Group: Ahmad Sharkawi

Zakaria Khatib

Mohammad Al Ghali

Adnan Khalil

Mohammad Jomaa

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:

1. **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.
2. **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.
3. **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.
4. **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.

References

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.

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

1. 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.

2. 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 tailor**ed 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 for this project will be impacted by several constraints, including team availability, project complexity, and deadlines. Key constraints include:

  - Limited Working Hours: Team members may have varying schedules or work part-time, which could limit the number of hours worked on the project each week.
  - Project Phases: Certain phases, such as design, development, testing, and deployment, will need to be completed in sequence. Delays in one phase could push back the overall timeline.
  - Deadlines: A fixed delivery deadline may limit the flexibility in adjusting the project scope or timeline during development.

Project Issues

- **Authentication Issues**

  - **Problem**: Difficulties with user login, session management, and role-based access.
  - **Impact**: Inconsistent login states and potential security concerns.
  - **Resolution**: Investigating improvements to session handling, JSON Web Token (JWT) implementation, and role-based authentication.

- Risks
  1. **Security Risks**:
     - Insecure authentication could lead to unauthorized access.
     - Mitigation: Implement strong encryption and secure authentication methods.

  2. **Dependency on External Libraries:**
     - Risks with deprecated or unsupported third-party tools.
     - Mitigation: Monitor updates and ensure fallback solutions.

3. **User Adoption Risks:**
    - Resistance from users to adopt the new system.
    - Mitigation: Conduct user testing, refine UI, provide 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.

- Building the Website to a Mobile app: Ahmad Sharkawi

- Deployment:
  Ahmad Sharkawi: Responsible for deploying the application as a fully functional website and ensuring its seamless publication on both Android and iOS platforms.

Ethical Issues

1. Data Privacy and Security:
    - Problem: The app handles sensitive student data, including attendance records, which must be protected from unauthorized access.
    - Ethical Concern: Users (students, teachers, admins) have a right to privacy, and any data breach or misuse of their personal information would be unethical.
    - Resolution: Implement strong encryption for data storage and transmission, ensure compliance with data protection laws, and set up proper access controls to safeguard sensitive data.

2. Bias in Attendance Monitoring:

- Problem: If the systems algorithms or manual input by teachers are biased, it could lead to unfair attendance marking
- Ethical Concern: Bias in how attendance is recorded could disproportionately affect students and lead to unfair treatment.
- Resolution: Develop clear guidelines for teachers on how to record attendance accurately. Regularly review the system to ensure fairness and transparency.

3. Informed Consent:
    - Problem: Users should be made fully aware of how their data is being collected, stored, and used.
    - Ethical Concern: Collecting data without user consent or failing to inform users about how their data will be used violates ethical guidelines.
    - Resolution: Provide clear terms of service and privacy policies. Ensure users give explicit consent before their data is collected and allow them to opt out where applicable.

Software Model Process

1. **Waterfall Model:**

    - **Description**: A sequential approach where each phase of development must be completed before moving to the next. It's a linear process that works well for projects with well-defined requirements.

    - **Phases**:
        - **Requirement Gathering**: Understand and document the full set of requirements.
        - **System Design**: Plan architecture and design based on the requirements.
        - **Implementation**: Develop the software according to the design.
        - **Testing**: Perform QA testing to ensure the software meets requirements.
        - **Deployment**: Release the product to users.
        - **Maintenance**: Ongoing bug fixes and updates after release.

2. **Incremental Model:**

- Description: In this approach, the software is developed in small, manageable pieces or increments. Each increment adds functionality to the system until the final product is complete.

- Phases:
  - Initial Planning: Define overall system requirements and high-level design.
  - Incremental Development: Develop small portions of the system with each increment.
  - Testing: Test each increment as it's developed.
  - Release: Deploy incremental updates to users.
  - Final Release: Once all increments are completed, the full system is finalized and released.

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**
    - Define project goals and scope.
    - Gather detailed requirements from stakeholders (teachers, students, admins).
    - Outline user stories and map out core features.

2. **System Design and Architecture**
    - Design the database schema (tables for students, attendance records, user roles).
    - Create wireframes or prototypes for the user interface.
    - Choose and set up the development environment (Laravel, Blade , MySQL).
3. **Frontend Development (Initial Phase)**
    - Develop the core UI components (login page, dashboard, attendance marking).
    - Implement responsive design using Tailwind CSS.
    - Set up Blade files for dynamic, client-side interaction.
4. **Backend Development (Initial Phase)**
    - Set up the Laravel backend with necessary routes and controllers.
    - Develop core features (user authentication, attendance CRUD operations).
    - Implement basic routes to interact with the frontend.
5. **User Authentication & Authorization**
    - Implement JWT or Laravel Passport for user authentication.
    - Set up roles and permissions (admin, teacher, student) using Laravel's middleware.
    - Implement session management and security features (password hashing, encryption).
6. **Attendance Tracking System Development**
    - Develop the feature for marking attendance (manual input, barcode scanning).

- Create real-time attendance updates (if applicable, using Laravel Echo and Pusher).
- Implement logic for tracking student attendance over time.

7. **Integration with External Services (if required)**
    - Integrate any external APIs (for notifications, real-time updates, data visualization).
    - Set up Gemini API (if using third-party APIs for additional functionalities).
8. **Frontend-Backend Integration**
    - Integrate the frontend with backend (attendance data, user profiles, notifications).
    - Ensure smooth communication between frontend and backend.
    - Test all routes and UI components for correctness.
9. **Testing Phase**
    - Perform unit tests for backend logic (using PHPUnit).
    - Run integration tests (Postman) for API endpoints.
    - Perform end-to-end testing with Laravel Dusk.
10. **Bug Fixing and Quality Assurance**
    - Fix any bugs found during the testing phase.
    - Review code for performance, security, and best practices.
    - Conduct user acceptance testing to ensure the app meets requirements.
11. **Deployment Preparation**
    - Set up the production environment (Hostinger).
    - Prepare and run CI/CD pipelines using GitHub Actions.
    - Prepare the app building to publish on both Android and iOS platforms.
12. **Go-Live (Production Deployment)**
    - Deploy the app to the production server.
    - Ensure the app is fully functional and accessible to users.
    - Monitor real-time performance and user feedback.
    - Deploy the app on android and iOS
13. **Post-Deployment Monitoring and Maintenance**

- Monitor app performance and user behavior (using Google Analytics or similar tools).
- Address any immediate bugs or issues post-launch.
- Regularly update the app with new features, improvements, and security patches.

14. Project Handover & Documentation

- Finalize project documentation (user manual, API documentation, codebase).
- Provide training to the stakeholders (teachers/admins) on how to use the app.
- Handover the project for ongoing maintenance and support.