May 5, 2025

UNIVERSITY OF BUEA
Buea, South West Region
Cameroon
P.O. Box 63,
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

NAME	MATRICULE
BILLA SOPHIA	FE22A176
EKANE METUGE AKAME FAVOUR	FE22A199
EYONG GODWILL NGANG	FE22A214
NEBOTA ISMAEL OWAMBA	FE22A256
ONYA MARTHA . O	FE22A292

Task 3

Course Master**:**
Dr. Nkemeni Valery
2024/2025

AuraCheck

REQUIREMENT ANALYSIS REPORT

Mobile-Based Attendance Management System using Geofencing and Facial Recognition

Group: 24
CEF440 - Internet Programming and Mobile Programming
Version: 1.0

TABLE OF CONTENTS

..... 0

1. Introduction 2

1.1 Purpose of Analysis 2

1.2 Scope of Analysis 2

1.3 Document Structure 2

2. Review and Analysis of Gathered Requirements 2

2.1 Completeness Assessment 2

2.2 Clarity Assessment 3

2.3 Technical Feasibility Assessment 4

2.4 Dependency Analysis 5

3. Identification of Issues from Gathered Requirements 6

3.1 Inconsistencies 6

3.2 Ambiguities 6

3.3 Missing Information 7

4. Requirement Prioritization Approach 7

5. Requirement Validation Strategy 8

6. Conclusion 9

1. Introduction

1.1 Purpose of Analysis

The purpose of this Requirement Analysis Report is to systematically review, evaluate, and refine the requirements gathered for the Mobile-Based Attendance Management System project. This analysis aims to ensure the requirements are complete, clear, consistent, technically feasible, and accurately reflect the needs of all stakeholders (students, instructors, administrators, IT department). It serves as a critical input for developing a robust Software Requirements Specification (SRS).

1.2 Scope of Analysis

This analysis covers the requirements gathered during the initial phases of the project (as documented in Task 1 and Task 2 reports, and the project description. It examines functional and non-functional aspects, including user interactions, system features (facial recognition, geofencing), performance expectations, security considerations, and technological constraints associated with developing a mobile attendance application using Flutter for the University of Buea context.

1.3 Document Structure

This document is organized as follows:

- Section 1: Introduction - Outlines the purpose, scope, and structure.
- Section 2: Review and Analysis - Assesses gathered requirements against quality criteria (completeness, clarity, feasibility, dependencies).
- Section 3: Identification of Issues - Details inconsistencies, ambiguities, and missing information found during the review.
- Section 4: Prioritization Approach - Describes the methodology used to prioritize requirements.
- Section 5: Validation Strategy - Outlines the plan for validating requirements with stakeholders.
- Section 6: Conclusion - Summarizes the findings of the analysis.

2. Review and Analysis of Gathered Requirements

This section assesses the requirements gathered from project documentation (Tasks 1 & 2) based on key quality attributes.

2.1 Completeness Assessment

- **Strengths:**
    - Core functionalities (facial recognition check-in, geofencing validation, instructor real-time view, student history access) are identified.
    - Key stakeholders (Students, Instructors, Administrators, IT Department) are listed.
    - Initial technology stack suggestions (Flutter, Google ML Kit, Geolocator plugin, Node.js/PostgreSQL backend) are present.

    - Key performance goals (e.g., <5 second check-in time) are stated.
    - User roles are defined with basic needs outlined.
- **Gaps/Areas Requiring Enhancement:**
    - **Error Handling:** Lack of detailed specifications for handling various error conditions (e.g., GPS permission denied, network loss during check-in, repeated facial recognition failures, server errors).
    - **Administrative Functions:** Requirements for administrative tasks beyond basic user/course management are underdeveloped (e.g., detailed audit logging, system configuration options, bulk data operations, system monitoring tools).
    - **Data Management:** Specific requirements for data backup frequency, recovery procedures (RPO/RTO), and data retention policies are missing.
    - **Security Details:** Security requirements need more depth beyond general statements (e.g., specific encryption algorithms, API security measures like rate limiting, handling of security vulnerabilities in dependencies).
    - **Scalability:** Quantitative requirements regarding the expected number of concurrent users, total user base, and data volume growth are absent.
    - **Maintenance & Updates:** Requirements related to system updates, patching, and long-term maintenance procedures are not specified.
    - **Edge Cases:** Handling of complex scenarios like students enrolled in conflicting/overlapping classes, temporary classroom changes affecting geofences, or significant user appearance changes needs explicit definition.
    - **Manual Override Process:** The exact workflow, logging requirements, and potential limitations for instructor manual overrides need clarification.
    - **Accessibility:** Specific accessibility standards or requirements are not mentioned.
- **Conclusion on Completeness:** The gathered requirements form a solid starting point but require significant elaboration, particularly in non-functional areas (security, scalability, reliability, maintainability), administrative features, detailed error handling, and edge case management, to achieve the level of completeness needed for robust system development.

2.2 Clarity Assessment

- **Strengths:**
    - The fundamental concepts of using facial recognition for identity and geofencing for location are clearly stated as the core premise.
    - Distinctions between user roles (Student, Instructor, Admin) are generally clear.
    - The primary performance target (<5s check-in) is explicitly stated.
- **Ambiguities Identified (See Section 3.2 for details):**
    - Terms like "real-time," "secure storage," "user-friendly," and "high accuracy" lack precise, measurable definitions.
    - The exact boundary conditions and tolerance for geofencing require clarification (e.g., handling GPS drift near the boundary).
    - The step-by-step user workflow for critical processes like enrollment and check-in needs more detailed description to avoid misinterpretation.
    - The scope and limitations of features like "manual override" need unambiguous definition.

    - The conditions under which the <5s check-in target applies (e.g., network quality, device type) are not specified.
- **Conclusion on Clarity:** While the high-level objectives are understood, several key requirements suffer from ambiguity. Resolving these ambiguities by providing precise definitions, measurable criteria, and detailed process descriptions is essential before proceeding to design and implementation.

2.3 Technical Feasibility Assessment

- **Facial Recognition (ML Kit):**
    - *Feasibility:* High. Libraries like Google ML Kit provide pre-trained models capable of running efficiently on modern smartphones. On-device processing is feasible and advantageous for privacy and speed.
    - *Challenges:* Performance variability across different device hardware; sensitivity to lighting conditions, face angles, occlusions (masks, glasses); potential for spoofing attacks (necessitating liveness detection). Accuracy tuning is required.
- **Geofencing (GPS/Location Services):**
    - *Feasibility:* High. Standard OS location services and plugins (e.g., Flutter `geolocator`) provide the necessary functionality.
    - *Challenges:* Significant accuracy limitations indoors where GPS signal is weak or unavailable; battery consumption associated with continuous location tracking (if implemented); dependency on user granting location permissions; defining appropriate geofence radii to balance accuracy and usability.
- **Cross-Platform Development (Flutter):**
    - *Feasibility:* High. Flutter is a mature framework well-suited for building visually consistent apps on Android and iOS with good performance and access to native features.
    - *Challenges:* Ensuring consistent behavior and performance across diverse device models and OS versions; managing platform-specific dependencies or workarounds if needed.
- **Backend System (Node.js/Express/PostgreSQL suggested):**
    - *Feasibility:* High. These are standard, robust technologies for building scalable web APIs and managing relational data.
    - *Challenges:* Requires proper architectural design for scalability and security; database schema design; API security implementation.
- **Integration:**
    - *Feasibility:* High. Integrating camera, location services, ML libraries, and backend APIs is standard mobile development practice.
    - *Challenges:* Requires careful handling of permissions, asynchronous operations, network communication, and error states across different components.
- **Performance Constraint (<5s Check-in):**
    - *Feasibility:* Moderate to High, but Challenging. Achieving this consistently requires optimizing each step: fast location acquisition, efficient on-device facial recognition, minimal network latency for confirmation/sync. Depends heavily on device capabilities and network conditions. Rigorous testing and optimization are mandatory.
- **Conclusion on Feasibility:** The core technologies required for the system are readily available and technically feasible to implement. However, significant challenges exist

concerning indoor location accuracy, ensuring robust facial recognition under varied conditions, achieving the strict performance target consistently, and mitigating security risks (especially spoofing). These challenges require careful consideration during design and thorough testing.

2.4 Dependency Analysis

- **Core Check-in Process:** This critical function has a chain of dependencies:
    1. User Authentication (Login Status)
    2. Active Class Session Identification
    3. Availability of Correct Geofence Definition for the Session
    4. Device Location Services Enabled & Permission Granted
    5. Sufficiently Accurate Location Fix Obtained
    6. Calculated Location Falls Within Geofence Boundary
    7. Device Camera Functional & Permission Granted
    8. Successful Facial Feature Extraction
    9. Successful Match Against Stored Template
    10. Network Connectivity (for real-time sync) OR Local Storage Capacity (for offline mode)
    - *Implication:* Failure at any mandatory step prevents successful check-in. The system must handle failures at each point gracefully.
- **System Components:**
    - Mobile App depends on the Backend API for data (user profiles, geofences, templates, history) and validation logic (template comparison).
    - Backend API depends on the Database for data persistence and retrieval.
    - Mobile App depends on OS Services (Camera, Location).
    - Mobile App depends on Third-Party Libraries (Flutter SDK, ML Kit, Geolocator).
- **Data Dependencies:**
    - Attendance records depend on accurate User, Course, and Session data.
    - Geofence validation depends on accurate classroom coordinate data configured by Admins.
    - Facial recognition depends on successfully enrolled facial templates.
- **Operational Dependencies:**
    - System usability depends on users having compatible devices and granting permissions.
    - Real-time functionality depends on reliable network connectivity on campus.
    - System administration (user/course/geofence setup) must be performed correctly for the system to function.
- **Conclusion on Dependencies:** The system exhibits significant dependencies, particularly within the check-in workflow. Robust error handling, clear user guidance (e.g., for permissions), and thorough integration testing are essential to manage these dependencies effectively. The reliance on external factors like network quality and GPS accuracy highlights the need for fallback mechanisms (offline mode) and realistic performance expectations.

3. Identification of Issues from Gathered Requirements

This section consolidates the inconsistencies, ambiguities, and missing information identified during the review process.

3.1 Inconsistencies

1. **Geofence Enforcement Rule:** Potential conflict noted in Task 3 draft regarding whether check-in is *always* blocked outside the geofence.
    - *Resolution:* Clarified in SRS (FR.4.4, FR.4.5) – Geofence validation is a strict prerequisite. Check-in is denied if outside.
2. **Facial Recognition "Training":** Task 3 draft mentioned the system being "trained" for facial changes, conflicting with the typical use of pre-trained ML Kit models.
    - *Resolution:* Assumed standard ML Kit usage (recognition, not continuous training). Significant appearance changes require user re-enrollment (SRS FR.2.7). If custom training/fine-tuning is desired, it constitutes a major scope change requiring new requirements.

3.2 Ambiguities

1. **"Real-time" Updates:** The exact mechanism and latency for updating the instructor's attendance view are unclear.
    - *Resolution Needed:* Specify if WebSockets (instant) or polling (near real-time) will be used and define the target update latency (e.g., < 5 seconds).
2. **"Secure Storage":** Lacks specifics on encryption algorithms, key management, and protection levels for biometric templates.
    - *Resolution Needed:* Define specific standards (e.g., AES-256 encryption at rest) in the SRS/Design.
3. **"User-friendly":** Subjective term needing measurable criteria.
    - *Resolution Needed:* Define specific usability goals (e.g., task completion success rate > 95% in testing, System Usability Scale score > 75).
4. **Geofence Boundary Handling:** Behavior at the exact boundary or with GPS inaccuracies is undefined.
    - *Resolution Needed:* Specify tolerance/buffer (e.g., allow check-in if within radius + 5 meters) and note dependency on GPS quality. Requires empirical tuning.
5. **"<5s Check-in" Conditions:** The context (network, device) for this target is missing.
    - *Resolution Needed:* Specify target as an average under defined "typical conditions" (e.g., specific network bandwidth/latency, mid-range device).
6. **Repeated Failure Handling:** The consequence of multiple failed check-in attempts (face or geo) is not defined.
    - *Resolution Needed:* Define policy (e.g., lockout after N attempts, require manual intervention).
7. **Manual Override Details:** Bidirectionality (Present<->Absent) and the exact workflow/logging need clarification.
    - *Resolution Needed:* Specify capabilities and mandatory justification logging.
8. **"High Accuracy" (Face/Geo):** Lacks specific, measurable targets (e.g., TAR/FAR percentages for face recognition, distance tolerance for geofencing).
    - *Resolution Needed:* Define quantitative accuracy targets in NFRs.

3.3 Missing Information

1. **Detailed Error Scenarios & Messages:** Specific handling and user messages for failures like permission denial, network timeout, server errors, no GPS fix, camera unavailable, etc.
2. **Audit Logging Requirements:** What specific events need to be logged, what data included in logs, log retention period, log security measures.
3. **Backup and Recovery Plan:** Specific RPO, RTO, backup frequency, storage location, recovery testing procedures.
4. **Scalability Targets:** Concrete numbers for peak concurrent users, total users over time, expected data volume/growth rate.
5. **Liveness Detection Specifics:** If required (recommended), the specific method and expected effectiveness need definition.
6. **Accessibility Standards:** Requirement for compliance with specific standards (e.g., WCAG AA for mobile).
7. **Deployment and Update Strategy:** How the app and backend will be initially deployed and subsequently updated/patched.
8. **Data Retention Policies:** Specific durations for retaining different data types (attendance logs, audit trails, inactive user accounts, biometric templates after user departure).
9. **System Configuration Details:** Which parameters should be configurable by administrators (e.g., thresholds, timeouts, radii).
10. **Third-Party Library Management:** Policy for selecting, vetting, and updating external libraries/SDKs.

4. Requirement Prioritization Approach

To manage scope and focus development efforts, requirements identified will be prioritized using the **MoSCoW** method:

- **Must Have (P1):** Requirements essential for the system to meet its core objectives and be considered functional. Without these, the system fails its primary purpose (e.g., facial check-in, geofence validation, basic viewing). These form the Minimum Viable Product (MVP).
- **Should Have (P2):** Important requirements that add significant value, usability, or robustness but are not strictly essential for the initial launch. The system can function without them, but would be less effective or complete (e.g., offline mode, detailed reporting, manual override).
- **Could Have (P3):** Desirable requirements that are "nice to have" and would improve the system further but have lower impact or higher implementation cost/complexity. These are typically addressed if time and resources permit after P1 and P2 items are complete (e.g., advanced analytics, push notifications, accessibility enhancements).
- **Won't Have (this version):** Requirements explicitly excluded from the scope of the current project iteration, potentially to be considered in future versions (e.g., ERP integration, parental portal).

Prioritization will be based on a combination of:

- **Business Value:** How critical is the requirement to achieving the project goals (accuracy, efficiency, security)?
- **Stakeholder Needs:** How important is the requirement to key user groups (students, instructors, admins)?
- **Technical Feasibility & Risk:** How complex or risky is the implementation?
- **Dependencies:** Does implementing this requirement enable other critical features?

The prioritized list will guide the development roadmap and release planning.

5. Requirement Validation Strategy

A multi-faceted approach will be used to validate the requirements documented in the SRS, ensuring they are correct, complete, unambiguous, consistent, and feasible:

1. **Stakeholder Reviews:**
    - **Target Audience:** Course Master (Dr. Nkemeni), representative students, instructors, administrators, and potentially IT staff.
    - **Method:** Circulate the SRS document for review. Conduct focused review meetings or workshops tailored to each stakeholder group to discuss requirements relevant to them. Use questionnaires or feedback forms to gather structured input.
    - **Focus:** Correctness (Does it meet the need?), Completeness (Is anything missing?), Clarity (Is it understandable?), Consistency (Do requirements contradict?).
2. **Prototyping and Usability Testing:**
    - **Method:** Develop interactive prototypes (low or high fidelity) demonstrating key user workflows (e.g., enrollment, check-in, attendance monitoring).
    - **Target Audience:** Representative end-users (students, instructors).
    - **Focus:** Validate UI/UX requirements, workflow logic, ease of use, clarity of instructions and feedback. Identify usability issues early.
3. **Technical Reviews:**
    - **Target Audience:** Development team members, technical lead/architect.
    - **Method:** Conduct internal reviews or walkthroughs of the requirements.
    - **Focus:** Assess technical feasibility, identify implementation risks or challenges, evaluate dependencies, check for technical inconsistencies or ambiguities.
4. **Requirements Traceability:**
    - **Method:** Create and maintain a Requirements Traceability Matrix (RTM).
    - **Focus:** Ensure each requirement has a source (origin), is linked to corresponding design elements, code components, and test cases. This helps verify that all requirements are addressed and tested.
5. **Formal Inspections/Walkthroughs:**
    - **Method:** Conduct structured walkthroughs of the SRS document involving key stakeholders and the development team. Assign roles (moderator, presenter, scribe, reviewers).
    - **Focus:** Systematically examine the SRS section by section to identify defects (ambiguities, inconsistencies, errors, omissions) before baselining.
    -

6. **Feedback Management:**
    - **Method:** Log all feedback received during validation activities. Discuss and resolve issues. Update the SRS document accordingly, maintaining version control.
    - **Focus:** Ensure that the validation process leads to tangible improvements in the quality of the requirements specification.

This comprehensive validation strategy aims to minimize errors and misunderstandings early in the lifecycle, increasing the likelihood of project success.

6. Conclusion

This analysis indicates that while the foundational requirements for the Mobile-Based Attendance Management System have been gathered, significant refinement is needed. Key areas requiring attention include adding detail to non-functional requirements (especially security, scalability, reliability), resolving identified ambiguities, filling information gaps related to error handling and administration, and carefully managing technical challenges (indoor GPS, facial recognition robustness, performance). The proposed prioritization and validation strategies provide a framework for addressing these issues and developing a high-quality SRS that will effectively guide the subsequent design and development phases.