STEO

by

Ahmad Sharkawi
Zakaria Khatib
Mohammad Al Ghali
Adnan Khalil
Mohammad Jomaa

Project

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor

in computer engineering

Department of Engineering

Faculty of Engineering

Year Spring- 2025

جـامعة بيروت العربية
BEIRUT ARAB UNIVERSITY

STEO

by

Ahmad Sharkawi
Zakaria Khatib
Mohammad Al Ghali
Adnan Khalil
Mohammad Jomaa

Project

Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelor

in computer engineering

Department of Engineering

Faculty of Engineering

Supervised by

Prof. /Dr Imane Hiadar

Contents

List of Figures

List of Tables

Project Description

Project Overview

Steo is an attendance management app designed for Beirut Arab University to streamline the attendance tracking process for both instructors and students. The app allows instructors to manage classes and students through a dashboard, while students can mark their attendance by scanning their university ID barcode. The app also includes features like automated notifications for absenteeism, emergency case submissions, and event announcements to enhance communication and efficiency within the university.

Project Objectives

1. Simplify attendance tracking for instructors and students using barcode scanning technology.
2. Provide instructors with a dashboard to manage classes and students (CRUD operations).
3. Automate notifications for repeated absenteeism and enable warning emails.
4. Allow students to submit emergency cases with supporting documents for missed attendance.
5. Facilitate communication between instructors and students through announcements and events.

Background

The field of educational technology (EdTech) is rapidly growing, with a focus on improving administrative efficiency and student engagement. Attendance tracking is a critical aspect of academic management, but traditional methods are often time-consuming and prone to errors. By applying software solutions like Steo, universities can automate attendance processes, reduce manual workload, and improve communication between instructors and students. This project aligns with the global trend of digitizing educational systems to enhance productivity and accountability.

Literature Review

Below is a summary of similar products and research in the field of attendance management systems:

Product: Attendify

- **Description**: A mobile app for event and class attendance tracking using QR codes.

- **Advantages**: Easy to use, real-time tracking, and integrates with event management tools.
- **Problems**: Limited customization for academic institutions and no support for barcode scanning.

Product: Smart Attendance (Lebanon)

- **Description**: A web-based attendance system used by some Lebanese universities for tracking student attendance.
- **Advantages**: Cloud-based, accessible from any device, and supports bulk data upload.
- **Problems**: Lacks mobile app support and barcode/QR code integration.

Research Paper: "Automated Attendance Management System Using Face Recognition"

- **Authors**: Kumar et al. (2020)
- **Description**: A system that uses facial recognition to mark attendance in classrooms.
- **Advantages**: Reduces manual effort and eliminates proxy attendance.
- **Problems**: High implementation cost and privacy concerns.

Research Paper: "Barcode-Based Attendance System for Universities"

- **Authors**: Ali et al. (2019)
- **Description**: A system that uses student ID barcodes to track attendance.
- **Advantages**: Cost-effective and easy to implement.
- **Problems**: Requires barcode scanners and lacks advanced features like notifications.

Literature Review Table

Ref. Nb	Authors	Description	Advantages	Problems
1	Attendify	Mobile app for attendance tracking using QR codes.	Easy to use, real-time tracking, integrates with event tools.	Limited customization, no barcode support.
2	Smart Attendance	Web-based system for Lebanese universities.	Cloud-based, accessible, supports bulk upload.	No mobile app or barcode integration.
3	Kumar et al.	Facial recognition-based attendance system.	Reduces manual effort, eliminates proxy attendance.	High cost, privacy concerns.
4	Ali et al.	Barcode-based attendance system for universities.	Cost-effective, easy to implement.	Requires barcode scanners, lacks advanced features.

Applications

The monitoring applications that track student attendance at a university are for use by the instructor. A student can log in through the web portal and check in for attendance. The instructor can record the attendance using a barcode, RFID, or biometrics. This helps the instructor to prevent cheating, save time, and keep the records accurate. The instructor can generate a report from the software, this improves the management of the class and makes it efficient.

Alternative Designs

Mobile app

This design focuses on a student-centric mobile application that handles attendance tracking. The mobile app serves as both a convenience for students and an efficient tool for instructors and administrators. Where instructors can record the attendance using a barcode and generate a report from the app.

Attendance Monitoring System

A Web-based System for Students and Faculty. The system is aimed at students and faculty to efficiently control and monitor attendance. The system is a responsive website and can be used on any device that is connected to the internet (desktop, laptop, mobile).

QR Code Check-In: A moving QR code will be shown on the webpage, and students can scan it with their mobile phones to check-in when they enter the classroom or seminar room

Project Planning

Constraints

- **Implementation environment of the current system**

Our project is based on standard web technologies (HTML, CSS, JavaScript, with a full Laravel backend). This environment is sufficient for building a basic attendance management system but limits our ability to implement more complex features or highly efficient interfaces.

  - Frontend: HTML, CSS, JavaScript (with Blade templating engine for dynamic views)
  - Backend: Laravel (PHP) for handling server-side logic, database interactions, and authentication
  - Database: MySQL for storing attendance and user data

- **Partners and collaboration**

Our team has basic to intermediate knowledge of the technologies used in the project, which means we may face challenges when collaborating on more advanced tasks. For instance, if an issue arises in the backend, the frontend developers might struggle to assist the backend developer, and vice versa.

- **Off-the-shelf Software**

The project does not heavily rely on off-the-shelf software, as we are building a custom solution tailored to our specific needs. However, we may integrate existing software tools for specific functionalities, which limits our flexibility in customization.

- **Anticipated Workplace Environment**

The workplace environment for the development of the attendance app will be a combination of both remote and university.

- **Schedule Constraints**

The schedule of the project will be impacted by many constraints, some being project complexity, deadline, and team availability. Some of the most significant constraints are:

  - **Limited Working Hours:** Project team members may have varying shifts or work on a part-time basis, therefore decreasing the work hours on the project per week.
  - **Project Phases:** Some of the phases, for instance, design, development, testing, and deployment, need to be carried out one after another. Delay in any phase could result in delay in the whole timeline.
  - **Deadlines:** A specified delivery date can limit the scope for modifying the project scope or timeline once the development process is initiated.

Project Issues

- **Authentication Issues**

    - **Problem:** Session management, user login, and access role challenge.
    - **Impact:** Arbitrary login scenarios and security threat probability.
    - **Solution:** Session management improvement, inclusion of JSON Web Token (JWT), and role validation.

- **Risks**

    1. **Security Risks**
        - Insecure authentication will lead to unauthorized usage.
        - Mitigation: Impregnable encryption and genuine authentication.

    2. Dependency on External Libraries
        - Vulnerability to unmaintained third-party tools risks.
        - Mitigation: Keep in line with updates and keep fallbacks ready.

    3. User Adoption Risks
        - Users adapting to changes for adopting new system.
        - Mitigation: User testing, error-free UI, offer training.

Team Members Tasks

- **UI/UX Design**

Ahmad Sharkawi: User Experience Specialist, dedicated to crafting seamless, intuitive, and user-friendly experiences that enhance engagement and efficiency.

- **Frontend developers**

Ahmad sharkawi, Adnan Khalil, Mohamad Jomaa: Responsible for user interface (UI), and the design of the app.

- **Backend developers**

Mohamad Al Ghali, Zakaria khatib: Responsible for creating function and API.

- **Database**

Mohamad Jomaa: responsible for storing attendance and user data.

Ethical Issues

1. **Data Privacy and Security**

- **Problem:** The app handles private information of students like attendance records that need to be kept out of reach of unauthorized persons.
- **Ethical Concern:** Users (admin, teachers, students) are entitled to privacy, and any leakage of data or usage of their own data would be unethical.
- **Resolution:** Use strong encryption when transmitting and storing data, adhere to data protection laws, and have proper access controls in effect to safeguard sensitive information.

2. **Bias in Attendance Monitoring**

    - **Problem:** f bias exists in teacher handwriting or in systems, then biased attendance marking occurs
    - **Ethical Concern:** Partial attendance marking could be treating the students differently and could result in discrimination.
    - **Resolution:** Strictly monitor the teachers to mark the attendance honestly. Review the system from time to time for transparency and fairness.
3. **Informed Consent:**
    - **Problem:** The users need to be well-informed about information collection, storage, and use.
    - **Ethical Concern:** It is unethical to collect information without users' consent or to fail to inform users of the use of their information.
    - **Resolution:** Publish conspicuous terms of service and privacy notices. Get explicit consent from the users prior to collecting their information and give them an option to opt out wherever possible.

Software Model Process

1. **Waterfall Model**

    - **Description:** Linear model where each development phase must be completed before moving on to the next one. It is a linear model that best fits projects with well-defined requirements.
    - **Phases**
        - **Requirement Gathering:** To identify and document the entire set of requirements.
        - **System Design:** System design and planning according to requirements.
        - **Implementation:** Install the software according to the design.
        - **Testing:** Execute QA testing to verify the software against requirements.
        - **Deployment:** Deploy the product with customers.
        - **Maintenance:** Bug fixes and maintenance on an ongoing basis following release.

2. **Incremental Model**

    - **Description:** Under this approach, the software is developed in small increments or pieces, usually called bites. Every increment adds functionality to the system and develops it until it is completed.
    - **Phases**

        - **First Planning:** Define overall system requirements and general design.
        - **Incremental Development:** Develop small parts of the system with each increment.
        - **Testing:** Test every increment as it is developed.
        - **Release:** Publish incremental updates to customers.
        - **Final Release:** When all increments are finished, the whole system is finished and published.

Feasibility study

- **Technical Feasibility:** The Laravel-based student attendance system is possible using ID card scanning and a MySQL database.
- **Economic Feasibility:** Costs are minimal, mainly for ID scanners and self-hosted server expenses.
- **Operational Feasibility:** The system is easy to use, scalable, and reduces manual attendance errors.
- **Schedule Feasibility:** The project can be completed within 10 weeks with proper planning and development.

Tools/Technology

- **Programming Languages**

    - HTML (Markup Language)
    - CSS
    - JavaScript
    - PHP
    - SQL

- Frameworks and Libraries

    - Laravel (PHP Framework)
    - Tailwind CSS
    - Axios (for API requests)
    - Laravel Passport (for Authentication)
    - Capacitor & ionic (to build the website as an app)

- API Keys
    - None (or specify if any third-party APIs are used for the app)
- Development Tools
    - GitHub
    - VS Code
    - Composer (for PHP dependency management)
    - XAMPP/WAMP (local development server for PHP/MySQL)
    - Postman (for API testing)
    - Hostinger (for server deployment)

Standards

1. **Coding Standards**
    - Write clean and well-organized code.
    - Use simple and consistent naming for files and variables.
    - Keep the code easy to read and understand.
2. **Security Standards**
    - Use strong passwords and authentication methods
    - Protect data from hacking by keeping it encrypted.
    - Make sure only the right people can access certain parts of the system.
3. **Design and UI Standards**
    - Make the app simple and easy to use.
    - Ensure it works well on both computers and mobile phones.
    - Use clear colors, buttons, and layouts for a better user experience.
4. **Testing Standards**
    - Check that everything works before launching the app.
    - Test all features to make sure they don’t break.
    - Fix any bugs before releasing the app.
5. **Development Process Standards**
    - Work in an organized way using GitHub to save progress.
    - Assign tasks clearly to team members.
    - Keep track of project updates and deadlines.
6. **Deployment and Maintenance Standards**

- Make sure the app runs smoothly when deployed.
- Regularly update and fix any issues.
- Keep backups to avoid losing important data.

Milestones

1. **Project Planning and Requirement Gathering**

    - Complete project purpose and scope.
    - Obtain proper requirements from stakeholders (admins, teachers, students).
    - User story decomposition and most important features.

2. **System Design and Architecture**

    - Create database schema structure (user roles, attendance tables, student tables).
    - Create wireframe or a prototype for the user interface.
    - Install and choose development framework (Laravel, Blade, MySQL)

3. **Frontend Development (Initial Phase)**

    - Create the fundamental UI components (login page, dashboard, mark attendance).
    - Implement responsive design using Tailwind CSS.
    - Implement dynamic client-side interaction using Blade files.

4. **Backend Development (First Phase)**

    - Implement the Laravel backend with critical routes and controllers.
    - Implement core features (user authentication, CRUD operations for attendance).
    - Implement basic routes to communicate with the frontend.

5. **User Authentication & Authorization**

    - Implement JWT or Laravel Passport for user authentication.
    - Implement role definition and permission (admin, teacher, student) using Laravel middleware.
    - Enable session management and security options (password hashing, encryption).

6. **Implementation of Attendance Tracking System**

    - Implement the attendance taking feature (manual entry, barcode scan).
    - Show real-time attendance updates (if necessary, using Laravel Echo and Pusher).
    - Implement logic to save student attendance history.

7. **External Service Integration (if necessary)**

    - Integrate with external APIs (for notifications, real-time updates, data visualization).
    - Integrate Gemini API (if third-party APIs are utilized for some other functionality).

8. **Frontend-Backend Integration**

    - Integrate backend and frontend (attendance record, notifications, user profiles).

- Integrate effective communication between frontend and backend.
- Test all UI components and routes for correctness.

9. **Testing Phase**

- Perform unit testing for backend logic (PHPUnit).
- Perform integration testing (Postman) on API endpoints.
- Perform end-to-end test through Laravel Dusk.

10. **Bug Fixing and Quality Assurance**

- Triage bugs picked up during testing.
- Security, performance, and best practice code review.
- User acceptance testing to validate app is hitting spec.

11. **Preparation for Deployment**

- Configure production environment (Hostinger).
- Execute code implementing and execute CI/CD pipelines in GitHub Actions.
- Stage app build to deploy on Android and iOS hardware.

12. **Go-Live (Deployment to Production)**

- Execute deploy the app on production server.
- Publish the app to public sight and test publicly with massive numbers of users.
- Monitor user performance and reaction in real-time.
- Publish the app on android and iOS

13. **Monitoring and Maintenance of Project after Release**

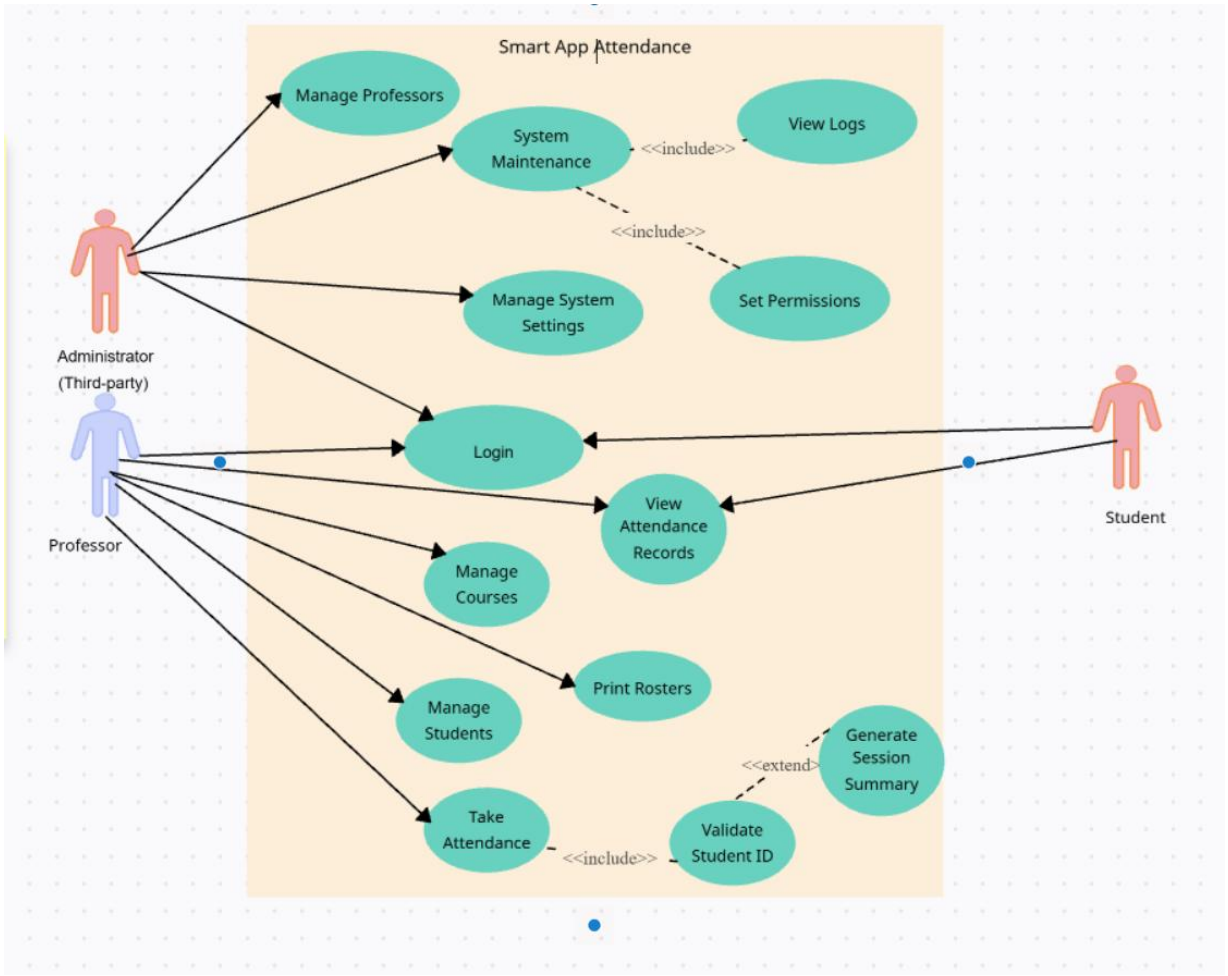
- Monitor app performance and usage by users (through Google Analytics or otherwise).
- Publish required bugs or deficiencies after release.
- Publish the app regularly with new features, updates, and patches.

14. **Handover & Documentation of Project**

- Master class codebase for preparing final project, user guide, API guide.
- Training management/Faculty for app using.
- Maintenance and support for project handover.

Requirements

Use Cases



Functional Requirements

These define what the system should do.

- Instructors can log in and access their dashboard.
- Instructors can perform CRUD operations on:
  - Students
  - Classes
  - Attendance
  - Announcements
- Students can log in and:
  - View their classes and attendance records
  - Mark attendance via barcode scanning
  - Submit emergency absence cases with documents
- Admins can manage both student and instructor data.
- System sends automated absentee warning emails after 3 missed sessions.
- Instructors can create and send university-wide announcements and events.

Data Requirements

These define what data the system stores and processes.

- Student data: ID, name, attendance records, emergency case documents.
- Instructor data: ID, name, classes handled.
- Attendance logs: Date, student ID, class ID, status.
- Emergency cases: Description, date, attached documents.
- Announcements and events: Title, description, target audience.

Non-Functional Requirements

These describe how the system performs.

- The system must be mobile and web responsive.
- The barcode scanner must function in real-time.
- Notifications and email systems must trigger automatically.
- Instructor dashboard must be intuitive and load in under 2 seconds.

4. Performance Requirements

- Attendance submission must be processed in less than 1 second.
- System should handle concurrent users (instructors and students) without lag.
- Reports generation should take under 5 seconds for up to 100 students.

5. Dependability Requirements

- System must have 99.5% uptime.
- Attendance records must never be lost (automatic backups enabled).
- In case of network failure, attendance can be cached and synced later.

6. Maintainability and Supportability Requirements

- System must be modular to allow easy updates and debugging.
- Documentation must be provided for all backend APIs and database schema.
- Admin panel for managing system-level settings and backups.

7. Security Requirements

- All data must be encrypted in transit and at rest.
- User roles and permissions must be enforced (admin, instructor, student).
- Session management using secure tokens.
- Uploaded documents scanned for potential malware.
- Email addresses verified before account activation.

8. Usability and Humanity Requirements

- Simple and user-friendly interface for both students and instructors.
- Instructions for scanning ID and submitting cases should be clear.

9. Look and Feel Requirements

- Clean and professional UI using responsive design.
- Dark and light mode options.
- Institutional branding (BAU logo/colors) must be present.

- Consistent layout across all pages and platforms.

10. Operational and Environmental Requirements

- Should run on all modern browsers and Android/iOS devices.
- Minimal processing power and bandwidth requirements.
- Web server should be Linux-compatible (used with Hostinger hosting).

11. Cultural and Political Requirements

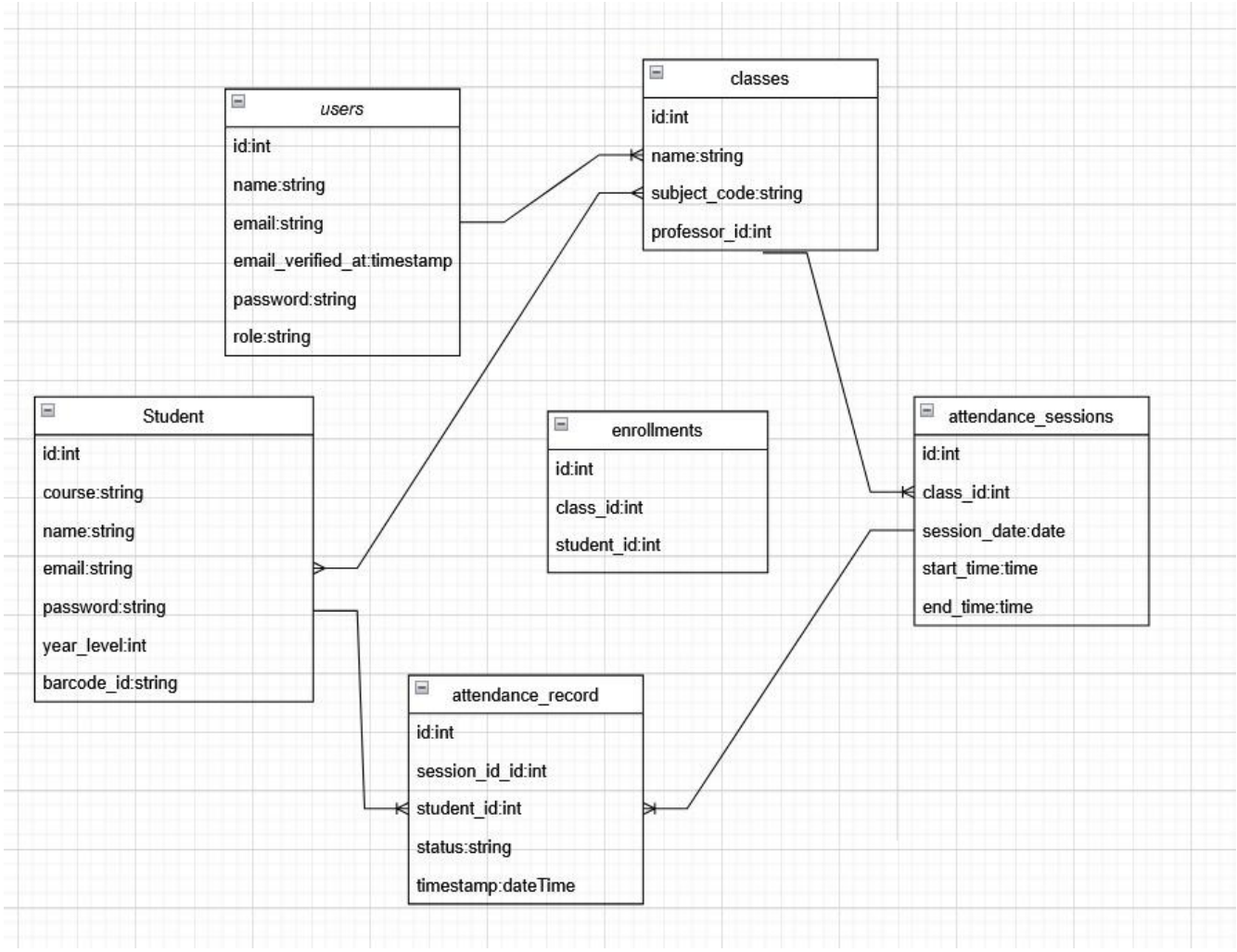
- Content (announcements, emergency options) should respect local cultural norms.
- Support for Hijri and Gregorian calendar if required.
- Clear privacy messaging to respect student data in line with regional expectations.

12. Legal Requirements

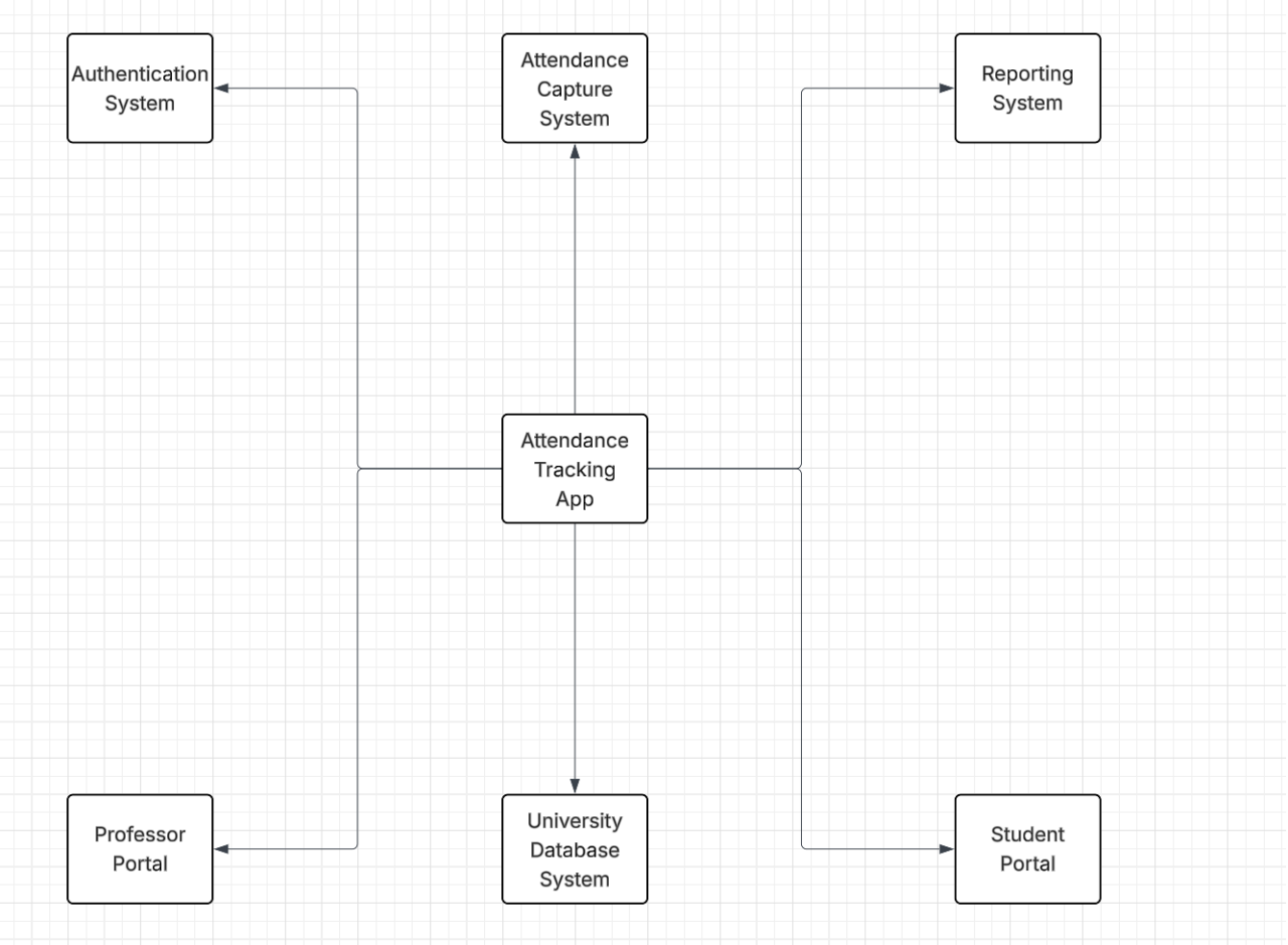
- Compliance with Lebanese university data protection and privacy regulations.
- Emergency absence handling must comply with university attendance policy.

Design

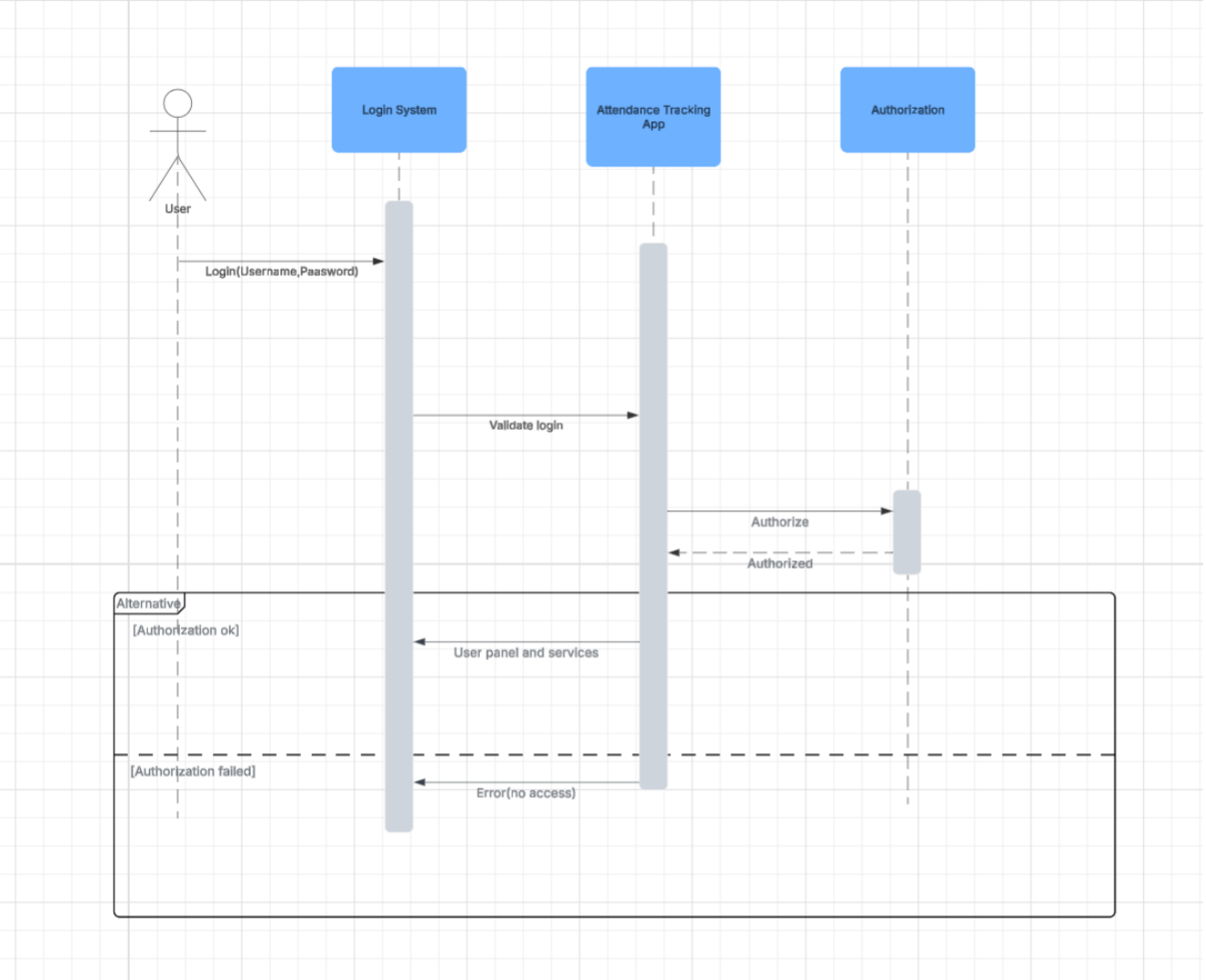
Class Diagrams



Context Diagram:



Sequence Diagram



Test Plans

Features to be tested / not to be tested
Pass/Fail Criteria
Approach
Suspension and resumption
Testing materials (hardware / software requirements)
Test cases
Testing schedule

Implementation

Output

Results Evaluation

Conclusion

Summary
Novelty
Integrity and Values
Future Work

References / Bibliography

1. Attendify. (n.d.). Retrieved from Attendify Website
2. Smart Attendance. (n.d.). Retrieved from Smart Attendance Website
3. Kumar, A., et al. (2020). "Automated Attendance Management System Using Face Recognition." *Journal of Educational Technology*, 12(3), 45-56.
4. Ali, B., et al. (2019). "Barcode-Based Attendance System for Universities." *International Journal of Computer Applications*, 178(15), 10-15.

Appendix

Glossary

Naming Conventions and Definitions

Code and links

User Manual