INTERNET PROGRAMMING AND MOBILE PROGRAMMING

CEF 440

Course instructor:
DR. Valery Nkemeni

Group 20

05th May 2025

Table of content.

REPORT ON TASK 3:

REQUIREMENT ANALYSIS.

INTRODUCTION:

Following the requirement gathering (done in task 2), the information gathered needs to be analysed for further processing and to detect error, uncompleted, missing or ambiguous data in the data collection process to save time and cost in the development of the project.

DEFINITION:

Requirement analysis is the process of refining and structuring the gathered requirements to ensure they are clear, feasible, and aligned with stakeholder goals. This step helps to avoid scope creep, ensure completeness, and prioritize development efforts effectively.

1. Review and Analysis of Gathered Requirements:

We reviewed the requirements gathered through surveys, interviews, and focus groups. Here's an evaluation based on key criteria:

- Completeness: The data captured both subjective (user feedback) and objective (network metrics) requirements.
- Clarity: Most user responses were clear, though some feedback required rephrasing for technical understanding.
- Technical Feasibility: The app's main features (background monitoring, prompting for feedback, logging network metrics) are feasible using Android's APIs and third-party libraries.
- Dependency Relationships :
    - Continuous background activity depends on battery optimization settings.
    - Real-time location tracking depends on GPS permissions.
    - Feedback prompts rely on user interaction at scheduled intervals.

The review confirmed that the gathered requirements are largely relevant, feasible and aligned with user needs. Some dependencies and technical considerations were noted, laying the groundwork for the next phase of design and development

2. Identified Inconsistencies, Ambiguities, and Missing Information:

During the analysis of the gathered requirements, several inconsistencies and unclear or missing details were identified, which could affect the accuracy and effectiveness of the final system if not addresses early.

a) Inconsistencies and Ambiguities:

- Target Platform Not Specified: Unclear if app is for Android, iOS, or both.
- User Feedback Frequency: Vague – how often will users be prompted?
- Data Collection Permissions: No clarity on consent for accessing sensitive data like location.
- Scope of Network Metrics: Collection methods for jitter, latency, and packet loss are not detailed.

- Data Security & Privacy: No mention of how data is protected or anonymized.

b) Missing Information:

- Target Users: Who will use the app? General public, volunteers, etc.
- Authentication: Is login required or is the app anonymous?
- Feedback Mechanism: What format is the feedback in (rating, text, etc.)?
- Analysis Tools: Will there be dashboards or reports for network operators?
- Operator Identification: How is the user's network operator detected or recorded?
- Data Collection Frequency: How often is passive data collected?
- Internet Usage: Will the app use mobile data for transmission, and is this disclosed to users?

3.Prioritizing Requirements for Quality of Experience Mobile Application:

Our Quality of Experience (QoE) app transforms how users and providers like MTN, Orange and CAMTEL tackle network issues by gathering feedback and tracking metrics like signal strength, call quality, and internet speed. By zeroing in on features that matter most to users—like signal strength, critical for 47.2%—and ensuring they're achievable with our Android tools in the time allocated for the work, we've crafted a focused plan. Guided by the survey responses, this prioritization delivers a user-friendly app that maximizes network performance for all.

Prioritized Requirements:

The requirements were prioritized into four categories—**Must Have**, **Should Have**, **Could Have**, and **Won't Have**—based on their importance to users and operators and their feasibility within our team's resources (five students, Android Studio, 3-6-month timeline).

a) Must Have Requirements:

These are essential for the application's core functionality and are technically feasible:

- Track Signal Strength: Monitor signal strength metrics, critical due to 47.2% of users reporting poor signal (Task 2, page 5).
- Track Slow Internet: Measure bandwidth and latency, vital as 43.4% face slow internet issues.
- Track Call Breakages: Capture jitter and latency for audio/video calls, with 58.5% reporting breakages.
- Star Ratings for Feedback: Allow users to rate satisfaction, speed, and usability, expected by users (survey question) and simple to implement via Android UI.

b) Should Have Requirements:

These are important but not critical, feasible with moderate effort:

- Location Tracking with Consent: Log feedback with location data, supported by high survey responses (page 6) noting location's influence, but requires user permission handling.
- Data Deletion Option: Enable users to delete their feedback, addressing privacy concerns (high demand, page 7) and feasible with database controls.

- Simple User Interface: Design a straightforward UI, requested by users ("simple, straightforward," page 7), achievable with Android layouts.
- Low Battery and RAM Usage: Optimize resource consumption, a user concern ("not battery draining," page 7), possible with profiling but secondary.

c) Could Have Requirements:

These enhance the app but are lower priority, still feasible:

- Weekly Feedback Prompts: Remind users to provide feedback weekly, preferred by 60% of respondents (page 7), implemented with simple timers.
- Free Data Donations: Not feasible and outside project scope.

d) Won't Have Requirements:

These are low importance or not feasible within constraints:

- Scam Detection: Suggested by users but unrelated to QoE and technically complex (requires AI, 9-12 months).
- iOS Support: Not viable as the project focuses on native Android (Task 1).
- Real-Time Notifications: High effort (complex networking) and less critical than core metrics.
- Global Access: Unaffected by climate/topography, not feasible within timeline and skills.

I. Justification of Prioritization:

The prioritization leverages the survey data and feasibility analysis:

➢ **Importance**:

- User Needs: The survey highlighted call breakages (58.5%), signal strength (47.2%), and slow internet (43.4%) as top issues, mapping to jitter, signal, and bandwidth metrics. Star ratings align with user expectations for feedback (page 4, satisfaction question).
- Operator Goals: Network operators (primary stakeholders, Task 2) require signal, bandwidth, and jitter data for optimization, making these must-haves.
- Privacy and Comfort: Location tracking (high influence, page 6) and data deletion (page 7) address user reluctance, ranked as should-haves to balance privacy.
- Lower Priorities: Suggestions like scam detection or free data lack QoE relevance, reducing their importance.

➢ **Feasibility**:

- Technical: Signal strength, bandwidth, jitter, and star ratings use standard Android APIs, fitting a 3-6-month timeline (Task 1, basic app). Location tracking requires permission APIs but is manageable. Scam detection or real-time notifications demand advanced skills (9-12 months, Task 1).
- Resources: Our team's moderate experience supports simple UI and weekly prompts but not complex features like iOS support or global access.
- MoSCoW Method: Must-haves (signal, internet, ratings) are high-impact and easy to implement. Should-haves (location, deletion) are valuable but need extra effort. Could-haves (prompts, UI) enhance usability but are secondary. Won't-haves are infeasible or irrelevant.

II. Impact on QoE App Development:

The prioritized requirements ensure the QoE app is:

- User-Centered: Focuses on top user issues (58.5% call breakages, 47.2% signal), enhancing satisfaction.
- Operator-Aligned: Delivers metrics (signal, bandwidth) for network optimization.
- Feasible: Fits the team's skills and timeline, using Android APIs for core features.
- Reluctance-Sensitive: Includes consent for location and data deletion to address user concerns (Task 2, page 8).
- SRS-Ready: Provides a clear feature list (e.g., "Must track signal strength") for the SRS, streamlining Task 3's next steps.

Prioritizing requirements based on importance and feasibility has refined our QoE app's focus, ensuring it addresses critical user needs (e.g., 47.2% signal issues) and operator goals while remaining achievable within our Android focused, 3-6-month project. The MoSCoW method ranked signal strength, slow internet, call breakages, and star ratings as must-haves, with location tracking and data deletion as should-haves, guiding a practical SRS. This prioritization minimizes scope creep, optimizes resources, and sets a foundation for a successful app that enhances network quality feedback for stakeholders.

4. classification of requirements:

To better organize and understand the system's expectations, the gathered requirements were categorized into functional and non-functional types, helping to define what the system should do and how it should perform.

A. Functional Requirements:

Functional requirements Functional Requirement that define what the system must do. These are based on the needs of stakeholders and the feedback collected from users.

1. User Registration and Authentication:

- Users should be able to register and log in securely.
- Include a feature to reset forgotten passwords.

2. Network Performance Monitoring:

- The app should collect and display network metrics such as:
  - Latency
  - Packet loss
  - Bandwidth
- Allow users to view their network performance history.

3. Feedback Collection:

- Provide a feature for users to submit feedback on network issues.
- Enable users to rate their network service providers.
- Enable users to choose the rate at which feedback are been asked.

4. Real-Time Notifications:

- Notify users of network issues in their area.
- Send alerts for improvements or updates on network performance.

5. ISP Integration:

- Link the app to Internet Service Providers (ISPs) to enable real-time feedback and data sharing.

6. Data Usage Tracking:

- Provide a feature to monitor data usage by other apps.
- Display network traffic information.

7. Cross-Platform Availability:

- Ensure the app is accessible on both Android and IOS platforms.

8. Offline Mode:

- Allow users to access basic app functionalities without an internet connection.
-Still to precise those app functionalities

9. Customizable User Settings:

- Provide options to enable or disable features like notifications, background data collection, and location tracking.

B. Non-Functional Requirements:

Non-functional requirements focus on the system's performance, usability, and other quality aspects.

1. Performance:

- The app should consume minimal battery power while running in the background.
- Ensure the app performs well even with large amounts of data.

2. Scalability:

- The system should handle an increasing number of users and data without degradation in performance.

3. Compatibility:

- The app should be compatible with different devices, operating systems, and screen sizes.
- Ensure support for older Android and IOS versions.

4. Security:

- Implement secure data storage and transmission (e.g., encryption).

- Ensure user privacy by not storing unnecessary personal data.

5. Reliability:

- The app should remain stable and functional even during heavy network use.

6. Data Storage and Accessibility:

- Minimize storage requirements on user devices.

- Store data securely in the cloud for easy retrieval.

7. Response Time:

- Provide near-instant responses for real-time notifications and updates.

C. User Experience (UX) Requirements:

User experience requirements focus on ensuring the app is intuitive, accessible, and enjoyable to use.

1. User Interface (UI):

- Provide a clean, simple, and intuitive design.

- Use minimalistic layouts with clear navigation.

2. Ease of Use:

- Simplify complex terms (e.g., latency) for non-technical users.

- Include tutorials to guide users.

3. Accessibility:

- Ensure the app is lightweight and does not occupy much device storage or RAM.

- Design for accessibility, including support for visually impaired users (e.g., screen reader compatibility).

4. Customizable Notifications:

- Allow users to customize notification preferences (frequency, type, etc.).

5. Feedback-Friendly Design:

- Simplify the process of submitting feedback.

- Use interactive elements, such as sliders and buttons, for user ratings.

6. Battery Efficiency:

- Optimize the app to minimize battery usage during background operations.

7. Real-Time Updates:

- Ensure real-time performance tracking and issue notifications.

- Provide live updates on network metrics.

8. Multi-Language Support:

- Include options for multiple languages to cater to diverse users.

9. Error Handling:

- Provide clear and helpful error messages when issues occur.

10. Engagement Features:

- Add optional features like weekly free data donations or gamified elements (e.g., rewards for feedback submissions).

By addressing these functional, non-functional, and user experience requirements, the app can meet user expectations, improve network performance, and encourage widespread of the mobile app.

5. **Software Requirements Specification:**

Based on the refined and validated requirements, a structured Software Requirement Specification (SRS) outline was developed to guide the system's design and development process.

Purpose: A mobile application for collecting real-time data from mobile subscribers for better network performance insights.

Scope: Its scope revolves around collecting device/network parameters and user feedback, uploaded to a centralized backend.

I. Overall Description:

Product Perspective: Standalone app with backend for data storage and analytics.

Product Functions: Passive network monitoring, user feedback, location logging, secure data upload.

User Classes: Subscribers, Admins.

Operating Environment: Android OS 8.0, Linux-based backend with REST API.

Constraints: Limited OS access, minimal battery/data use, privacy law compliance.

Assumptions: Users grant permissions and have internet access.

II. System Features and Requirements:

- Passive Monitoring Module:
  - Collect signal strength periodically.
  - Log network type (3G/4G/5G).
  - Measure latency and jitter via ping.

- Active Feedback Module:

- Prompt users for feedback every 24 hours if no network issue and every 12 hours if network issues.
- Allow 1-5-star ratings.
- Log feedback with timestamp and location.

- Data Storage and Transmission:

- Store logs locally before upload.
- Upload logs every 60 minutes.
  Encrypt data before transmission.

- Permissions and Privacy:

- Request permission for sensitive data.
- Let users opt out of metrics.
- Anonymize user data.

III. External Interface Requirements:

- **User Interfaces**: Simple UI, feedback screen, and settings.
- **Software Interfaces**: Android APIs & IOS API
- **Communication Interfaces**: REST API over HTTPS, JSON data format.

IV. Non-Functional Requirements:

- Performance: CPU usage under 5% when idle.
- Security: Encrypted data transmission.
- Usability: Feedback submission under 30 seconds.
- Reliability: Auto-restart on crash.
- Battery: Optimized background performance.
- Response time: provide quick responses.
- Data storage and accessibility
- Scalability

6. **Validation of Requirements with Stakeholders :**

To ensure that the documented requirements accurately reflect stakeholder expectations are feasible, and can be implemented without misunderstandings. This is a crucial phase to avoid costly changes later in the software development life cycle Steps

steps 1: Identify Stakeholders;

List all key stakeholders involved in or affected by the system. These may include:

- Mobile Network Operator (MNO)
- End users (e.g., mobile app users)
- Software development team (developers, UI/UX designers)

- Technical Stakeholders: App Developers and Network Engineers

Step 2: Schedule a Validation Session;

Set up a requirement review meeting or validation workshop with all stakeholders.

- Tools used: In-person meetings.
- Material shared: Software Requirements Specification (SRS), Use case diagrams, and User stories
- Present each requirement
- Collect stakeholder feedback
- Validate the feasibility and relevance

Step 3: Present and Discuss Requirements:

Used the SRS document to walk stakeholders through:

- Functional Requirements: What the system should do
- Non-Functional Requirements: Performance, security, usability, etc.
- Constraints and Assumptions

Technique Used:

- Requirement Walkthrough: We explained each requirement. Stakeholders can raise issues or ask questions.

Step 4: Collect and Document Feedback:

We created a feedback log or validation matrix to record stakeholder input.

Step 5: Update the SRS Document:

Based on the validated feedback, we

- Modified unclear or incorrect requirements
- Added new requirements were requested
- Removed irrelevant or redundant ones
- Highlighted all changes using version control or change tracking

Step 6: Obtain Formal Approval:

After revisions:

- We shared the updated SRS with stakeholders.

Design and implementation of a mobile app for collection of user experience data for mobile network subscribers.

Final Deliverables:

1.Validated SRS Document (version-controlled, updated with stakeholder feedback)

2.Feedback Log / Validation Matrix.

Validating requirements with stakeholders ensures alignment between developers and users, reduces the risk of project failure, and increases satisfaction. It is a collaborative activity that builds trust and shared understanding.

Conclusion:

Through this task, we successfully transformed user feedback and survey insights into structured, validated, and actionable requirements. This sets the stage for effective system modelling and design.

GROUP 20 MEMBERS:

Name	Matricule
TIOKENG SAMUEL	**FE19A110**
KUE KOUOKAM GILLES BRYTON	**FE22A235**
LUM BLESSING NFORBE	**FE22A239**
NWETBE NJIWUNG LORDWILL	**FE22A285**
TALLA TIZA AGYNUI	**FE22A304**