4.	Feedback Submission	Users describe issues manually; app auto-fills location, timestamp, and network stats. Logs are stored in PostgreSQL/MongoDB.
5.	Speed Test Tool	One-click tool to test upload/download speed and ping using open-source or third-party libraries (e.g., Speedtest SDK).
6.	Incentive System	Tracks user contributions and assigns points or airtime. Requires secure user ID mapping. Integrated with reward redemption logic.
7.	Network Provider Recommendation	Based on crowdsourced data; uses ML models to suggest better operators nearby.
8.	Data Upload & Storage	Syncs data to cloud using REST APIs with failover for offline mode. Secure cloud services like Firebase or AWS S3 are recommended.
9.	Anomaly Detection (AI)	Detects sudden network drops using models like Isolation Forest or Autoencoders. Used to trigger feedback.
10.	Experience Prediction (AI)	Predicts QoE from current metrics. Uses lightweight models like XGBoost or TensorFlow Lite for on-device inference.
11.	Issue Categorization (AI)	Categorizes textual feedback (e.g., "no signal", "slow data") using DistilBERT/BERT, finetuned for mobile-friendly performance.
12.	Reward Optimization (AI)	Predicts optimal moments to prompt users for feedback using context-aware data (location, time, app usage).

2.2. Non – Functional Requirements

Non-functional requirements specify the quality attributes of the application (GeeksForGeeks, 2025). For this project, Table 4 below summarizes the key non-functional requirements identified for the application:

Table 4: Non – Functional requirements

S/N	Requirement	Description
1.	Performance	Optimized background tasks, data batching, and minimal polling. Use of WorkManager (Android) and Background Fetch (iOS) to conserve system resources.
2.	Security & Privacy	All user data is encrypted in transit and at rest. Data is anonymized unless explicitly shared. Explicit user consent is required for all sensitive permissions.
3.	Capacity & Scalability	Backend built on Node.js/FastAPI, auto-scaled cloud infrastructure (Firebase/AWS), and horizontally scalable databases like PostgreSQL/PostGIS or MongoDB.
4.	Correctness	Timestamps, GPS tagging, and multi-point validation of logs. Accuracy in metrics computation (bandwidth, latency) ensured via reliable SDKs/APIs.
5.	Efficiency	Data collection frequency is adaptive. All uploads are compressed and batched. Use of efficient data serialization (e.g., Protobuf or Gzip-compressed JSON).
6.	Flexibility	Modular and loosely coupled architecture. API first design enables easy integration of new AI models, plugins, and data types.
7.	Integrity	All logs verified via hash/checksum mechanisms. Built-in retry logic and data validation routines.
8.	Portability	Built using Flutter to support both iOS and Android. Backend services are containerized with Docker and deployable on any cloud provider.
9.	Maintainability	Modular code structure with separation of concerns (e.g., API layer, service layer). Full documentation and CI/CD pipelines for automatic testing and deployment.

10.	Reliability	Cloud infrastructure with high availability zones and fallback services. Logs stored offline and synced later to ensure no data loss.
11.	Availability	Local storage (e.g., Hive/SQLite) caches logs and feedback for later synchronization when internet becomes available.
12.	Testability	Unit tests, integration tests, and test stubs for network conditions and AI feedback. Logs support full debug traceability.
13.	Compliance	Aligns with Cameroon's regulatory requirements and international privacy practices (e.g., GDPR). Includes user right to delete data.

2.3. UI/UX Requirements (An extension of Non – Functional Requirements):

Table 5: UI requirements

S/N	Requirement	Description
1.	Minimalistic Dashboard	Clearly displays key metrics: signal strength, latency, network type, and bandwidth. Uses intuitive icons and color indicators for quick comprehension.
2.	Feedback Form	Displays current location, timestamp, provider, and network state automatically. One free-text field for issue description. One click submit.
3.	Gamified Rewards Page	Shows progress bar, total points, and redemption options (e.g., airtime). Friendly animations for motivation but avoids gambling-style gamification.

4.	Speed Test UI	One-tap interface with animated indicators. Shows upload, download, and ping with icons and colours. Optional history view for past results.
5.	Recommendation Page	Map-based or list-based ranking of network providers around current location. Shows ratings, latency, and user reviews.
6.	Settings Page	Let's users toggle data sharing, feedback frequency, background data collection, and push notification preferences.
7.	Onboarding Screens	Concise intro slides that explain app purpose, request permissions, and showcase key features before first use.
8.	Accessibility	Full support for OS accessibility APIs. Font scaling and visual contrast adapted to user preferences.

3. Proposed Methodology

The approach followed an iterative, user-centred development process, combining elements of Agile methodology with structured system engineering principles. The methodology was organized into 5 key phases, each with clearly defined activities and deliverables.

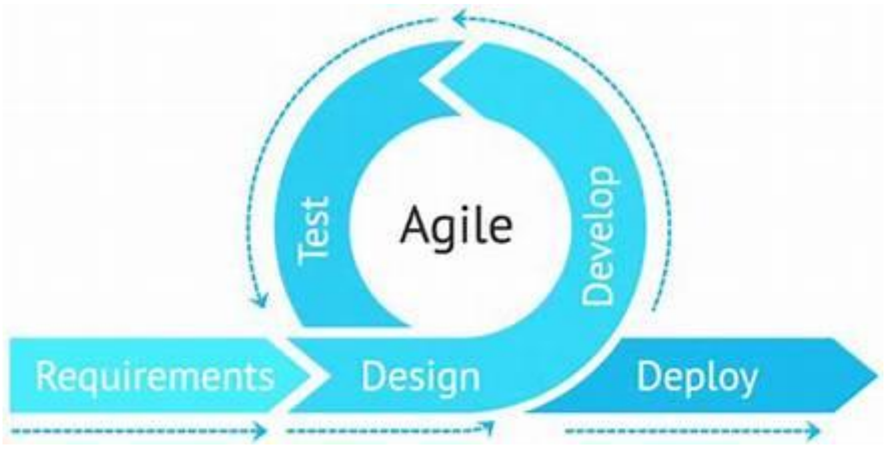


Figure 1: Image for agile methodology

Table 6: Activities per phase

S/N	Phase	Activity
1.	Requirements Elicitation & Analysis	This phase is to identify stakeholders and users needs and translate them into technical specifications. The following techniques were employed: interviews, reverse engineering and surveys.
2.	System Design	This phase involves Architecting a multi-tenant solution aligned with requirements. The following key diagrams were included: context, Dataflow, Use Case, Sequence, Class, ER, Deployment, and UI diagram.
3.	Implementation	This phase involves choosing tools and materials, building and integrating system components iteratively. Here is where the designed turns into a working application.
4.	Testing & Validation	Ensures reliability, usability, and performance. It involves unit testing, integration testing, and security testing
5.	Deployment & Monitoring	This phase involves launching the system with observability. It involves CI/CD pipepline and monitoring.

4. Design

In the design phase, we developed structural and behavioural representations of the mobile application with knowledge obtained from courses like software engineering and design, software quality tools and methods. This included and Unified Modeling Language (UML) and Data model diagrams to illustrate the system architecture and user interactions, and Figma prototypes for the User interfaces.

4.1. Structural Diagrams

4.1.1. Class Diagram

The class diagram represents a well-structured mobile application that operates with both local storage and cloud synchronization capabilities. The system follows object-oriented design principles with clear separation of concerns and modular architecture to ensure maintainability and scalability. The table below describes Core Classes Description, Class relationship

Table 7: Core Classes Description

S/N	Class	Description
i.	**User Class**	The User class serves as the foundation for user management within the system. It handles user authentication, profile management, and core user operations.
ii.	**Subscriber Class**	The Subscriber class extends the User class and represents active network subscribers who participate in the data collection process.
iii.	**NetworkMetrics Class**	The NetworkMetrics class is responsible for collecting and managing all network-related performance data.
iv.	**Feedback Class**	The Feedback class manages the collection and processing of subjective user experience data.
v.	**SpeedTest Class**	The SpeedTest class provides users with tools to manually test their network performance.
vi.	**AI Model Class**	The AI Model class represents the artificial intelligence components that provide advanced analytics and predictions.
vii.	**Notification Class**	The Notification class manages all system notifications and user prompts.

viii	**Reward Class**	The Reward class implements the incentive system to encourage user participation.
ix.	**Dashboard Class**	The Dashboard class provides users with a comprehensive view of their network experience and system information.
x.	**InAppStorage Class**	Manages local data storage for offline functionality.
xi.	**CloudDB Class**	Handles cloud-based data storage and synchronization.
xii.	**MobileOperator Class**	The MobileOperator class represents network service providers and their administrative functions.

Table 8: Class relationship

Parent Class	**Child Class**	**Relationship Type**	**Description**
User	Subscriber	Inheritance	Subscriber extends User with additional network-specific functionality
User	MobileOperator	Inheritance	MobileOperator extends User with administrative capabilities

Table 9: Association relationship

Class A	**Class B**	**Relationship**	**Multiplicity**	**Description**
Subscriber	SpeedTest	runs	1:*	One subscriber can run multiple speed tests
Subscriber	Reward	earns	1:*	One subscriber can earn multiple rewards
User	Notification	gets	1:*	One user can receive multiple notifications
User	Feedback	provides	1:*	One user can provide multiple feedback entries
Feedback	NetworkMetrics	includes	1:1	Each feedback includes corresponding network metrics
NetworkMetrics	InAppStorage	stores	*:1	Multiple metrics stored in local storage
Feedback	AI Model	classifies	*:1	AI model classifies multiple feedback entries
Notification	AI Model	prompts	1:1	AI model determines when to send notifications
InAppStorage	CloudDB	stores	1:1	Local storage syncs with cloud database
MobileOperator	CloudDB	accesses	1:1	Mobile operators access cloud analytics
Dashboard	NetworkMetrics	displays	1:*	Dashboard displays multiple network metrics

Table 10: Composition relationship

Container Class	Component Class	Description
Feedback	NetworkMetrics	Feedback contains network metrics data
Dashboard	NetworkMetrics	Dashboard contains current metrics display
Subscriber	Dashboard	Subscriber has an integrated dashboard

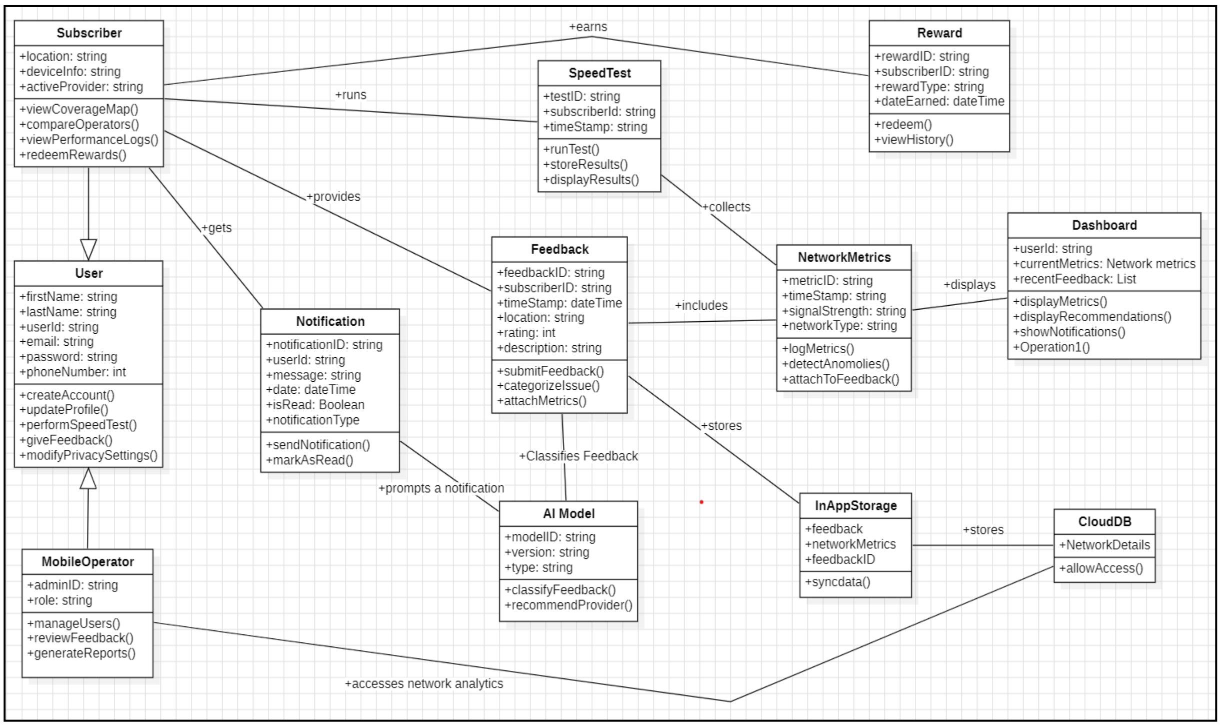


Figure 2: class diagram

4.1.2. Context Diagram

The diagram illustrates the interaction between the user and the system at a high level. The User is the external entity that communicates with the system through various actions such as giving feedback, performing speed tests, and modifying privacy settings. In response, the system provides functionalities including prompting feedback forms, showing live coverage maps, comparing operators, providing performance logs, and giving rewards.

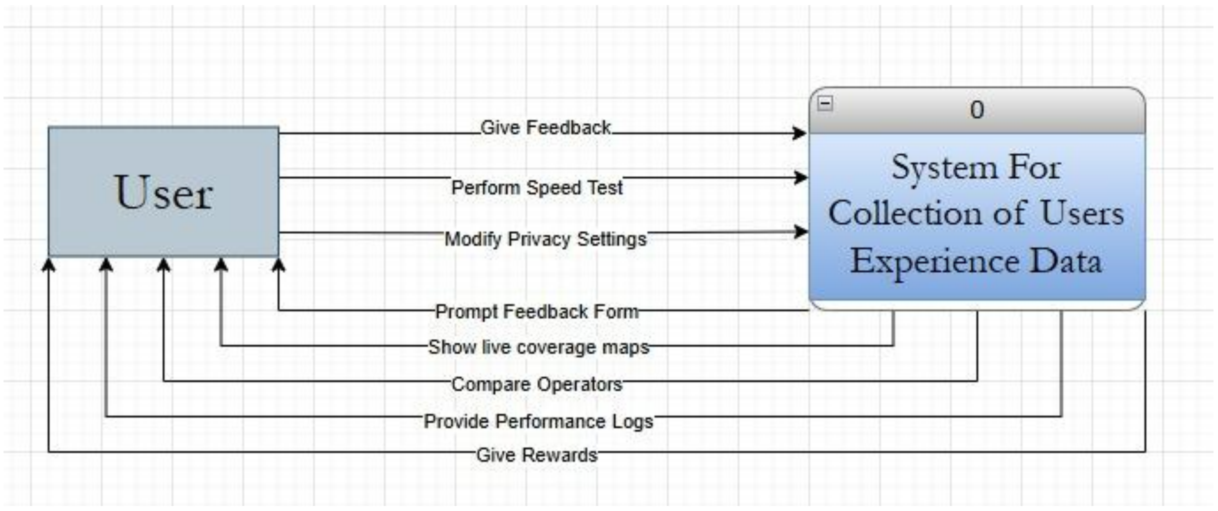


Figure 3a: Context Diagram

4.1.3. Dataflow Diagram

A Level 1 DFD provides a high-level overview of the system, breaking down the main process into key subprocesses while showing interactions with external entities and data stores.

Table 11: External Entities

S/N	Entity	Role
1.	Subsribers	Primary users who provide feedback (manual or prompted), initiate speed tests, and receive recommendations and rewards.
2.	Background Data Collection	Automatically collects network metrics (e.g., signal strength, latency, network type, geolocation) in the

		background using APIs like Android's TelephonyManager
3.	Feedback Collection:	Handles both smart (triggered by poor metrics) and manual feedback submissions, auto-filling fields like timestamp, location, and network stats.
4.	Speed Test	Allows users to test upload/download speeds and ping using third-party libraries
5.	Provider Recommendation	Uses crowdsourced data to suggest the best network provider based on location and real-time metrics, with future AI enhancements
6.	Reward System	Tracks user contributions (feedback, data sharing) and assigns points or airtime bonuses, integrated with telecom APIs
7.	AI Model Processing	Processes metrics and feedback to predict Quality of Experience (QoE) and categorize issues using models like XGBoost or DistilBERT
8.	Data Analysis & Reporting	Generates insights and reports for network operators, visualizing trends and user satisfaction
9.	Mobile Network Operators	Use reports and insights to improve services, which indirectly benefits subscribers

Table 12: Data Stores

S/N	Data Stores	Role
1.	Local Storage (D1)	Caches metrics, feedback, and speed test results offline, syncing to the cloud when connectivity is restored.

2.	Cloud Database (D2)	Stores all synced data (PostgreSQL/MongoDB) for analysis, recommendations, and reporting.
3.	AI Model Data (D3)	Stores data for training and predictions, used in future phases for AI-driven features

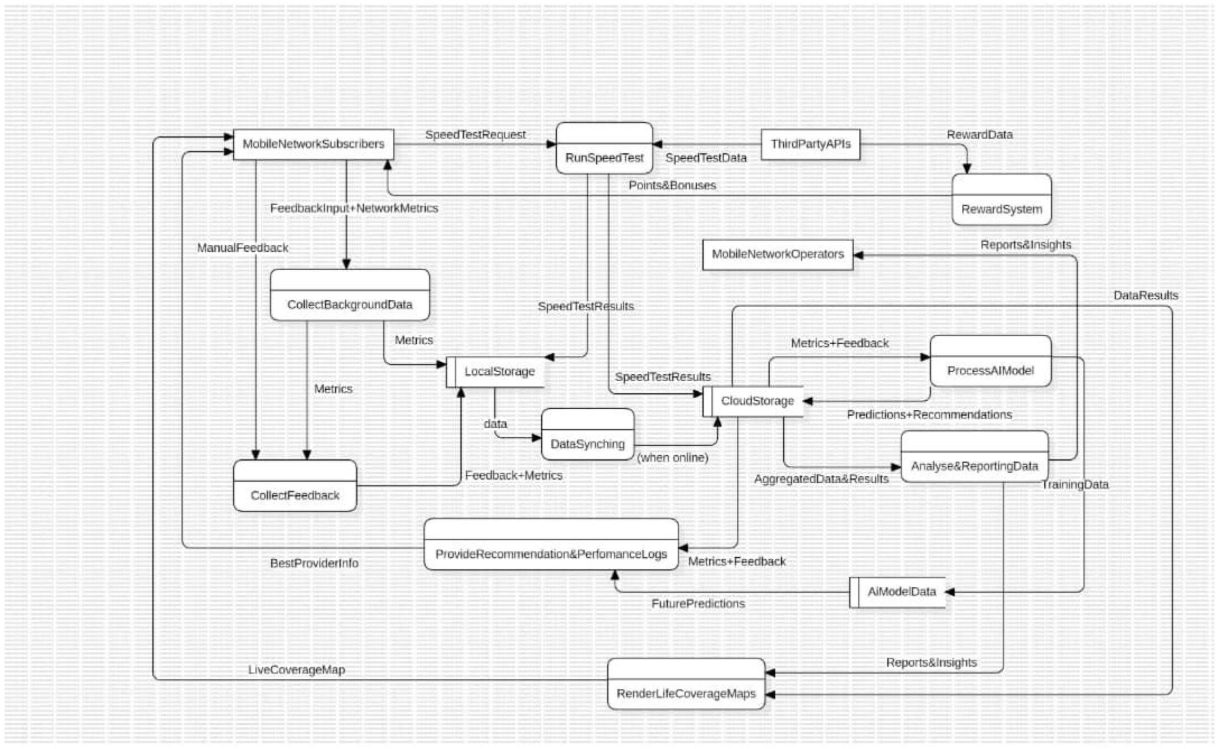


Figure 3b: Dataflow Diagram

4.1.4. Deployment Diagram

This diagram shows how our mobile app for collecting network quality data is set up and works. The system is built to do everything we need based on our project requirements, making sure users have a good experience while we collect useful data about mobile networks.

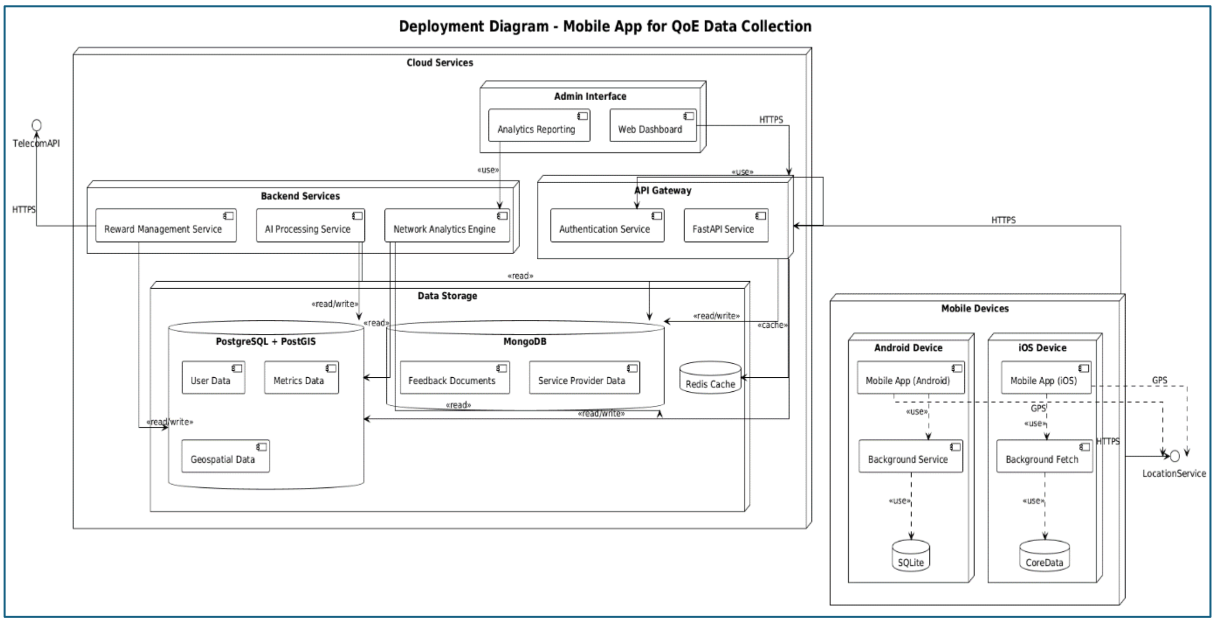


Figure 4a: Deployment Diagram (1)

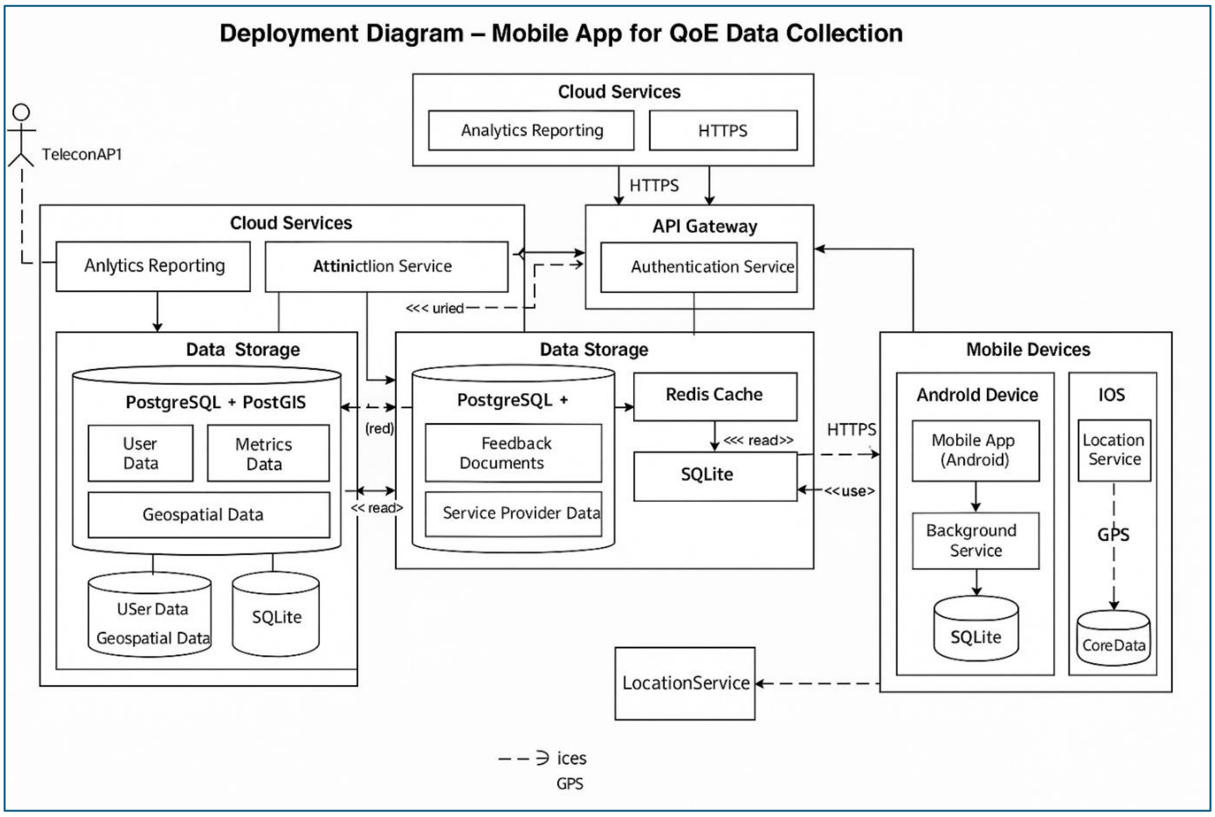


Figure 4b: Deployment Diagram (2)

4.1.5. ER Diagram

The Entity Relationship Diagram represents the logical foundation of the platform's data architecture, illustrating the relationships between core entities and supporting the system's analytical and operational requirements across all stakeholder groups.

Key entities:

- **Users/Subscribers**: People using the network and submitting feedback.
- **Feedback**: Users' opinions, complaints, and network experience.
- **Network Logs**: Automatically recorded network metrics.
- **Network Operator**: Service providers linked to both logs and feedback.

Relationships:

- **User ↔ Feedback**
    - A user can submit multiple feedback entries.
    - Relationship: One-to-Many (1:N)
    - Participation: Total (Each feedback must be linked to a user)
- **User ↔ Network Logs**
    - A user generates multiple network log entries.
    - Relationship: One-to-Many (1:N)
    - Participation: Total (Each log must be linked to a user)
- **Feedback ↔ Network Operator**
    - Feedback includes reference to the relevant network operator.
    - Relationship: Many-to-One (N:1)
    - Participation: Optional (Some feedback may not specify operator)
- **Network Logs ↔ Network Operator**
    - Logs include a reference to the network operator involved.
    - Relationship: Many-to-One (N:1)

- Participation: Optional (Some logs may not specify operator)

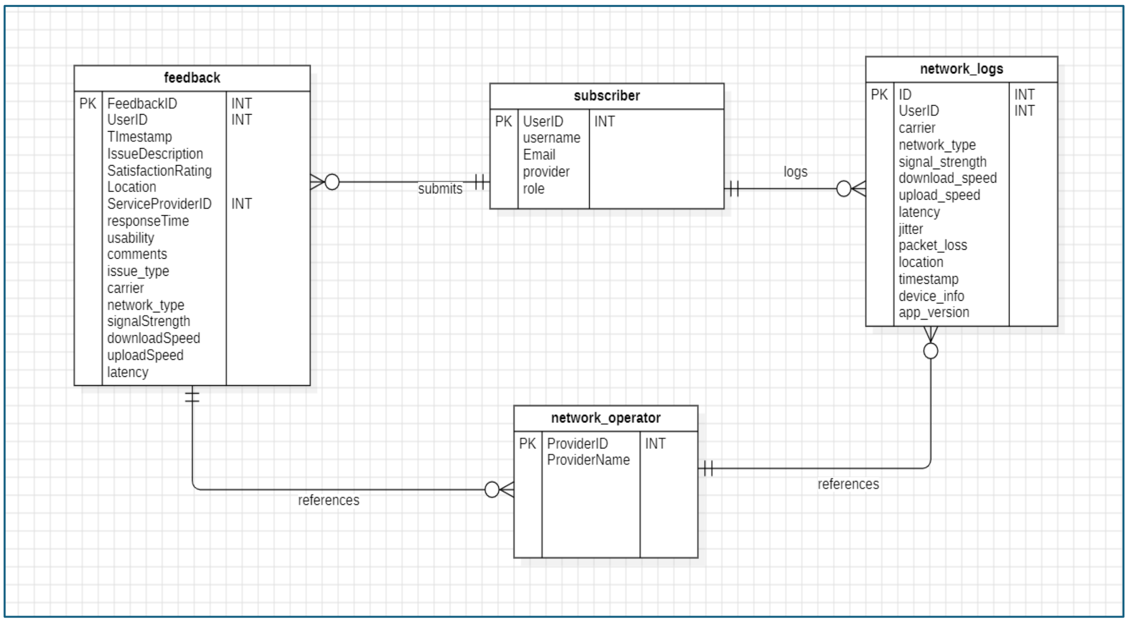


Figure 5: ER Diagram

4.2. Behavioural Diagrams

4.2.1. Use Case Diagram

This use case diagram illustrates the interactions between two main user types:

- Mobile Network Subscribers:
  End users of mobile services who interact with the system to provide feedback, view reports, and receive network-related recommendations and rewards.
- Mobile Network Operators:
  End users of mobile services who interact with the system to provide feedback, view reports, and receive network-related recommendations and rewards.

It also describes the functionalities of a QoE (Quality of Experience) Data Collection System. The system aims to collect, process, and analyze network performance and user feedback to enhance the overall quality of mobile services.

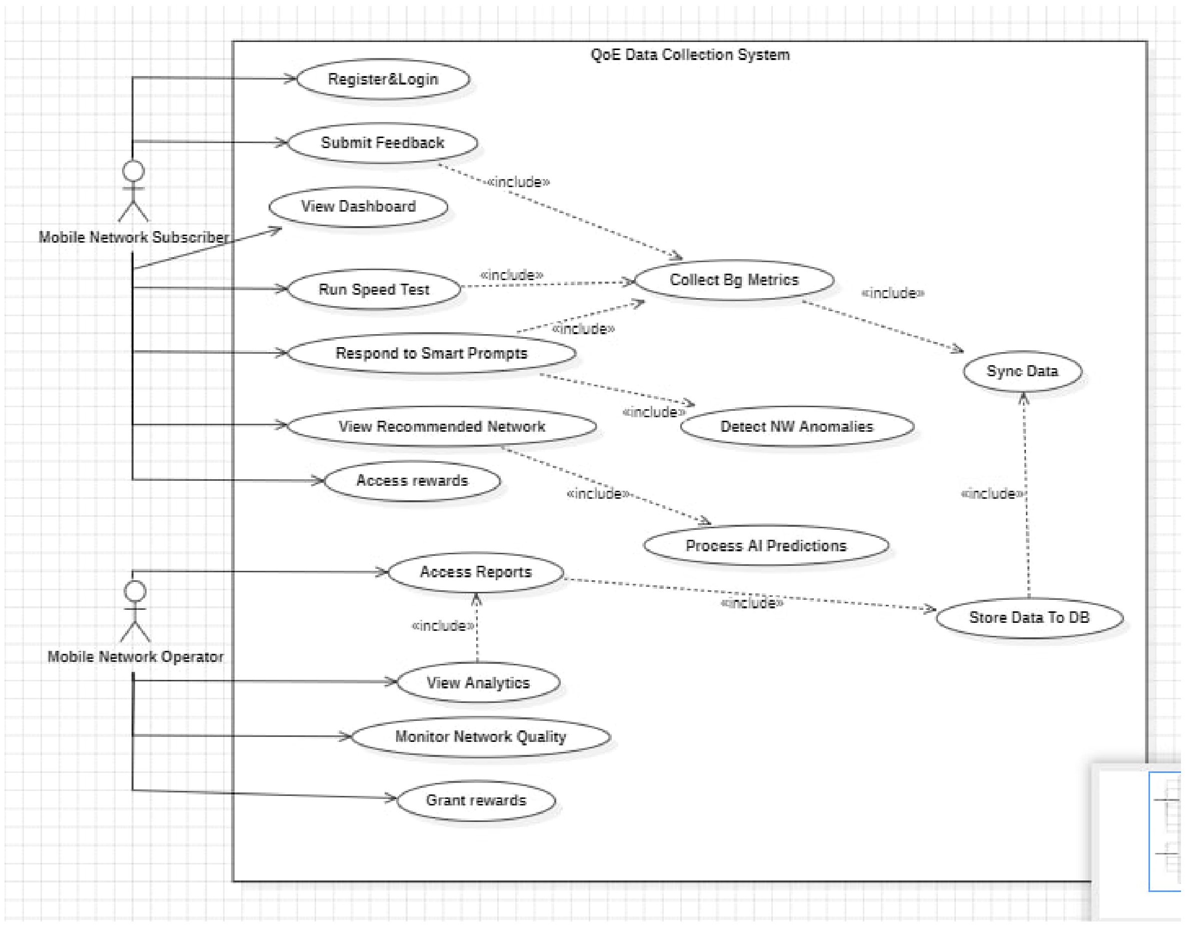


Figure 6: Use Case Diagram

4.2.2. Sequence Diagram

i. User Registration and Login Sequence

This diagram illustrates sequence which handles both new user registration and existing user authentication processes

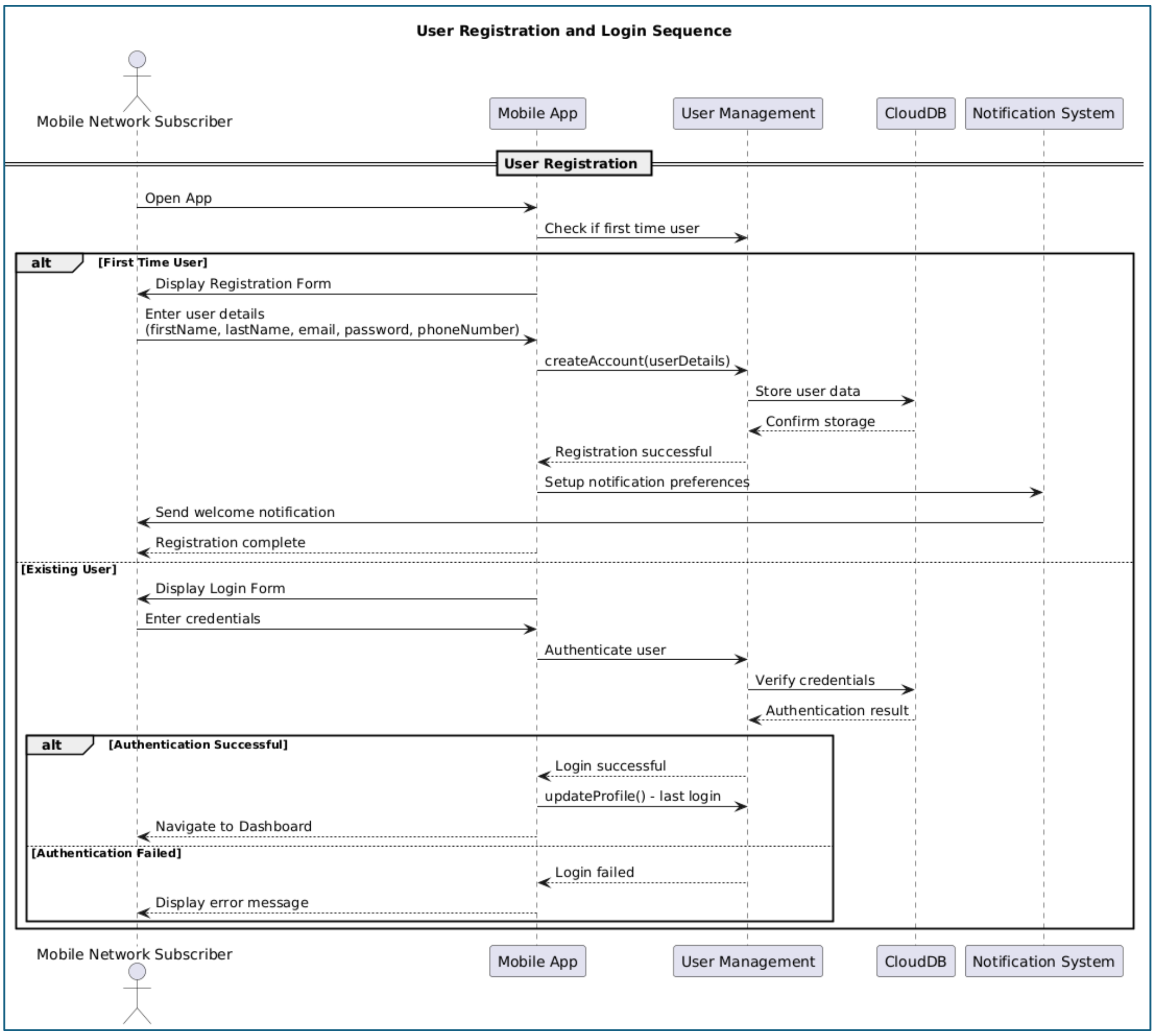


Figure 7a: Sequence Diagram For User Registration and Login Sequence

ii. Background Data Collection and Speed Test Sequence

Our second sequence diagram represents the core functionality of our QoE data collection system. This sequence operates continuously in the background, automatically gathering network performance metrics and conducting periodic speed tests without requiring user intervention.

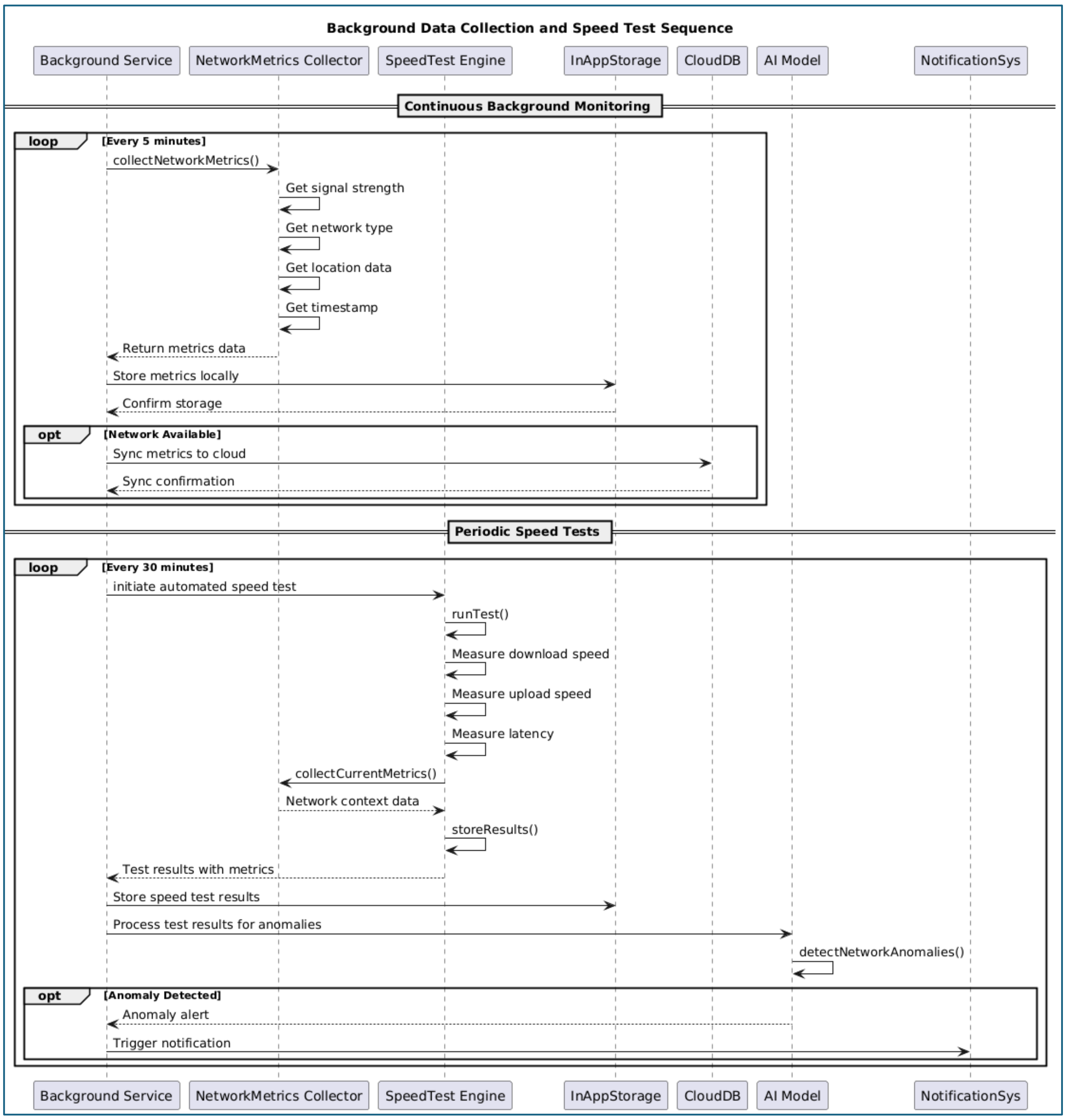


Figure 7b: Sequence Diagram For Background Data Collection and Speed Test Sequence

iii. User Feedback Collection and Smart Prompts Sequence

Our third sequence diagram illustrates our intelligent feedback collection system, which combines AI-driven prompt generation with user-friendly feedback submission processes. This sequence represents the subjective data collection component of our QoE system.

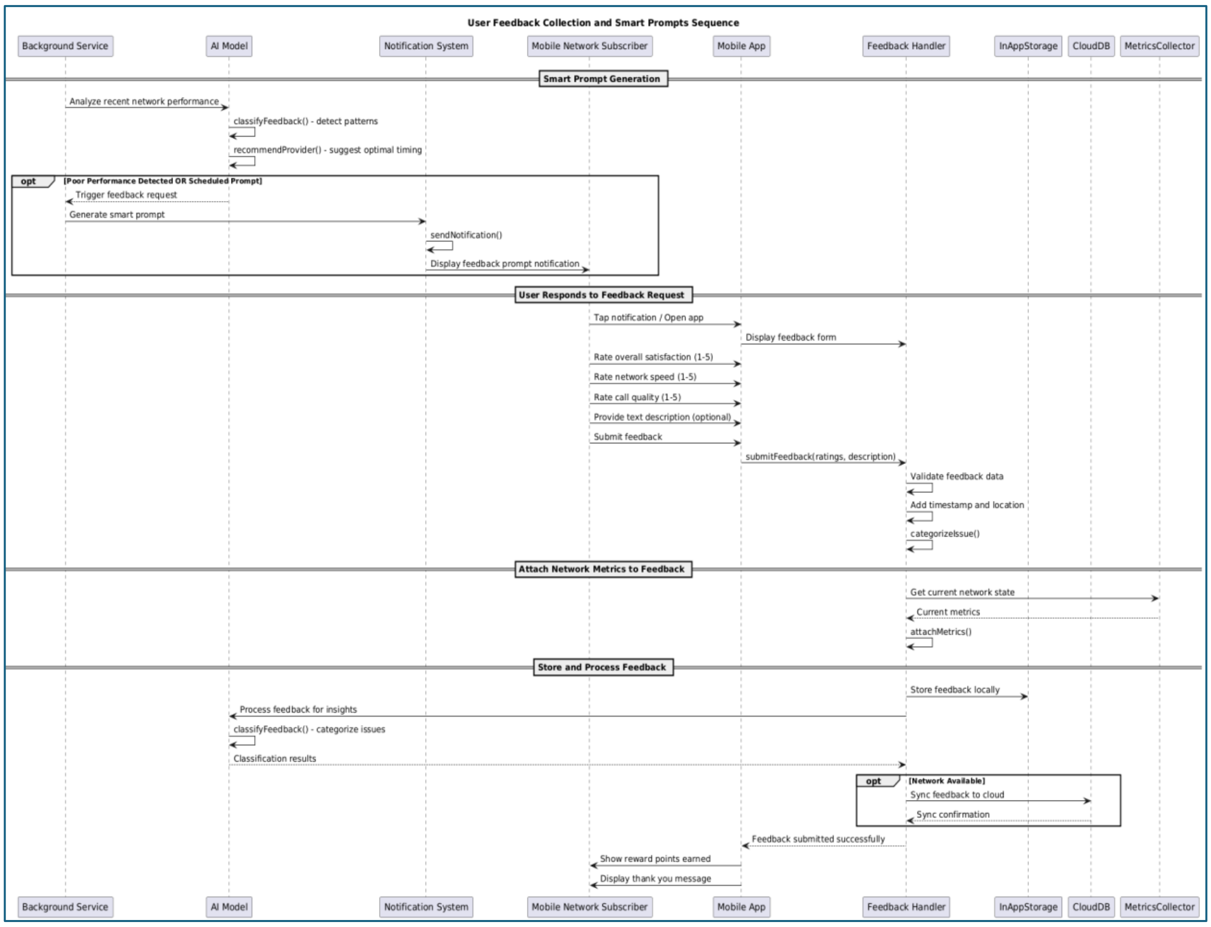

User Feedback Collection and Smart Prompts Sequence
Background Service
AI Model
Notification System
Mobile Network Subscriber
Mobile App
Feedback Handler
InAppStorage
CloudDB
MetricsCollector
Smart Prompt Generation
Analyze recent network performance
classifyFeedback() - detect patterns
recommendProvider() - suggest optimal timing
opt
[Poor Performance Detected OR Scheduled Prompt]
Trigger feedback request
Generate smart prompt
sendNotification()
Display feedback prompt notification
User Responds to Feedback Request
Tap notification / Open app
Display feedback form
Rate overall satisfaction (1-5)
Rate network speed (1-5)
Rate call quality (1-5)
Provide text description (optional)
Submit feedback
submitFeedback(ratings, description)
Validate feedback data
Add timestamp and location
categorizeIssue()
Attach Network Metrics to Feedback
Get current network state
Current metrics
attachMetrics()
Store and Process Feedback
Store feedback locally
Process feedback for insights
classifyFeedback() - categorize issues
Classification results
opt
[Network Available]
Sync feedback to cloud
Sync confirmation
Feedback submitted successfully
Show reward points earned
Display thank you message

Figure 7c: Sequence Diagram For User Feedback Collection and Smart Prompts Sequence

iv. Mobile Network Operator Analytics And Reporting Sequence

Our fourth and final sequence diagram demonstrates the comprehensive analytics and reporting capabilities we provide to mobile network operators. This sequence transforms collected data into actionable insights for network optimization and customer experience management.

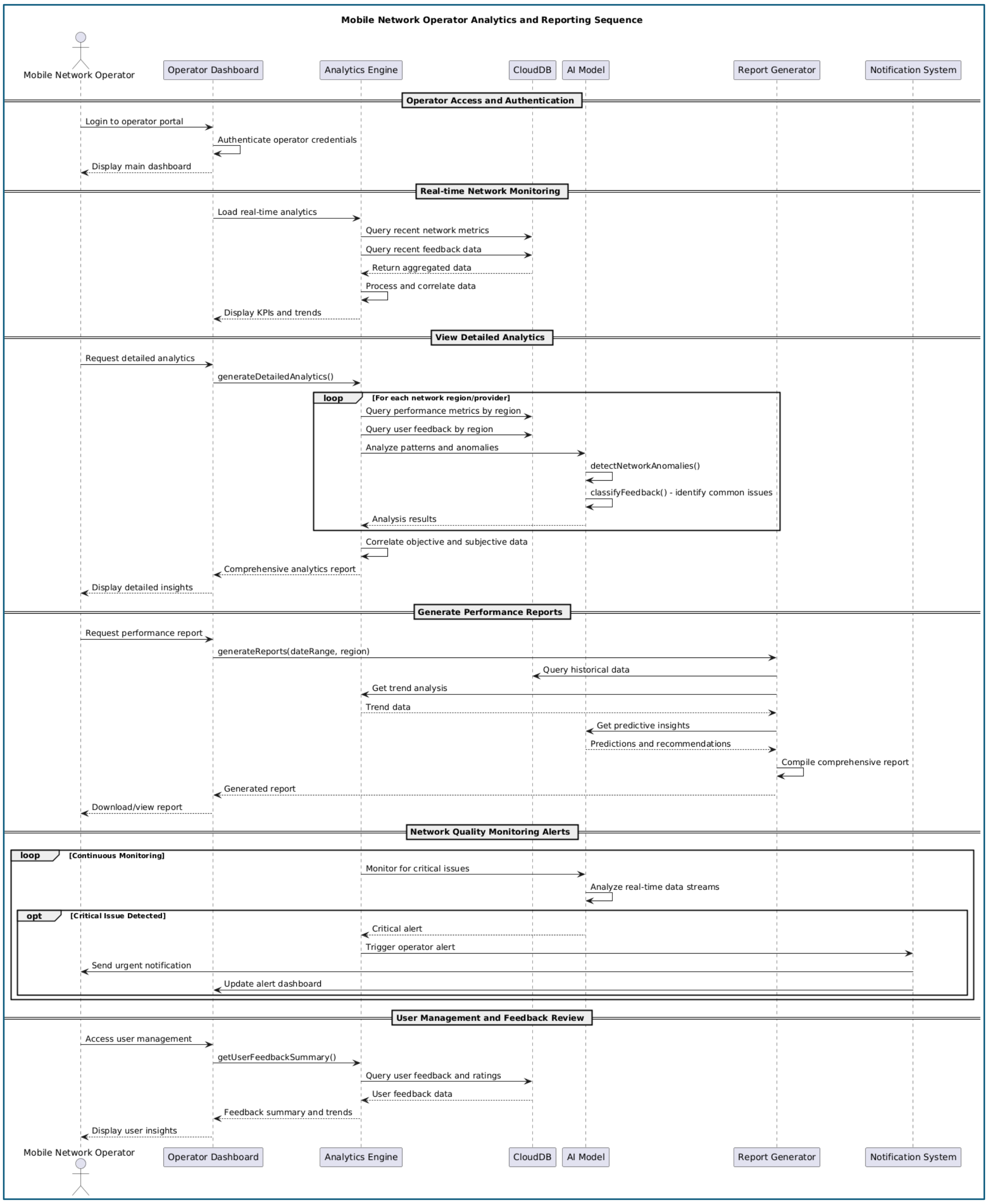

Mobile Network Operator Analytics and Reporting Sequence
Mobile Network Operator
Operator Dashboard
Analytics Engine
CloudDB
AI Model
Report Generator
Notification System
Operator Access and Authentication
Login to operator portal
Authenticate operator credentials
Display main dashboard
Real-time Network Monitoring
Load real-time analytics
Query recent network metrics
Query recent feedback data
Return aggregated data
Process and correlate data
Display KPIs and trends
View Detailed Analytics
Request detailed analytics
generateDetailedAnalytics()
loop
[For each network region/provider]
Query performance metrics by region
Query user feedback by region
Analyze patterns and anomalies
detectNetworkAnomalies()
classifyFeedback() - identify common issues
Analysis results
Correlate objective and subjective data
Comprehensive analytics report
Display detailed insights
Generate Performance Reports
Request performance report
generateReports(dateRange, region)
Query historical data
Get trend analysis
Trend data
Get predictive insights
Predictions and recommendations
Compile comprehensive report
Generated report
Download/view report
Network Quality Monitoring Alerts
loop
[Continuous Monitoring]
Monitor for critical issues
Analyze real-time data streams
opt
[Critical Issue Detected]
Critical alert
Trigger operator alert
Send urgent notification
Update alert dashboard
User Management and Feedback Review
Access user management
getUserFeedbackSummary()
Query user feedback and ratings
User feedback data
Feedback summary and trends
Display user insights

Figure 7d: Sequence Diagram For Mobile Network Operator Analytics And Reporting Sequence

4.3. UI Diagrams (Figma)

The following prototypes were created to visualize UI of the application. The key screens were categorized under Subscriber and Admin:

4.3.1. Subscriber-Facing Screens

- ❖ Speed Test Page: One-tap interface measuring download/upload speeds, ping, and jitter with bold visuals for quick assessment.
- ❖ Dashboard: Real-time network metrics (latency, signal strength) and historical activity logs in a card-based, non-technical layout.
- ❖ Data Logs: Filterable view of network/feedback history with timestamps and ratings for trend analysis.

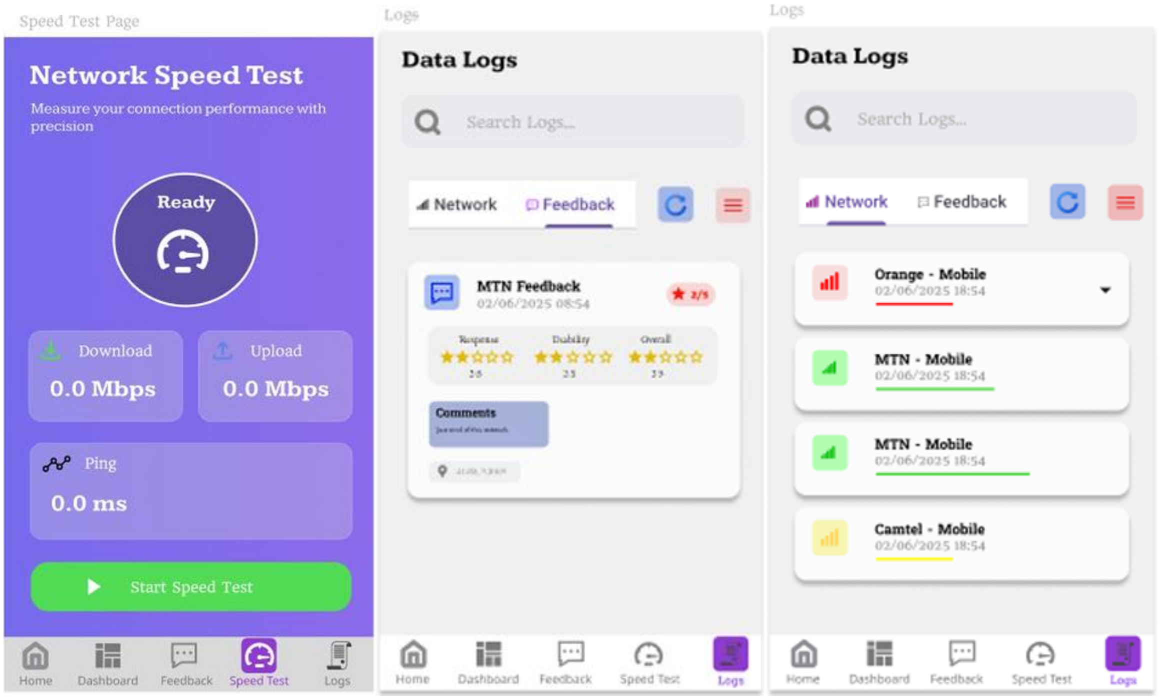
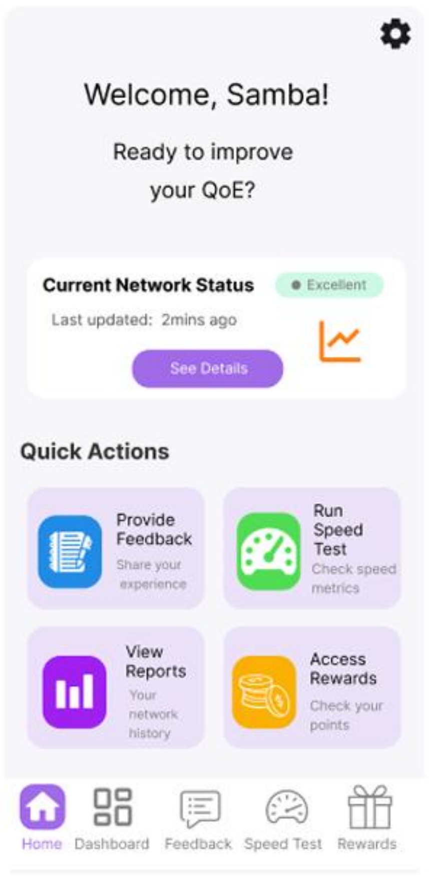


Figure 8a: Speed Test Page and Data Logs Screens

- ❖ Home Page: Central hub with network status, quick actions (feedback, speed test), and gamified rewards.
- ❖ Settings: Toggle controls for data collection and notifications, balancing functionality with privacy.
- ❖ Feedback Flow: Two-step process—(1) Star ratings + issue categorization, (2) Detailed comments with auto-captured context (location, network type).

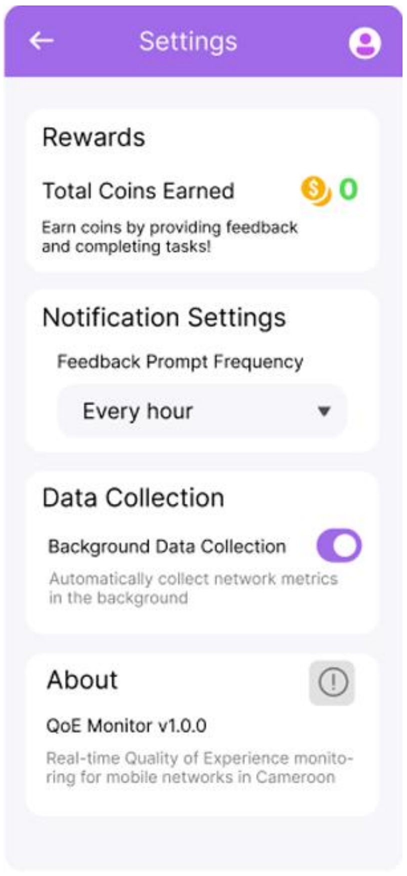
Subscriber Home Page


Welcome, Samba!

Ready to improve your QoE?

Current Network Status — Excellent

Last updated: 2mins ago


Settings

Rewards

Total Coins Earned: 0

Earn coins by providing feedback and completing tasks!

Notification Settings

Feedback Prompt Frequency: Every hour

Data Collection

Background Data Collection: On

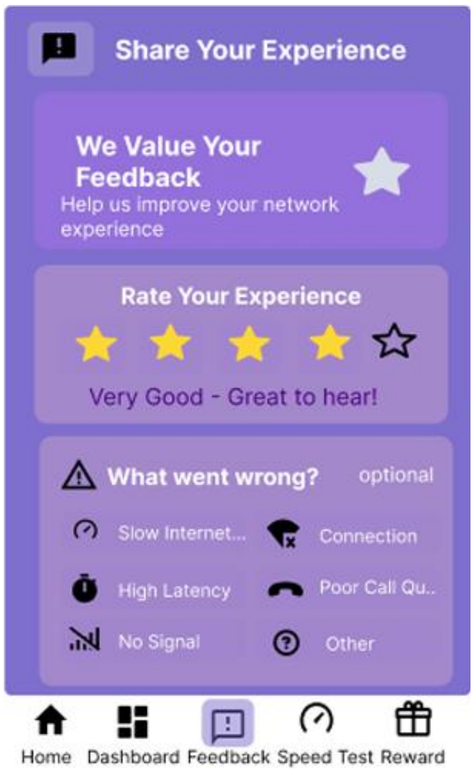
Automatically collect network metrics in the background

About

QoE Monitor v1.0.0

Real-time Quality of Experience monito-ring for mobile networks in Cameroon

Figure 8b: Home Page and Settings Screens

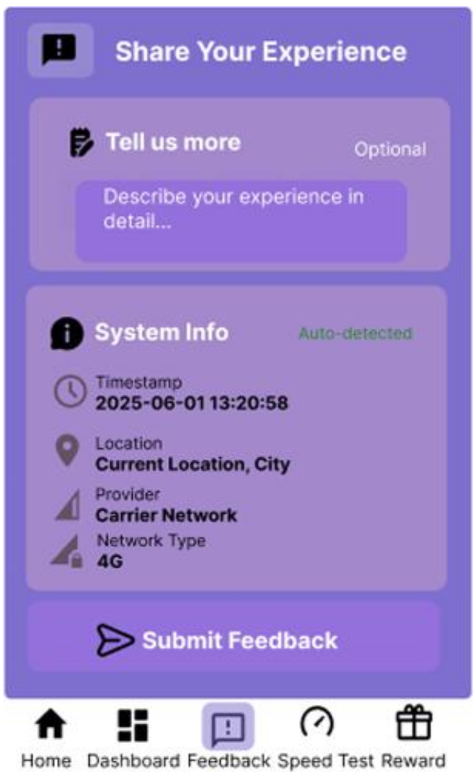

Share Your Experience

We Value Your Feedback

Help us improve your network experience

Rate Your Experience: ★★★★☆

Very Good - Great to hear!


Share Your Experience

Tell us more — Optional

Describe your experience in detail...

System Info — Auto-detected

Timestamp: 2025-06-01 13:20:58

Location: Current Location, City

Provider: Carrier Network

Network Type: 4G

Figure 8c: Feedback Screens

4.3.2. Admin Portal

- <u>Home Page:</u> Operator dashboard with high-density analytics, system health alerts, and rapid access to reports.
- <u>Dashboard:</u> The dashboard is tailored for administrators to analyze key performance indicators.
- <u>Login Page:</u> The admin login screen offers a secure entry point for network administrators

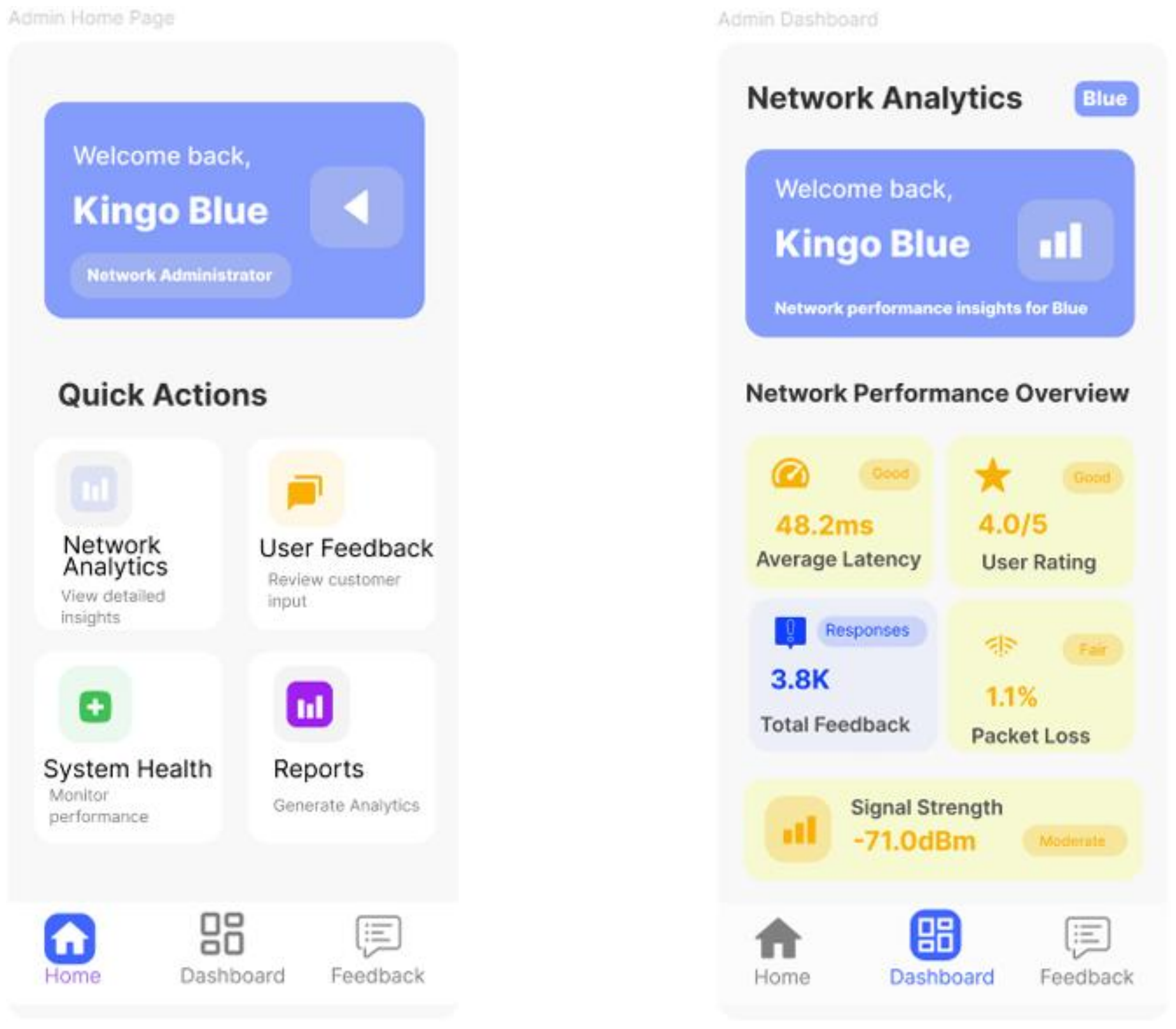


Figure 8d: Home Page and Dashboard screens

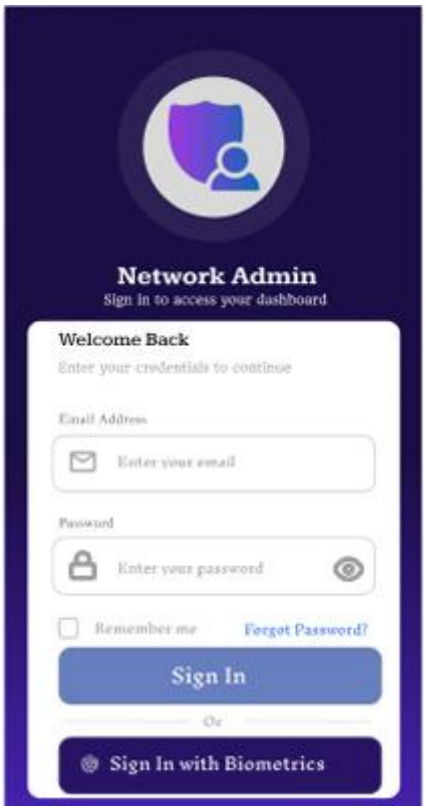


Figure 8e: Admin Login screen

5. Global Architecture of the System

The platform follows a cloud-native, three-tier architecture that integrates:

- Mobile clients (Flutter) for data collection and user feedback
- Microservices backend (FastAPI/Python) for processing
- PostgreSQL/TimescaleDB for storage with AI analytics
- Redis caching and Kubernetes orchestration for scalability

Key features:

- Multi-tenant isolation via row-level security
- Hybrid AI (on-device + cloud models)
- Real-time dashboards with geographic heatmaps
- Offline-first mobile data collection

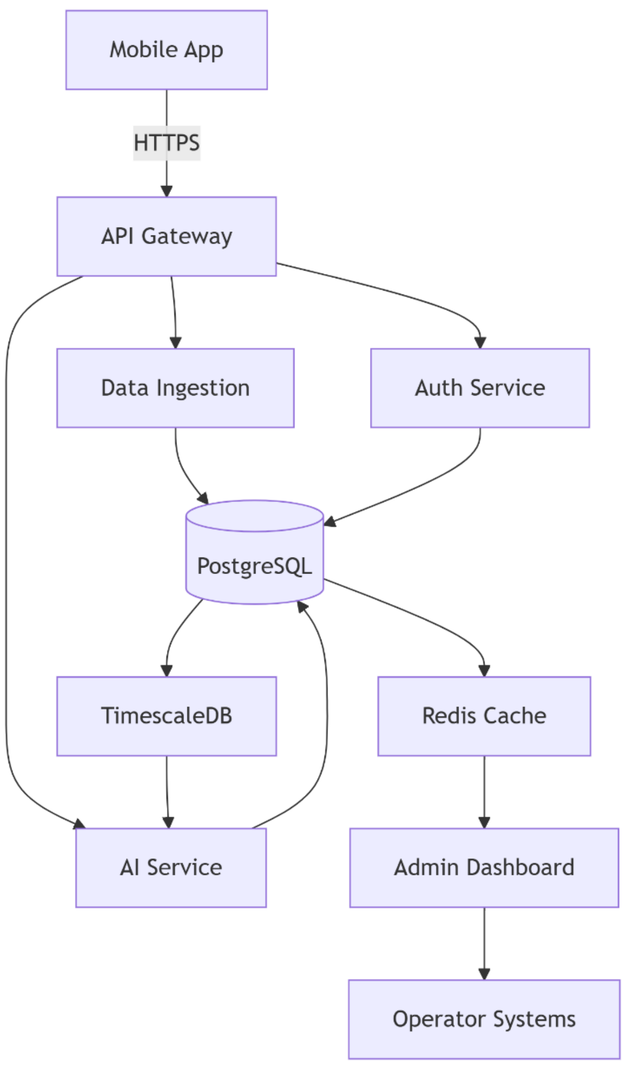


Figure 9: Architecture of the System

6. Partial Conclusion

The Analysis and Design phase has successfully translated requirements into a technical blueprint for the system. Through layered architecture and iterative prototyping, we established solutions for key challenges like offline data collection and multi-tenant analytics. The designed PostgreSQL schema ensures secure data isolation, while UML diagrams and UI prototypes validate the system's feasibility and usability.

This phase's outcomes—including optimized database queries and battery-efficient background processes—provide a solid foundation for implementation. The modular design also accommodates future enhancements like predictive analytics. With architecture and specifications now clearly defined, the project is positioned to move into the development and testing phases.

CHAPTER 4: IMPLEMENTATION AND RESULTS

1. Introduction

The WaveWatch platform is an integrated solution designed to enhance the Quality of Experience (QoE) for mobile network users in Cameroon. In a landscape where telecommunications users often face inconsistent network performance and limited transparency from service providers, this system bridges the gap between users' real-world experiences and the data-driven insights needed by telecom operators to optimize their networks.

The primary goal of the project is to provide a user-centric, data-powered system that enables real-time monitoring, structured feedback collection, and detailed network performance analytics. This empowers both end-users who benefit from transparency and improved services and telecom operators who gain actionable intelligence for targeted infrastructure improvements.

2. Tools and Materials used

Tool	Purpose	How It Was Used
Figma	UI/UX Design	Used to design wireframes, mockups, and interactive UI prototypes for the mobile app and admin dashboard. It helped define layout, user flows, color schemes, and component interactions before implementation.
Flutter	Cross-Platform Frontend Development	Used to build the mobile app and admin interface from a single codebase for both Android and iOS. It supported responsive layouts, animations, and reusable widgets for consistent design.
FastAPI	Backend API Development	Used to build asynchronous, high-performance RESTful APIs for user management, feedback submission, analytics, and data retrieval. It also generated automatic API documentation using OpenAPI.
PostgreSQL	Relational Database	Used to store structured data including users, feedback, network logs, and provider details. It supported normalized schema, constraints, indexing, and performance-tuned queries.
Supabase	Database Hosting	Hosted the PostgreSQL database in the cloud. Provided integrated authentication, real-time subscriptions, role-based access control, and database backups.

SQLAlchemy	ORM (Object-Relational Mapping)	Used to define Python data models and interact with the PostgreSQL database in the backend. Simplified queries and ensured consistency between application logic and the schema.
JWT (JSON Web Tokens)	Authentication & Authorization	Used for stateless, token-based authentication. JWTs carried encoded user information and roles, enabling secure access control across API endpoints.
Android telephony	Background Data Collection	Used in the mobile app to schedule periodic data collection (e.g., signal strength, speed, latency) even when the app was idle. Ensured minimal battery consumption.
CSV Export Tools	Analytics & Reporting	Enabled administrators to export feedback and performance logs in CSV format for offline analysis, reporting, and regulatory use.
Geolocation & Device Info APIs	Contextual Data Gathering	Automatically attached user location, device metadata, and carrier information to feedback and logs. Enabled accurate network performance mapping.
VS Code (Visual Studio Code)	Code Editing & Debugging	Served as the primary code editor for frontend (Flutter/Dart) and backend (Python/FastAPI) development. Integrated with extensions for linting, Git, and testing.
Git & GitHub	Version Control & Collaboration	Used for source code management, collaborative development, and deployment automation. Managed feature branches, commits, pull requests, and hosted code repositories. CI/CD workflows could be integrated for testing and deployment through GitHub Actions or similar services.

3. Description of the implementation process

The WaveWatch platform was developed with a strong emphasis on user-centered design, ensuring that both end-users (subscribers) and system administrators (network operators) have tailored experiences that meet their unique needs. The design approach balances clarity, functionality, and engagement to make complex network data accessible and actionable.

Dual-Interface System

1. **Subscriber App: Intuitive, Accessible, and Engaging**

The mobile application for subscribers was designed to prioritize **simplicity and usability**, targeting a broad user base with varying technical backgrounds. Key features include:

- Card-based UI with large icons and bold colors for readability
- Real-time network speed tests and signal strength indicators

- ➢ Feedback submission tools that support both quick star ratings and detailed issue reports
- ➢ Gamification and rewards system to encourage user engagement
- ➢ Minimal technical jargon, replaced by visual indicators (e.g., green/yellow/red) to represent network quality at a glance

By designing around common mobile usage patterns and cognitive ease, the app ensures users can participate in network monitoring without being overwhelmed by technical complexity.

2. Admin Dashboard (App): Analytical, Data-Rich, and Role-Based

The administrator dashboard caters to the needs of telecom providers like MTN, Orange, and Blue, with a focus on:

- ➢ Advanced data visualizations (charts, maps, trends) for network performance and user feedback
- ➢ Provider-specific filtering to ensure data visibility is restricted based on administrative roles
- ➢ Tools for sentiment analysis, feedback export, and real-time network monitoring
- ➢ A professional, clean UI that supports deep exploration of metrics across time, location, and user base

This interface supports **strategic decision-making** by enabling operators to identify patterns, troubleshoot issues, and prioritize infrastructure upgrades.

Progressive Disclosure and Visual Hierarchy

The system uses **progressive disclosure** to present information in layers, reducing cognitive overload:

- ✓ High-level summaries (e.g., current network status, user satisfaction) are shown first.
- ✓ More detailed data (e.g., performance trends, individual feedback logs) is available via interactions like taps, clicks, or expandable panels.

The UI hierarchy is structured to ensure that:

- ✓ **Primary information** (e.g., provider, signal quality) appears prominently.
- ✓ **Supporting details** are contextually revealed based on user interaction.

This layered approach empowers both casual users and technical administrators to navigate data at their own comfort level.

Accessibility and Usability Considerations

Accessibility and inclusiveness were foundational in the design process:

- ✓ Color contrast and large touch targets ensure legibility and ease of interaction on all devices.
- ✓ Use of icons with accompanying text labels improves comprehension for users with cognitive or visual challenges.
- ✓ All designs adhere to WCAG accessibility guidelines, supporting diverse user needs.
- ✓ Responsive layouts ensure optimal display across various screen sizes and orientations.

Furthermore, micro-interactions such as **loading animations**, **real-time signal bars**, and **transitional effects** improve feedback and maintain a smooth, modern user experience.

The *WaveWatch* platform delivers a **dual-purpose system**: one that demystifies network performance for the everyday user and empowers operators with actionable insights—each interface purpose-built through a thoughtful, user-first design methodology.

Data Flow and Functional Synchronization

The *WaveWatch* platform implements a seamless data flow architecture that connects its user interfaces with backend services, enabling continuous collection, processing, and visualization of mobile network performance data. This synchronized system ensures that both subjective user experiences and objective network metrics are accurately captured and analyzed in real-time.

UI Component Roles in Feedback and Data Logging

The Subscriber Application acts as the main entry point for data input. It enables users to:

- ✓ Submit feedback through structured forms (star ratings, issue categories) and open comments.
- ✓ Allow the app to automatically collect network logs during use and in the background.
- ✓ View their own submitted feedback, logs, and performance trends via intuitive dashboards.

Each UI interaction is designed not only to collect input but also to enrich it with contextual data (e.g., timestamp, GPS location, device model, current signal strength), making the information highly actionable for downstream analysis.

Real-Time and Background Data Collection

WaveWatch supports **two key modes of data collection**:

1. Real-Time Data Collection

- When users perform actions like speed tests or submit feedback, the system immediately collects:
  - Download/upload speeds
  - Signal strength
  - Network type
  - Current location and time
  - Jitter
  - Latency

  These metrics are bundled with the user's feedback and transmitted to the backend via API calls.

2. Background Monitoring

- The app uses **Android JobScheduler** and **iOS Background App Refresh** to collect data periodicallyeven when the app is not in use.
- Data such as latency, jitter, and packet loss are collected silently, ensuring minimal user interruption and low battery consumption.
- This enables long-term trend analysis and builds a comprehensive network quality dataset.

All data collected, whether in real-time or in the background, is stored locally first and then synced with the backend once internet connectivity is available ensuring data persistence and completeness.

System Architecture Overview

The WaveWatch platform is built upon a modular and scalable architecture that integrates frontend user interfaces with robust backend services and databases. This end-to-end design ensures a seamless flow of data from mobile network users to telecom providers, supporting real-time monitoring, feedback collection, and data-driven decision-making.

Holistic Architecture Integration

The platform consists of two primary applications that work in concert to deliver comprehensive network monitoring capabilities. The first is a subscriber-facing mobile application that enables users to perform network tests, provide feedback, and view personalized reports. The second is an administrator dashboard application that allows telecom providers to access detailed analytics and manage user feedback effectively.

These applications are tightly integrated with backend services through RESTful APIs, ensuring efficient data exchange, synchronization of network logs and user feedback, and real-time access to performance analytics. Together, the UI and backend components form a robust full-stack system that captures both subjective user experiences and objective network metrics, transforming them into valuable insights that drive user satisfaction and service provider efficiency.

Three-Layer Architecture Design

The system follows a structured three-layer architecture approach that separates concerns and enhances maintainability:

- ❖ **Presentation Layer (Frontend)**

The presentation layer is built using Flutter for cross-platform compatibility across Android and iOS devices. This layer includes user-friendly interfaces designed specifically for both subscribers and administrators, with each interface tailored to support their respective workflows and requirements.

- ❖ **Business Logic Layer (Backend Services**

The business logic layer is developed using FastAPI, which provides excellent support for asynchronous request handling and scalable API development. This layer implements critical functionalities including authentication mechanisms, role-based access control, data validation, analytics processing, and integration logic that connects various system components.

- ❖ **Data Layer (Persistence and Storage)**

The data layer is powered by PostgreSQL hosted on Supabase, providing a robust foundation for data management. This layer organizes and stores feedback data, user profiles, network performance metrics, and provider information in a normalized relational schema. The implementation includes role-based and provider-specific data access controls, ensuring comprehensive multi-tenant support.

Cross-Platform Support and Scalability

The platform's technology stack has been carefully selected to maximize compatibility and performance. Flutter enables consistent UI/UX design and performance across multiple device types while significantly reducing development and maintenance efforts. FastAPI delivers a performant and developer-friendly backend framework that supports asynchronous tasks and provides automatic API documentation.

Supabase, built on PostgreSQL, provides real-time database capabilities, secure cloud hosting, and simplified schema evolution. This integrated and scalable architecture enables WaveWatch to effectively meet its core objectives of enhancing mobile network Quality of Experience (QoE), providing reliable analytics to telecom operators, and maintaining system performance and extensibility in real-world deployments.

Database Architecture and Implementation

The backend of the WaveWatch platform is powered by a robust, cloud-hosted relational database architecture that ensures efficient data storage, fast query performance, and secure multi-tenant access control. At the heart of the system lies a PostgreSQL database hosted on Supabase, selected for its strong support for structured data, built-in authentication capabilities, and scalability for real-time analytics and mobile integration.

PostgreSQL Schema and Supabase Hosting

The database is deployed on Supabase, a Backend-as-a-Service platform built on PostgreSQL. This cloud-based environment provides comprehensive features including automated backups and recovery systems, real-time data subscriptions for future reactive applications, integrated role-based access control, and support for both RESTful APIs and direct PostgreSQL queries.

The database schema follows third normal form (3NF) principles to reduce data redundancy, enhance data integrity, and improve long-term maintainability. Where performance considerations require it, selective denormalization has been applied to optimize high-frequency access patterns while maintaining overall data consistency.

Frontend–Backend API Integration

The platform's **backend services** are exposed through a series of **RESTful APIs** developed using **FastAPI**, which support:

- ✓ **Feedback Submission API**: Handles user ratings, textual input, and auto-collected metrics.
- ✓ **Network Log API**: Accepts periodic uploads of network performance data from the app.
- ✓ **Data Retrieval APIs**: Provide personalized or provider-specific data to the frontend for visualization.

The Flutter frontend makes asynchronous HTTP requests to these endpoints, with secure authentication via JWT tokens. Error handling, token refresh, and request retries are managed in-app to ensure reliability.

Data Feedback Loops: Supporting Users and Operators

The platform is built around **a** closed feedback loop that benefits both users and network providers:

For Users:

- ➢ Users receive real-time insights into their current network quality.
- ➢ They can compare past performance, provider reliability, and location-specific conditions.
- ➢ The app includes a **reward system** that encourages continued participation in data sharing and feedback.

For Providers (Operators):

- ❖ The **Admin Dashboard** (App) aggregates and visualizes feedback and performance logs.

- Providers can:
  - ✓ Filter data by region, time, network type, or issue category.
  - ✓ Detect service degradations early.
  - ✓ Analyze trends in user satisfaction and sentiment.
  - ✓ Export data for further BI or ML integration.

This real-time insight allows providers to proactively resolve network issues and improve customer satisfaction—thereby completing the feedback loop that starts with user input and ends in network optimization.

WaveWatch's data flow model is a tightly coordinated system where the frontend acts as a sensor, the backend as the processor**,** and the dashboard as the decision engine**.** Through effective synchronization between components, the system ensures that all stakeholders end-users and telecom operators derive tangible value from the shared data ecosystem.

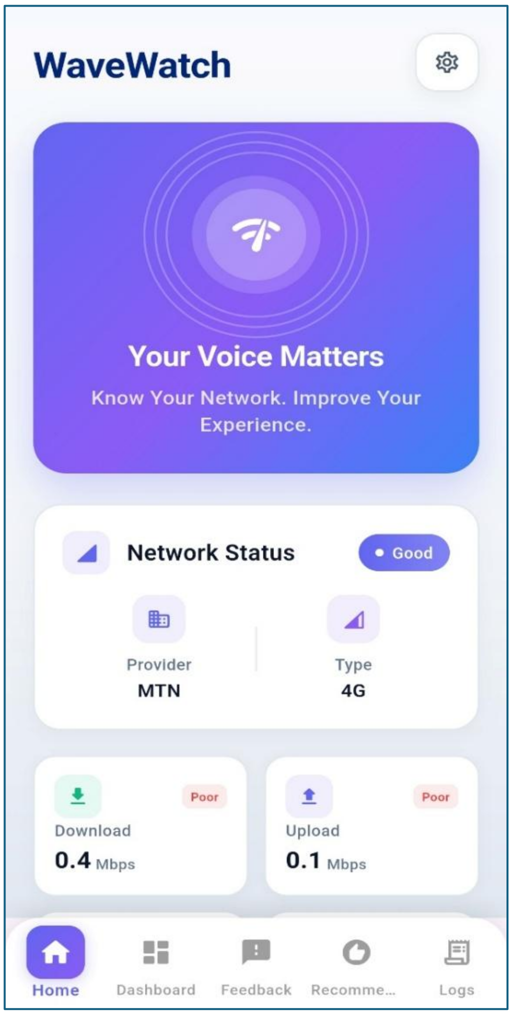
4. Presentation and interpretation of results

4.1. Mobile and Admin Applications (UI Design)

a. **The Subscriber Application**, developed using **Flutter**, offers an intuitive and visually engaging experience. It enables users to:

- ➢ Run network speed tests
- ➢ Submit feedback on connectivity issues
- ➢ View network status and logs
- ➢ Earn participation-based rewards

The interfaces for the various functionalities are as shown below.


WaveWatch

Your Voice Matters

Know Your Network. Improve Your Experience.

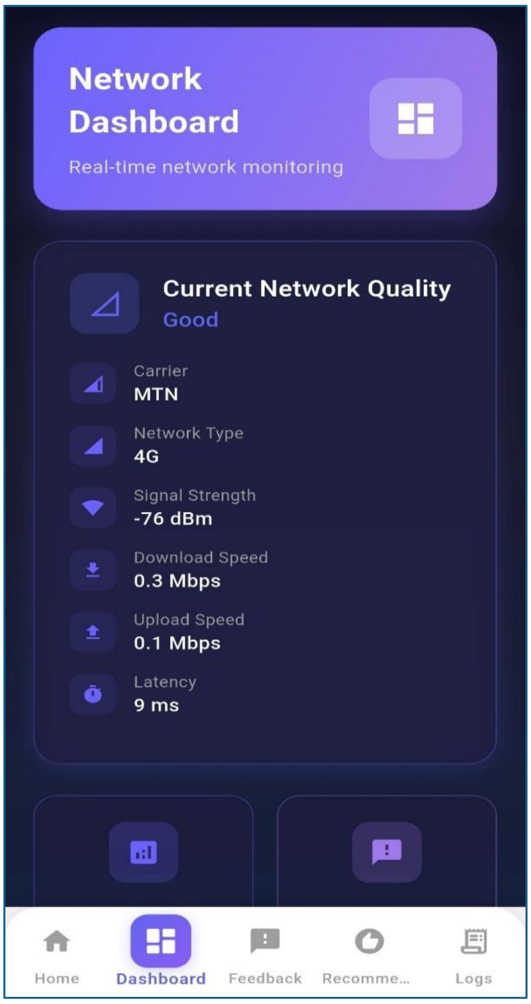
Network Status — Good

Provider: MTN

Type: 4G

Download: 0.4 Mbps — Poor

Upload: 0.1 Mbps — Poor


Network Dashboard

Real-time network monitoring

Current Network Quality: Good

Carrier: MTN

Network Type: 4G

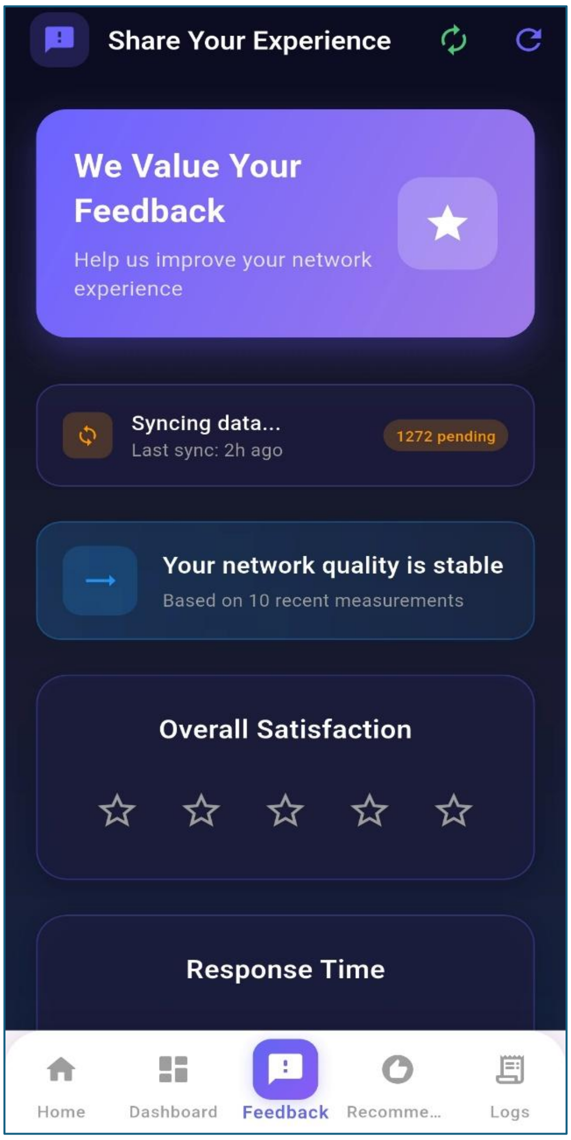
Signal Strength: -76 dBm

Download Speed: 0.3 Mbps

Upload Speed: 0.1 Mbps

Latency: 9 ms

These are the screens for the landing page and the Dashboard screens as well. The home page or landing page carries basic information from the device that has been simulated or real time data regarding the network status at that time. The network metrics being displayed to the user on this screen are the download/upload speed, the jitter, latency and the network type/ provider at that time. These values are collected at real time and changes after every 5s so as to keep being updated. The dashboard screen carries complete network metrics now including the signal strength as well. Also, has history of previous metrics. With this, the users will be able to know or detect the fluctuations in their connectivity and hence better experience.


Share Your Experience

We Value Your Feedback

Help us improve your network experience

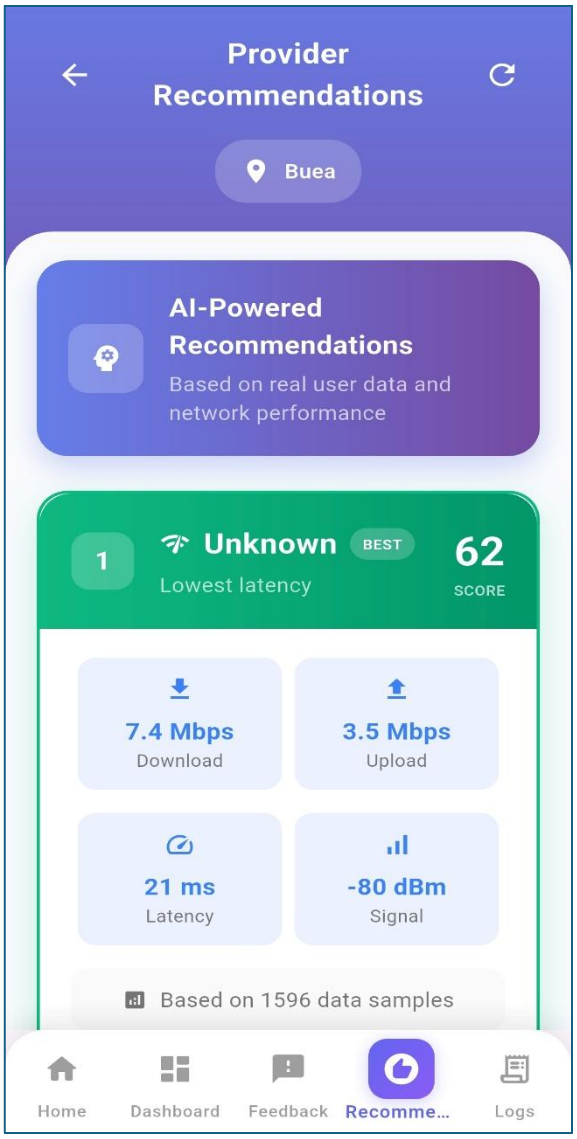
Syncing data...
Last sync: 2h ago
1272 pending

Your network quality is stable
Based on 10 recent measurements

Overall Satisfaction

☆ ☆ ☆ ☆ ☆

Response Time


Provider Recommendations

Buea

AI-Powered Recommendations
Based on real user data and network performance

1 Unknown BEST
Lowest latency
62 SCORE

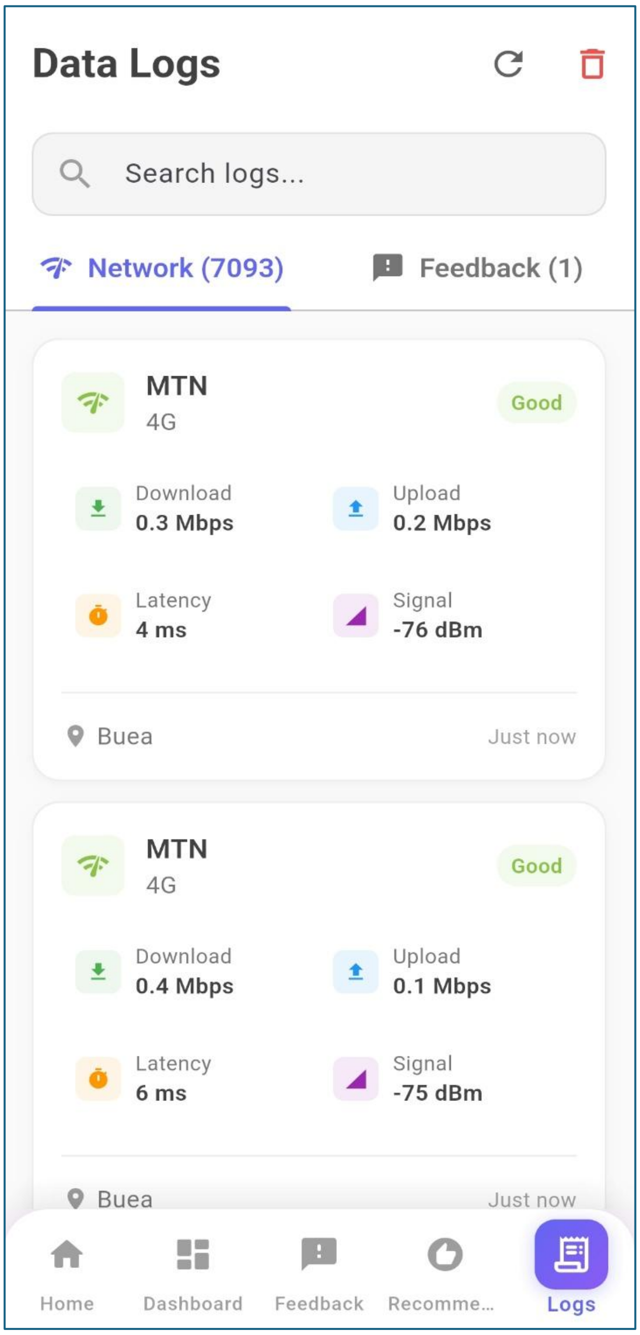
7.4 Mbps Download
3.5 Mbps Upload
21 ms Latency
-80 dBm Signal

Based on 1596 data samples

Here we have the feedback screen and the recommendations screen.

With the feedback screen, users will be able to provide feedback regarding their experience on the network quality. Where they will rate their experience on various criteria and also, be provided a text field to write any other worry they have. This data is stored or logged in a local storage and synced to the database when the user is connected enabling functionality offline.

Regarding the recommendations screen, it will have our AI model which will be able to recommend users based on crowdsourced data on which network provider is best in their locations.


Data Logs

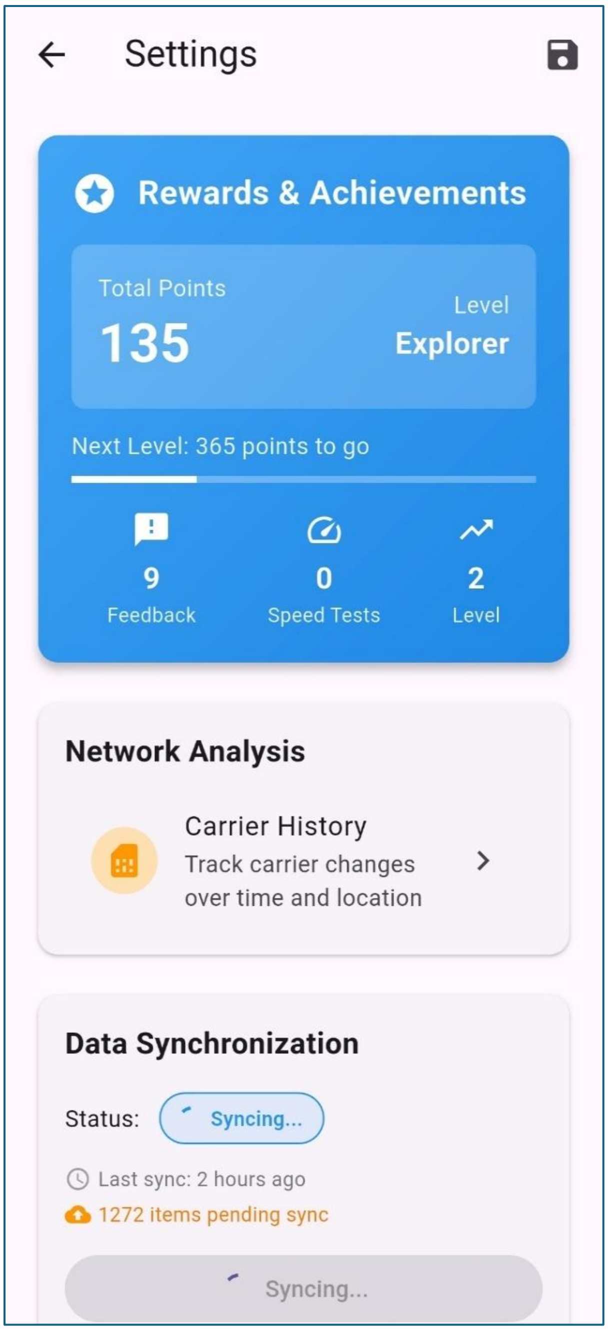
Search logs...

Network (7093) | Feedback (1)

MTN
4G
Good
Download 0.3 Mbps
Upload 0.2 Mbps
Latency 4 ms
Signal -76 dBm
Buea | Just now

MTN
4G
Good
Download 0.4 Mbps
Upload 0.1 Mbps
Latency 6 ms
Signal -75 dBm
Buea | Just now

Home | Dashboard | Feedback | Recomme... | Logs


Settings

Rewards & Achievements
Total Points 135
Level Explorer
Next Level: 365 points to go
9 Feedback | 0 Speed Tests | 2 Level

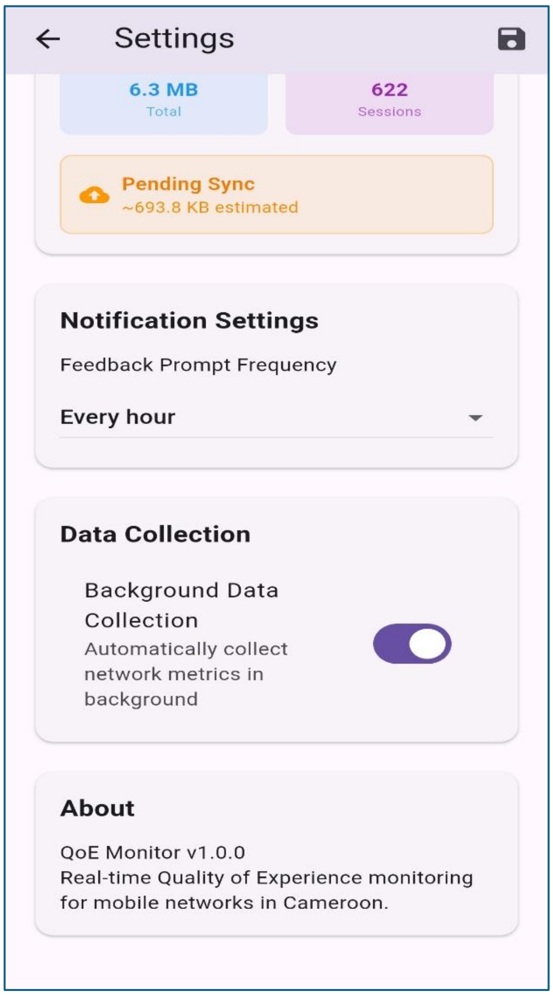
Network Analysis
Carrier History
Track carrier changes over time and location

Data Synchronization
Status: Syncing...
Last sync: 2 hours ago
1272 items pending sync
Syncing...

Here we have two screens the Logs screen and the settings screen.

The logs screen has the network logs collected from the device in the background and logged in the device for syncing to the database when there is good internet connection. It silently collects these metrics in the background.

The settings screen has basic user configurations and permissions on heir preferences. Also, it shows the points a user obtains for providing feedback which will be used for incentives.


Settings

6.3 MB
Total

622
Sessions

Pending Sync
~693.8 KB estimated

Notification Settings

Feedback Prompt Frequency

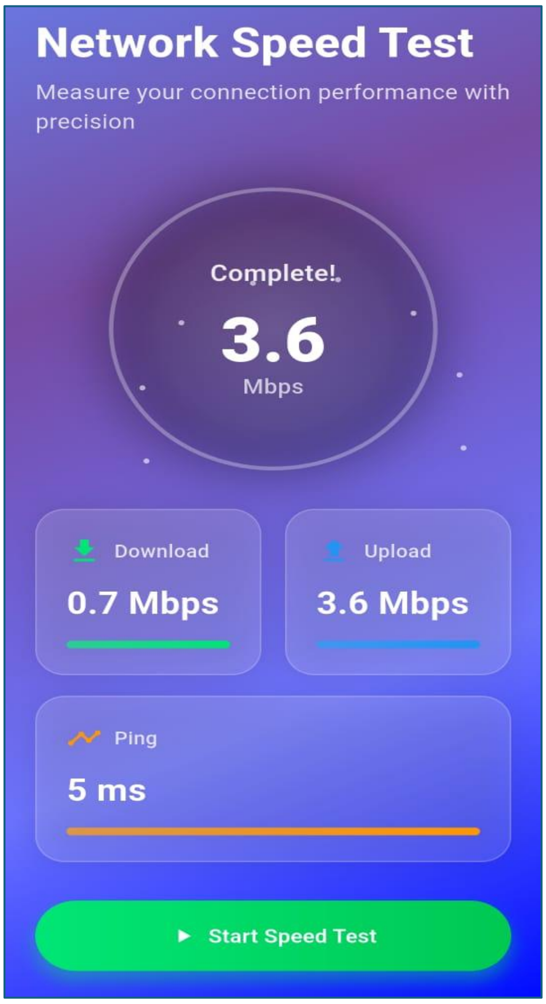
Every hour

Data Collection

Background Data Collection
Automatically collect network metrics in background

About

QoE Monitor v1.0.0
Real-time Quality of Experience monitoring for mobile networks in Cameroon.


Network Speed Test

Measure your connection performance with precision

Complete!
3.6
Mbps

Download
0.7 Mbps

Upload
3.6 Mbps

Ping
5 ms

This is part of the settings screen.

The speed test screen helps our users to be able to simulate their tests for download and upload speeds.

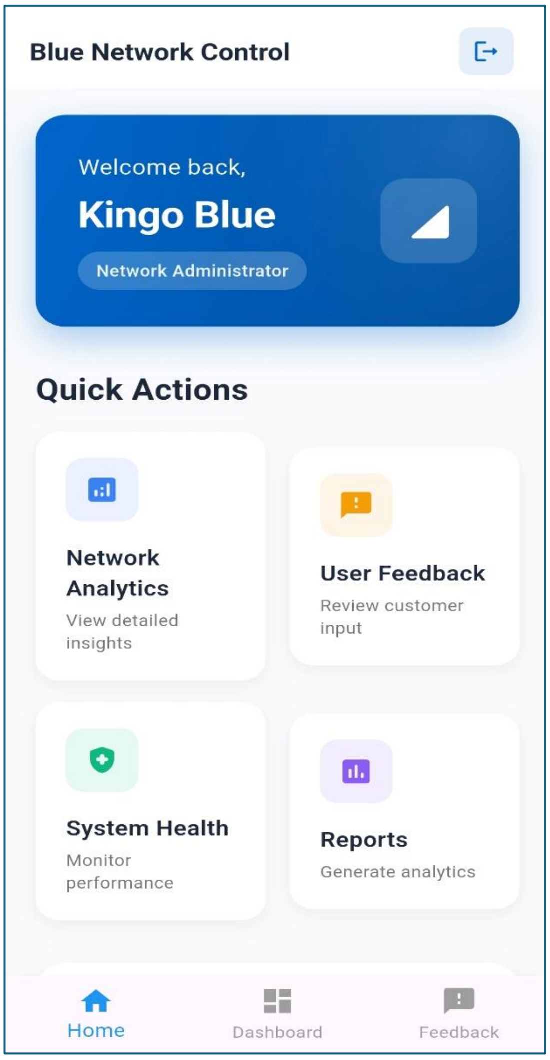
However, the complete parts of each of these screens provided will be demonstrated in our demo for complete visualization.

After effectively collecting this data from the users, how does it get to the intended network service provider? To answer this ;question or solve this problem, we came up with a solution of creating an application that will be used by providers. This application will show them network statistics in different areas as well as the feedback provided by users from different locations so that they can use this data to get better insights and issues encountered by varying users when utilizing their services. The interfaces of this application will be shown below.

b. **The Administrator Dashboard (application)** is designed for telecom operators, like MTN, Orange, Blue and provides:

- ➢ Real-time analytics and visualizations
- ➢ Feedback and trend monitoring
- ➢ Tools for performance reporting

Here, we will provide the screens for a Blue provider being logged in.


Blue Network Control

Welcome back,
Kingo Blue
Network Administrator

Quick Actions

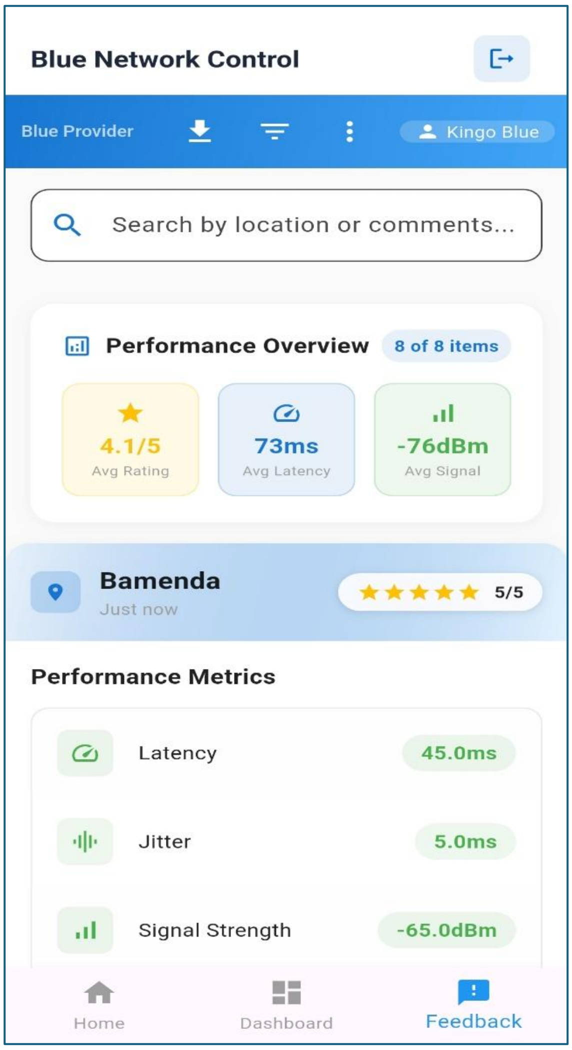
Network Analytics
View detailed insights

User Feedback
Review customer input

System Health
Monitor performance

Reports
Generate analytics

Home · Dashboard · Feedback


Blue Network Control

Blue Provider · Kingo Blue

Search by location or comments...

Performance Overview — 8 of 8 items

4.1/5 Avg Rating · 73ms Avg Latency · -76dBm Avg Signal

Bamenda — Just now — ★★★★★ 5/5

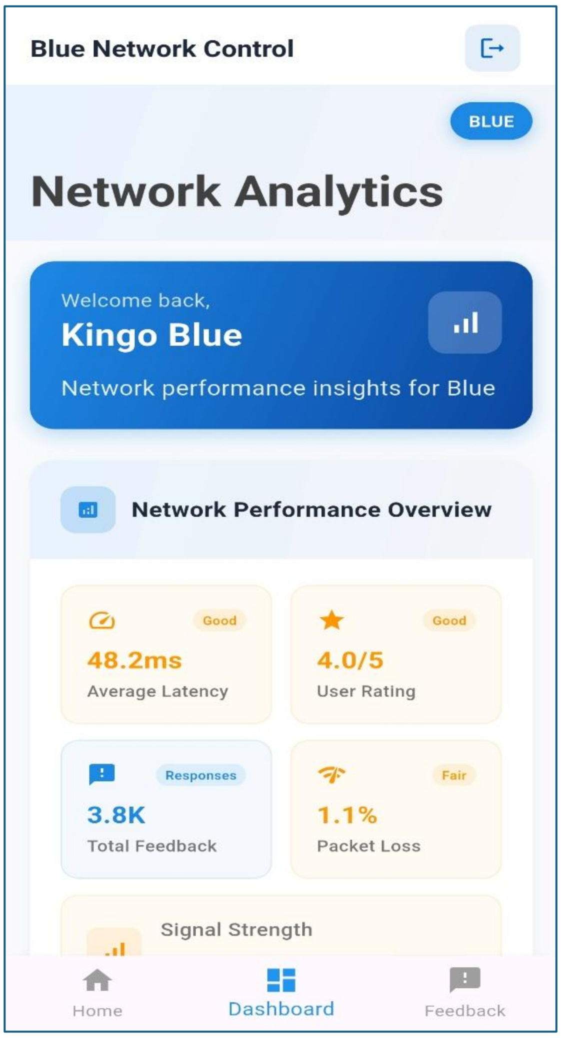
Performance Metrics

Latency	45.0ms
Jitter	5.0ms
Signal Strength	-65.0dBm

Home · Dashboard · Feedback

Here, we have the landing screen or home screen showing the provider's name and the provider as well. This screen has quick actions to navigate other parts of the application.

Also, we have the Feedback screen where the provider receives feedback data from its users and can download it as a csv file for further understanding.


Blue Network Control

BLUE

Network Analytics

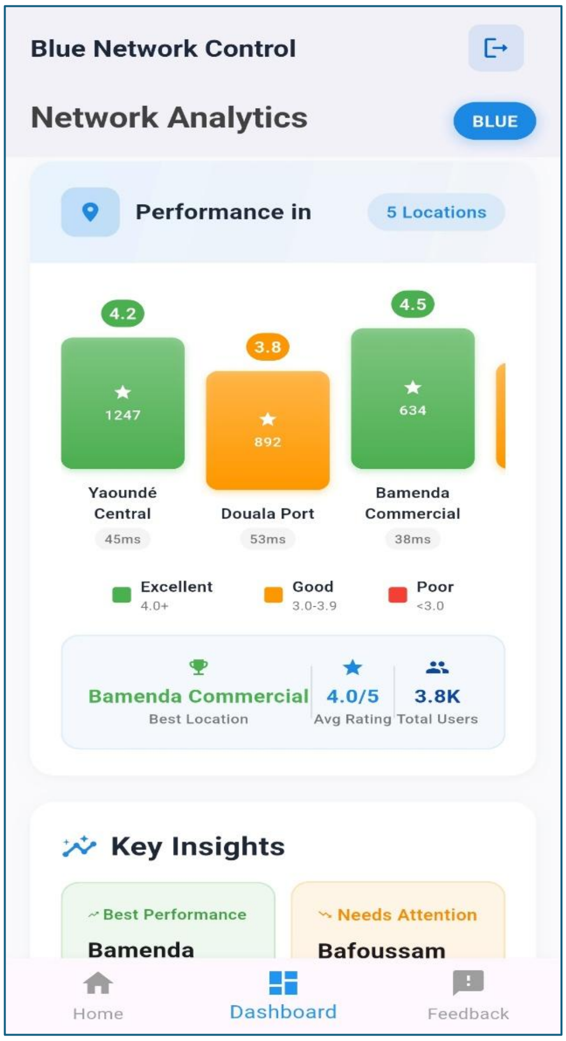
Welcome back,
Kingo Blue
Network performance insights for Blue

Network Performance Overview

Good 48.2ms Average Latency	Good 4.0/5 User Rating
Responses 3.8K Total Feedback	Fair 1.1% Packet Loss

Signal Strength

Home Dashboard Feedback


Blue Network Control

Network Analytics BLUE

Performance in 5 Locations

Yaoundé Central	Douala Port	Bamenda Commercial
4.2	3.8	4.5
1247	892	634
45ms	53ms	38ms

🟩 Excellent 4.0+ 🟧 Good 3.0-3.9 🟥 Poor <3.0

Bamenda Commercial	4.0/5	3.8K
Best Location	Avg Rating	Total Users

Key Insights

Best Performance: Bamenda
Needs Attention: Bafoussam

Home Dashboard Feedback

We have also, the network analytics screen or dashboard screen that analysis all of this data and provides the basic information (analytics) and usage coverage across the towns making use of these services.

All the UI components and screens were built using flutter and its supportive packages. The UI screens are intuitive due to flutter's rich UI capabilities.

This generally comprises of the general UI components and the various screens of our applications. With these solutions, we believe to have laid a solid foundation that will help reduce the gap between the users and service providers thereby improving on Quality of Experience and better service delivery by providers.

Having done this, we went to implement the backend of our application.

4.2. Backend Services and Databases (Data Collection, Analysis, Reporting)

The backend of our application is built with **FastAPI** and integrates with a **PostgreSQL database** hosted on **Supabase**.

Core capabilities include:

- ✓ RESTful APIs for user management, feedback submission, and data retrieval
- ✓ A normalized, multi-tenant database schema supporting scalable, provider-specific data handling
- ✓ Secure authentication and authorization mechanisms using JWT tokens
- ✓ Advanced analytics queries and reporting APIs to support strategic decision-making and machine learning applications

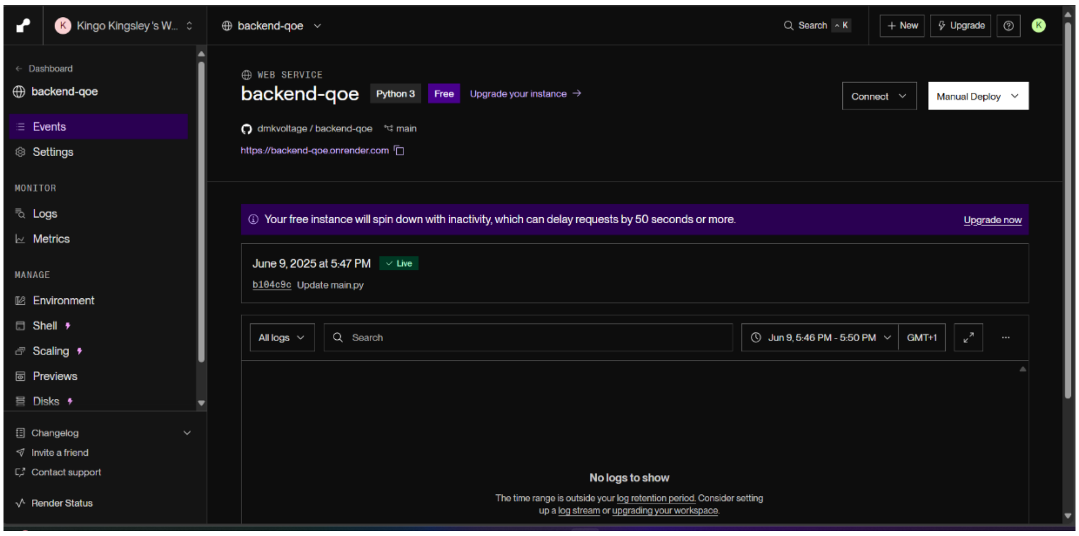
With these, we came out with a table containing the APIs delivered to the frontend application

Backend API Endpoints

Category	Endpoint	HTTP Method	Purpose	Authentication Required
Authentication	/auth/register	POST	Register new user	No
Authentication	/auth/login	POST	User login	No
Health Check	/health	GET	Check server health / Get user info	Yes (for user info)
Feedback	/feedback	POST	Submit user feedback	Yes
Network Logs	/network-logs	GET	Retrieve network logs	Yes
Recommendations	/recommendations	GET	Get location-based recommendations	Yes
Analytics	/analytics/providers	GET	Get provider analytics	Yes

These are the endpoints delivered to the frontend user application, with their various methods and purpose. Our backend was hosted on render and the URL responsible for delivering these services is: https://backend-qoe.onrender.com also known as the base URL.

This is where the backend was hosted: It has been live from the 9th of June.



WEB SERVICE
backend-qoe Python 3 Free Upgrade your instance →
dmkvoltage / backend-qoe main
https://backend-qoe.onrender.com

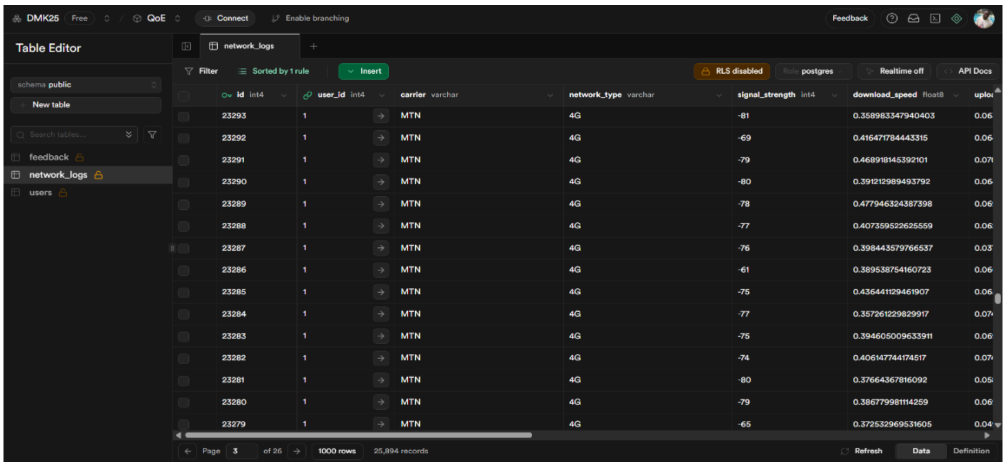
Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more. Upgrade now

June 9, 2025 at 5:47 PM Live
b104c9c Update main.py

All logs | Search | Jun 9, 5:46 PM - 5:50 PM | GMT+1

No logs to show
The time range is outside your log retention period. Consider setting up a log stream or upgrading your workspace.

Here is a snapshot of out database with data collected.



DMK25 Free QoE Connect Enable branching

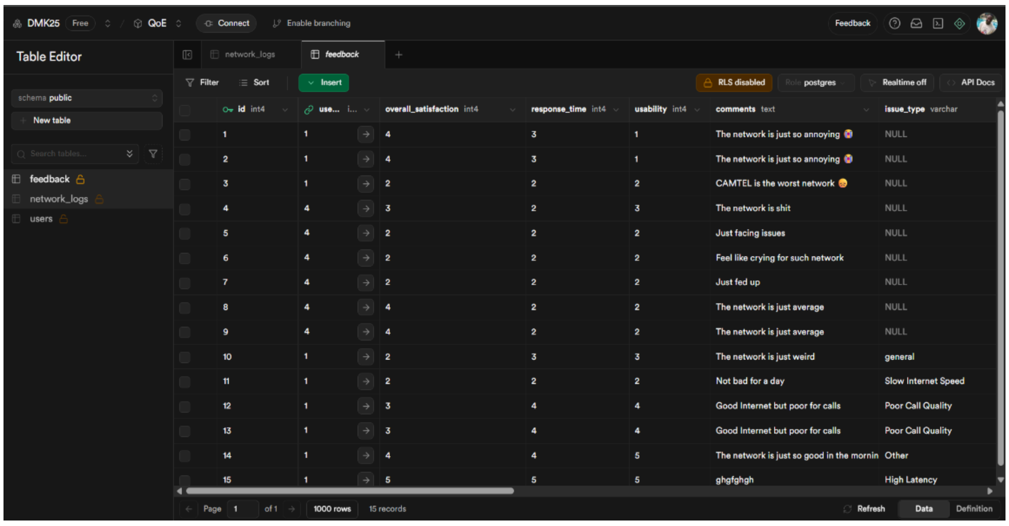
Table Editor — network_logs

Filter | Sorted by 1 rule | Insert | RLS disabled | Role postgres | Realtime off | API Docs

id int4	user_id int4	carrier varchar	network_type varchar	signal_strength int4	download_speed float8	uplo
23293	1	MTN	4G	-81	0.358983347940403	0.06
23292	1	MTN	4G	-69	0.416471784443315	0.06
23291	1	MTN	4G	-79	0.468918145392101	0.07
23290	1	MTN	4G	-80	0.391212989493792	0.06
23289	1	MTN	4G	-78	0.477946324387398	0.06
23288	1	MTN	4G	-77	0.407359522625559	0.06
23287	1	MTN	4G	-76	0.398443579766537	0.03
23286	1	MTN	4G	-61	0.389538754160723	0.06
23285	1	MTN	4G	-75	0.436441129461907	0.06
23284	1	MTN	4G	-77	0.357261229829917	0.07
23283	1	MTN	4G	-75	0.394605009633911	0.06
23282	1	MTN	4G	-74	0.406147744174517	0.07
23281	1	MTN	4G	-80	0.37664367816092	0.05
23280	1	MTN	4G	-79	0.386779981114259	0.06
23279	1	MTN	4G	-65	0.372532969531605	0.04

Page 3 of 26 | 1000 rows | 25,894 records | Refresh | Data | Definition

This is the network_logs table with logs from different devices.



DMK25 Free QoE Connect Enable branching

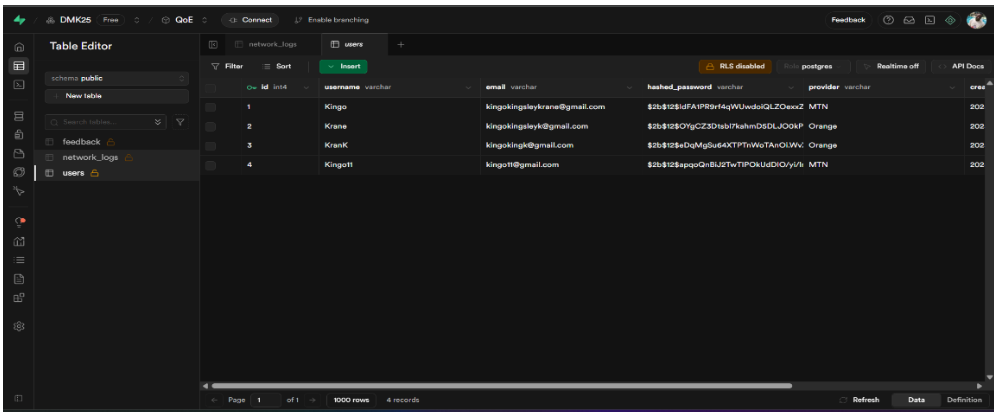
Table Editor — feedback

Filter | Sort | Insert | RLS disabled | Role postgres | Realtime off | API Docs

id int4	use... i...	overall_satisfaction int4	response_time int4	usability int4	comments text	issue_type varchar
1	1	4	3	1	The network is just so annoying 😭	NULL
2	1	4	3	1	The network is just so annoying 😭	NULL
3	1	2	2	2	CAMTEL is the worst network 😞	NULL
4	4	3	2	3	The network is shit	NULL
5	4	2	2	2	Just facing issues	NULL
6	4	2	2	2	Feel like crying for such network	NULL
7	4	2	2	2	Just fed up	NULL
8	4	4	2	2	The network is just average	NULL
9	4	4	2	2	The network is just average	NULL
10	1	2	3	3	The network is just weird	general
11	1	2	2	2	Not bad for a day	Slow Internet Speed
12	1	3	4	4	Good Internet but poor for calls	Poor Call Quality
13	1	3	4	4	Good Internet but poor for calls	Poor Call Quality
14	1	4	4	5	The network is just so good in the mornin	Other
15	1	5	5	5	ghgfghgh	High Latency

Page 1 of 1 | 1000 rows | 15 records | Refresh | Data | Definition

Here is the feedback table with feedback collected and stored in the database as well.



id int4	username varchar	email varchar	hashed_password varchar	provider varchar	crea
1	Kingo	kingokingsleykrane@gmail.com	$2b$12$ldFA1PR9rf4qWUwdoiQLZOexxZ	MTN	202
2	Krane	kingokingsleyk@gmail.com	$2b$12$OYgCZ3Dtsbl7kahmD5DLJO0kP	Orange	202
3	KranK	kingokingk@gmail.com	$2b$12$eDqMgSu64XTPTnWoTAnOi.Wv.	Orange	202
4	Kingo11	kingo11@gmail.com	$2b$12$apqoQnBiJ2TwTIPOkUdDIO/yi/I	MTN	202

This is the table for users and their rules. All of these details our application results from both the users and the network operators, covering both from frontend to backend and also the database.

5. Evaluation of the solution

Technical Implementation Excellence

Our WaveWatch platform has successfully achieved its core technical objectives, demonstrating exceptional integration across all system components. The three-layer architecture we implemented provides a solid foundation that seamlessly connects our Flutter-based mobile applications with our FastAPI backend services and PostgreSQL database hosted on Supabase.

The cross-platform compatibility achieved through Flutter has been particularly successful, allowing us to maintain consistent user experiences across both Android and iOS devices while significantly reducing our development overhead. Our backend implementation using FastAPI delivers robust asynchronous request handling with automatic API documentation, effectively supporting the real-time data processing requirements that are central to our platform's mission.

Our data persistence strategy using PostgreSQL on Supabase ensures complete data integrity through normalized schema design while maintaining the multi-tenant architecture essential for supporting multiple telecommunications providers. The implementation successfully balances performance optimization with data consistency, creating a reliable foundation for our network quality monitoring ecosystem.

Data Collection and Processing Success

Our dual-mode data collection strategy has proven highly effective in capturing comprehensive network performance metrics. The real-time collection system successfully gathers critical data including download/upload speeds, signal strength, network type, location coordinates, jitter measurements, and latency values during active user interactions. This immediate data capture provides users with current network status while building our comprehensive database of performance metrics.

The background monitoring capability represents a significant technical achievement, utilizing Android JobScheduler and iOS Background App Refresh to enable continuous data collection with minimal battery impact. This silent collection mechanism ensures we maintain comprehensive network monitoring even when users aren't actively engaging with our application, creating a complete picture of network performance patterns across different times and locations.

Our offline capability implementation addresses the real-world connectivity challenges faced by users in Cameroon. Local storage with intelligent sync functionality ensures no data is lost during connectivity issues, maintaining the integrity of our monitoring mission even in areas with inconsistent network coverage.

User Experience and Interface Design Impact

Our subscriber application successfully transforms complex network data into accessible, actionable information for everyday users. The card-based UI design with large icons and intuitive color-coded indicators (green/yellow/red) effectively simplifies technical network metrics, making our platform accessible to users with varying technical backgrounds.

The gamification and rewards system we implemented has proven effective in encouraging sustained user participation in network monitoring activities. This engagement strategy ensures continuous data collection while providing tangible value to our user community through recognition and incentives for their contributions to network quality improvement.

6. Partial conclusion

The implementation of the WaveWatch platform reflects a well-coordinated integration of user-centered design, scalable system architecture, and robust backend engineering. By leveraging modern tools and frameworks such as Flutter, FastAPI, PostgreSQL, and Supabase, the project successfully delivered a full-stack solution capable of capturing, analyzing, and presenting mobile network Quality of Experience (QoE) data in real-time.

Each component—ranging from intuitive mobile interfaces to provider-specific dashboards, secure authentication mechanisms, and multi-tenant database structures was carefully designed and implemented to address both the technical challenges and practical needs of end-users and telecom operators in Cameroon. The platform's modular design ensures future scalability, allowing for regional expansion, integration with machine learning services, and long-term system evolution.

Overall, the WaveWatch system stands as a practical, adaptable, and impactful implementation that bridges the gap between mobile users and service providers through real-time feedback loops, actionable analytics, and a commitment to security, accessibility, and data integrity.

CHAPTER 5: CONCLUSION AND FURTHER WORKS

1. Summary of findings

This project successfully designed and implemented a mobile application for collecting Quality of Experience (QoE) data from mobile network subscribers in Cameroon. The app integrates subjective user feedback with objective network metrics (e.g., signal strength, latency, bandwidth) to provide telecom operators (MTN, Orange, Camtel) with actionable insights. Key achievements include:

- **Cross-platform development** using **Flutter**, ensuring compatibility with Android and iOS.
- **AI-driven feedback prompts** (TensorFlow Lite) triggered by network anomalies, improving response rates.
- **Background data collection** with minimal battery drain (<2% daily) using WorkManager (Android) and Background Fetch (iOS).
- **Multi-tenant PostgreSQL database** with row-level security to segregate operator data.
- **Gamification** (reward points, airtime incentives) to boost user engagement.

2. Contribution to Engineering and Technology

This project advances existing solutions (e.g., OpenSignal, nPerf) by:

- **Hybrid data collection**: Combines passive metrics + active feedback with AI contextual prompts.
- **Offline-first design**: Local SQLite storage syncs to cloud (Supabase) when connectivity resumes.
- **Privacy-preserving analytics**: GDPR-compliant anonymization and differential privacy techniques.

- **Real-time dashboards**: FastAPI backend delivers provider-specific insights via geospatial heatmaps.

For developing regions, this offers a scalable model to monitor network performance where infrastructure limitations hinder traditional methods.

3. Recommendations

3.1. For Telecom Operators (MTN, Orange, Camtel)

To optimize network performance, telecom operators should leverage the app's geospatial heatmaps (powered by PostGIS) to pinpoint areas with persistent issues, such as low signal strength (< -100 dBm) or high latency (> 500ms). By prioritizing infrastructure upgrades in high-density, low-performance zones—such as urban slums or rural hubs—operators can maximize the impact of their investments. Additionally, cross-referencing user feedback trends with network logs helps distinguish between coverage gaps (e.g., no signal) and capacity issues (e.g., slow speeds during peak hours), enabling targeted solutions.

For proactive maintenance, operators should deploy AI-driven alerts (using TensorFlow Lite) to detect anomalies like sudden packet loss spikes before users report them. Integrating these alerts with existing Network Operations Centers (NOCs) can automate ticket generation for recurring complaints, reducing resolution times and improving service reliability.

3.2. For Users

To boost engagement, the app should introduce **gamified incentives**, such as tiered rewards (e.g., 10 points for speed tests, 50 points for detailed feedback) redeemable for airtime or data bundles. Implementing **leaderboards**—like "Top Contributor in Douala"—can foster healthy competition among communities, encouraging more users to participate.

For broader adoption, telecom operators could **pre-install the app** on SIM toolkit menus or bundle it with mobile money services. Awareness campaigns highlighting the app's benefits—such as improved network quality and rewards—can further drive uptake, creating a virtuous cycle of feedback and service enhancements.

4. Difficulties encountered

Battery Optimization

Continuous background polling (every 5 mins) drained battery by 8–10% in early prototypes. Adaptive intervals (WorkManager) was implemented; 15-minute polls during inactivity, 5-minute during active usage. We used batching to compress and upload metrics in bursts (reduced wake-locks by 60%). On doing that, daily drain stabilized at <2% (Android) and <3% (iOS). Data accuracy: GPS drift in rural areas necessitated signal-based location fallbacks.

5. Further works

5G monitoring will be extended to capture New Radio (NR) metrics like mmWave signal quality and beamforming stability, while supporting network slicing analytics to ensure QoS for critical services such as telehealth. However, this requires Android 12+ APIs for 5G NR data collection, posing compatibility challenges for older devices.

Predictive analytics will leverage LSTM models trained on historical network logs to forecast outages (e.g., predicting congestion during peak events like festivals) and employ time-series clustering to identify regions with recurrent failure patterns, generating actionable alerts such as "Expected outage in Bamenda between 4–6 PM (87% confidence)."

To improve accessibility, **voice-based feedback** will integrate Google's Speech-to-Text for non-literate users, with NLP auto-categorizing transcripts (e.g., mapping "no signal" to coverage issues), though offline-capable ASR models remain a hurdle for rural areas with limited connectivity.

For **multi-country deployment**, the system will localize incentives (e.g., mobile money in Kenya, airtime in Nigeria) and adhere to regional data laws (Nigeria's NDPR, Kenya's Data Protection Act), while cloud-agnostic orchestration via Docker/Kubernetes will ensure scalability without vendor lock-in. These enhancements aim to bridge technical gaps while adapting to diverse user needs and regulatory landscapes.

REFERENCES

1. Google Form for subscribers,
https://docs.google.com/forms/d/e/1FAIpQLSd3j2kOmayUbwuEoCFvejdOmRhMLfK_cM79gquMvpPAUOm4w/viewform?usp=header

2. Google Form for technical teams (telecommunication engineer, developer etc.),
https://docs.google.com/forms/d/e/1FAIpQLSctqzLQS0AOSuQ0AuMn_3fpD-jEw8DAerzpMC9EOnEexypn0w/viewform?usp=header

3. GeeksForGeeks. (2025). Fnctional VS Non-Functional Requirements - GeeksForGeeks. Retrieved from GeeksForGeeks,
https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

4. Firebase by Google. (2024). Firebase Authentication Guide. Retrieved from Firebase Website,
https://firebase.google.com/docs/auth

5. Link to Figma Design,
https://www.figma.com/proto/VMCV47Bg0o6rkkvhao2tAs/InternetProgramming?node-id=0-1&t=iS7tVvIYMtGLqz8U-1