**UNIVERSITY OF BUEA**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CEF440: Internet Programming and Mobile Programming**

**COURSE INSTUCTOR: Dr. Valery Nkemeni**

# TASK 3: REQUIREMENT ANALYSIS

**BY GROUP 3**

|  |  |
| --- | --- |
| **NAMES** | **MATRICULE** |
| 1. **ABONGWA CALEB NFORNGWA** | **FE22A133** |
| 1. **BEKONO AKONO MARTHE LUNYA LAWRICA** | **FE22A171** |
| 1. **MUKETE ATIA HANS MASANGO** | **FE22A250** |
| 1. **NGUIMFACK AZAFACK JOREL** | **FE22A267** |

**Academic Year (2024/2025)**

**TABLE OF CONTENTS**

[INTRODUCTION 1](#_Toc3341)

[1. Review and Analysis of Requirements 1](#_Toc17434)

[2. Identified Inconsistencies, Ambiguities, and Missing Information 2](#_Toc12938)

[3. Prioritization of Requirements 2](#_Toc20576)

[4. Classification of Requirements 3](#_Toc3994)

[Summary 3](#_Toc19245)

**Introduction**

**Requirement Analysis** is a critical phase in the software development lifecycle that involves reviewing, interpreting, and refining the requirements gathered during the earlier stages. The goal is to ensure that all stakeholder needs are clearly understood, well-documented, and feasible for implementation. This task acts as a bridge between gathering raw requirements and designing a viable solution.

During requirement analysis, each requirement is evaluated for **completeness, clarity, consistency, technical feasibility, and dependencies**. Any ambiguities, contradictions, or missing information are identified and resolved. This process also includes classifying the requirements into **functional** (what the system should do) and **non-functional** (how the system should perform), and prioritizing them based on their importance and implementation effort.

Requirement analysis is essential because it:

* **Reduces misunderstandings and project risks** early on.
* **Helps in building realistic timelines and budgets.**
* **Ensures all stakeholders have a shared understanding** of what the system is intended to do.
* **Serves as a solid foundation** for system design, development, and testing.

Ultimately, thorough requirement analysis contributes to the success of the project by ensuring that the final software product meets user expectations and performs reliably in real-world scenarios.

## 1. Review and Analysis of Requirements

|  |  |
| --- | --- |
| Criterion | Observations |
| Completeness | Requirements cover most user and system needs: real-time monitoring, feedback collection, privacy controls, alerts, multilingual support, etc. However, deeper backend analytics and admin panel needs are not specified. |
| Clarity | Functional requirements are fairly clear. Some non-functional goals like "high reliability" could be more precisely measured (e.g., 99.9% uptime). |
| Technical Feasibility | Technically feasible with current mobile development technologies (Android/iOS SDKs, backend servers, analytics platforms). Privacy concerns need careful implementation to comply with laws. |
| Dependency Relationships | Several features depend on others: e.g., privacy settings must be in place before geolocation feedback can be safely collected; the notification system depends on real-time monitoring availability. |

## 2. Identified Inconsistencies, Ambiguities, and Missing Information

|  |  |
| --- | --- |
| Issue Type | Description |
| Ambiguity | "Minimal CPU and battery usage" is subjective , could be clearer (e.g., use no more than 2% battery per hour). |
| Missing Information | No detailed description of server/backend requirements for storing and processing the collected data. |
| Inconsistency | Field interviews suggest MTN is reluctant to share data with third parties, but user feedback submission in the app suggests external reporting without clear MTN integration strategy. |
| Ambiguity | No explanation of how anonymous data vs. identifiable data will be handled technically. |
| Missing Information | No explicit mention of testing plans (e.g., beta testing phase, feedback collection post-launch). |
| Dependency Concern | Offline caching requires proper synchronization strategy when users regain connectivity — not clearly defined. |

## 3. Prioritization of Requirements

|  |  |
| --- | --- |
| Priority Level | Requirements |
| High Priority | Real-time network speed monitoring, privacy and data control settings, network quality alerts, user feedback submission, low battery consumption, security measures. |
| Medium Priority | Service history logs, multilingual interface (English and French), offline data caching. |
| Low Priority | Support integration with third-party systems (e.g., MTN Zigi), loyalty/reward systems, community impact dashboard, regional signal benchmarking. |

## 4. Classification of Requirements

|  |  |
| --- | --- |
| Type | Requirements |
| Functional Requirements | - Real-time speed monitoring - User-submitted feedback - Network quality alerts - History and logs of issues - Multilingual support (English/French) - Support link to MTN Zigi/chatbots - Privacy control settings - Adaptive notification settings |
| Non-Functional Requirements | - Minimal CPU/battery usage - High reliability (near 100% uptime) - Data security (encryption, anonymization) - App scalability (thousands of users) - Usability (simple, intuitive UI) - Maintainability (easy updates) - Compatibility (Android 8+, iOS 12+) - Localization (urban vs rural optimization) |

## Summary

* The gathered requirements are generally complete and feasible, but some areas like backend design and data handling policies, need further detailing.
* We identified ambiguities around resource usage, offline synchronization, and data privacy handling.
* Requirements were prioritized based on how crucial they are to user satisfaction and technical implementation difficulty.
* A clear classification into functional and non-functional requirements was done to guide development focus.