UNIVERSITY OF BUEA **REPUBLIC OF CAMEROON**
P.O Box 63, PEACE – WORK - FATHERLAND
Buea_South West Region
Cameroon
Tel: (+237) 674354327
Fax: (+237) 3332 22 72

FACULTY OF ENGINEERING AND TECHNOLOGY
COMPUTER ENGINEERING DEPARTMENT
CEF 440: INTERNET PROGRAMMING (J2EE) AND MOBILE PROGRAMMING

TASK 2: SYSTEM MODELING AND DESIGN FOR A BIOMETRIC STUDENT ATTENDNCE SYSTEM

SN	NAMES	MATRICULE
1	NGULEFAC JERRY MBUOH	FE21A265
2	NEBA PRINCEWILL AMBE	FE21A251
3	NKENGBEZA DERICK	FE21A277
4	KAH JOSPEN NGUM	FE21A207
5	NYOCHENBENG ENZO NKENGAFACK	FE21A293

COURSE INSTRUCTOR: NKEMENI VALERY

Academic year 2023/2024

Presented by Group 11

Contents

List Of Figures

List of Tables

1. Introduction

1.1. Scope

The system is required to support mobile application development, biometric authentication integration, real-time attendance tracking, scalability, customization, automated attendance reporting, and notifications. Stakeholders include administrators, instructors, students, and the IT department. Key requirements include biometric data capture, enrollment and authentication modules, real-time attendance tracking, comprehensive reporting, and notifications.

The system modeling and design of a biometric student attendance system involve creating an advanced, secure, and efficient method for recording and managing student attendance in educational institutions. Utilizing biometric technology, such as fingerprint recognition, this system ensures accurate identification of students, thereby eliminating the issues of proxy attendance and manual errors. The design phase incorporates hardware components, including biometric scanners and a central processing unit, as well as software for data management and user interfaces. The system also features real-time data processing. Robust security protocols are implemented to protect sensitive biometric data. Additionally, the system is designed to be user-friendly, ensuring seamless adoption by both students and administrative staff. Overall, the biometric attendance system enhances accountability, streamlines administrative tasks, and improves overall attendance tracking efficiency.

1.2. Overview

A biometric student attendance system is a modern solution designed to streamline and enhance the process of tracking student attendance in educational institutions. This system leverages biometric technology, such as fingerprint for scanning, to accurately and uniquely identify each student. The core components of the system include biometric sensors for capturing student data, a database for storing and managing attendance records, and software for processing and reporting attendance information.

2. System Architecture

2.1. Architectural Design

2.1.1. Architecture Overview

Below outlines the proposed system architecture for a mobile attendance tracking application with biometric authentication. This application aims to provide real-time attendance tracking, scalability, customization, and automated reporting features while ensuring security and user-friendliness. A multi-tier architecture with microservice features like modularity would be most suitable

- **Presentation Tier (Client-side)**
  - **Mobile Application:** Developed using React Native for cross-platform compatibility (iOS and Android).
  - **UI Components:** User interfaces for students, instructors, and administrators.
  - **Biometric Authentication**: Integrates with device's biometric sensors (fingerprint, facial recognition).
  - **Real-time Attendance Tracking:** Interface to mark and view attendance.

    - **Notifications:** Handles push, in-app, email, and SMS notifications.
    - **Devices:** Mobile phones, laptops, and devices with fingerprint sensors.
- **Application Tier (Server-side Logic)**
    - **Backend Server:** Built with Node.js and Express.js.
        - **Authentication Module:** Uses Auth0 for secure login and biometric authentication.
        - **Attendance Tracking Module:** Handles real-time attendance marking and GPS verification.
        - **Notification Module:** Manages the sending of notifications and alerts.
        - **Reporting Module:** Generates and exports attendance reports.
        - **User Management:** Secure login, access control, and account management.
    - **APIs:**
        - **Biometric Data Capture API**: Interfaces for capturing and validating biometric data.
        - **GPS Verification API:** Integrates with Google Maps for location-based verification.
        - **Email Service API**: Node-mailer or emailjs for sending email notifications.
        - **Data Encryption API**: Ensures secure data transmission and storage.
- **Data Tier (Database and Storage)**
    - **Database**: Cloud-based MongoDB for scalability and flexibility.
    - **Attendance Data:** Stores real-time attendance records.
    - **User Data:** Stores user profiles, biometric templates, and authentication logs.
    - **Notification Data:** Stores notification and alert logs.
    - **Storage:** Cloud storage for large data sets and backups.
- **Security:**
    - **Data Security**: Encryption of data at rest and in transit
    - **Access Control**: Strict access control policies using RBAC
    - **Secure Communication:** HTTPS protocol for secure communication

2.1.2. High-Level System Architecture Diagram

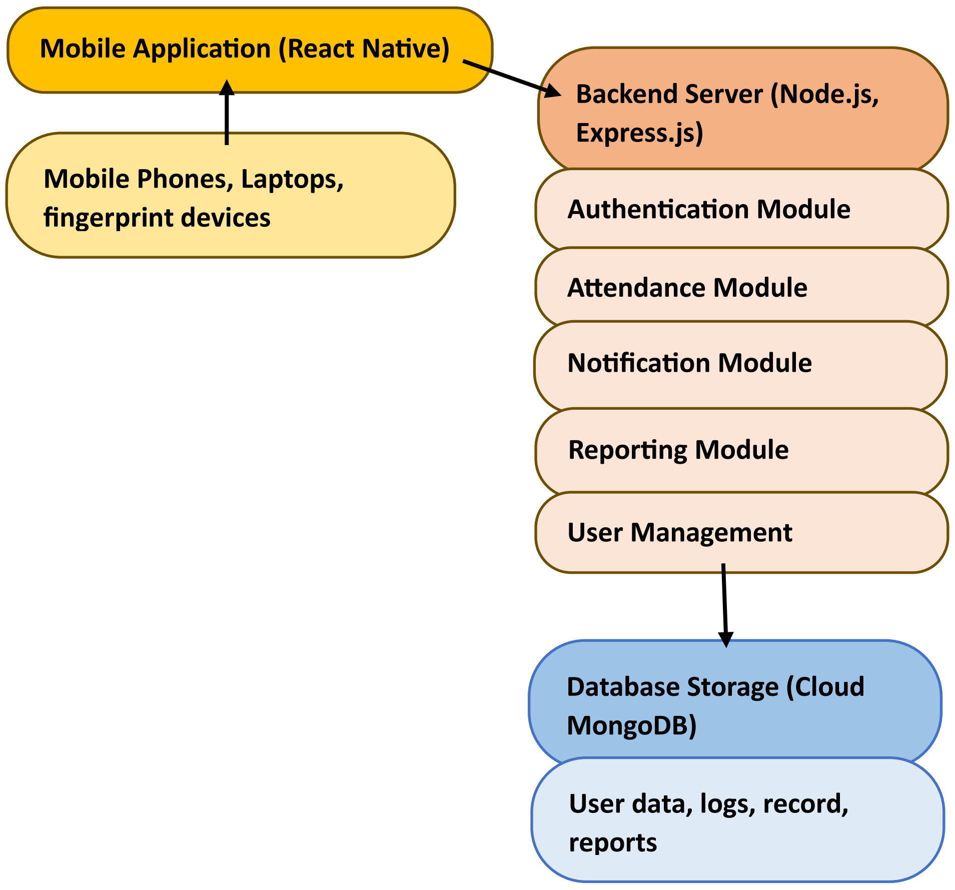
Mobile Application (React Native)
Mobile Phones, Laptops, fingerprint devices
Backend Server (Node.js, Express.js)
Authentication Module
Attendance Module
Notification Module
Reporting Module
User Management
Database Storage (Cloud MongoDB)
User data, logs, record, reports

Figure 1: Architecture Diagram

System Workflow

- **User Authentication:** Users log in via the mobile app using credentials or biometrics, authenticated by Auth0.
- **Initiating Attendance:** Instructors initiate attendance sessions, verified by GPS.
- **Marking Attendance:** Students mark attendance using biometrics, recorded in real-time with GPS data.
- **Notifications and Alerts:** System sends various notifications for attendance sessions.
- **Reporting:** Administrators generate and export detailed attendance reports.
- **User Management and Customization:** Users manage their profiles and settings, and admins configure security and privacy settings.

2.1.3. Components and their Interactions

- **Client Devices (Students & Instructors)**
    - **Biometric Capture:** Captures biometric data (e.g., fingerprints).
    - **GPS Verification:** Verifies the location during attendance marking.
    - **Attendance Tracking:** Marks and tracks attendance in real-time.
    - **Notifications:** Receives push notifications, SMS, and emails.
- **Admin Workstations**

- **Biometric App Web Interface:** Accesses the system via a web browser for administrative tasks.

- **Cloud Infrastructure**
  - **API Gateway Services:** Central hub for routing requests to functional servers.
  - **Reporting Module:** Manages and generates attendance reports.
  - **Attendance Module:** Handles real-time attendance tracking and verification.
  - **Notification Module: Manages** notifications and alerts.
  - **Authentication Module:** Handles secure login and biometric authentication
- **Functional Servers**
  - **Reporting Server:** Generates and manages comprehensive attendance reports.
  - **Attendance Server:** Manages real-time attendance tracking.
  - **Notification Server:** Sends notifications, emails, and SMS alerts.
  - **Authentication Server**: Manages user authentication and biometric verification.
- **Database Server (MongoDB)**
  - **User Info:** Stores user information.
  - **Attendance Data:** Stores attendance records.
  - **Courses and Session Details:** Stores course and session information.

2.2. Hardware and Software Configurations

2.2.1. Hardware Requirements

- **Client Devices**
  - **Students & Instructors:** Smartphones or tablets with biometric capabilities and GPS functionality.
  - **Admin Workstations:** Laptops or desktops with a web browser.
- **Cloud Infrastructure**
  - **Severs:** Cloud-based servers for hosting API Gateway Services, functional servers, and the database.

2.2.2. Software Requirements

- **Client Devices**
  - **Operating Systems:** iOS, Android.
  - **Biometric Client App:** Mobile application for attendance tracking.
- **Admin Workstations**
  - **Web Browser:** Modern web browsers (Chrome, Firefox, Safari, Edge).
- **Cloud Infrastructure**
  - **API Gateway Services:** Node.js, Express.js.
  - **Functional Servers:** Node.js, Express.js, APIs for specific functionalities.
  - **Database:** MongoDB.
  - **Authentication**: Auth0 for secure login and authentication.
  - **Notification Services:** emailjs for email services, SMS APIs.

2.3. Deployment Diagram

2.3.1. Node Configuration

- ❖ **Client Devices**
  - o **Nodes:** Smartphones, tablets.
  - o **Components:** Biometric capture module, GPS verification, attendance tracking, notification receiver.
- ❖ **Admin Workstations**
  - o **Nodes**: Laptops, desktops.
  - o **Components**: Web browser accessing the Biometric Web Interface.
- ❖ **Cloud Infrastructure**
  - o **Nodes**: Virtual servers hosted in the cloud.
  - o **Components**: API Gateway Services, Reporting Server, Attendance Server, Notification Server, Authentication Server, Database Server.

2.3.2. Physical Deployment

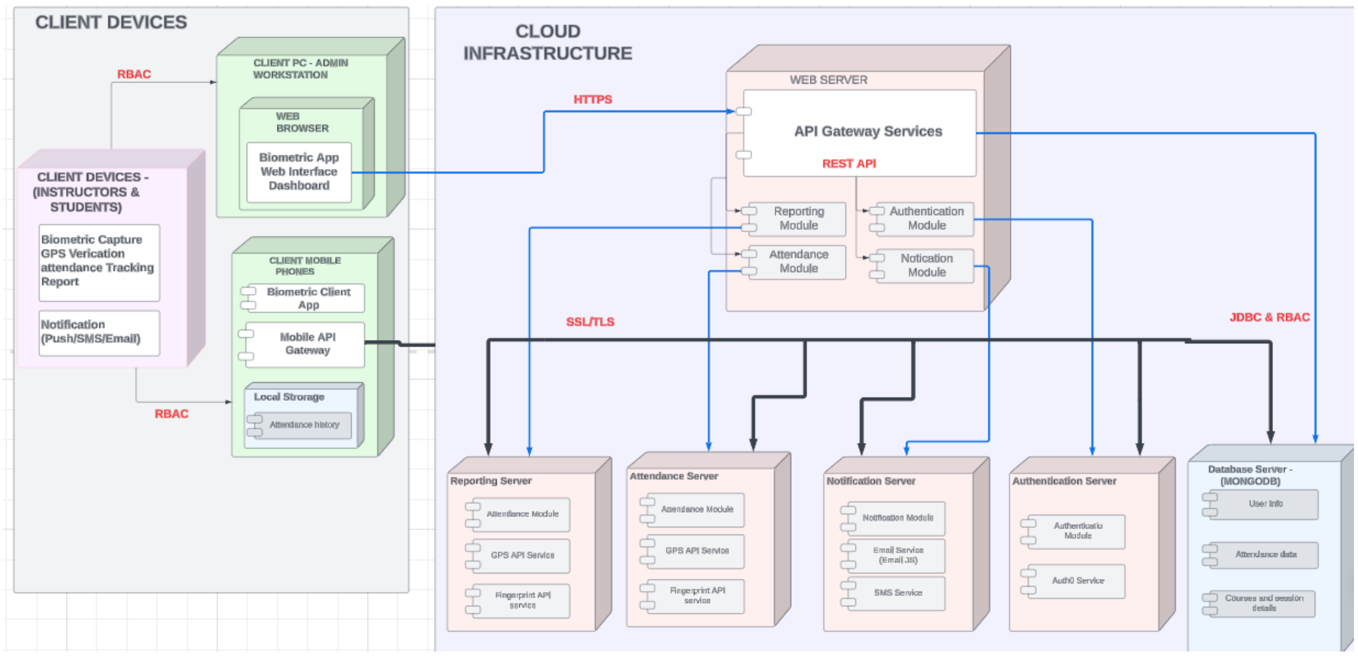

Figure 2: Physical Deployment Diagram

3. Data Flow and Context Diagram

3.1. Overview of Context and data flow Diagram

The context and data flow diagram (DFD) for the Biometric Attendance Application provides a visual representation of the system's interactions with external entities and the data flows between these entities and the system. The diagram helps in understanding the scope of the system, the major external entities involved, and the main data processes within the system.

3.2. External Entities and System Boundaries

- ❖ **Exterior Entities**
  - o **Students:** Provide authentication data and check-in for attendance. Receive attendance confirmation, history, and notifications.

    - **Instructors:** Initiate attendance sessions, request reports, configure notifications, and receive attendance data along with alerts and reminders.
    - **Administrators:** Request attendance reports and manage overall system functions.
    - **IT Staff:** Handle system updates, provide support, and manage performance reports.
- **Interior Entities**
    - **Biometric Module**: Captures and validates biometric data.
    - **Notification Module:** Sends notifications based on requests from the application.
    - **GPS System**: Provides location verification data for attendance.
    - **Feedback System**: Receives and processes feedback related to the application.
- **System Boundaries:**
    - The core of the system is the Biometric Attendance Application, which includes internal modules for biometric authentication, attendance tracking, attendance reporting, notifications, GPS location verification, and feedback processing.

3.3. Data Flow between External Entities and System

The data flows in the context diagram illustrate the interactions between the external entities and the Biometric Attendance Application:

- Students provide authentication data and attendance check-ins to the application, and receive attendance confirmations, history, and notifications.
- Instructors initiate attendance sessions, request reports, configure notifications, and receive attendance data and reminders.
- Administrators request and receive generated reports.
- IT Staff provide system updates, support, and performance reports.
- The Biometric Module receives biometric data and returns validation results.
- The Notification Module processes notification requests and sends notifications.
- The GPS System receives location data requests and provides location verification data.
- The Feedback System receives feedback and sends it to the application

3.4. Data Flow Diagrams

3.4.1. Level 0 DFD (Context Diagram)

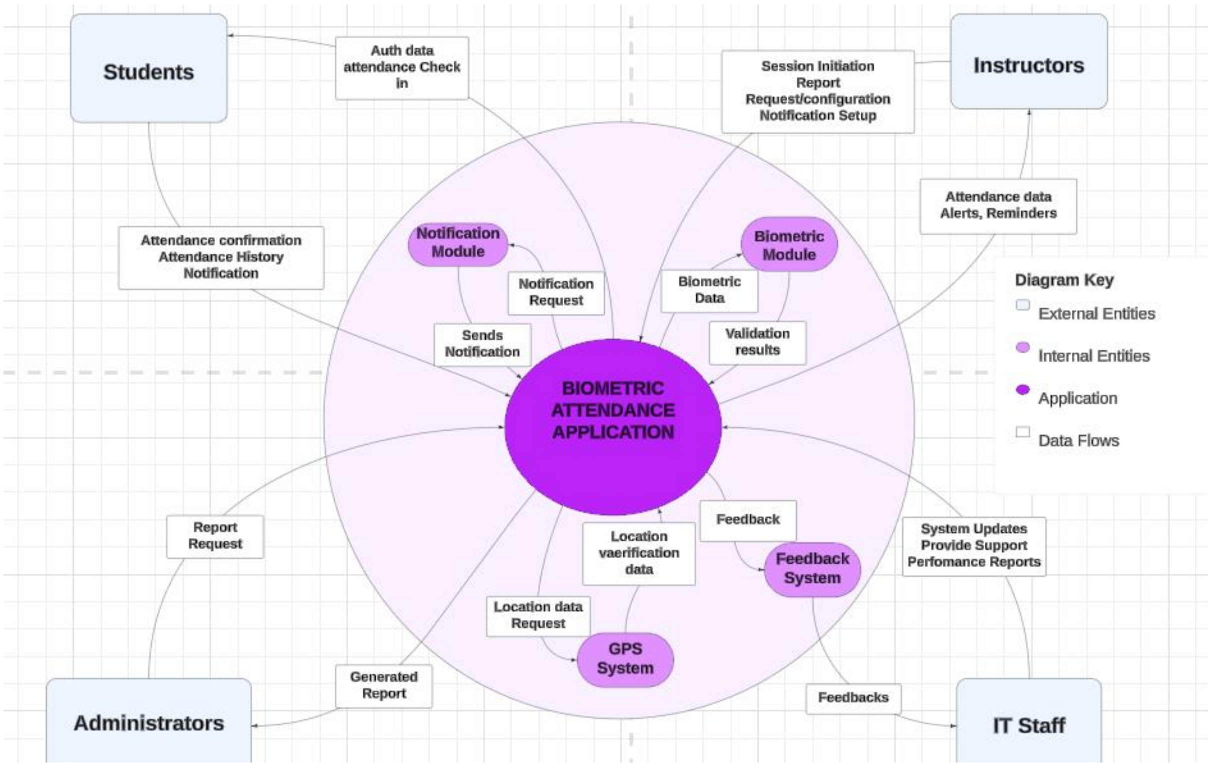

Figure 3: Context Level Data Flow Diagram

Detailed Description of Entities with Inputs and Outputs

- **Students**
  - **Inputs:**
    - Attendance confirmations and attendance history from the Biometric Attendance Application.
    - Notifications from the Notification Module.
  - **Outputs:**
    - Authentication data for check-in to the Biometric Attendance Application.
- **Instructors**
  - **Inputs:**
    - Attendance data from the Biometric Attendance Application.
    - Alerts and reminders from the Notification Module.
  - **Outputs:**
    - Session initiation requests to the Biometric Attendance Application.
    - Report requests and configuration settings to the Biometric Attendance Application.
    - Notification setup requests to the Notification Module.
- **Administrators**
  - **Inputs:**
    - Generated reports from the Biometric Attendance Application.
  - **Outputs:**
    - Report requests to the Biometric Attendance Application.

- **IT Staff**
  - **Inputs:**
    - System updates and performance reports from the Biometric Attendance Application.
  - **Outputs:**
    - System maintenance and support to the Biometric Attendance Application.
- **Biometric Module**
  - **Inputs**:
    - Biometric data requests from the Biometric Attendance Application.
  - **Outputs:**
    - Validation results to the Biometric Attendance Application.
- **Notification Module**
  - **Inputs:**
    - Notification requests from the Biometric Attendance Application.
  - **Outputs:**
    - Sends notifications to students and instructors.
- **GPS System**
  - **Inputs:**
    - Location data requests from the Biometric Attendance Application.
  - **Outputs:**
    - Location verification data to the Biometric Attendance Application.
- **Feedback System**
  - **Inputs:**
    - Feedback data from students, instructors, and administrators.
  - **Outputs:**
    - Processed feedback results to the Biometric Attendance Application.

4. Actors and Functions

4.1. Use Case Diagram

4.1.1. Use Case Identification

A use case is a description of how users interact with a system or product to achieve specific goals.

Below is our system's use case diagram.

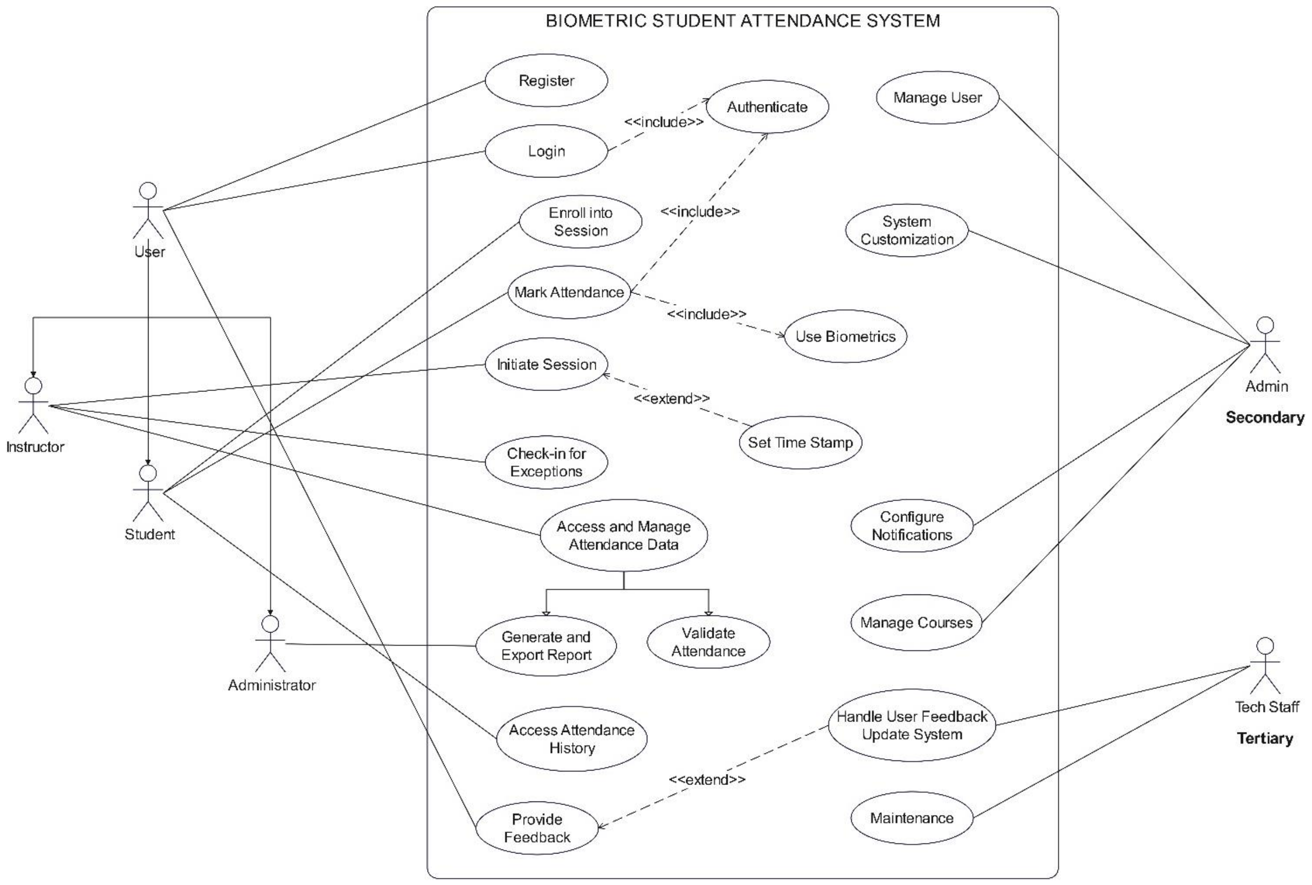

Figure 4: Use Case Diagram

4.1.2. Use Case Descriptions

A use case diagram in the Unified Modelling Language (UML) visually represents the interactions between actors (users, organizations, or external systems) and a system.

Purpose:

A use case diagram for a biometric student attendance system provides a high-level overview of how actors interact with the system.

It helps identify system functionalities and their relationships.

Components:

Actors: Represent users or external entities interacting with the system. They are depicted as stick figures.

The actors can be primary actors, secondary actors or tertiary actors.

The primary actor is a user or external entity who initiates a use case. They have a specific goal that the system helps fulfill.

Secondary actors provide services to the system but do not directly initiate use cases.

Tertiary actors would theoretically represent entities that interact indirectly with the system.

<u>Our primary actors are instructors and students.</u>

<u>Secondary actors are admins.</u>

<u>Use Cases</u>:

Represent specific functionalities or actions that the system performs. Use cases are shown as labelled ovals.

<u>System Boundary</u>:

Drawn around the use cases to define the system's scope.

<u>Notation</u>:

Actors connect to use cases with lines to show their involvement.

Goals (end results) of use cases are described within the diagram.

4.2. Actors and Their Interactions

Below are some of the use cases of our actors.

Student:

- Register information plus Biometric:
  Here, the students input their student details and biometric feature(fingerprint) for his/her registration.
- Login:
  The student has to login so as to avoid long query search. This is so as to improve latency. This is for comparison between inputted fingerprint during attendance taking and the existing fingerprint for the logged in student rather than all fingerprints in the database.
- Mark Attendance:
  The student inputs his/her fingerprint for authentication and attendance is validated if positive result after authentication.

Instructor:

- Login:
  Here, the Instructor inputs a credential related their course to have access to their course(s) only.
- Initiate Attendance:
  The instructor initiates any attendance session and students can only take attendance during this period.
- Set time stamp:
  This is an extend of instantiate attendance because there will be a default time to stop attendance if the instructor does not decide when to stop the attendance session on his own.
- Update Attendance (check-in for exceptions):

The instructor has the ability to update(modify) the attendance of any student depending on the scenario.

The instructor can mark a student as present in case of permission or health issues.

The instructor can also cancel the attendance of a student in case of bad behaviors or some other negative reason.

- Generate Report:

  The instructor generates reports for specific occasions and time.
- View Attendance Record:

  The instructor views the attendance record of his/her course(s).

Admin:

- Manage User:

  The admin validates the registration of users(students, instructors) and creates their account.
- Manage course:

  The admin manages the life cycle of courses into the database.

4.3. Functional Requirements Mapping

Functional requirements describe what a software system should do. They focus on defining the system's behaviour and functionality.

The table below tries to map our functional requirements to our use cases.

Functional Requirement	**Use Case**
Biometric Data Capture	Registration -> use Biometric
Enrollment and Authentication Module:	Registration Login
Real-time Attendance Tracking	Initiate Attendance Mark Attendance Stop Attendance
Reporting	Generate Report

Table 1: Functional Requirements to Use Case Mapping

4.4. Activity Diagram

An activity diagram helps to visualize a certain use case at a more detailed level. It is a behavioural diagram that illustrates the flow of activities through a system. Its purpose is to model the behaviour of a system or process in a clear and structured way, making it easier to understand and analyse.

Below are the activity diagrams for the biometric student attendance recording system.

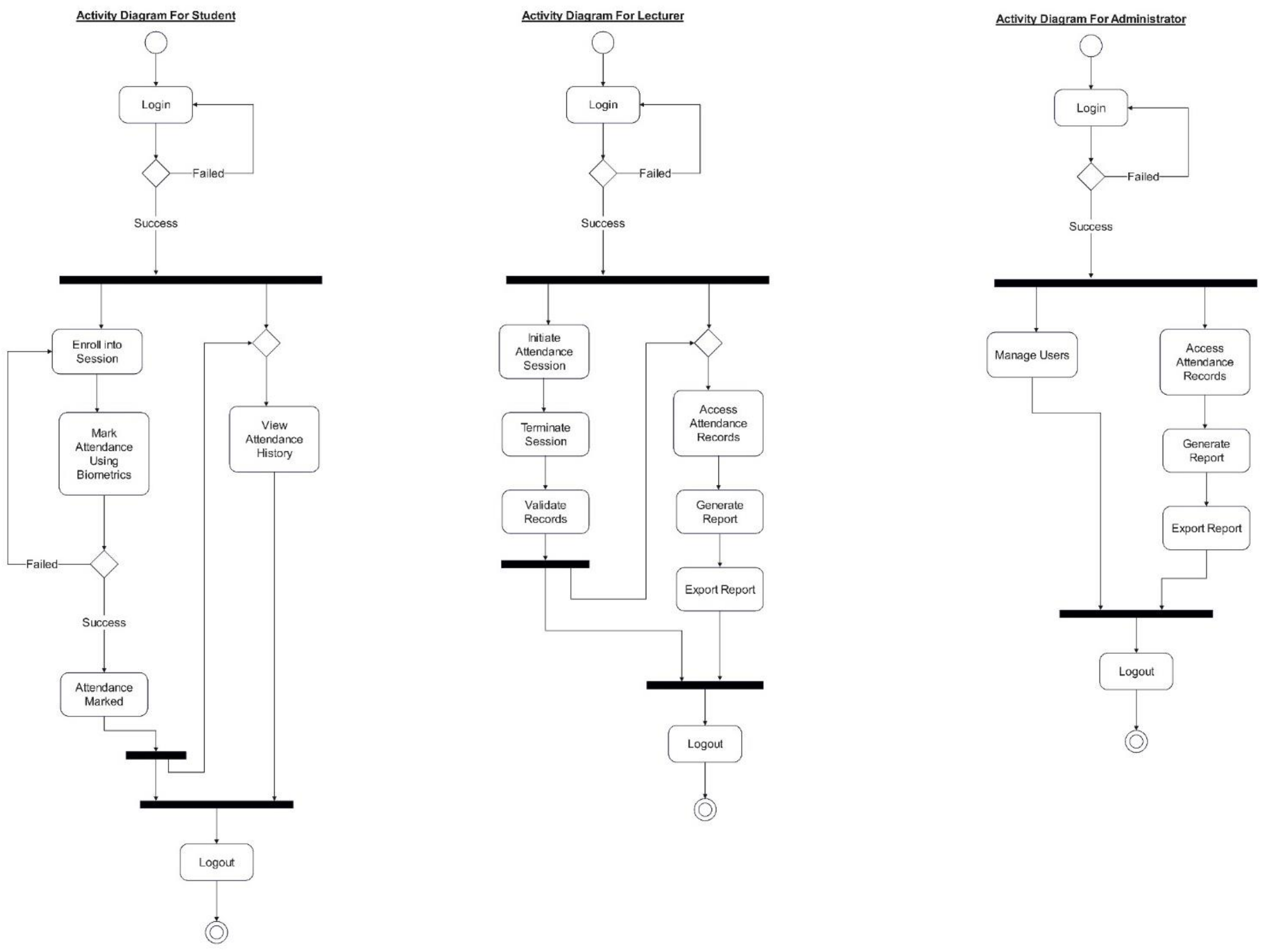

Figure 5: Activity diagrams

Overview of User Activities

1. **Lecturer:**
   - Lecturer attempts to login.
   - If login is successful, then lecturer proceeds with either initiating session, or accessing old attendance records.
   - After initiating a session, he terminates the session and then validates the records.
   - After accessing attendance records, he can generate a report and then export the report.
   - The lecturer then logs out of the system.
2. **Student:**
   - Student attempts to login.
   - If login is successful, he can either enroll into an attendance session or view attendance history for courses.
   - When he enrolls into a session, he uses his biometrics to mark himself present and if biometric authentication is successful, he is marked present.
   - The student then logs out of the system.

3. **Administrator:**
    - Administrator attempts to log into the system.
    - If login is successful, he can either go ahead managing user (lecturers and students) or accessing attendance records.
    - After accessing attendance records, he can generate a report and then export the report.
    - The admin then logs out of the system.

5. Interactions

5.1. Sequence Diagrams

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

Creating a sequence diagram for user authentication in a biometric attendance system involves illustrating the interactions between students, instructors, school administrators (admins), the biometric system, and the authentication system. Below, are the processes in detail.

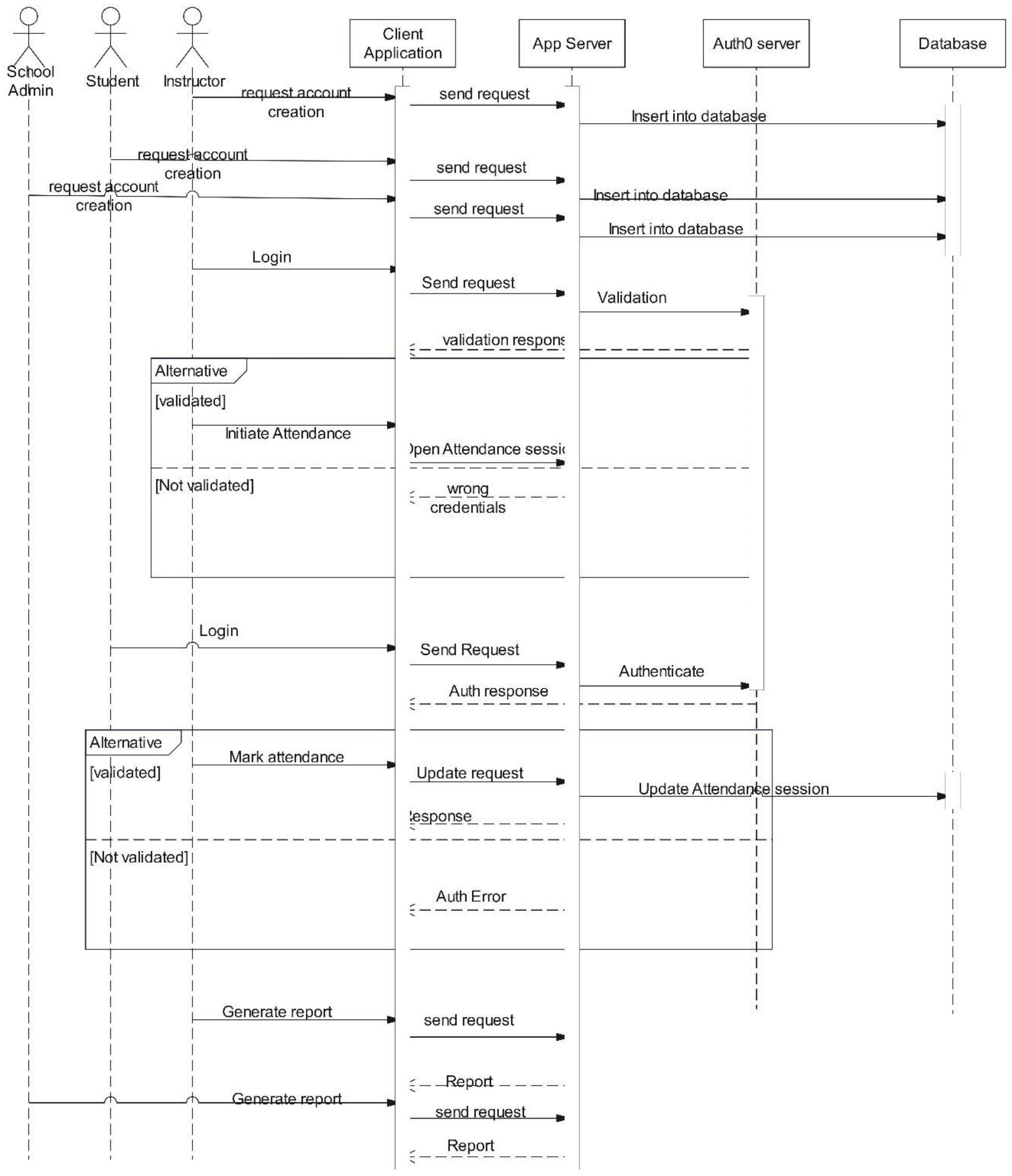

Figure 6: Sequence Diagram

5.1.1. User Authentication Sequence

The user (student, instructor, or admin) approaches the Biometric System and is later asked to register by providing their credentials like name, email, matricule, and biometric information for students. The students must register with their biometric information and matricule to uniquely identify each student.

The information is sent from the client application to the server which is authenticated validated and stored in the database.

After a successful login, users are prompted to log into the system with they provided.

5.1.2. Attendance Marking Sequence

Instructor initiates Attendance session

The student arrives at the biometric scanner and initiates the attendance process by providing a biometric input (e.g., fingerprint scan, facial recognition).

Capture Biometric Data:

The Biometric System captures the student's biometric data. The captured biometric data is sent to the Authentication Server for verification.

Verify Biometric Data:

The Authentication Server compares the received biometric data with the stored data in its database to identify the student.

The Authentication Server sends a verification result (success or failure) back to the Biometric System.

Send Verification Result to Biometric System:

The Biometric System receives the verification result from the Authentication Server.

Grant Access or Deny Access:

If the verification result is successful (i.e., the biometric data matches), the Biometric System grants access and proceeds to mark the student as present.

If the verification result is unsuccessful (i.e., the biometric data does not match), the Biometric System denies access, and attendance is not marked.

Log Attendance in Attendance System:

If access is granted, the Biometric System sends the student's attendance data to the database, where it is logged.

The Attendance System updates the records to reflect that the student is present for the class/session.

6. Object-Oriented Solutions

6.1. Class Diagrams

A class diagram in Unified Modelling Language (UML) is a type of static structure diagram that visually represents the structure of a system. It provides an overview of the system's classes, their attributes, operations (or methods), and the relationships among objects.

6.1.1. Class Identification

A class is a blueprint or template for creating objects within an object-oriented system.

A class defines the structure and behavior of objects. It specifies what an object can do and what attributes it possesses.

Our student biometric attendance system has the following classes:

Key Components of a Class:

A class is visually represented by a rectangle in UML diagrams. This rectangle includes three compartments:

Class Name: The name of the class appears in the top compartment and is typically centered and bold.

Attributes: These represent the data members (properties) of the class. Attributes describe the characteristics or properties of objects created from the class.

Methods (Operations): Methods define the behaviors or actions that objects can perform. They represent the functionality associated with the class.

Visibility Notations:

Visibility notations indicate the access level of attributes and methods within the class:

+: Public visibility (visible to all classes).

-: Private visibility (visible only within the class).

#: Protected visibility (visible to subclasses).

~: Package or default visibility (visible to classes in the same package).

Our system's classes.

- Instructors
- Students
- School administrators
- Courses

6.1.2. Relationships and Inheritance

Relationships help us understand how different classes collaborate and interact with each other.

Inheritance is a paradigm which allows a class to reuse or acquire attributes or methods of another class.

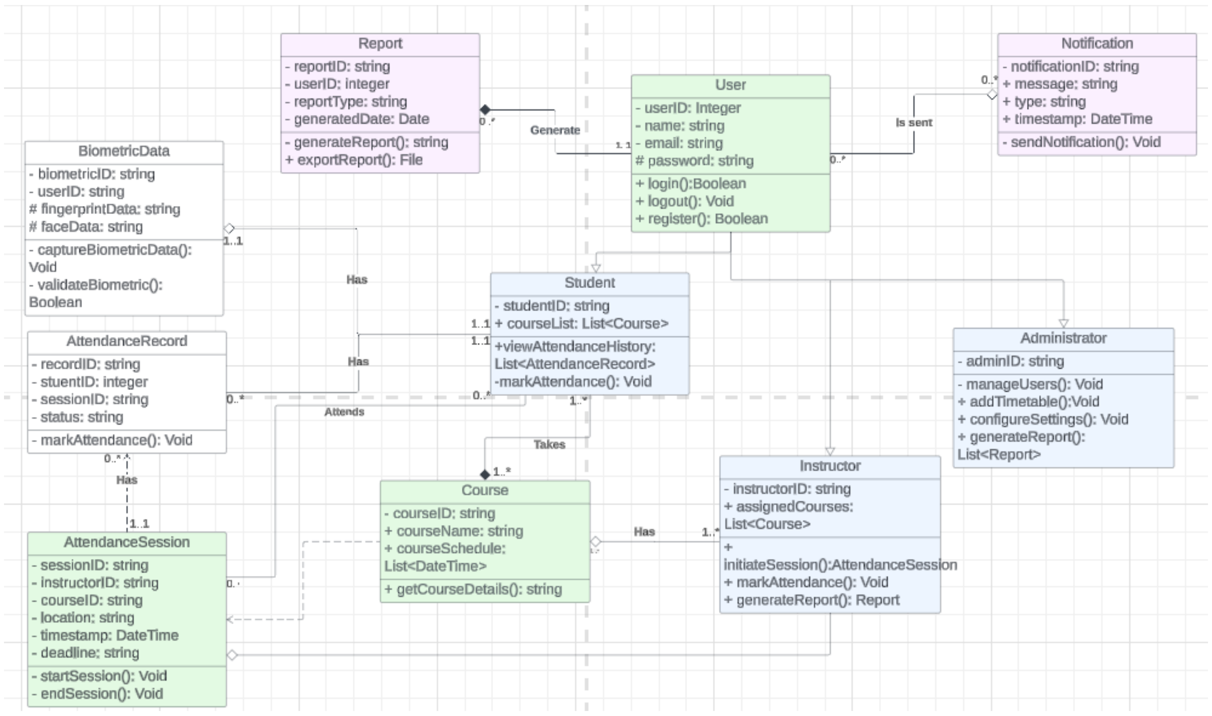

Figure 7: Class Diagram and Relationships

7. Security and Performance Considerations

7.1. Security Architecture

When designing a biometric attendance system, it's crucial to consider both security and performance to ensure the system is robust, reliable, and efficient. These are the key strategies we considered.

Biometric Data Protection:

Encryption: Encrypt biometric data both in transit and at rest using strong encryption standards (e.g., AES-256) to prevent unauthorized access.

Secure Transmission: Use secure communication protocols like HTTPS to transmit biometric data to the authentication server.

Authentication Server Security:

Access Controls: Implement strict access controls to ensure only authorized personnel can access the server and its data.

Regular Audits: Conduct regular security audits and vulnerability assessments to identify and mitigate potential threats.

Multi-Factor Authentication (MFA): Require MFA for administrators and other users accessing sensitive parts of the system.

User Privacy:

Consent: we obtained explicit consent from users before capturing their biometric data and collected only the necessary biometric data required for the attendance system and avoid over-collection.

Compliance: we have to ensure compliance with data protection regulations like GDPR, CCPA, and other relevant laws.

Liveness Detection: Implement liveness detection techniques to ensure the biometric input is from a live person and not a fake or spoofed sample.

Multi-Biometric Systems: Use multi-modal biometric systems (e.g., combining fingerprint and facial recognition) to enhance security.

Logging and Monitoring:

Audit Logs: Maintain detailed audit logs of all biometric authentication attempts and administrative actions.

Real-Time Monitoring: Implement real-time monitoring to detect and respond to suspicious activities promptly.

7.2. Performance Optimization Strategies

System Scalability:

Scalable Architecture: Design the system to handle varying loads by using scalable architecture (e.g., microservices, load balancers).

Cloud Services: Utilize cloud-based services to dynamically scale resources based on demand.

Response Time:

Efficient Algorithms: Use efficient biometric matching algorithms to ensure quick verification.

Caching: Implement caching mechanisms for frequent operations to reduce processing time.

System Reliability:

Redundancy: Ensure high availability by incorporating redundancy in critical components such as servers, storage, and network paths.

Failover Mechanisms: Implement failover mechanisms to maintain service continuity during hardware or software failures.

User Throughput:

Concurrent Processing: Design the system to handle multiple concurrent biometric scans and verifications to avoid bottlenecks.

Optimized Workflow: Streamline the user interaction workflow to minimize the time required for each biometric scan.

Accuracy:

High-Quality Sensors: Use high-quality biometric sensors to capture accurate and reliable data.

Threshold Tuning: Adjust matching thresholds to balance between false acceptance rate (FAR) and false rejection rate (FRR) based on the specific use case.

7.3. Implementation Strategies:

Security Enhancements:

Encryption Example: Use libraries like OpenSSL for encrypting biometric data.

Liveness Detection Example: Implement techniques like blinking detection for facial recognition or pulse detection for fingerprints.

Performance Enhancements:

Scalability Example: Deploy the system on a cloud platform like AWS, which offers auto-scaling and load balancing.

Caching Example: Use Redis or Memcached to cache frequently accessed data, reducing database load.

8. Conclusion

8.1. Summary of Design

The development of the Mobile Attendance System aims to revolutionize the way attendance is tracked and managed in educational and professional environments. By integrating cutting-edge technologies such as biometric authentication, real-time attendance tracking, and comprehensive reporting, this system offers a robust solution to modern attendance challenges.

Key Insights from UML Models

- **Context Diagram:**
  The context diagram provided a clear boundary for our system, defining how it interacts with external entities such as administrators, instructors, students, and the IT department. This high-level view ensured that all necessary interactions were identified early in the design process. Also allowed us to visualize the flow of data through various processes within the system, ensuring that data handling and transformations are efficient and secure. This detailed mapping helps in pinpointing potential bottlenecks and areas for optimization.
- **Use Case Diagram:**

The use case diagram outlined the primary functionalities required by different users. By focusing on user needs, we ensured that the system is both functional and user-friendly, addressing specific roles and their interactions with the system.

- **Deployment Diagram:**
  The deployment diagram provided a clear picture of the physical deployment of the system components, ensuring that all hardware and software requirements are well understood. This clarity is crucial for successful implementation and integration with existing infrastructure.
- **Class Diagram:**
  The class diagram detailed the static structure of the system, highlighting key classes and their relationships. This structural blueprint is vital for developers to understand the core components and their interactions, facilitating a smoother development process.
- **Sequence Diagram:**
  The sequence diagrams mapped out the interactions between objects over time for critical scenarios, such as user authentication and attendance marking. These dynamic models help in understanding the real-time operations of the system, ensuring efficient process flows and quick response times.

8.2. Future Enhancements and UI Design Plan

Moving forward, our focus will shift to designing an intuitive and user-friendly interface that caters to all stakeholders. By prioritizing user-centric design principles, we aim to develop an interface that is not only functional but also visually appealing and easy to navigate.

Key steps include:

- **User Centric Design:** Conducting user research and usability testing to gather insights and feedback.
- **Wireframing and Prototyping:** Developing wireframes and interactive prototypes to visualize user flows.
- **Theme and Visual Design:** Develop a style guide and color scheme
- Ensuring **accessibility, performance** and **responsiveness** across multiple devices and platforms.
- **Iterating** on the design based on continuous **feedback** and evolving requirements.

8.3. References

https://stackoverflow.com/questions/1988500/how-relations-in-uml-class-diagram-inherit.

https://www.uml-diagrams.org/class.html.

https://lucid.app/lucidchart.

https://creately.com/diagram/example/ifpdd632/attendance-system-using-biometric-system-classic.

https://www.pinterest.com/pin/activity-diagram-for-student-attendance-management-system--635148353705202005/