UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

REPULBLIC OF CAMEROON

PEACE - WORK - FATHERLAND

MINISTRY OF HIGHER EDUCATION

Course Code: CEF440

Course Title: Internet Programming and Mobile Programming

Course Instructor: Dr. Nkemeni Valery

ANALSYS AND SYSTEM REQUIREMENTS AND SPECIFICATIONS OF A MOBILE NETWORK MONITORING APPLICATION

PRESENTED BY:

GROUP 23

2025 EDITION

GROUP MEMBERS

NAMES	MATRICULE
MBIGHA KINENUI STEPH	FE22A244
TIAYA FOTSEU JOSUE	FE22A315
NGANYU BRANDON YUNIWO	FE22A258
NGULEFAC FOLEFAC FOBELLA	FE19A081
NTUV TCHAPTSA JAMISON LII	FE22A284

Table of Contents

I. Introduction

The Requirement Analysis phase is a critical step in the software development lifecycle (SDLC), serving as a bridge between raw user input and the formal design and implementation of a working system. Building on the foundation laid in Task 2: Requirement Gathering, this task involves a comprehensive and structured analysis of the collected user requirements for the proposed mobile application designed to monitor and report mobile network performance in Cameroon.

1. Context and Relevance

Cameroon had approximately 25.5 million cellular mobile connections at the start of 2025, reflecting the country's heavy reliance on mobile data for communication, commerce, and education (DataReportal – Global Digital Insights). Yet signal strength and bandwidth vary significantly across the major operators—MTN, Orange, and Camtel—leading to frequent user frustration and productivity loss (Agence Ecofin). In response to regulatory pressure, Cameroonian authorities have even ordered MTN, Orange, and Camtel to implement real-time network monitoring to anticipate and mitigate service disruptions (Business in Cameroon). Despite these mandates, existing solutions remain generic and fail to address local QoE (Quality of Experience) nuances, especially in semi-urban and rural areas where coverage gaps persist. A localized, user-centered mobile application can empower end users to monitor real-time performance, report issues, and contribute to network planning, thereby improving transparency and service quality.

2. Description of Existing System

Currently available network diagnostic tools (e.g., Network Analyzer) focus on low-level connectivity tests—ping, traceroute, Wi-Fi analysis—but offer limited support for cellular network monitoring and lack features for user-driven reporting (Android Apps on Google Play). These apps also rarely include privacy controls or data anonymization options, raising concerns about personal data exposure when diagnosing network problems. In practice, users must switch between multiple disparate apps to run speed tests, log signal readings, and manually send feedback to operators, resulting in poor adoption and fragmented data.

Consequently, neither users nor network providers gain a holistic view of field-level performance trends or user sentiment.

3. Problem Statement

Despite growing mobile connectivity, Cameroonian users face unpredictable network performance—fluctuating signal strength, variable bandwidth, and intermittent outages—that directly impacts their daily activities and economic opportunities (GSMA). The absence of an integrated, real-time feedback mechanism prevents both users and operators from identifying and resolving localized service degradations promptly (Business in Cameroon). Without such a system, end users remain disempowered, and network providers lack actionable, crowd-sourced data necessary for targeted infrastructure improvements.

4. Objectives

This SRS aims to:

- ❖ **Elicit and specify functional requirements** (e.g., real-time signal monitoring, issue reporting).
- ❖ **Define non-functional requirements** (e.g., performance, security, privacy, and reliability).
- ❖ **Produce a comprehensive specification** to guide design, development, and validation of the mobile network monitoring application.

5. Methodology

Requirements were synthesized from stakeholder input (end users, network providers, and administrators), structured surveys of 65 respondents across Buea, Yaoundé, and Douala, open-ended interviews, and competitive analysis of existing diagnostic tools. Detailed methodology (instruments, sampling, and analysis procedures) is provided below:

5.1 Survey Overview

To address this need, we conducted a detailed survey involving over 65 mobile network users from different regions including Buea, Yaoundé, Douala, and others. The survey targeted users aged 18 and above, with the majority (over 60%) falling within the 18–34 age range — the group most affected by mobile connectivity issues due to their high internet usage.

Key highlights from the gathered data include:

- Over 70% of respondents experience frequent network issues, particularly slow internet speed and dropped calls.
- MTN and Orange are the most used networks, though both suffer from regional inconsistencies.
- Users reported a high level of interest in a mobile app that could monitor network performance and offer a way to report problems.

5.2 Scope of Work

This task includes:

- A systematic review of all collected requirements for gaps, redundancies, or inconsistencies.
- The classification of requirements into functional and non-functional categories.
- A prioritization strategy based on feasibility and user value.
- The development of a formal Software Requirements Specification (SRS) document.
- A reflection on stakeholder validation, using the original survey data to ensure alignment with expectations.

By the end of this task, the project will have a validated, structured, and technically reviewed list of requirements, ready for design and prototyping.

5.3 Review and Analysis of Requirements

This section focuses on reviewing the requirements gathered from mobile network subscribers during the data collection phase and assessing their **completeness**, **clarity**, **technical feasibility**, and **dependency relationships**. The goal is to ensure that the gathered requirements are well-understood, practically implementable, and structured for smooth system design.

5.3.1 Completeness

The survey included structured questions covering user demographics, mobile device types, network usage patterns, problems encountered, satisfaction levels, interest in the proposed app, and feature preferences.

Complete Aspects:

- User expectations and pain points (e.g., slow internet, dropped calls).
- Key functional desires like signal strength monitoring and issue reporting.
- Privacy and resource (battery/data) concerns.
- Willingness to use the app and conditions for adoption.

Areas with Missing Information:

- Not much feedback was gathered from actual telecom engineers or network providers, just casual information as most of them were reluctant to speak to us
- No responses on how users would like to visualize performance data (e.g., charts, maps).
- Some technical expectations (like frequency of background data collection) were not addressed explicitly.

Conclusion: The requirement set is largely complete from the **end-user perspective**, but further input from **technical stakeholders** (e.g., developers, some telecoms) will be good for full technical implementation planning.

5.3.2 Clarity

Most of the responses were well-structured through multiple-choice formats and consistent language. Open-ended feedback, although limited, also revealed clear sentiments such as privacy concerns and the need for simplicity.

Clear Aspects:

- Users clearly want the app to be **lightweight**, **privacy-respecting**, and **useful**.
- Users prefer **event-triggered feedback prompts** over frequent, scheduled ones.

- There’s strong support for passive monitoring but only if transparent and optional.

Ambiguities Found:

- Terms like “manual reporting” were mentioned by users but not clearly defined (e.g., is it a text box, a complaint form, or a button?).
- Some responses mentioned “see my signal” without specifying the desired detail (bars, dBm, graphs?).

Conclusion: Minor ambiguities exist, mostly around **UI expectations** and **level of technical depth** for features. These should be resolved through prototyping or user story workshops.

5.3.3 Technical Feasibility

The features requested by users were reviewed against what is currently feasible in **React Native** and **native Android development**:

FEATURE	FEASIBILITY	NOTES
Signal Strength Logging	Android native only	Requires Java module integration
Network Type (3G/4G)	Easily available	Use NetInfo
Feedback Prompts	Standard UI	Can be daily or event-driven
Speed Test	With APIs or libraries	Use react-native-speed-test or custom download test
Privacy Features	Configurable	Opt-in/out for location and data collection
Call Quality Monitoring	Complex	Not natively accessible without deep system hooks

Conclusion: Most requested features are feasible using **React Native + native Android modules**. Some advanced features (like call quality scores) may not be possible without telecom partnerships or low-level access.

5.3.4 Dependency Relationships

Understanding which features depend on others helps structure development:

Requirements	Depends On
Signal logging with location	GPS access, network permission
Event-triggered feedback	Real-time error detection module
Data upload to cloud	Stable internet + background sync logic
Feedback visualization	Data storage + UI design logic

Conclusion: Certain requirements will need to be developed together. For instance, background data collection must be in place before meaningful feedback can be linked with network issues.

6. Technology Stack/Tools

For the project, I'll structure the frontend and backend details in a table, we'll choose React Native for its JavaScript familiarity and ecosystem support, including Expo, Redux, React Navigation, and Recharts. The backend will use Node.js + Express, MongoDB, JWT, AWS S3, and Socket.io, backed by citations from relevant resources. React Native's advantages will be supported with context-based citations.

6.1 Frontend Technology Stack

Technology	Purpose in This Project
React Native	Cross-platform UI framework delivering near-native performance and broad community support (Reddit, DEV Community).
Expo	Managed React Native workflow for streamlined builds, OTA updates, and easy testing.
React Navigation	Routing and navigation between app screens with native-feeling transitions.
Redux	Global state management for user session, settings, and cached logs.

6.2 Backend Technology Stack

Component	**Technology Stack**	**Purpose in This Project**
Server Runtime	Node.js + Express	Lightweight, event-driven server ideal for real-time monitoring APIs and microservices (Imaginary Cloud, DEV Community).
Database	MongoDB	Schema-flexible NoSQL DB for storing time-series signal logs and user reports (DEV Community)
BaaS (optional)	Firebase	Out-of-the-box authentication, analytics, and realtime database to speed up MVP development
Authentication	JWT (JSON Web Tokens)	Stateless, secure user authentication for API access (standard best practice).
Hosting / Infra	AWS (EC2/Lambda) or Google Cloud	Scalable compute for APIs and serverless functions handling background tasks (e.g., data aggregation).

These choices leverage mature ecosystems, support offline-first patterns, and scale as user demand grows.

7. Assumptions and Dependencies Overview

We assume users possess 3G/4G-capable smartphones with GPS and sufficient storage, and that network operators expose basic metrics via device APIs. Dependencies include third-party mapping and analytics services, JWT-based authentication infrastructure, and compliance with Cameroon's data-protection guidelines. Where required, data anonymization and encryption will be enforced.

II. Requirements Analysis Specifications

Below is a detailed breakdown of the system's requirements, organized into an overview of its architecture, the major components, user roles, and both functional and non-functional requirements.

1. System Overview

The proposed solution comprises two primary subsystems: (1) a cross-platform mobile application (built in React Native) that runs on 3G/4G Android and iOS devices and (2) a backend service stack including a Node.js/Express API layer, a MongoDB time-series database for logging network metrics, and optional Firebase services for authentication and push notifications (Perforce, SolarWinds Documentation). The mobile app captures real-time signal strength, allows manual issue reporting, and caches historical data locally for offline use; the backend aggregates, analyzes, and serves this data to both end users and network providers via RESTful endpoints (SmartGrid Resource Center). Third-party dependencies include a mapping API for geo-visualization and JWT-based auth infrastructure to secure all client–server communications.

2. System Requirements

The system's high-level components include:

- **Mobile Client:**
  - Captures network metrics (signal strength, latency) using Android/iOS native APIs.
  - Implements local caching via SQLite for offline functionality.
  - Renders UI elements (charts, maps) using React Native libraries.
- **API Server:**
  - Manages user authentication, data synchronization, and role-based access control (RBAC).
  - Integrates with Firebase for push notifications and real-time alerts.
- **Database:**
  - Stores time-stamped network logs, user preferences, and issue reports in MongoDB collections.

- **External Services:**
    - Mapbox API for visualizing signal strength heatmaps.
    - Speed-test libraries (e.g., react-native-speed-test) for on-demand network diagnostics.

3. User Roles / Actors

Actor	Responsibilities	Interaction Examples
End User	Monitors network performance, submits issue reports, configures privacy settings.	FR-1 (view signal strength), FR-3 (submit report).
System Administrator	Manages backend infrastructure, monitors API health, assigns user roles.	Configures data retention policies, audits access logs.
Network Provider	Analyzes aggregated reports to identify regional network gaps.	Accesses anonymized heatmaps via admin dashboard (NFR-10).

4. Functional Requirements

ID	Requirement	Use Case Mapping
FR-1	Display real-time signal strength (dBm), network type (3G/4G), and carrier.	UC-01: Monitor Network Performance
FR-2	Log GPS location, timestamp, and signal metrics during background monitoring.	UC-02: Passive Data Collection
FR-3	Allow users to submit structured reports (e.g., “dropped call,” “slow speed”).	UC-03: Manual Issue Reporting
FR-4	Sync reports to the backend within 5 seconds of submission (online mode).	UC-04: Data Synchronization

FR-5	Render historical data as interactive time-series charts.	UC-05: View Usage History
FR-6	Authenticate users via JWT tokens with Firebase Auth integration.	UC-06: User Authentication
FR-7	Provide opt-in/out toggles for GPS and anonymized data sharing.	UC-07: Privacy Configuration
FR-8	Cache up to 7 days of data locally for offline access.	UC-08: Offline Functionality
FR-9	Enable users to select preferred network operator (MTN, Orange, Camtel).	UC-09: Network Selection
FR-10	Auto-sync cached data when connectivity is restored.	UC-10: Background Sync

5. Non-Functional Requirements

- **Performance**:
  - The signal-monitoring API shall respond within 2 s on a 3G connection.
  - The history view shall load and render up to 100 data points in ≤3 s.
- **Security & Privacy**:
  - All API calls must use HTTPS with TLS 1.2+.
  - User credentials and JWT tokens shall be stored encrypted in device secure storage (CLG Global).
  - Personal data (e.g., GPS coordinates) must be anonymized if the user opts out of location sharing.
- **Reliability & Availability**:
  - The backend must achieve ≥99.5 % uptime.
  - The mobile app shall cache up to 7 days of logs for offline use.

- **Maintainability & Extensibility**:
    - Code shall follow industry standards (ESLint, Prettier, PEP 8 where applicable) and be modularized into clear feature modules.
    - The system shall expose OpenAPI-compliant documentation for all endpoints.
- **Compliance & Legal**:
    - Must adhere to Cameroon’s 2025 Data Protection Act regarding sensitive personal data (Home - Tech Hive Advisory, DataGuidance).
    - Shall implement data-retention policies as per national regulations (e.g., purge logs older than 90 days unless consented).

6. Prioritization of Requirements

In a real-world project with limited time and resources, it's important to identify which requirements should be implemented **first**, which can be **added later**, and which are **optional or deferred**. This section uses the **MoSCoW prioritization method**:

MoSCoW stands for:

- **M**ust Have
- **S**hould Have
- **C**ould Have
- **W**on’t Have (for now)

6.1 Functional Requirements Prioritization

Priority	Req. ID	Requirement	Justification
Must Have	FR-1	Display real-time signal strength (dBm), network type (3G/4G), and carrier.	Core functionality for users to monitor network performance; foundational to the app's purpose.
	FR-2	Log GPS location, timestamp, and signal metrics during background monitoring.	Enables geospatial analysis and historical tracking; essential for actionable insights.
	FR-3	Allow users to submit structured reports (e.g., "dropped call," "slow speed").	Empowers user-driven feedback; critical for crowd-sourced network evaluation.
	FR-6	Authenticate users via JWT tokens with Firebase Auth integration.	Mandatory for securing user data and ensuring privacy compliance.
	FR-8	Cache up to 7 days of data locally for offline access.	Critical for regions with intermittent connectivity; ensures data continuity.
	FR-10	Auto-sync cached data when connectivity is restored.	Guarantees data integrity and reliability, even in low-network areas.
Should Have	FR-4	Sync reports to the backend within 5 seconds of submission (online mode).	Enhances user experience but can tolerate minor delays during network instability.

	FR-5	Render historical data as interactive time-series charts.	Improves data transparency but secondary to core monitoring features.
Could Have	FR-7	Provide opt-in/out toggles for GPS and anonymized data sharing.	Addresses privacy preferences but can be phased post-MVP after core security is established.
	FR-9	Enable users to select preferred network operator (MTN, Orange, Camtel).	Adds customization value but not critical for initial deployment.
Won't Have	–	Predictive analytics for network anomalies (deferred to Phase 2).	Advanced feature requiring additional data and stakeholder feedback; out of MVP scope.

6.2 Non-Functional Requirements Prioritization

Priority	**Category**	**Requirement**	**Justification**
Must Have	**Performance**	API response time ≤2s on 3G connections.	Ensures usability in low-bandwidth regions; core to user retention.
	Security	HTTPS/TLS 1.2+ encryption; credentials stored in secure storage.	Non-negotiable for protecting sensitive user data and compliance.
	Compliance	Adhere to Cameroon's 2025 Data Protection	Legal requirement; failure to comply risks penalties and loss of trust.

		Act (e.g., user consent, anonymization).	
Should Have	**Reliability**	Backend uptime ≥99.5%; app remains functional in offline mode.	High reliability expected for monitoring tools; offline mode critical for low-connectivity users.
	Maintainability	Code follows ESLint/Prettier standards; OpenAPI documentation for endpoints.	Facilitates scalability and collaboration but can be refined post-MVP.
Could Have	**Performance**	Load 100 data points in ≤3 s for history view.	Enhances UX but secondary to real-time monitoring performance.
	Localization	Support French language alongside English.	Important for inclusivity but can follow initial English-only release.
Won't Have	**Advanced Features**	Push notifications for network anomalies.	Requires backend analytics infrastructure; deferred until MVP validation.

Conclusion

This prioritization ensures the MVP delivers **core functionality** (signal monitoring, issue reporting, offline support) while adhering to legal and security standards. Subsequent phases will address enhancements like advanced analytics and multilingual support, guided by user feedback and resource availability.

7. Glossary of Terms and Acronyms

Term/Acronym	Definition
QoE	Quality of Experience – user-perceived quality of service, often derived from subjective feedback.
QoS	Quality of Service – objective network-level performance metrics (e.g., latency, bandwidth, jitter).
Signal Strength	The power level received by the device from the network tower, measured in dBm or shown as signal bars.
ISP	Internet Service Provider – the company providing internet services, which can include mobile data.
FR	Functional Requirement – specific system behaviors or services that must be implemented.
NFR	Non-Functional Requirement – constraints on system operation (e.g., performance, usability, security).
GPS	Global Positioning System – satellite-based location service used to tag data geographically.
MVP	Minimum Viable Product – a product version with just enough features to be usable and gather feedback.
Manual Reporting	A feature allowing users to submit network issues directly via the app interface.
Event-Triggered Feedback	User feedback initiated by predefined triggers (e.g., call drop or poor speed).
Background Sync	Automated uploading of data collected locally once internet access is available.

NetInfo	A React Native library/API for monitoring device network status (e.g., connection type, reachability).

A high-level system architecture clarifies the major subsystems (mobile client, API server, database, and external services) and their interactions, ensuring scalability, maintainability, and security. The UML use-case diagram captures actor–system interactions for the six primary scenarios. Component-level views (component and block diagrams) then decompose the solution into modular units—UI, data, auth, reporting—highlighting interfaces and dependencies. Finally, the implementation section outlines how these components translate into modules (Signal Monitor, Issue Reporter, History Viewer, Settings), with accompanying wireframes to guide UI development and a data schema overview for persistence.

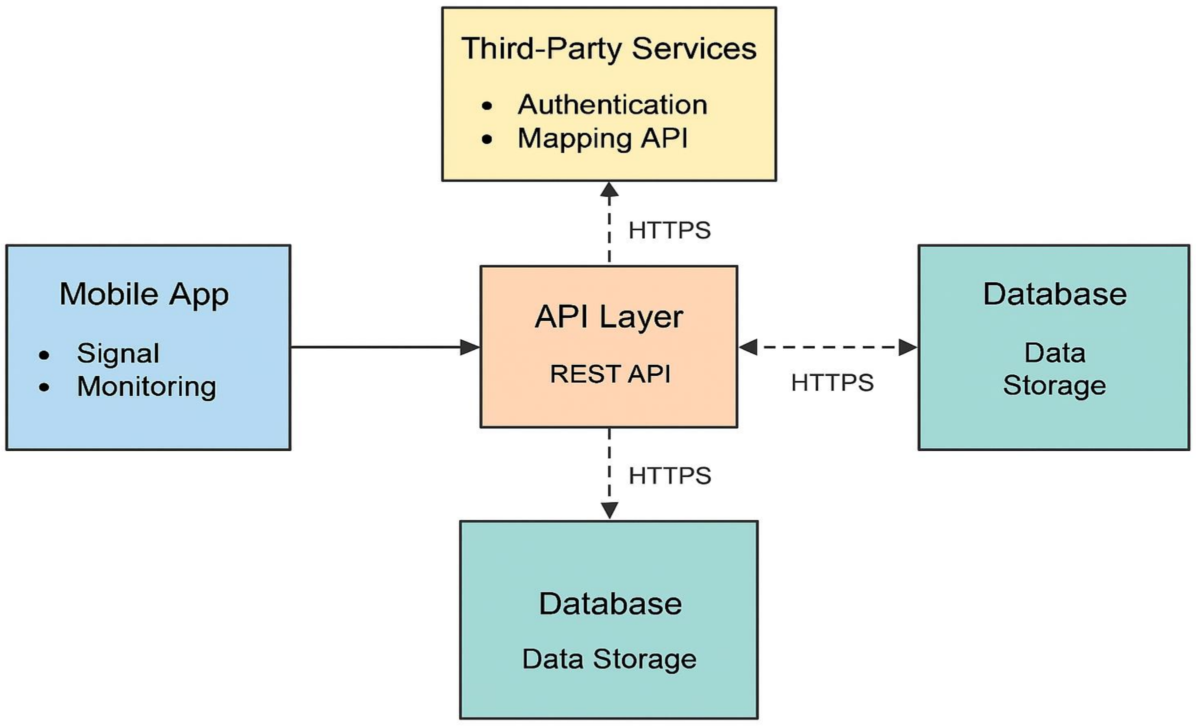
III. System Analysis and Design

1. System Architecture

At the highest level, the system comprises four layers:

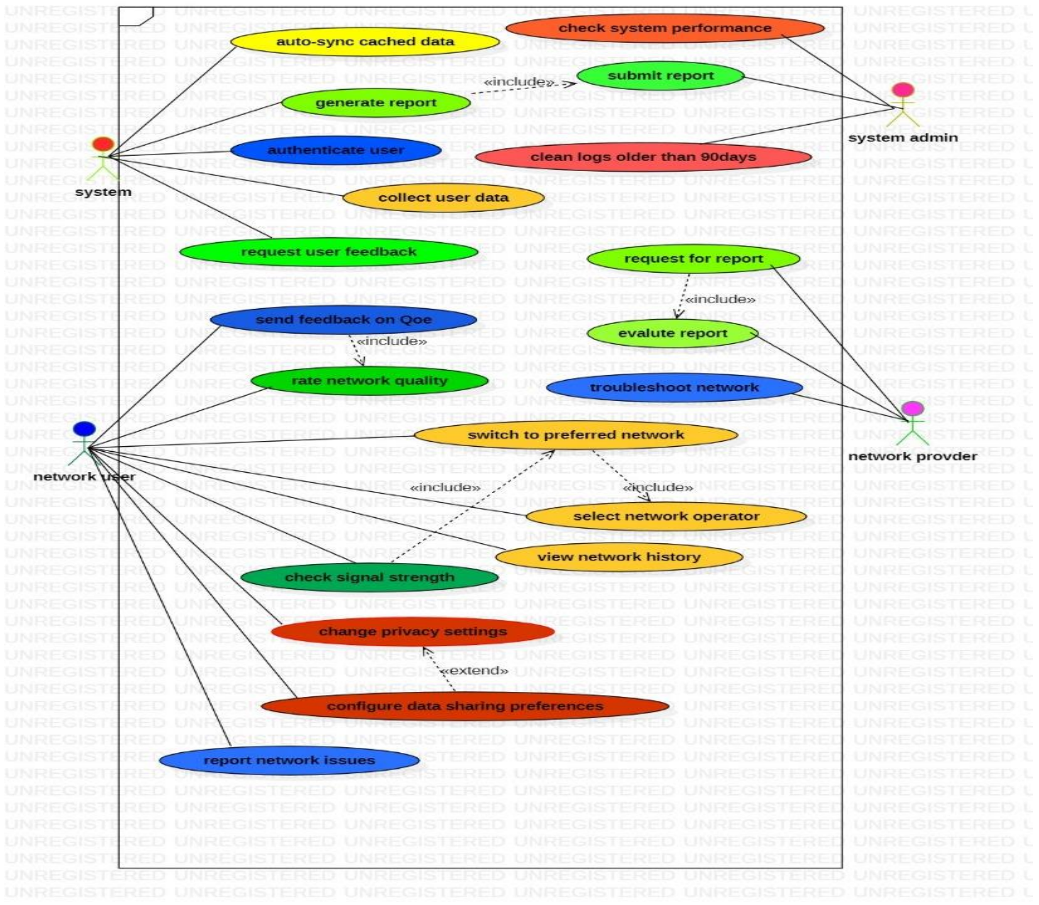
- **Presentation Layer (Mobile Client):** A React Native app handling UI, local caching, and device API calls for signal metrics.
- **API Layer (Backend Server):** A Node.js/Express service exposing RESTful endpoints for authentication, data ingestion, and retrieval.
- **Data Layer (Database):** A MongoDB cluster storing time-series logs, user profiles, and issue reports.
- **External Services:** JWT-based auth (possibly via Firebase Auth), mapping API (e.g., Mapbox), and push-notification gateways.

A block diagram can be used to illustrates these layers and data flows between them, highlighting HTTPS channels, secure token exchanges, and asynchronous sync mechanisms for offline support.



2. UML Diagram

The Use Case Diagram depicts three actors—Mobile Network User, System Administrator, Network Provider—and six use cases (Monitor Signal, Report Issue, View History, Authenticate, Configure Privacy, Switch Network). This behavior-centric view aligns with UML's standard for capturing stakeholder interactions.

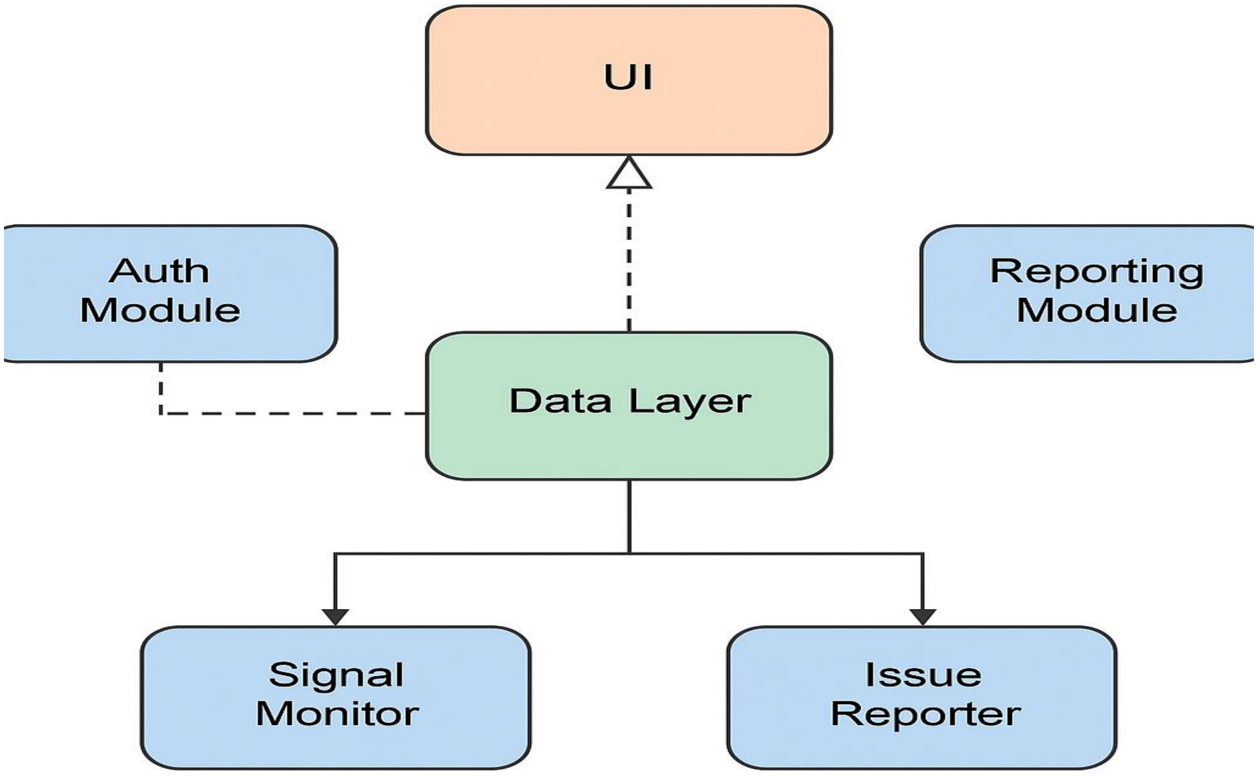


IV. Implementation

1. Component Diagram

Component diagram refines the system into implementation-level modules:

- **UI Module:** Screens (Home, Monitor, Report, History, Settings) implemented via React Native components.
- **Data Layer Module:** Local storage (AsyncStorage/SQLite) for caching and MongoDB client wrappers for remote operations.
- **Auth Module:** JWT token manager with secure storage and Firebase SDK integration.
- **Reporting Module:** Form validation, media upload handler, and API client for issue submission.



2. Breakdown of Modules

Module	Responsibility
Signal Monitor	Invokes device APIs (telephony manager) to fetch dBm, network type, and carrier; updates UI and logs data locally.
Issue Reporter	Provides structured form UI, enforces validation rules, handles optional media attachments, and calls reporting API endpoint.
History Viewer	Retrieves cached logs, transforms them into data points, and renders interactive time-series charts via a charting library (e.g., Recharts).
Settings & Privacy	Manages user preferences for network selection, data-sharing toggles, and triggers config persistence both locally and on the backend.

3. User Interface Mockups

Wireframes showcase:

- **Home Screen:** Dashboard with real-time signal gauge and quick-access buttons.
- **Monitor Screen:** Detailed signal metrics and "Start/Stop Monitoring" control.
- **Report Screen:** Issue form with dropdown types, text area, and camera picker.
- **History Screen:** Scrollable chart with tappable data points.
- **Settings Screen:** Toggle switches for privacy options and network preference selector.

4. Data Schema Overview

The MongoDB schema defines:

- **SignalLog:** { userId, timestamp, latitude, longitude, networkType, signalStrength }
- **IssueReport:** { userId, timestamp, issueType, description, mediaUrl }
- **UserProfile:** { userId, authToken, preferences: {network, anonymize}}.

This schema supports efficient time-series queries and indexing on location fields, optimizing map-based visualizations and analytics.

These analysis, design, and implementation sections ensure a clear, modular architecture and provide the development team with concrete blueprints for coding, testing, and deployment.

V. Verification and Validation Methodology

Verification and validation (V&V) ensure that the system is built correctly (verification) and meets user needs (validation) through systematic reviews and testing at multiple levels. Adopting an **Agile iterative approach** allows continuous feedback, early defect detection, and alignment with evolving requirements.

1. Development Approach

We employ a **Scrum-**based Agile lifecycle with 2-week sprints. Each sprint delivers a potentially shippable increment—covering feature development, code reviews, and automated tests. Sprint retrospectives and backlog grooming sessions incorporate V&V feedback into future iterations.

2. Verification Methods

Per **IEEE 1028** (Software Reviews and Audits), we conduct:

- **Technical Reviews**: Peers examine design documents and code for conformance to requirements (IEEE Standards Association).
- **Walkthroughs**: Authors guide stakeholders through artifacts to identify defects early.
- **Inspections**: Formal, role-based inspection teams evaluate artifacts (e.g., SRS, test plans) using defined checklists (IEEE Standards Association).

Each review produces an inspection report documenting issues, severity, and resolution status.

3. Validation Methods

Aligned with **IEEE 829** (Test Documentation), we execute:

- **Unit Testing**: Developers write and run automated tests for each module (e.g., Signal Monitor, Issue Reporter) to verify correctness.
- **Integration Testing**: Combined modules are tested via CI pipelines to validate inter-module interfaces (e.g., API calls, data caching).
- **System Testing**: End-to-end tests simulate real user workflows on emulators and physical devices, covering all six use cases.
- **Acceptance Testing**: Key stakeholders (end users, network providers) perform user-acceptance tests to validate that the product meets business goals.

4. Traceability Checks

Continuous traceability audits verify that each requirement (FR-1…FR-10, NFRs) maps to one or more test cases. Discrepancies trigger refinement of requirements or tests, preventing coverage gaps.

5. Test Environments and Tools

- **Environments**: Android and iOS emulators (latest two OS versions), physical devices representing urban and rural connectivity.
- **Tools**: Jest and Mocha for unit tests; Postman/Newman for API tests; Appium for automated UI testing; SonarQube for static code analysis (QA Madness).
- **CI/CD**: GitHub Actions pipelines enforce linting, build, test, and deployment steps on every pull request.

VI. Requirements Tracing and Test Plan

A robust traceability and test plan ensure that all requirements are validated, and defects are managed efficiently.

1. Proposed Requirements Traceability Table

Stakeholder Need	Use Case	Req. ID	Test Cases
Real-time monitoring of signal metrics	UC-01: Monitor Network	FR-1	TC-01: Verify display of dBm within 2 s
Background logging of location & signal	UC-02: Passive Collection	FR-2	TC-02: Log entries queued when offline
Structured issue reporting	UC-03: Manual Reporting	FR-3, FR-4	TC-03: Submit report; TC-04: Acknowledge

Historical data visualization	UC-05: View History	FR-5	TC-05: Chart loads ≤3 s with 100 points
Secure user authentication	UC-06: Auth	FR-6	TC-06: JWT login success/failure paths
Privacy configuration & data anonymization	UC-07: Privacy Config	FR-7, FR-8	TC-07: Toggle opt-in; TC-08: Anonymize GPS
Preferring a specific network operator	UC-09: Network Selection	FR-9	TC-09: Preference applies immediately
Offline data synchronization	UC-10: Background Sync	FR-10	TC-10: Auto-sync after connectivity resume

2. Test Analysis (Anticipated Challenges)

- **Offline Sync Reliability**: Ensuring data integrity across intermittent connectivity may lead to race conditions; require robust queueing and retry logic.
- **Device Permissions**: Variations in Android/iOS permission flows can disrupt feature access; implement adaptive permission requests with clear UX prompts.
- **Network Variability**: Testing under simulated 3G/4G conditions demands network throttling tools to validate performance NFRs (Microsoft Learn).
- **Data Privacy Compliance**: Verifying anonymization logic against legal requirements requires automated checks and manual audits (Parasoft).

3. Test Data Management Approach

- **Synthetic Data Generation**: Create realistic signal logs spanning diverse geographies and network conditions.
- **Data Seeding**: Preload test accounts with mock profiles and historical logs.
- **Data Cleanup**: Automate purging of test artifacts post-sprint to maintain environment consistency.
- **Security**: Mask any real user data; employ secure test keys for external services.

VII. User Interface and Experience Validation

1. UI/UX Best Practices

Consistent use of UI components—such as buttons, form fields, and navigation patterns—reduces cognitive load and fosters intuitive interactions, which is vital for mobile applications where screen real estate is limited.

Adhering to accessibility guidelines—ensuring sufficient color contrast, scalable font sizes, and support for screen readers—makes the app usable by users with visual impairments or color-blindness (Accessibility Blog).

Localization for bilingual support (English and French) requires room for text expansion and culturally appropriate iconography, preventing truncated strings and misinterpretation (Gridly).

Employing plain-language principles—writing at a low reading level with concise, well-organized text—benefits users with low literacy or non-native speakers, improving comprehension (WCAG). Designing with consistent layout grids and ample whitespace enhances readability and allows users to identify related content groups quickly (Harvard Accessibility).

2. Usability Testing Plan

We will conduct **heuristic evaluations** using Nielsen's 10 usability heuristics to identify UI issues early; expert evaluators will score each screen for compliance and severity of violations. **User workshops** with 8–10 participants per session will gather qualitative feedback on prototypes; tasks will include monitoring signal, reporting an issue, and adjusting privacy settings (Contentsquare).

A/B testing will compare two interface variants (e.g., different chart styles for history view) to quantitatively measure task completion time and user preference (Nielsen Norman Group).

All sessions will be recorded (with consent) and analyzed to extract pain points, navigational patterns, and satisfaction scores using standardized questionnaires (e.g., SUS) (Looppanel).

VIII. Conclusion

1. Assessment

The SRS provides a complete blueprint for design and prototyping, with clearly defined functional and non-functional requirements, traceability matrices, and validation plans that align with both stakeholder needs and industry standards (Cprime). With prioritized MoSCoW categorization, the team is ready to initiate the first sprint focusing on "Must Have" features—signal monitoring, background logging, report submission, and authentication—with V&V processes embedded in each iteration (Harvard Accessibility).

2. Summary Table of Achievements, Limitations, and Challenges

Category	**Achievements**	**Limitations**	**Challenges**
Requirements Defined	FR-1 to FR-10; NFRs covering performance, security, privacy	Detailed wireframes and component diagrams pending	Ensuring complete traceability across evolving requirements
Validation Plans	V&V methodology per IEEE 829/1028; UI/UX testing strategies	Toolchain setup for automated UI tests in progress	Recruiting representative users from rural areas
Design Artifacts	Use-case, component, and architecture diagrams specified	Prototypes not yet user-validated	Balancing simplicity with rich functionality
Compliance & Legal	Data-protection compliance plan for Cameroon 2025 Act	Localization beyond English/French not scoped	Verifying legal requirements under evolving regulations

3. Future Roadmap Note

Phase 2 will expand on **"Could Have"** features—privacy toggles refinement, network-switch predictions, and advanced analytics—while integrating feedback from Phase 1 usability tests. Post-MVP, we will explore partnerships with operators for direct API integration and pilot deployments in select regions (8allocate).

References

1. Nielsen Norman Group. “Heuristic Evaluations: How to Conduct.” June 25, 2023. (Nielsen Norman Group)
2. Gridly. “7 Best Practices for Effective App UI/UX Localization.” 2023. (Gridly)
3. Glassbox. “Mobile App UX Design: 16 Must-Know Best Practices.” 2024. (Glassbox)
4. Digital Accessibility – Harvard University. “Design for Readability.” 2024. (Harvard Accessibility)
5. UK Government Accessibility Blog. “Dos and Don’ts on Designing for Accessibility.” 2016. (Accessibility Blog)
6. Contentsquare. “8 Usability Testing Methods That Work.” 2023. (Contentsquare)
7. NN/g. “A/B Testing 101.” 2024. (Nielsen Norman Group)
8. Sendbird. “Top 20 Must-Know Mobile App UX Best Practices [2024].” 2023. (Sendbird)
9. Interaction Design Foundation. “Heuristic Evaluation: How to Conduct.” 2024. (The Interaction Design Foundation)
10. WCAG. “Less Is More: Writing in Plain Language.” 2020. (WCAG)
11. Submit.com. “Empowering Low-Literacy Applicants with Accessibility Features.” 2023. (Submit.com)
12. GeeksforGeeks. “Software Requirement Specification (SRS) Format.” 2023. (GeeksforGeeks)
13. TestFort Blog. “SRS, FRS, and BRS in Software Testing.” 2024. (testfort.com)
14. Cprime. “How to Create a Clear High-Quality SRS Document.” 2020. (Cprime)
15. 8allocate. “The Ultimate Guide to Writing Software Requirements Specification.” 2023. (8allocate)