REPUBLIQUE DU CAMEROUN

PAIX – TRAVAIL – PATRIE

MINISTERE DE L'ENSEIGNEMENT SUPERIEURE

UNIVERSITE DE BUEA

REPUBLIC OF CAMEROON

PEACE – WORK – FATHERLAND

MINISTRY OF HIGHER EDUCATION

UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

PO BOX 63

BUEA, SOUTH WEST REGION

DEPARTMENT OF COMPUTER ENGINEERING

COURSE TITLE/CODE: Internet Programming (J2EE) and Mobile Programming – CEF440

COURSE INSTRUCTOR: Dr. NKEMENI Valery

REPORT FOR THE REQUIREMENT ANALYSIS PROCESS (TASK 3)

Presented by **GROUP 13**:

AFAYI MUNSFU OBASE NGALA – FE22A136

ATANKEU TCHAKOUTE ANGE N. – FE22A158

BETNESSI GRACE BLESSING – FE22A446

CHESSEU NJIPDI TERTULLIEN – FE22A181

JAMES ARISTIDE MASSANGO – FE22A226

May 2025

ACADEMIC YEAR: 2024/2025

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

This document presents the detailed analysis of the requirements for the project titled "**Design and Implementation of a mobile app for collecting Quality of Experience (QoE) data from mobile network subscribers**". It aims to ensure that all requirements gathered are clear, complete, consistent, technically feasible, and aligned with the stakeholders' expectations. The requirement analysis phase is a very critical phase because it helps to shape the system (system design) and by identifying the system behaviors and user needs.

1.2 SCOPE OF THE PROJECT

The project involves designing and developing a mobile application capable of collecting both subjective (example: "Are you satisfied with the call quality?") and objective QoE data (example: signal strength, network type, latency time, …) from mobile network subscribers in Cameroon. The app is intended to run in the background, capture user feedback on network performance, and log contextual device/network data (example: signal strength, latency, battery level). The collected data will support mobile network operators in evaluating the real-time quality of their services, enhancing user satisfaction, and driving data-informed infrastructure decisions.

1.3 OBJECTIVE OF THE REQUIREMENT ANALYSIS PHASE

1.3.1 GENERAL OBJECTIVES

The general objective of the requirement analysis phase is to understand what the system is supposed to do and ensure that all stakeholders share a common, precise understanding of the product to be built

1.3.2 SPECIFIC OBJECTIVES

- Ensure requirement completeness.
- Improve clarity and understanding.
- Check technical feasibility.
- Detect and resolve conflicts.
- Prioritize features or requirements based on their importance.
- Classify functional and non-functional requirements.
- Formulate a concise and validated Software Requirements Specification (SRS).
- Validate the requirements with stakeholders.

2.0 REVIEW & ANALYSIS OF GATHERED REQUIREMENTS

In this section of the document, we are going to review and analyze the gathered requirements then assess the completeness, clarity, technical feasibility and dependency relationships.

The systematic review and analysis of the collected responses from the survey form was done in a mixed method – quantitatively and qualitatively, in order to obtain valuable insights about the collected data.

After the data was gathered and later on, cleaned, we performed clustering on the cleaned data in order to obtain quantitative insights of the data. Later on, we performed sentiment analysis on the cleaned data in order to obtain the qualitative insights of the cleaned data.

- **Sentiment analysis – Qualitative analysis**

Tool used: **Pandas** for dataset storage, **Matplotlib** and **Seaborn** for visualizations, **nltk (Natural Language Tool Kit)** for sentiment analysis and **WordCloud** for visualization of words on the same interface.

From the sentiment analysis performed, we obtained insights by generating a word cloud about the expectation of user, user frustration and also a histogram of the common frustration themes. Note that, on the word clouds, the size of a word corresponds to its recurrence in the responses obtained from the survey.

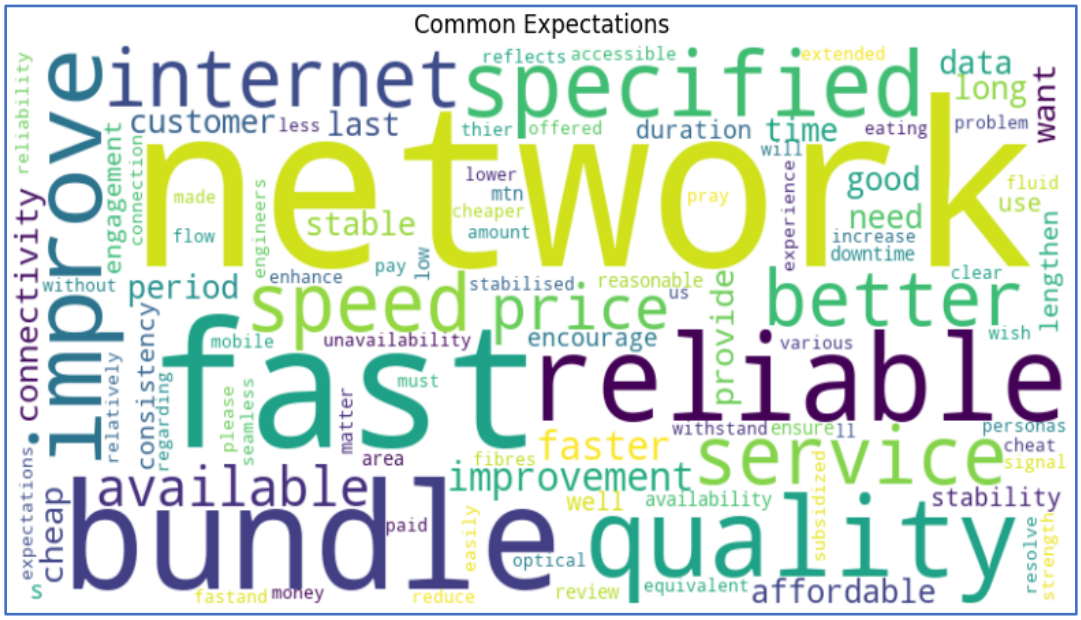


Figure 1: Word cloud of common user expectations

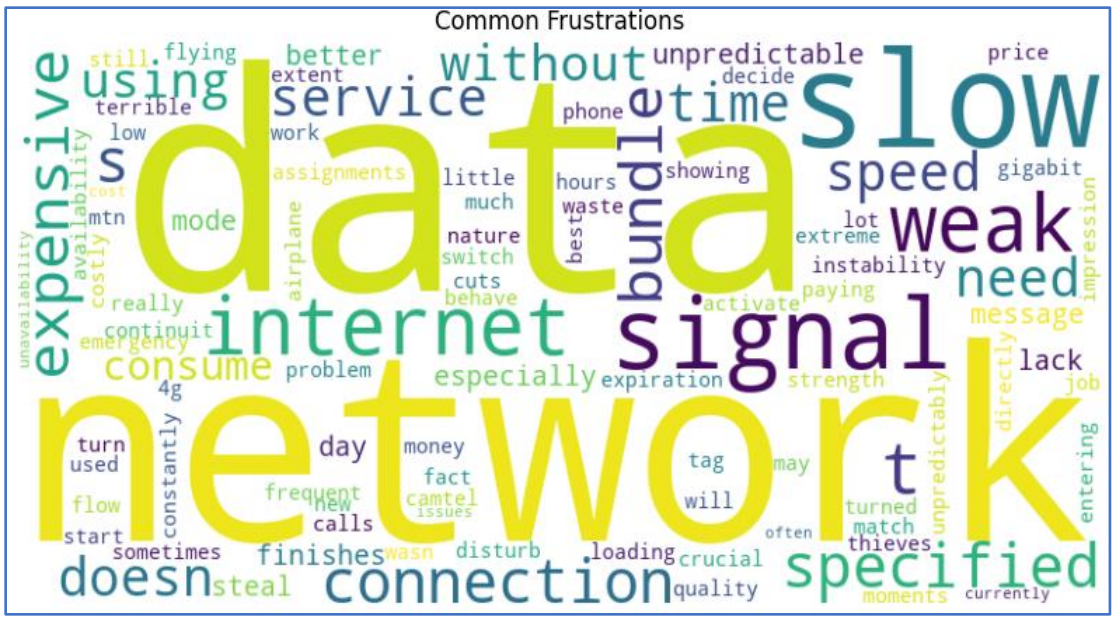


Figure 2: Word cloud of the common user frustration

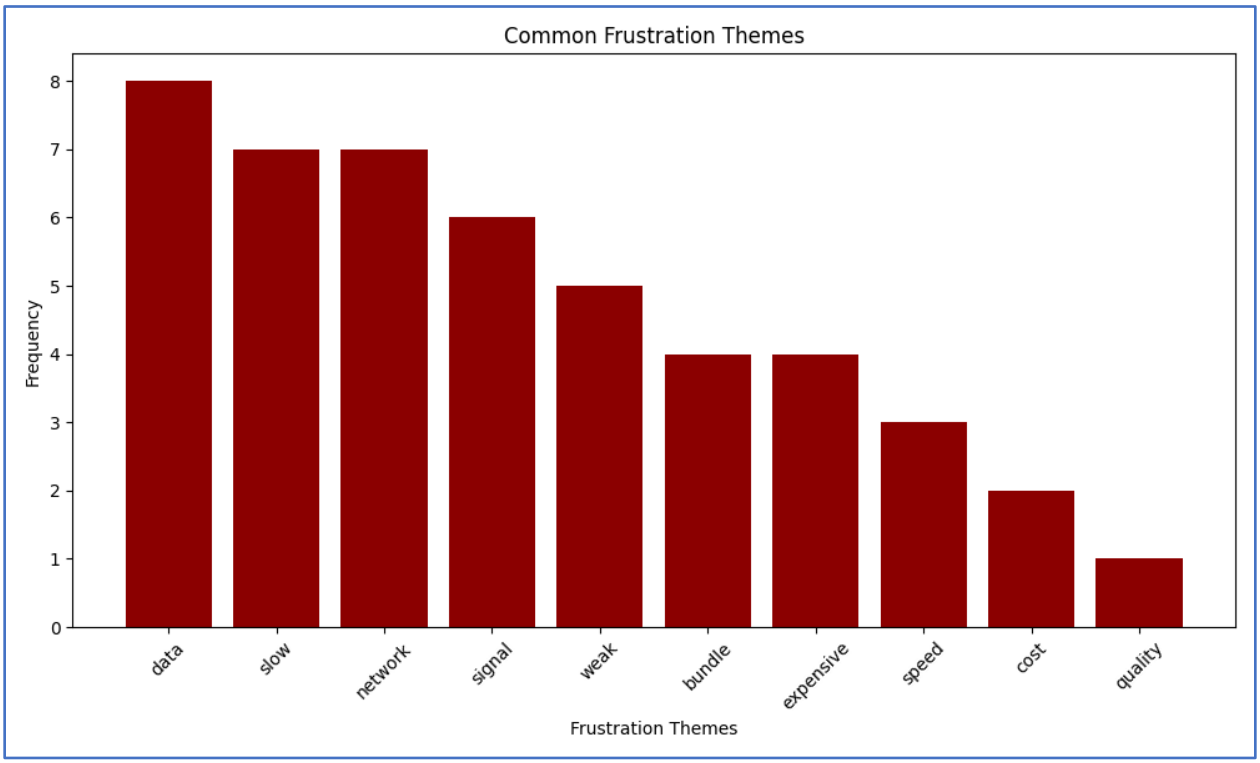


Figure 3: Histogram of common frustration themes

- **Clustering – Quantitative analysis**

Tool used: **Pandas**, **NumPy** to store data, **Matplotlib, Seaborn** and **Scikit-learn** for clustering.

A **cluster** is a group of survey respondents with similar patterns in their responses. This analysis has as aim to group users in cluster based on some characteristics like network operators which they use, challenges faced, location, internet activities, just to name a few.

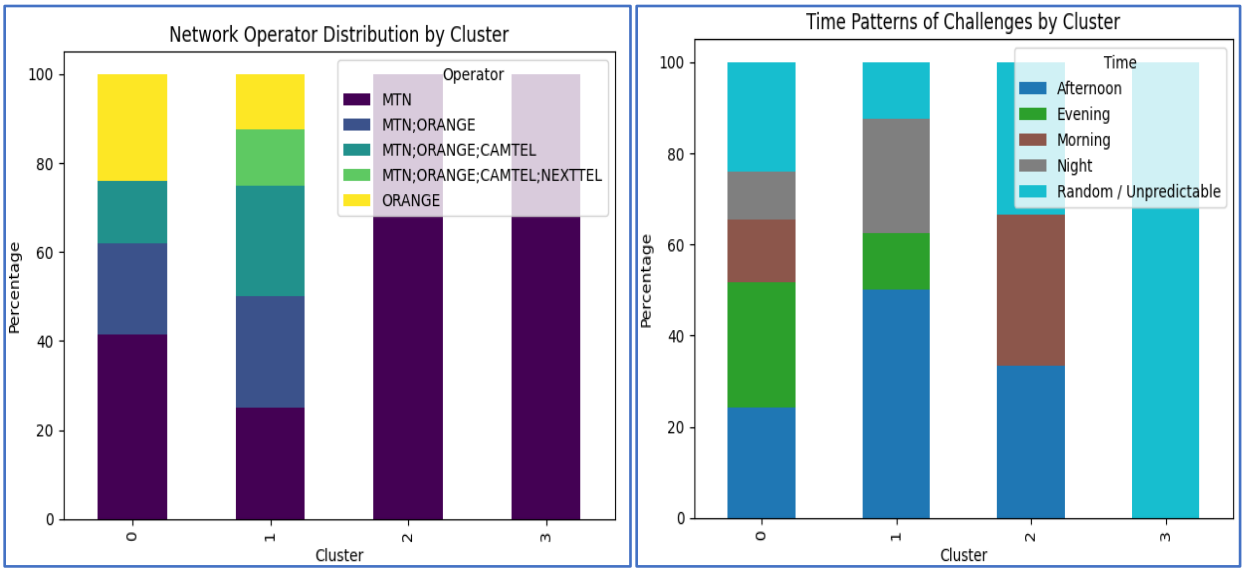


Figure 4: Clusters showing network operator distribution & time pattern of network challenges

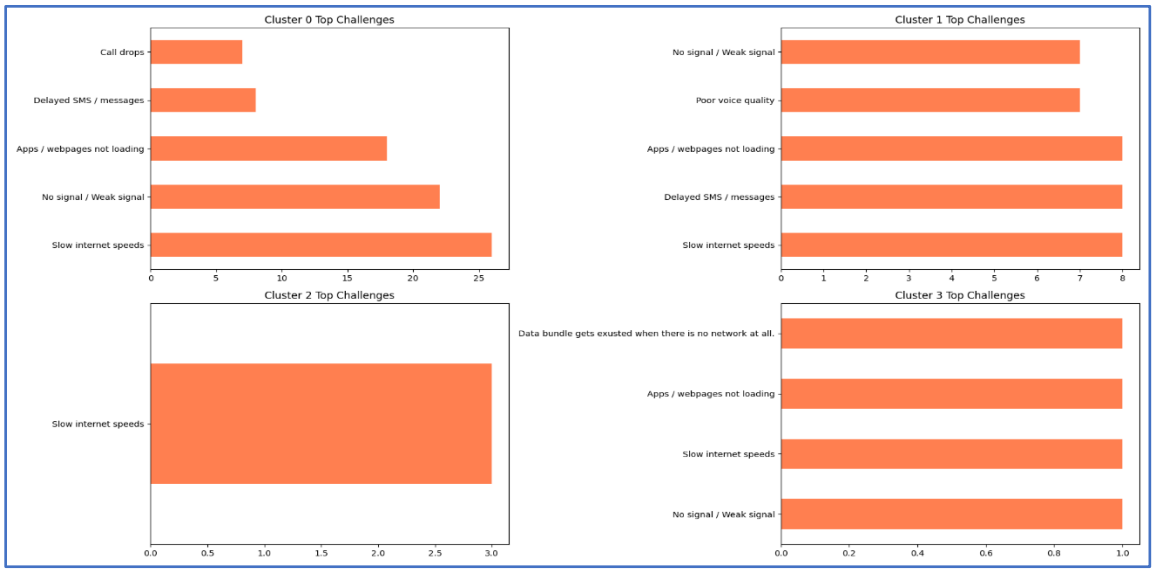


Figure 5: Clusters of challenges faced by network subscribers

2.1 COMPLETENESS

The requirements were gathered through user surveys and stakeholder interviews, focusing on network performance pain points in Cameroon.

Evidence of Completeness:

- **User-Covered Needs:**
  - 85% of users reported frequent slow speeds and signal.
  - 78% emphasized data transparency.
  - 90% supported a feedback app to report issues
- **Core Features Addressed:**
  - Real-time issue reporting (FR-01, FR-02).
  - Automated metric collection (FR-03, FR-04).
  - Operator dashboards (FR-07, FR-08).
- **User Roles Supported:**
  - Subscribers (report issues, view alerts).
  - Operators (analyze trends, resolve complaints).

During the analysis, we identified the following gaps:

- Integration with operator APIs for tower data.
- Handling of rural vs. urban network disparities**.**

2.2 CLARITY AND UNAMBIGUITY

Table 1: Clarity & unambiguity table with resolution and validation method

Requirement	Ambiguity/Clarity Issue	Resolution	Validation Method	Impact
Real-time alerts	Subjective definition of "real-time"	Defined as **≤5s delay** from detection to alert.	Stakeholder agreement with MTN/Orange.	Ensures timely user notifications.

Requirement	Ambiguity/Clarity Issue	Resolution	Validation Method	Impact
Slow internet speeds	No quantitative threshold	Set as **<1Mbps download** or **>500ms latency**.	Technical benchmarking with Ookla.	Standardizes performance complaints.
Data anomaly detection	Vague "unusual depletion" claims	Threshold: **>20% bundle loss in 1h** triggers alerts.	Operator data analysis.	Reduces false positives in reporting.
User-friendly UI	Subjective interpretation	**≤3 taps to submit reports**; WCAG 2.1 AA compliant.	User testing with low-literacy groups.	Improves accessibility and adoption.
Secure data transmission	Unspecified encryption standards	**TLS 1.3+** for transit; **SHA-256** for storage.	Compliance audit.	Mitigates data breach risks.
Battery efficiency	"Minimal drain" not quantified	**≤3% battery/hour** in background operation.	Lab tests on mid-range smartphone devices.	Balances functionality and device impact.
Network availability	No clear uptime metric	**≥95% uptime** during peak hours.	Historical operator outage reports.	Sets measurable service-level expectations.

2.3 TECHNICAL FEASIBILITY

☑ 1. Will Flutter support background functionality to collect network status data?

✔ **Yes, with limitations and plugins.**

- **Flutter** itself is not low-level enough to run complex background services **natively**, but:
    - You can use packages like flutter_background_service to run Dart code in the background.
    - On Android, you'll rely on **Platform Channels** to use native Android features (like TelephonyManager, ConnectivityManager) through Kotlin or Java.
    - On iOS, background execution is **limited** — especially for network metrics. You may use **background fetch** APIs for periodic tasks, but not continuous monitoring.

☑ **Conclusion**:
Yes, Flutter **can** be used for background tasks **with plugin support and native integration**, especially on **Android**, which is more open to background services.

☑ 2. Can Flutter help reduce battery consumption for tasks like ping tests, jitter, speed tests?

✔ **Yes, indirectly — by controlling frequency and method.**

- In Flutter, **you control how frequently background tasks run** (e.g., every 1–2 hours vs. every 5 mins).
- You can **avoid high-battery-impact operations** like speed tests or ping unless the device is charging or on Wi-Fi.
- Use **WorkManager plugin for Android** with constraints like:
    - Only run when charging
    - Only run-on unmetered connections (Wi-Fi)
- Avoid full speed tests — instead, **estimate bandwidth** using file download size over time.

☑ **Conclusion**:
Flutter lets you **build logic to avoid excessive resource use**, helping minimize battery consumption. But the power-saving implementation logic is **your responsibility** as a developer.

☑ 3. Can Flutter be used with GPS location tracking? What free API can be used?

✔ **Yes.** Flutter supports GPS tracking via plugins. Here's how to do it:

☑ Use These Plugins:

- geolocator: Get device's location.
- location: Full-featured alternative with permission handling.
- google_maps_flutter: Display location on a map (optional).

☑ Free Location API (Optional for Enrichment)

If you want to get **location context** (e.g., city, country from coordinates), use:

- **OpenCage Geocoding API** – Free tier: 2,500 requests/day
- **LocationIQ** – Free tier: 5,000 requests/day
- **Nominatim (OpenStreetMap)** – Free and open source (usage policies apply)

🚨 **Important**: For compliance, avoid storing or sharing precise user location without consent. Consider using **coarse granularity** (e.g., town or neighborhood level).

☑ Summary of the Setup Validity

Table 2: Summary of setup validity

Item	Validity	Notes
Flutter for background	☑ Yes (with plugins and native channels)	Works best on Android; limited on iOS
Flutter for battery optimization	☑ Yes	You define frequency, constraints, and logic
GPS location tracking	☑ Yes	Use geolocator or location plugin
Free location API	☑ Yes	Use OpenCage, LocationIQ, or Nominatim

☑ Stack technology Evaluation

Table 3: Stack technology evaluation

Tool/Technology	Role in the App	Why It's Suitable
Flutter	Frontend (UI/UX) & some logics	Cross-platform, highly customizable UI, plugin ecosystem for device access (e.g., GPS, background services)
Android support plugins (e.g., WorkManager, TelephonyManager)	Native background services	Enables signal/log collection, background syncs, network monitoring, especially on Android
SQLite	Local/offline data storage	Stores user feedback and network logs when offline, syncs later with Firebase
Firebase	Cloud backend, authentication, analytics	Provides real-time database (optional), Firestore, authentication, cloud storage, analytics, and notifications
Figma	UI/UX design	Helps prototype user flows, design intuitive UI for tech/non-tech users, enables team collaboration
GitHub Actions	Continuous Integration/Deployment (CI/CD)	Automates building, testing, and deploying your app to Play Store/TestFlight, ensuring quality and fast updates

☑ User Requirement alignment

Table 4: User requirement alignment

Requirement	Can This Stack Handle It?	Notes
Background signal & metric collection	☑ Yes (Flutter + native Android plugins)	Fully supported on Android, iOS limited to fetch/background sync
Real-time feedback forms	☑ Yes	Flutter UI + Firebase submission logic
Local storage (offline support)	☑ Yes (SQLite)	Sync with Firebase when online
Periodic reminders/surveys	☑ Yes	Use WorkManager (Android), Local Notifications
User authentication	☑ Yes (Firebase Auth)	OTP login by phone number or email
Secure, scalable backend	☑ Yes (Firebase)	Scales well with Firestore + Firebase Functions
Operator dashboard	☑ Yes	Build separately as a Flutter web or integrate with Firebase Admin SDK
Multilingual support	☑ Yes	Flutter has excellent internationalization support
Gamification (optional)	☑ Yes	Easy to add through conditional logic and UI badges

Acknowledgement: Our team is still gaining proficiency in the tech skills mentioned above but we've identified resources (tutorials, templates, mentorship) to bridge knowledge gaps. We prioritize feasibility by choosing tools with strong community support and scalability for our skill level.

2.4 DEPENDENCY RELATIONSHIP

Dependency relationships define how different system components rely on one another to function correctly. Understanding these dependencies ensures:

- **Correct development sequencing** (what needs to be built first).
- **Risk mitigation** (avoiding bottlenecks or cascading failures).
- **Efficient testing** (validating dependencies early).

2.4.1 Functional Dependency Matrix

Table 5: Functional Dependency matrix

Feature (FR-ID)	Depends On	Dependency Type	Impact if Unmet
FR-01: User Issue Reporting	FR-09 (Auth), NFR-03 (GPS Accuracy)	Technical + Data	Reports lack location context
FR-03: Speed Tests	NFR-10 (Battery Optimization)	Resource Constraint	Excessive battery drains
FR-07: Operator Dashboard	FR-03/FR-04 (Data Collection)	Data Pipeline	Empty/incorrect analytics
FR-06: Real-time Alerts	FR-05 (Correlation Engine)	Processing Dependency	Delayed/irrelevant alerts

2.4.2 Cross-Module Dependencies

Table 6: Cross-module dependencies

Module	Upstream Dependencies (What we need first)	Downstream Impacts (Who depends on us)
Authentication	NFR-05 (Security)	All user-submitted data
Data Collection	NFR-11 (Network Availability)	Dashboard accuracy (FR-07/FR-08)
Notification System	FR-12 (Location Tracking)	Alert relevance (NFR-09)

2.4.3 Risk-Linked Dependencies

Table 7: Risk-linked dependencies

Dependency Chain	Risk	Mitigation Strategy
FR-04 → FR-07 → NFR-04 (Anomaly Detection → Dashboard → Scalability)	High-volume data crashes system	Implement data sampling for MVP
NFR-03 → FR-02 → FR-01 (GPS → Contextual Reports → User Adoption)	Inaccurate location tagging	Fallback to cell-tower triangulation

3.0 IDENTIFICATION OF INCONSISTENCIES, AMBIGUITIES, AND MISSING INFORMATIONS

This section has as objective to ensure the mobile network quality feedback system meets network user and operator needs. So, in order to realize this, we rigorously analyzed requirement in order to identify: **inconsistencies, ambiguity** and **missing information**.

This step is very crucial because it helps to identified and resolve these identified issues. We need to note that, these issues lead to:

- Inconsistencies erode the trust between users and operators.
- Ambiguity leads to rework.
- Missing information causes delays.

3.1 INCONSISTENCIES

- **Operator naming**: There were variations like "MTN", "MTN - ORANGE", "ORANGE" without standardization.
- **Bundle descriptions**: Inconsistent formats (e.g., "500f for 1g", "1000=2.5 GB", "*220# (yamo surf) 1,149U for 2.2G /30days")
- **Time formats**: Some use "Evening" while others use specific times like "5:00 AM".
- **Currency notations**: Mix of "frs", "FCFA", "U", and no currency indicators

3.2 AMBIGUITY

- **"Random/Unpredictable" timing**: Too vague for meaningful analysis of when issues occur.
- **Location descriptions**: Terms like "In my Area" and "Everywhere" are not specific
- **Bundle durations**: Unclear durations (e.g., "500frs-1.12Gb/ per every after 2days")
- **Quality assessments**: Subjective terms like "trash", "good", "fast" without quantification

3.3 MISSING INFORMATION

- **Blank responses**: Several entries have missing data for key questions.
- **Reason for use**: Several entries lack explanations for why they chose their operator.
- **Frustration points**: Several blanks about user frustration with their mobile network.

4.0 PRIORITIZATION OF REQUIREMENTS BASED ON IMPORTANCE AND FEASIBILITY

Based on the survey data, we can categorize user frustrations and expectations to define **scope-aligned requirements** for the app to be develop. In order to realize this specific step, we identified the key user frustrations, user expectations, then proceeded with prioritizing requirements based on the app's scope.

So, from our analysis, we identified the following:

4.1 KEY USER FRUSTRATION

- **Network Performance Issues**
  - Weak/no signal (most frequent complaint)
  - Slow internet speeds (affects work, streaming, classes)
  - Call drops & delayed SMS (communication disruptions)
  - Apps/webpages not loading
- **Data & Pricing Problems**
  - Data bundles exhaust too quickly
  - High cost for limited data
  - "Silent data consumption" (data used without activity)
  - Unfair pricing relative to quality
- **Service Reliability**
  - Unpredictable network availability
  - Poor service during peak hours (evening/night)
  - Need to frequently toggle airplane mode
- **Customer Experience**
  - Lack of transparency in billing
  - No effective feedback mechanism
  - Operators don't act on complaints

4.2 USER EXPECTATION

User want:

- Stable & fast connectivity
- Affordable, longer-lasting data bundles
- Transparency in data usage and billing
- Better coverage in weak zones
- A way to report issues and get resolutions

4.3 APP SCOPE & PRIORITIZED REQUIREMENTS

4.3.1 HIGH PRIORITY (CORE FEATURES)

Table 8: High priority requirements

Requirement	Why?
Real-time network issue reporting	Users want to log weak signals, slow speeds, etc.
Data usage transparency dashboard	Track consumption & detect "silent data" leaks
Direct feedback channel to operators	Users demand accountability
Crowdsourced network quality map	Identify weak zones (schools, homes)
Bundle recommendations	Suggest cost-effective plans based on usag

4.3.2 MEDIUM PRIORITY (ENHANCEMENTS)

Table 9: Medium priority requirements

Requirement	Why?
Automated complaint escalation	Ensure operators acknowledge reports
Peak-hour performance alerts	Warn users of expected slowdowns
User education on network optimization	Tips to improve connectivity
Operator response tracking	Show if complaints are resolved

4.3.3 LOW PRIORITY (FUTURE SCOPE)

Table 10: Low priority requirements

Requirement	Why?
Integration with operator APIs	For real-time resolution updates
AI-driven network diagnostics	Advanced troubleshooting
Community forums for users	Share tips & collective bargaining

PRIORITIZATION JUSTIFICATION

The prioritization of these requirements were done based on the following:

Why High Priority?

- **Immediate pain points**: Addresses top frustrations (signal, speed, data fairness).
- **Feasible**: Basic reporting & tracking features are technically simple.
- **User trust**: Transparency in data usage builds credibility.

Why Medium Priority?

- **Improves UX**: Alerts and education reduce frustration.
- **Encourages operator engagement**: Complaint tracking pushes accountability.

Why Low Priority?

- **High development effort**: API integration/AI requires partnerships.
- **Nice-to-have**: Community features are secondary to core reporting.

5.0 CLASSIFICATION OF REQUIREMENTS

Based on the survey data and project description, here are the categorized requirements for the mobile application.

5.1 FUNCTIONAL REQUIREMENTS

Table 11: Functional requirements elicitation

Requirement ID	Module	Description
FR-01	User reporting and feedback	The system shall provide users with capability to manually report network issues and submit periodic network satisfaction surveys.
FR-02	User reporting and feedback	The system shall enable contextual feedback by capturing GPS location of reported issues, timestamp of the issue occurrence, network operator and connection type.
FR-03	Automated network monitoring	The system shall continuously measure and log metrics such as signal strength, network speed, call/SMS reliability metrics and monitor data usage anomalies.
FR-04	Automated network monitoring	The system shall detect and flag anomalous data consumption patterns (example: “Data bundle exhausted prematurely”) and geographical coverage gaps.
FR-05	User experience management	The system shall correlate network performance with active applications/services during issues and also update users about network status through notifications.
FR-06	User experience management	The system shall provide real-time network status indicators, service disruption notifications and resolution updates to the network subscribers.
FR-07	Operator analytics	The system shall provide the network operators with interactive QoE dashboards for network monitoring, automated reports and data export functionality.
FR-08	Operator analytics	The system shall generate heatmaps of areas having network issues and an update section in order to notify the users of a resolved network issue.
FR-09	Authentication and security	The system shall authenticate users via mobile number as their identifier. On the other hand, network operator will authenticate via network admin credentials.
FR-10	Authentication and security	The system shall maintain secure session management (by enabling network subscriber and operators to logout) and implement role-based access control.

5.2 NON-FUNCTIONAL REQUIREMENTS

Table 12: Non-functional requirement elicitation

Non-Functional Requirement ID	Aspect	Description
NFR-01	Performance	The system shall make use of minimal battery/data usage for background operation. It shall also implement low-latency feedback submission to capture real-time issues.
NFR-02	Usability	The system shall provide a simple and intuitive UI for users with varying tech literacy. The systema may also comprise of a bilingual support in order to enhance the usability.
NFR-03	Privacy & Security	The system shall anonymize sensitive data (for example: providing location granularly instead of exact GPS). The system shall secure data transmission to prevent misuse.
NFR-04	Scalability	The system shall support concurrent usage by 25+ users during peak hours (example: evenings) and handle $\geq$ 10 issue reports per minutes during network outage.
NFR-05	Data integrity	The system shall ensure proper (100%) accuracy in the transmission of data specifically the network metrics in order to prevent data corruption which may lead to extra major issues.

6.0 SOFTWARE REQUIREMENTS AND SPECIFICATIONS

Software Requirements Specification (SRS) Document

Project Title: Design and Implementation of a Mobile App for Collecting QoE Data from Mobile Network Subscribers

Standard: IEEE Std 830-1998

Prepared by: Group 13

Date: May 2025

Version: V1.0

---

1. Introduction

1.1 Purpose

The purpose of this section of the document is to define the software requirements for the development of a mobile application aimed at collecting Quality of Experience (QoE) data from mobile network subscribers in real-time and periodically. The app will assist network providers and analysts in understanding network issues from the user's perspective to improve overall service delivery.

1.2 Scope

This mobile application will allow users to report mobile network problems (signal loss, slow speed, dropped calls), complete periodic surveys, and provide feedback in both structured and open-ended forms. It will also log background data such as signal strength, GPS location, internet speed, and timestamp data. An admin dashboard will be included for network operators to visualize and analyze the collected data.

1.3 Definitions, Acronyms, and Abbreviations

- **QoE**: Quality of Experience
- **GPS**: Global Positioning System
- **REST API**: Representational State Transfer Application Programming Interface
- **SRS**: Software Requirement and Specifications
- **FR**: Functional Requirement
- **NFR**: Non-functional Requirement

1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

The rest of this document is structured to detail the functional, non-functional, business, and survey-based requirements of the application. It also outlines the app environment, user interfaces, performance expectations, and external interface requirements.

---

2. Overall Description

2.1 Product Functions

- Allow users to report network issues manually
- Collect background data (signal strength, speed, GPS)
- Present periodic surveys
- Enable open-ended feedback
- Provide real-time network status
- Store and sync data for offline mode
- Admin dashboard for data visualization

2.2 User Characteristics

- **Mobile Users:** Non-technical individuals with smartphones
- **Admin Users:** Network operators with moderate technical skills

2.3 Constraints

- App must not excessively drain battery or consume data.
- Must comply with data privacy regulations.
- App must be built using appropriate OS compatibility.

---

3. Specific Requirements

3.1 Functional Requirements

FR-01: Manual Issue Reporting

- **Description:** The system shall allow users to report network issues via in-app forms.
- **Inputs:** Issue type (e.g., signal, speed), optional free-text comments.
- **Processing:** Store locally (if offline), tag with timestamp and sync when connected.
- **Outputs:** Confirmation message to user, stored report in backend.

FR-02: Contextual Feedback Logging

- **Description**: Automatically capture GPS location, timestamp, operator, and connection type.
- **Inputs:** Device context.
- **Processing:** Tag data to associated report submission.
- **Outputs:** Enhanced metadata with report.

FR-03: Background Network Monitoring

- **Description:** Continuously monitor signal strength, call/SMS reliability, data speed
- **Inputs:** Network APIs
- **Processing:** Periodic logging
- **Outputs:** Log entries, available to backend

FR-04: Anomaly Detection in Data Usage

- **Description:** Detect patterns like premature data exhaustion
- **Inputs:** Usage metrics
- **Processing:** Trend analysis, pattern matching
- **Outputs:** User notification

FR-05: Application Activity Correlation

- **Description:** Correlate issue with current user activity
- **Inputs:** Dropdown-selected activity (e.g., streaming)
- **Processing:** Link activity with report
- **Outputs:** Context-aware logs

FR-06: Real-time Network Status & Notifications

- **Description:** Show signal indicator and send resolution updates
- **Inputs:** Real-time signal data
- **Processing:** Display, alert generation
- **Outputs:** In-app visual feedback, push notifications

FR-07: Operator Dashboard

- **Description:** Web-based analytics and export functionality for network data
- **Inputs:** Aggregated user reports
- **Processing:** Data visualization and filtering
- **Outputs:** Graphs, tables, CSV exports

FR-08: Issue Heatmaps and Notifications

- **Description:** Create geographic heatmaps and notify users about resolved issues
- **Inputs:** Location-tagged issue data
- **Processing:** Aggregation and visualization
- **Outputs:** Map overlays, resolution alerts

FR-09: Authentication

- **Description:** Verify users via phone numbers and admins via credentials
- **Inputs:** Mobile number or admin ID
- **Processing:** Credential validation via Firebase/Auth service
- **Outputs:** Access token or error

FR-10: Session Management & Access Control

- **Description:** Manage sessions with timeout and role-based permissions
- **Inputs:** Session activity data
- **Processing:** Timeout check, permission check
- **Outputs:** Session state, access enforcement

3.2 Optional additional requirements

- Dashboard analytics (location/time/type filters)
- Alerts for recurring issues via backend
- Signal outage heatmaps
- Offline mode for data collection and later sync

---

4. External Interface Requirements

4.1 User Interfaces

- Mobile app UI: Tabs for issues, surveys, feedback, status widget.

4.2 Hardware Interfaces

- Smartphone sensors: GPS, telephony manager, network state

4.3 Software Interfaces

- Android OS, iOS, Firebase Auth, REST API endpoints

4.4 Communication Interfaces

- HTTPS for secure transmission
- Background sync services with retry logic

---

5. Other optional requirements

- The app must provide notification settings for user control
- Feedback forms should be adaptive and localized based on the user's region
- Logs shall be kept for up to 30 days on device before syncing or deletion

7.0 VALIDATION OF USER REQUIREMENTS

The **validation of User Requirements** phase ensures that all specified requirements truly reflect the needs, expectations, and constraints of end-users and stakeholders. This phase involves confirming the accuracy, completeness, clarity, and feasibility of the requirements through methods such as surveys, interviews, prototyping, and expert reviews. It acts as a quality gate to prevent misunderstandings, reduce rework during development, and ensure the final system delivers real-world value.

The techniques that were used in this phase:

7.1 USER SURVEYS

We established and distributed structures questionnaires via Google forms to the network subscribers. The surveys help to obtain user validation of proposed requirements. In a similar manner, the network operator requirements were validated by making use of a Google form. The questionnaires samples will be shown in the appendices section.

7.2 USE CASE SCENARIOS

To simulate real-world interaction with the app, we created detailed use case scenarios representing typical user activities (e.g., reporting a signal outage while commuting, receiving resolution notifications, or submitting periodic feedback). These scenarios helped validate whether the functional requirements aligned with user expectations and day-to-day behaviors.

The use case scenario table is shown below:

Table 13: Use case scenario development

Use Case	Actor(s)	Pre-condition	Main Flow	Post-condition
Submit a complaint	User	Logged in with phone number	Selects issue type, adds location/time, submits	Complaint is saved and visible to operator
Complete a feedback survey	User	Receives survey notification	Rates different parameters, submits form	Data is stored for analysis
Check real-time network status	User	App is installed	Opens app, views signal type, latency, call drop, signal strength, speed etc.	User gets informed view of current service
Track call performance	App (background)	Active SIM detected	App tracks dropped or failed calls	Logs are added to performance DB

Authenticate user	User	Installs app	Provides phone number and username to logs in	Session starts with user's identity
View analytics dashboard	Operator	Logged into dashboard portal	Views graphs by time, region, issue	Operator identifies problem zones
Tag network operator	App	SIM is active	Reads mobile operator name	Operator tagged with all feedback
Submit feedback offline	User	No internet	Fills out issue form	Data is stored locally and synced later

7.3 REQUIREMENT TRACEABILITY

We developed a requirement traceability matrix to track each requirement from its origin (survey, scenario, or stakeholder feedback) through its implementation in the system. This ensured consistency, prevented requirement loss, and facilitated future maintenance by clearly linking each feature to its source of validation.

Table 14: Requirement traceability matrix

Requirement Category	Requirement	Expected Feature(s)
Functional	Manual issue reporting (issue reported in forms)	In-app issue reporting form with categories (e.g., signal, speed, call drops)
Functional	Periodic surveys	Push notifications with rating forms (collects quality of experience based on fixed issue and also app ratings)
Functional	Open-ended feedback	Free-text input field (will be periodic and in a pop-up manner of display)
Functional	Si8.0nal strength logging	Background signal logging module
Functional	Internet speed test	Built-in speed test utility
Functional	Call/SMS performance tracking	Passive logs of dropped calls, delayed SMS
Functional	Timestamp logging	Time-stamped reports for all issues
Functional	Location tagging	Use of GPS or region-based tags
Functional	Operator tagging	Auto-association with SIM card provider (associates issues to particular network operators)
Functional	Activity correlation	Drop-down selector of current activity (e.g., "Streaming", "Work"). This feature will be linked to open-ended feedback feature
Functional	Real-time network status	In-app widget for live signal indicator
Functional	Issue resolution alerts	Push notifications for resolved issues
Functional	Phone number-based authentication	OTP-based login

Functional	Operator dashboard	Web admin panel with charts and reports
Functional	Logout functionality	Logout button in settings menu
Non-Functional	Performance (battery/data)	Lightweight background services
Non-Functional	Usability	Clean UI, multilingual (EN/FR)
Non-Functional	Privacy/Security	Anonymized GPS data, HTTPS
Non-Functional	Scalability & Integrity	Cloud-based, load-balanced infrastructure
Business	Analytics dashboards	Admin-facing graphs by time, location, issue type
Business	Alerts for recurring issues	Automated backend alerts for patterns
Business	Operator API integration	REST API for operator backend systems
Survey-Based	"No signal / Weak signal"	Signal strength heatmaps + outage alerts
Survey-Based	"Slow speeds during classes"	Priority tagging for educational activities
Survey-Based	"Random/unpredictable issues"	Background monitoring with timestamps/location
Additional	Offline Mode	Store feedback when offline, sync later

8.0 CONCLUSION

The **Requirement Analysis** process has systematically transformed user frustrations and expectations into a clear, actionable roadmap for the mobile app. By rigorously evaluating, prioritizing, and validating requirements, we have ensured that the solution aligns with both **user needs** and **technical feasibility**.

This phase ensured that important outcomes should be met. Some of these outcomes, include:

- **User-centric focus**

The analysis identified critical pain points (e.g., unstable connectivity, data transparency) and translated them into **high-priority features** like real-time reporting and usage dashboards.

- **Structured prioritization**

Using priority levels (high – medium – low), we defined an MVP (Minimum Viable Product) that delivers immediate value while planning scalable enhancements.

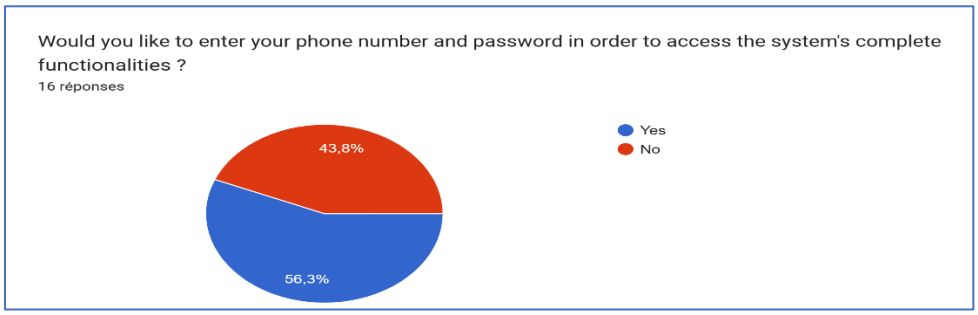
- **Clear specifications**

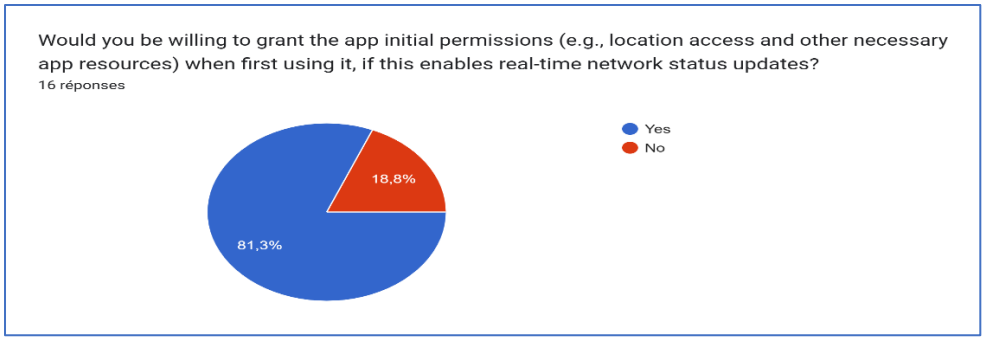
The **SRS document** presents a comprehensive blueprint for the design of the system and provides developers with precise functional and non-functional requirements, reducing ambiguity during implementation.

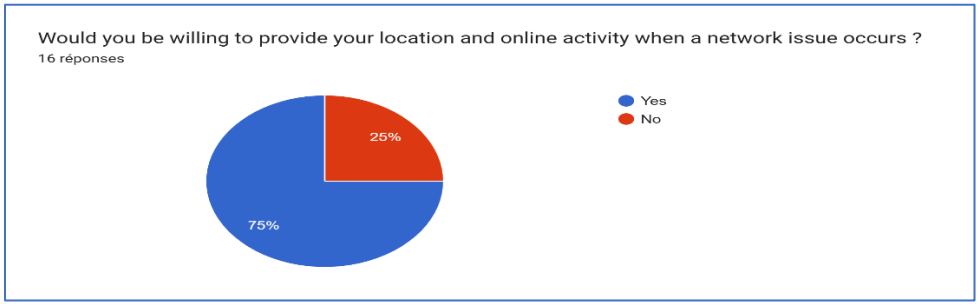
APPENDICES

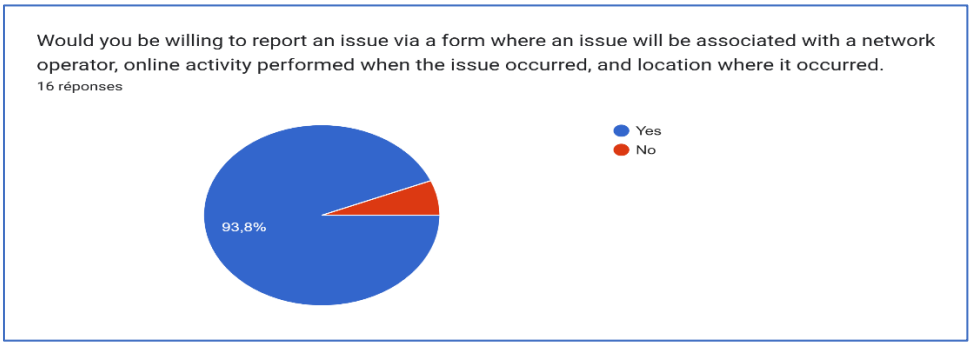
APPENDIX A: Requirement validation questionnaire for the network subscribers

Google form link: https://forms.gle/eHtJ7JfhgBszKwqy5



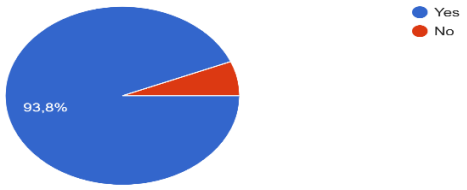






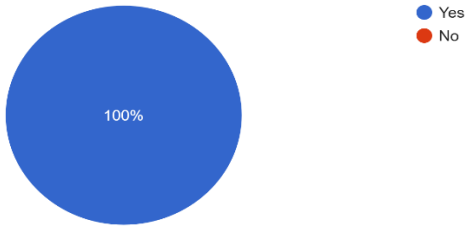
Should the app have a dashboard showing your current (real-time) network status (like network type, latency, call drop, network speed, signal strength, ...) ?

16 réponses



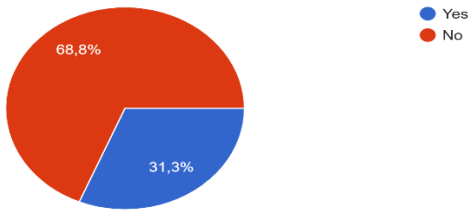
Should the app to provide you with a Help section (e.g. for user request assistance) ?

16 réponses



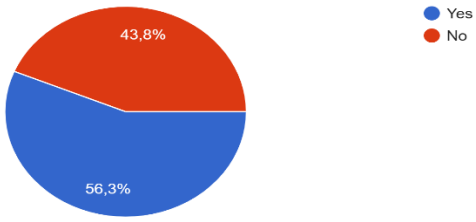
Would you be willing to use to the app even if it drains the battery of your phone passively (gradual consumption of a phone's battery even when the app is not actively in use) ?

16 réponses



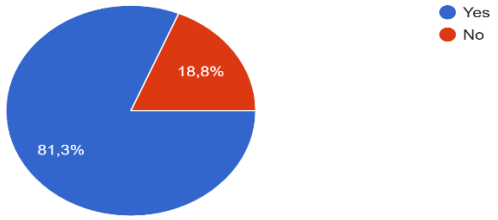
Should the app prompt an open-ended feedback periodically in a pop-up mechanism whenever the user stays for a while without using the app ?

16 réponses



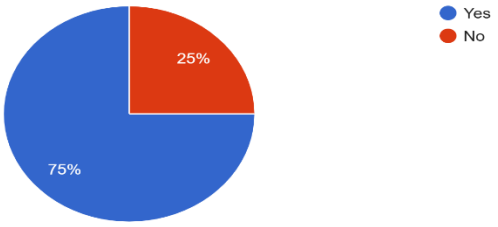
Should the app send the users with two type of notifications - a resolved issue notification and a periodic survey notification ?

16 réponses



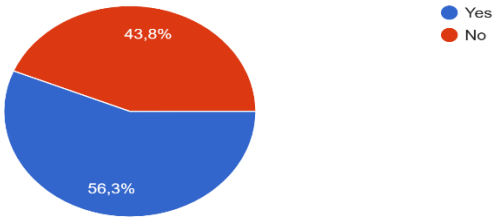
Should the app provide you with a dashboard where you can be able to view resolved network issues through text and heatmaps (for zones with more stable internet connection) ?

16 réponses



Would you like to provide feedback periodically about your current network status to through the app ?

16 réponses



On a scale of 1 - 5, how likely are you to use this app (1 = Not all all - 5 = Very likely)

16 réponses