A Project Report on
Design and Development of Attendance Management System

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Electronics and Communication Engineering

Submitted by
Roll Number: 1900097
Registration Number: 18508005379
Session: 2018-2019
Department of Electronics and Communication Engineering (ECE)
National University, Bangladesh

Declaration

We declare that this project report entitled "Design and Development of Attendance Management System" is the result of our original work and that all sources of information have been duly acknowledged and referenced in the report.

We hereby affirm that this project report has not been submitted in whole or in part for any other academic degree or professional qualification. Furthermore, we confirm that all data, results, and findings presented in this report are authentic and accurate to the best of my knowledge.

We also declare that we have adhered to the ethical principles of academic research and have obtained the necessary ethical clearance from the relevant authorities, where applicable

Submitted by:

____

Reg: 18508005379

Supervised by:

____

Abu Hena Md. Mustafa Kamal
Assistant Professor
Department of ECE
Institute of Science and Technology

Acknowledgments

First of all, we would like to express my sincere gratitude and appreciation to the Almighty God and then all those who have contributed to the successful completion of this project report.

We would like to thank our supervisor, **Abu Hena Md. Mustafa Kamal** assistant professor, of the Institute of Science and Technology, for his invaluable guidance and support throughout the project. His expertise and encouragement have been instrumental in shaping the direction of this work and ensuring its quality.

We would also like to acknowledge the contributions of our friends and peers, who have provided us with valuable feedback, insights, and support at various stages of the project. Their collaboration has been crucial in ensuring the project's success.

Furthermore, we extend our thanks to the all-office staff of the **Institute of Science and Technology** for providing us with the resources and facilities necessary for the completion of this project. We are grateful for the opportunities and resources that the organization has provided us with.

Lastly, we would like to express our gratitude to our family for their unwavering support, encouragement, and understanding throughout this project. Their love and support have been a constant source of motivation and inspiration for us.

Once again, thank you to all those who have contributed to this project report. Your support and contributions have been instrumental in making this project a success.

Abstract

The University Attendance Management Project is a powerful, modern solution designed to streamline and improve student attendance tracking in educational institutions. By combining hardware like the ESP8266 and RFID technology with a robust, Laravel-based web application, this system aims to replace traditional attendance methods with a faster, more reliable digital approach.

The system uses RFID-enabled smart cards, allowing students to quickly and accurately log their attendance. These smart cards interact with RFID readers connected to ESP8266 modules, which enable real-time Wi-Fi transmission of attendance data to a central server. This immediate transfer of information ensures up-to-date and reliable attendance records.

The core of this project is a Laravel-powered web application, offering a simple, user-friendly interface for administrators to manage attendance data. With this application, administrators can view and update records, generate reports, and gain insights through analytics. Designed with HTML, CSS, and JavaScript, the interface is intuitive and accessible, making it easy to use for all users.

Security is a top priority in this system. Data is protected through encryption during transmission, and strict access controls prevent unauthorized access, ensuring the safety and privacy of student information.

This project represents a major advancement in attendance management. By integrating advanced technology with a seamless design, it brings efficiency, accuracy, and scalability to the way institutions manage attendance, offering a tailored solution that meets the evolving needs of today’s academic environments.

TABLE OF CONTENTS

List of Abbreviation

HTML	= Hypertext markup language
CSS	= Cascading style sheet
JS	= JavaScript
DFD	= Data Flow Diagram
API	= Application Programming Interface
RFID	= Radio Frequency Identification

List of Figures

CHAPTER 1: INTRODUCTION

1.1 Introduction

Our world is constantly progressing and becoming more advanced day by day. Along with this, our daily tasks are becoming easier and more error-free. Consequently, various institutions and educational establishments are employing smart card attendance systems to streamline their operations. Therefore, we are conducting further research on this method to make it more efficient and cost-effective. Our main objective is to create a cost-effective and efficient smart card attendance system. Simultaneously, we aim to make this method portable and more effective in its functionality.

Furthermore, we want to enable attendance monitoring through the Internet from any location. To achieve this, we plan to utilize both hardware and software components. For capturing attendance, we will initially use hardware components such as the ESP8266, Arduino nano, batteries, smart card modules, and some leads, etc. For software development, we will employ HTML, CSS, JavaScript, and Laravel to ensure that attendance is displayed and monitored attractively and efficiently

1.2 Motivation

The integration of smart card attendance systems into educational and institutional environments represents a significant leap forward in efficiency and accuracy. By embarking on this project, we are not only embracing technological advancement but also addressing the pressing need for streamlined operations and error-free attendance tracking. Our motivation stems from the desire to revolutionize traditional attendance methods, making them more cost-effective, portable, and accessible. With a focus on leveraging both hardware and software components, we aim to create a solution that not only captures attendance seamlessly but also enables monitoring from any location via the Internet. By harnessing the power of ESP8266, Arduino Nano, smart card modules, and innovative software technologies like HTML, CSS, JavaScript, and Laravel, we aspire to not just improve attendance tracking but to enhance overall operational efficiency and provide a user-friendly experience for administrators. This project represents our commitment to embracing innovation and harnessing technology to drive progress in educational and institutional settings.[01]

1.3 Project Objectives

The primary goal of our project is to design and implement a cost-effective and efficient smart card attendance system. This system aims to streamline the attendance tracking process through the integration of hardware components and robust software development.

- **Cost-Effective Solution:** Develop a smart card attendance system that is budget-friendly, utilizing economical hardware components without compromising functionality.
- **User-Friendly Interface:** Employ HTML, CSS, and JavaScript in software development to create an attractive and user-friendly interface for displaying and monitoring attendance data.
- **Laravel Integration:** The Laravel framework enhances the software's functionality, providing administrators with a comprehensive tool for managing attendance records efficiently.
- **Reliable Power Management:** Implement a reliable power management system to ensure continuous operation and longevity of the smart card attendance system.[02]

1.4 Expected Outcome

Upon completion of this project, we anticipate the development and implementation of a highly efficient and cost-effective smart card attendance system. This system will streamline the attendance tracking process through the integration of hardware components and robust software solutions. The anticipated outcomes include:

- **Enhanced Efficiency:** The implementation of the smart card attendance system is expected to significantly improve the efficiency of attendance tracking processes within educational and institutional environments.
- **Cost Savings:** By utilizing budget-friendly hardware components and implementing efficient software solutions, the project aims to achieve cost savings for institutions in terms of attendance management.
- **Improved User Experience:** The incorporation of a user-friendly interface using HTML, CSS, and JavaScript will enhance the overall user experience for both administrators and users, making attendance tracking more intuitive and accessible.
- **Comprehensive Attendance Management:** Integration with the Laravel framework will provide administrators with a comprehensive toolset for managing attendance records efficiently, ensuring accuracy and ease of use.
- **Sustainable Operations:** The implementation of a reliable power management system will ensure continuous operation and longevity of the smart card attendance system, contributing to sustainable practices within educational and institutional settings.[3]

1.5 Report Layout

Chapter 1: Introduction

In this chapter, an introduction to the project is presented, covering the core motivation behind its inception, objectives, and the expected outcomes that the project aims to achieve. The motivation section highlights the problems or gaps this project seeks to address, outlining why the project holds relevance in its field. The objectives define the specific goals that guide the project's development, establishing a clear direction for each phase. Lastly, the expected outcome section outlines the anticipated achievements or impacts resulting from the project's successful implementation. This chapter concludes by describing the overall structure of the report, providing readers with a roadmap for navigating the following sections.

Chapter 2: Literature Review

This chapter explores the foundational background and context for the project. It discusses existing research and previous work related to the project domain, providing insights into the problem scope and potential challenges the project may face. The literature review helps identify the project's position within the broader field, highlighting how it builds on or diverges from existing solutions. This chapter also examines any existing gaps in technology or methodology that this project aims to address, thereby establishing the necessity and relevance of the work undertaken.

Chapter 3: Technology Stack

Chapter 3 provides a comprehensive overview of the technology stack utilized in the project. It details the chosen tools, platforms, and frameworks critical for implementing the project's specifications and meeting functional requirements. Key components include RESTful API design principles that facilitate communication between hardware and software elements, as well as hardware components like the RFID module, which enable data collection and authentication. Wi-Fi connectivity is achieved using the ESP8266 module, supporting real-time data transfer,

while Laravel serves as the primary backend framework, allowing efficient
data handling and storage. This chapter lays the technical groundwork for the project's implementation, describing how each component fits into the overall system architecture.

Chapter 4: Design Specification

In this chapter, the design of the system is thoroughly documented, beginning with the conceptual model and progressing to detailed technical diagrams. A conceptual model provides an abstract view of the system, capturing the primary entities and their interactions. This is followed by an entity-relationship model (ER model), which illustrates data organization and relationships between various system components, ensuring a structured approach to data management. Additionally, flowcharts visually represent the processes and workflows within the system. Together, these design specifications offer a blueprint for implementing the project in a structured and coherent manner, ensuring alignment with the functional requirements.

Chapter 5: Implementation

This chapter provides a detailed account of the project's implementation, focusing on the step-by-step processes involved in setting up and configuring the core hardware components, including the ESP8266, and RFID module. Practical instructions are provided for uploading code to these devices, enabling them to perform their designated functions within the system. The chapter also covers the process of sending data via HTTP requests and using APIs to store attendance data, facilitating seamless data transfer between devices and the backend. Additionally, this section includes the creation and configuration of APIs to enable efficient data storage, serving as the backbone of the system's attendance management functionality.

Chapter 6: Software Implementation

Chapter 6 centers on the software design aspects of the project, with a particular focus on the user interface for the admin panel. It provides an in-depth look at the principles and considerations that guided the interface design, ensuring it is both intuitive and functional for administrators. Attention is given to creating a layout that simplifies navigation, enhances user experience, and offers necessary tools for managing and monitoring data effectively. This chapter outlines the steps taken to ensure that the software interface aligns with the project's overall objectives, catering to user needs and ensuring efficient system operation.

Chapter 7: User Interface

The Home Page Interface introduces users to the system with a clear and organized layout, allowing for straightforward navigation to the primary sections. This interface serves as the gateway to the platform, offering a welcoming and functional design that guides users to essential features with ease.

The Login Page Interface provides a secure entry point where users input their credentials to access the system. Designed for simplicity and functionality, it includes fields for username and password, along with options for password recovery to ensure a smooth, secure login process for all users.

Once logged in, users are directed to the Dashboard Interface which serves as the control center. Here, users can view essential data summaries and access links to frequently used features. The dashboard's design focuses on efficiency, presenting a comprehensive yet organized snapshot of system activities to streamline the user experience.

The General Website Settings Interface allows administrators to configure basic settings within the platform, such as adjusting site preferences and updating information.

This interface offers the flexibility needed to align the system’s foundational settings with specific organizational requirements, enhancing customization options for administrators.

The Session Management Interface offers tools to oversee user sessions, allowing administrators to set session durations and view active sessions. This feature is designed to manage access control effectively, optimizing both security and system performance by ensuring that sessions are appropriately monitored and regulated.

In the Department Management Interface, administrators can manage organizational structure by adding, editing, or deleting departments. This interface provides a simple yet powerful tool to maintain an up-to-date department hierarchy, supporting clear user role assignments within the system.

The Student and Attendance Management Interface is a central hub for organizing student records and tracking attendance data. It enables users to search for specific student profiles, view detailed records, and update attendance information, ensuring that data is accurate and easily accessible.

The Student Addition Interface is designed for adding new student entries to the system. This form captures essential information, including identification and contact details, to ensure that each student’s record is comprehensive and accurately maintained within the system’s database

Chapter 8: Conclusion and Future Scope

The final chapter provides a summary of the project, reflecting on its achievements, limitations, and broader implications. The conclusion discusses the key outcomes, evaluating how well the project met its original objectives and addressing any challenges that arose during development. In addition, the future scope section explores potential avenues for further development, suggesting enhancements or additional features that could expand the project’s functionality or improve its performance. This closing chapter provides a comprehensive wrap-up, highlighting the project’s contributions and possibilities for future advancements in the field.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The Attendance Management System (AMS) is an indispensable tool in today's organizational landscape, offering a sophisticated and automated approach to monitor and oversee student attendance. In educational institutions, AMS plays a pivotal role in ensuring that students are present for their classes, lectures, and other academic activities. By leveraging advanced technologies such as biometric recognition, RFID, or mobile applications, AMS provides real-time updates on attendance, enabling educators to identify patterns, address absenteeism, and make informed decisions to enhance student engagement and academic performance.
In conclusion, the Attendance Management System represents a transformative solution that revolutionizes the way organizations manage attendance, whether in educational institutions or professional workplaces. By embracing automation, precision, and efficiency, AMS empowers stakeholders to optimize resource allocation, enhance operational efficiency, and foster a culture of accountability, ultimately driving organizational success in today's dynamic and competitive landscape.

- **Accurate Data Collection:** Attendance Management Systems (AMS) leverage advanced technologies such as RFID, biometric scanners, and automated check-ins to minimize human error associated with manual record-keeping. This ensures that attendance data is accurate, reducing discrepancies and enhancing reliability for academic and administrative use.

- **Time Efficiency:** By automating the attendance process, AMS drastically reduces the time teachers and administrators spend on attendance-related tasks. This efficiency allows educators to focus on core responsibilities like lesson planning, student engagement, and personalized instruction, ultimately contributing to a more productive learning environment.

- **Security and Privacy:** With technologies such as biometric authentication and encrypted data storage, AMS enhances the security of attendance records. Only authorized personnel have access to sensitive data, which not only ensures compliance with privacy regulations but also safeguards student information from unauthorized access.

- **Informed Decision-Making:** AMS systems provide detailed attendance analytics, enabling educators to monitor attendance trends over time. By identifying patterns in student attendance, such as frequent absenteeism, AMS allows for proactive intervention. This data-driven approach supports more targeted engagement strategies, helping schools implement support programs to improve attendance and academic performance.

- **Resource Optimization:** With streamlined processes, AMS enables schools to allocate resources more efficiently. Reduced paperwork and administrative overhead mean that staff can dedicate more time and resources to educational activities, extracurricular programs, and student support services, enhancing the overall educational experience.

In conclusion, the **Attendance Management System** represents a transformative shift, offering enhanced **efficiency**, **accuracy**, and **security** in attendance tracking. By adopting AMS, institutions and organizations can optimize resources, enhance operational efficiency, and foster accountability. In a competitive and data-driven environment, AMS has become a crucial tool for **organizational success**.

2.2 Literature Survey

The literature on Attendance Management Systems spans several domains, including educational technology, organizational management, and data security. This section explores traditional attendance methods, the limitations they present, and the technological advancements in modern AMS.

Historically, attendance tracking has relied on manual processes like attendance registers, roll-calls, or sign-in sheets. While widely used, these traditional methods have significant limitations:

- **Inaccuracy:** Manual data entry can lead to human errors, such as incorrect entries or missing information, compromising data reliability.
- **Time-Consuming Process:** Collecting attendance manually is a repetitive task that consumes valuable time during class or organizational meetings.
- **Susceptibility to Manipulation:** Manual systems are vulnerable to fraudulent practices, like "proxy attendance," where one person marks attendance on behalf of another.

Traditional attendance methods struggle to meet the accuracy, efficiency, and security demands of today's academic and organizational environments. Some key drawbacks include:

- **Data Inconsistency:** Discrepancies in attendance data often arise when transferring or compiling records, impacting analysis and reporting accuracy.
- **Lack of Real-Time Monitoring:** Manual methods do not offer real-time data, which delays the identification of absenteeism patterns and limits timely intervention.
- **Inefficient Record Management:** With physical registers, managing and accessing historical data can be cumbersome and time-intensive.

SAMS systems are designed to integrate seamlessly .This integration offers several benefits:

- **Enhanced Data Accessibility:** Allows data to be viewed, analyzed, and reported across various departments and systems.
- **Customization:** SAMS can be tailored to meet the specific needs of institutions, from class-level tracking to organizational-level analytics.
- **Scalability:** Systems are designed to handle varying levels of demand, from small classes to large institutions, ensuring long-term usability as institutions grow.

CHAPTER 3. TECHNOLOGY STACK

3.1 Application Specification

The Application Specifications for this development of an Attendance Management System are as follows:

- **Enhanced Efficiency:** Implementing the smart card attendance system is expected to significantly improve the efficiency of attendance tracking processes within educational and institutional environments.
- **Cost Savings:** By utilizing budget-friendly hardware components and implementing efficient software solutions, the project aims to achieve cost savings for institutions in terms of attendance management.
- **Improved User Experience:** The incorporation of a user-friendly interface using HTML, CSS, and JavaScript will enhance the overall user experience for both administrators and users, making attendance tracking more intuitive and accessible.
- **Comprehensive Attendance Management:** Integration with the Laravel framework will provide administrators with a comprehensive toolset for managing attendance records efficiently, ensuring accuracy and ease of use.

3.2 Functional Requirements

- **User Authentication**: The system should authenticate users (administrators) securely to ensure that only authorized can access and manage attendance data.
- **Attendance Tracking**:
    - The system should be able to capture attendance data accurately using RFID modules for smart card reading and also capture attendance in the admin panel.
    - It should record the date and time of attendance for each student.
- **Real-time Data Updates**: Attendance data should be updated in real time to reflect any changes or additions immediately.
- **User Management**: Administrators should have the ability to add, or modify user accounts (students, etc.) as needed.
- **Reporting and Analytics**: The system should generate comprehensive reports on attendance statistics, including daily attendance summaries, individual attendance records, and trends over time.
- **Customization and Configuration**: Administrators should have the flexibility to customize system settings, such as attendance thresholds, and preferences.

1. **Scalability**: The system should be scalable to accommodate varying numbers of users and locations, whether it's a small classroom or a large organization with multiple branches.
2. **Security**: The system should adhere to best practices for data security, including encryption of sensitive information, protection against unauthorized access, and regular security audits.

3.2 Functional Requirements

3.3.1 RESTful API Design Principles:

This section emphasizes the significance of following RESTful API design principles. It delineates the need for designing APIs that adhere to the REST architectural style, promoting simplicity, scalability, and interoperability. By adhering to these principles, the system can facilitate seamless communication and data exchange between various components, ensuring efficiency and flexibility in system integration.

3.3.2 ESP8266 and RFID Module:

In this section, the ESP8266 is utilized as the microcontroller platform. The ESP8266 offers Wi-Fi connectivity, making it ideal for wireless data transmission. The RFID module is responsible for reading RFID smart cards, enabling the efficient capture of attendance data. Integrating the ESP8266 with the RFID module creates a streamlined system for attendance tracking, where the data can be processed and transmitted in real time for logging or remote monitoring purposes. [05]

3.3.3 ESP8266 and Wi-Fi Connectivity:

This subsection underscores the importance of ESP8266 and Wi-Fi connectivity in enabling remote communication and data transfer. ESP8266, integrated with the system, facilitates wireless connectivity, allowing for real-time monitoring and management of attendance data from any location with internet access. Wi-Fi connectivity ensures accessibility and enhances the system's capability to adapt to modern networking requirements.[06]

3.3.4 Laravel Framework:

The adoption of the Laravel framework for backend development is highlighted in this section. Laravel provides a robust and feature-rich framework for building scalable and maintainable web applications. Leveraging Laravel's extensive capabilities, the system can efficiently manage data, create APIs, and implement business logic, thereby enhancing the system's reliability, performance, and maintainability.

3.3.5 HTML, CSS, and JavaScript:

This portion focuses on the front-end development aspect of the project, emphasizing the use of HTML, CSS, and JavaScript. HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript collectively form the foundation for creating an intuitive and visually appealing user interface. HTML structures the content, CSS styles the presentation, and JavaScript adds interactivity, resulting in a seamless user experience.[09]

3.3.6 MySQL

The utilization of MySQL for database management is implied in this subsection. MySQL, a popular relational database management system (RDBMS), provides the necessary infrastructure for storing, retrieving, and manipulating attendance records and related data.Byleveragin

MySQL, the system ensures efficient data storage, retrieval, and management, facilitating seamless integration with the application's backend logic.

CHAPTER 4. SYSTEM DESIGN

4.1 Design of Attendance Manegement System

The Conceptual Model section provides a foundational understanding of the conceptual structure and relationships within the attendance management system. It offers a high-level overview of the system's key components, interactions, and functionalities, serving as a blueprint for further development and implementation.

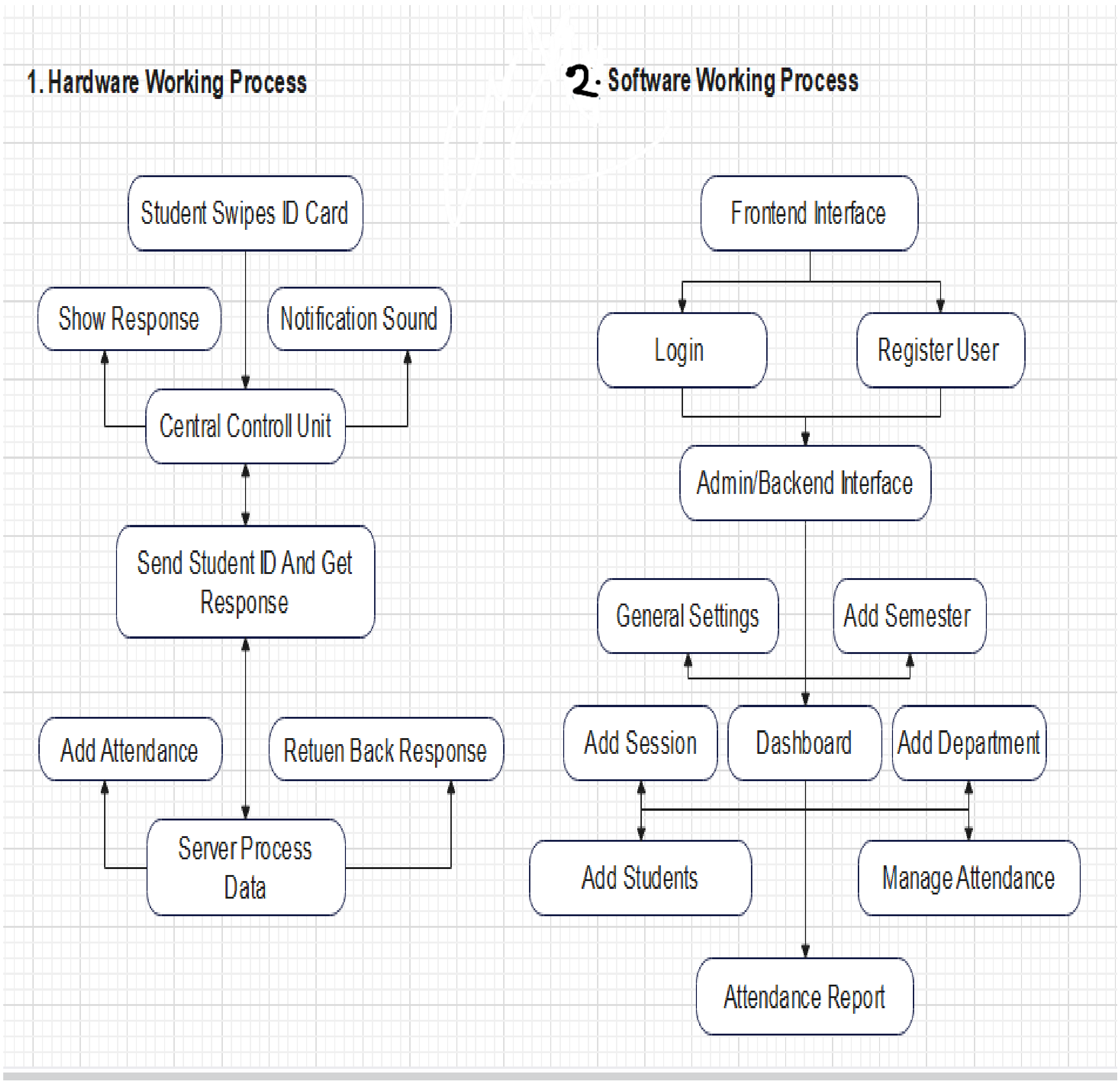


Fig-4.1: System Design of The Attendance Management System

4.2 Circuit Diagram

The Circuit Diagram section provides a visual representation of the hardware configuration and connectivity within the smart card attendance system. It offers a detailed schematic depicting the interconnections between different hardware components, including RFID modules, ESP8266, LCDs, and buzzers. The figure is shown in below:

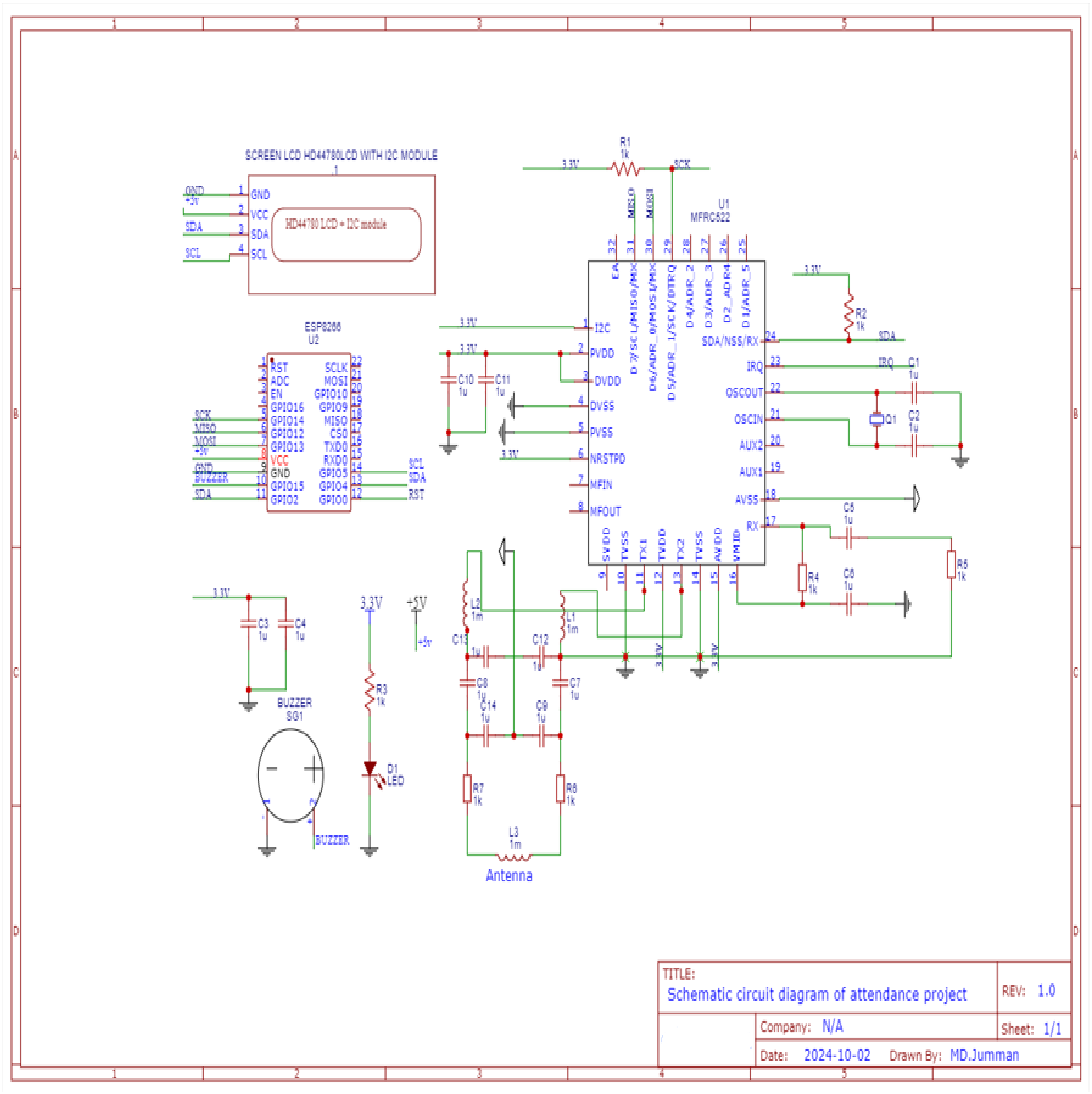


Fig 4.2: Circuit diagram of The Attendance Management System

4.3 Data flow Diagram (DFD)

The Data Flow Diagram (DFD) section presents a graphical representation of the flow of data within the smart card attendance system. It outlines the movement of information between various components, including input sources, processing units, storage systems, and output destinations**.**

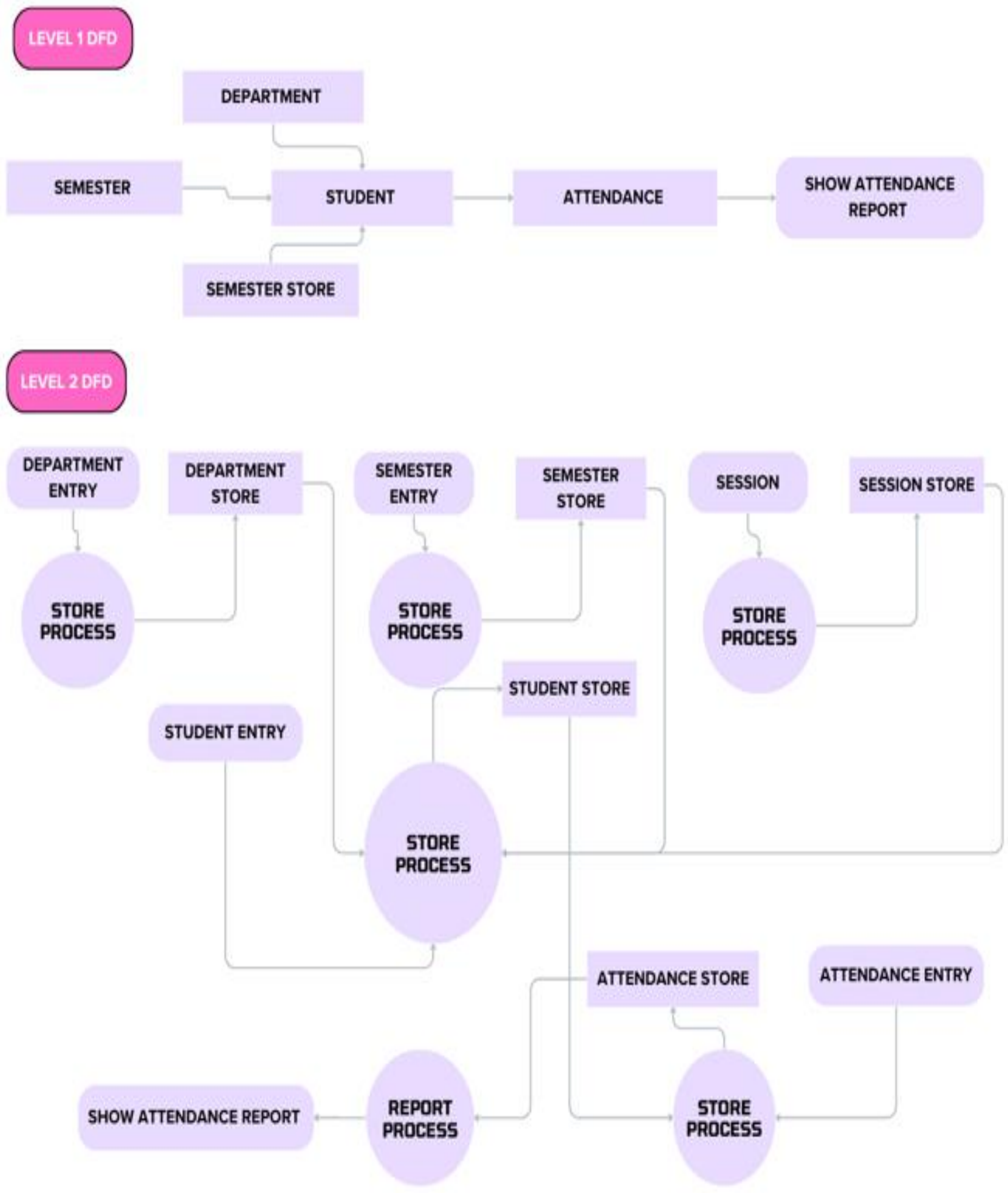


Fig4.3: DFD diagram of The Attendance Management System

4.4 Entity-Relationship Model

The Entity-Relationship Model (ER Model) provides a structured representation of the data schema and relationships within the smart card attendance system. It defines the entities, attributes, and relationships involved in storing and managing attendance data.

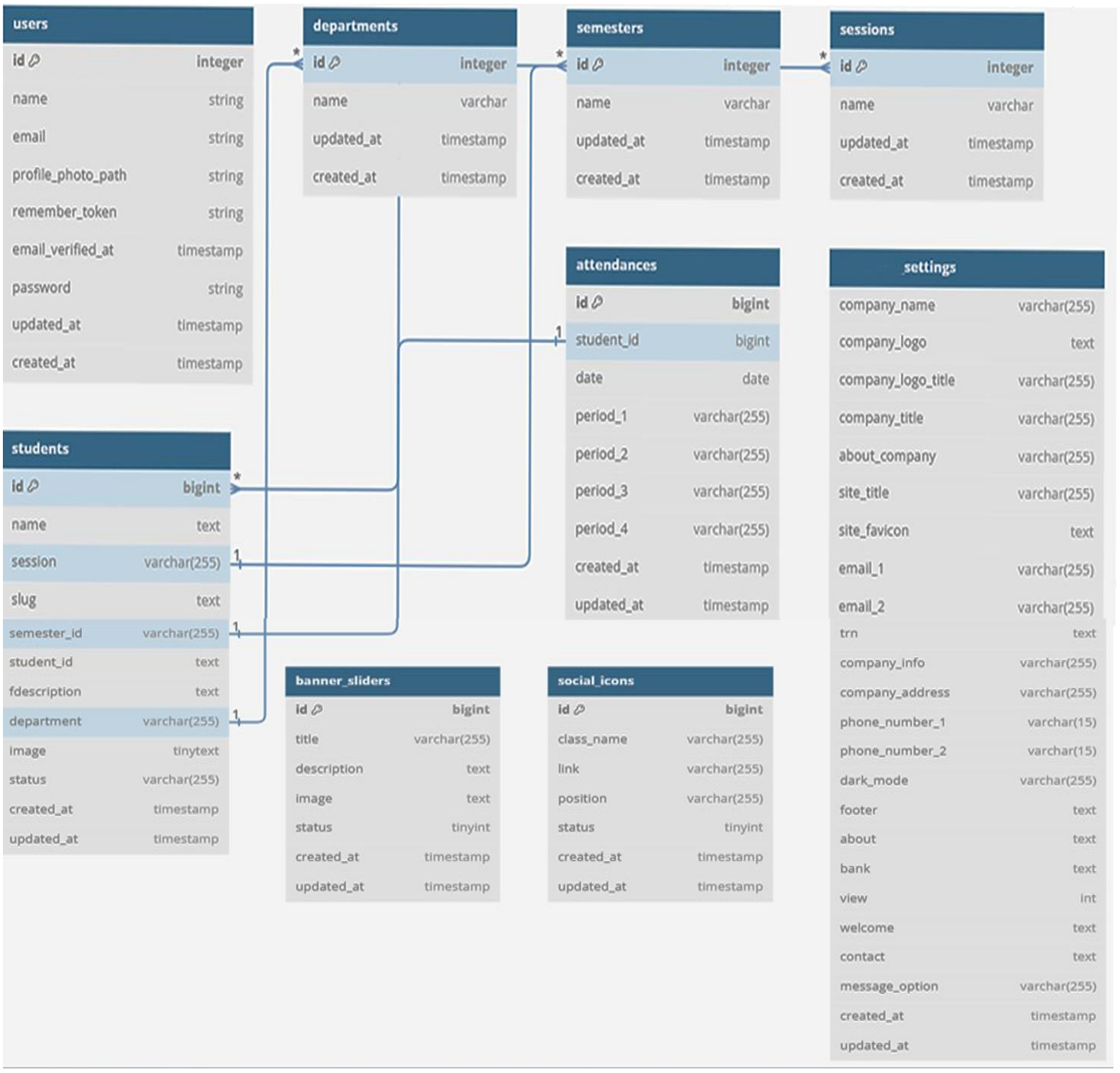


Fig4.4: ER diagram of The Attendance Management System

4.4 Flowchart Diagram

Creating a flowchart diagram for a Student Attendance Management System (SAMS) involves visually representing the steps and decision points in the process. Below is a simplified example of a flowchart for a generic SAMS. Keep in mind that the complexity of the flowchart can vary based on the specific features and functionalities of the system you are designing.

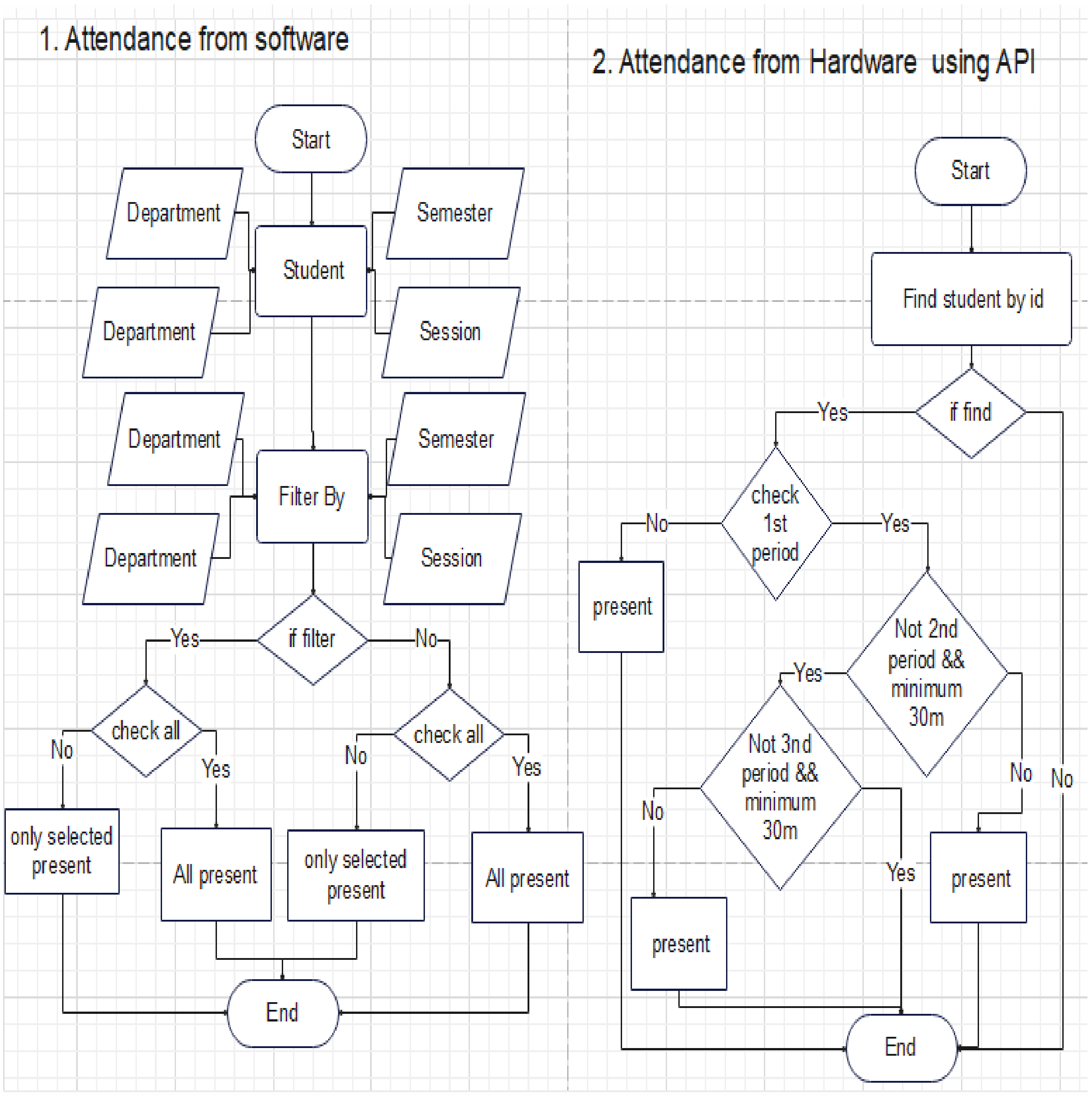


Fig-4.5: flowchart diagram of The Attendance Management System

CHAPTER 5. IMPLEMENTATION

5.1 Hardware Implementation

The Hardware Implementation section outlines the practical setup and integration of the core hardware components used in the attendance management system. Each component plays a crucial role in ensuring the system functions reliably and efficiently. The following subsections provide a step-by-step guide to the configuration and wiring of the ESP8266 Wi-Fi module, MFRC522 RFID reader, LiquidCrystal_I2C display, buzzer, and battery status monitoring system.

5.1.1 Setting up the ESP8266 Wi-Fi Module

The ESP8266 Wi-Fi module is essential for enabling wireless communication between the attendance system and the server, facilitating real-time data transfer and synchronization. By configuring the module to operate in *station mode*, it connects to a local Wi-Fi network using the designated SSID and password. This allows the system to transmit attendance records and user data to the server without the need for wired connections.

The setup involves programming the ESP8266 to connect to the Wi-Fi network automatically at startup, ensuring a seamless connection every time the system powers on. Once connected, the module is responsible for ensuring the reliable and continuous flow of data between the system and the server, making it possible for real-time updates on attendance records and status reports.

This configuration is critical for maintaining the accuracy and timeliness of the attendance data, allowing administrators to monitor and retrieve information instantly. The figure below provides a visual representation of how the ESP8266 Wi-Fi module is wired and integrated with the microcontroller and other system components, offering a clear guide for hardware setup.

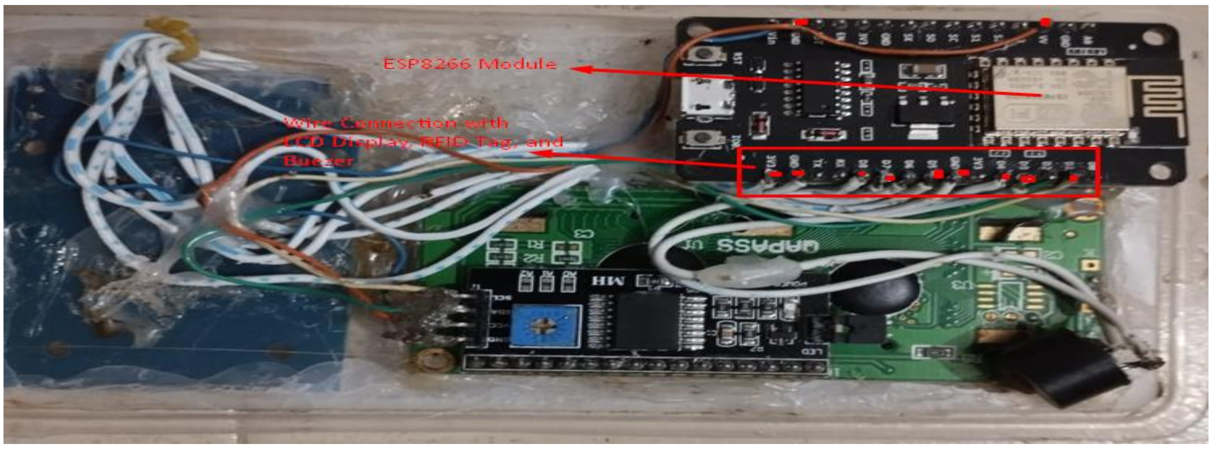


Figure 5.1: ESP8266 Wi-Fi Module Setup

5.1.2 Configuring the MFRC522 RFID Reader

The MFRC522 RFID reader plays a vital role in reading RFID tags, which serve as user identification cards for the system. This section covers the wiring process, connecting the RFID reader's SPI interface to the microcontroller, allowing smooth data transmission. Once wired, the software is configured to detect and authenticate each unique tag ID. The collected data is then used to log attendance and verify user identity in real time. The figure below shows the wiring setup of the MFRC522, illustrating its integration within the system.

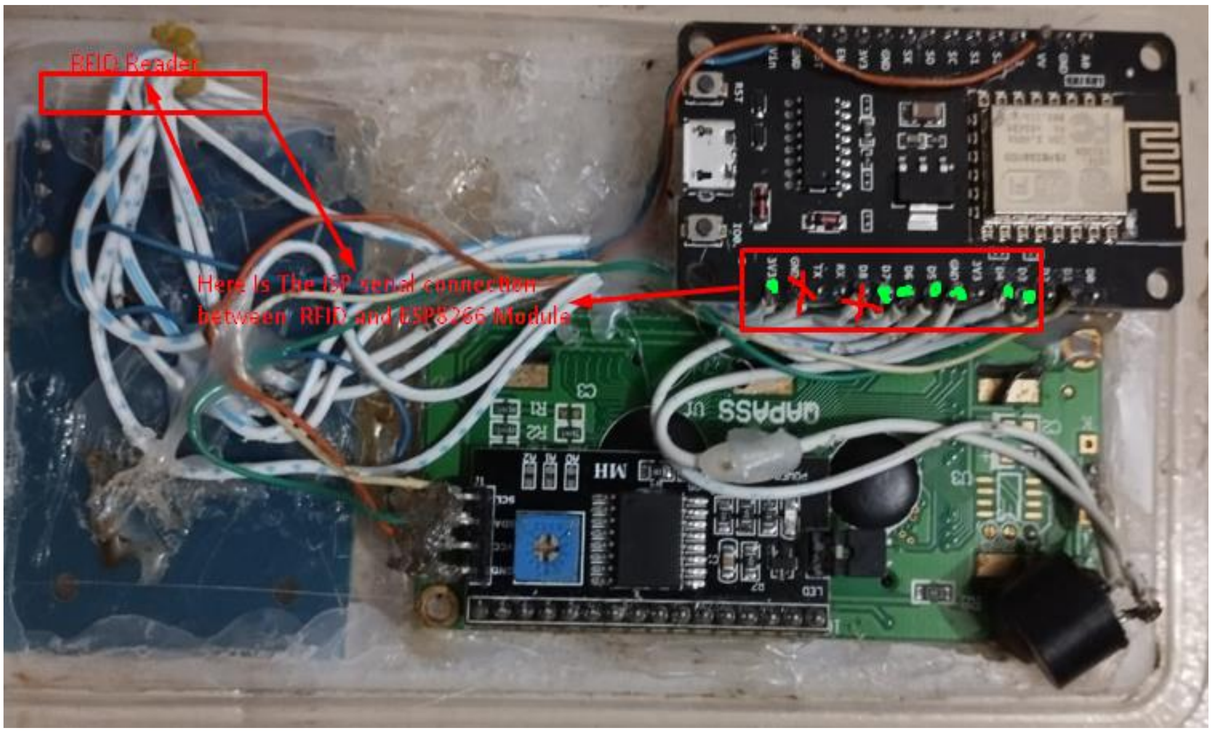

RFID Reader
Here Is The ISP serial connection between RFID and ESP8266 Module

Figure 5.2: MFRC522 RFID Reader Setu

5.1.3 Integrating the LiquidCrystal_I2C Display

The LiquidCrystal_I2C display serves as a user-friendly interface for the system, showing essential information such as system status, and notifications. This section explains how to integrate the display by wiring it to the microcontroller using the I2C interface, which reduces the number of pins needed for connection. After the hardware setup, the microcontroller is programmed to send relevant data to the display in real-time. This feature allows users to view critical system alerts and updates on-screen as they occur. The figure below illustrates the wiring setup of the LiquidCrystal_I2C display within the system.

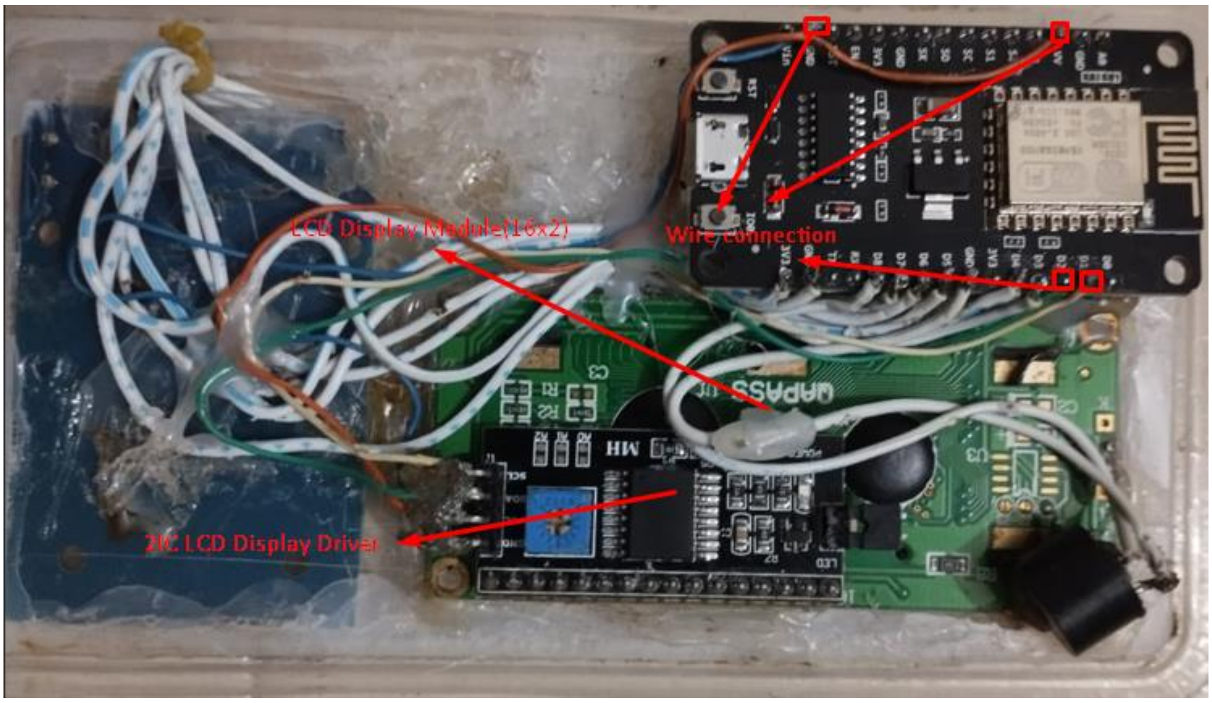


Figure 5.3: LiquidCrystal_I2C Display Setup

5.1.4 Wiring the Buzzer

The buzzer provides essential audio feedback, signaling the success or failure of RFID scans and alerting users to system errors. This section outlines how to wire the buzzer to the microcontroller and configure it to respond to specific events, such as correct tag identification or invalid entries. The buzzer enhances the user experience by offering immediate auditory feedback, helping users interact with the system efficiently. The figure below illustrates the wiring setup of the buzzer within the system.

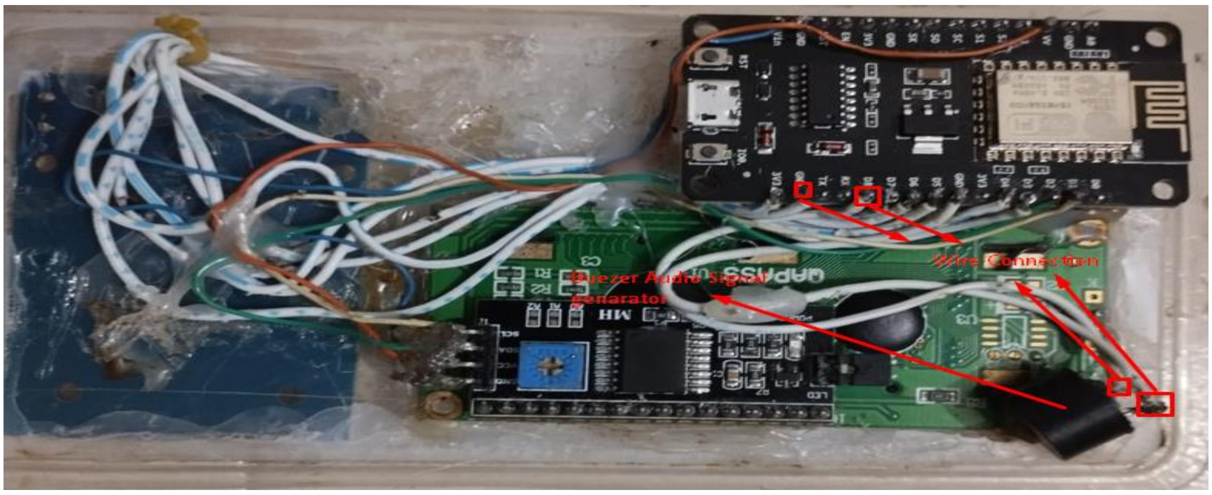


Figure 5.4: Buzzer Wiring Setup

5.1.5 Battery

For continuous operation, especially in portable setups, the system uses a Li-1850 battery. This section describes how the battery is integrated into the system, providing the necessary power for operation. The wiring setup connects the battery to the microcontroller and other components, ensuring reliable power delivery. This configuration allows for efficient energy use and supports the system's performance. The figure below illustrates the wiring setup for the Li-1850 battery.

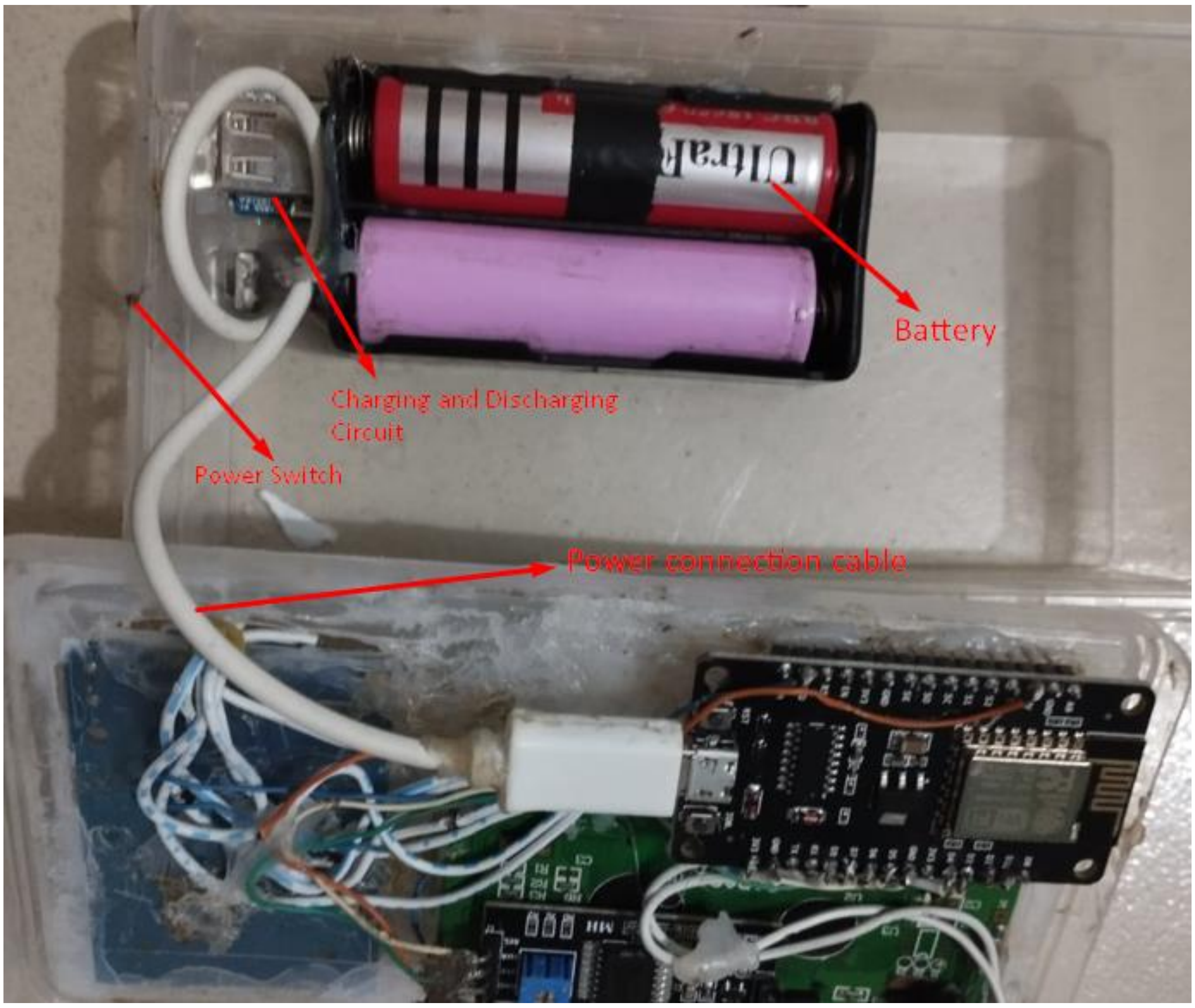


Figure 5.5: Li-1850 Battery Setup

5.2 Software Implementation

This section covers the software implementation for the attendance system, focusing on writing and uploading the code, managing Wi-Fi connections, processing RFID data, handling HTTP communication, and providing user feedback through display and audio. The software ensures seamless interaction between the hardware components and the system, facilitating real-time data processing and communication.

5.2.1 Writing and Uploading Code to the Microcontroller

This section details the process of writing the code required for the microcontroller to control various components like the ESP8266, RFID reader, and display. The code is developed in the Arduino IDE and uploaded to the microcontroller via a USB connection. Proper libraries and functions are implemented to ensure efficient data handling and communication between components.

(Source code: page 30)

5.2.2 Wi-Fi Connection and Reconnection Logic

To ensure continuous data transmission, the system is programmed to automatically connect to the specified Wi-Fi network upon startup. This section explains the Wi-Fi connection logic, including how the system handles reconnections in case of network interruptions. The code includes checks for network availability and reconnects the ESP8266 module when the connection is lost. The figure below demonstrates the Wi-Fi connection process and reconnection logic.

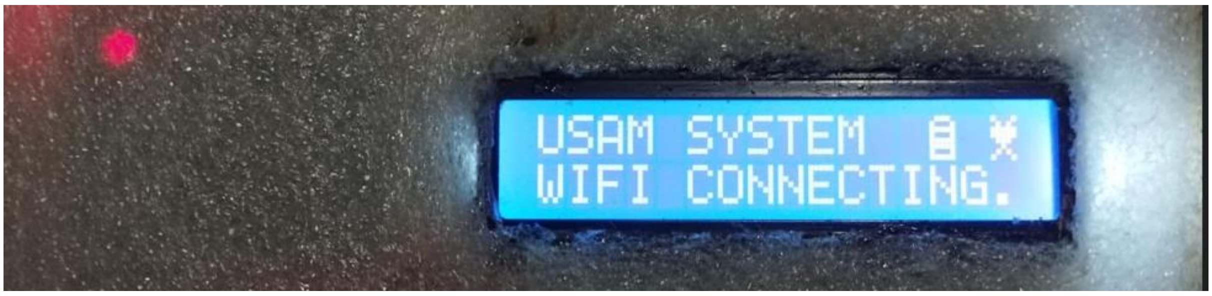


Figure 5.6: Wi-Fi Connection and Reconnection (Source code: page 30)

5.2.3 RFID Data Reading and Processing

In this section, the code for reading and processing RFID data is discussed. The microcontroller reads unique RFID tag IDs from the MFRC522 RFID reader and processes this data to log attendance and verify user identity. The figure below provides a visual representation of the RFID data reading process and its flow through the system.

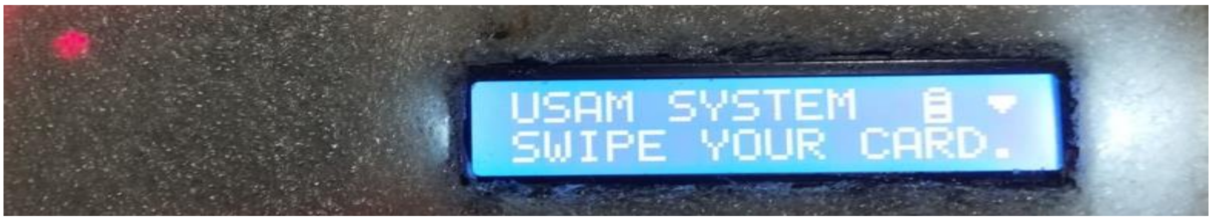


Figure 5.7: RFID Data Reading and Processing (Source code: page 30)

5.2.4 HTTP Client and Server Communication

This section describes how the system communicates with the server through HTTP requests and responses. The microcontroller acts as an HTTP client, sending data (such as RFID scans) to the server and receiving responses. The code ensures that communication is reliable, with error-handling mechanisms in place for network issues. The figure below shows the flow of HTTP communication between the client and server.

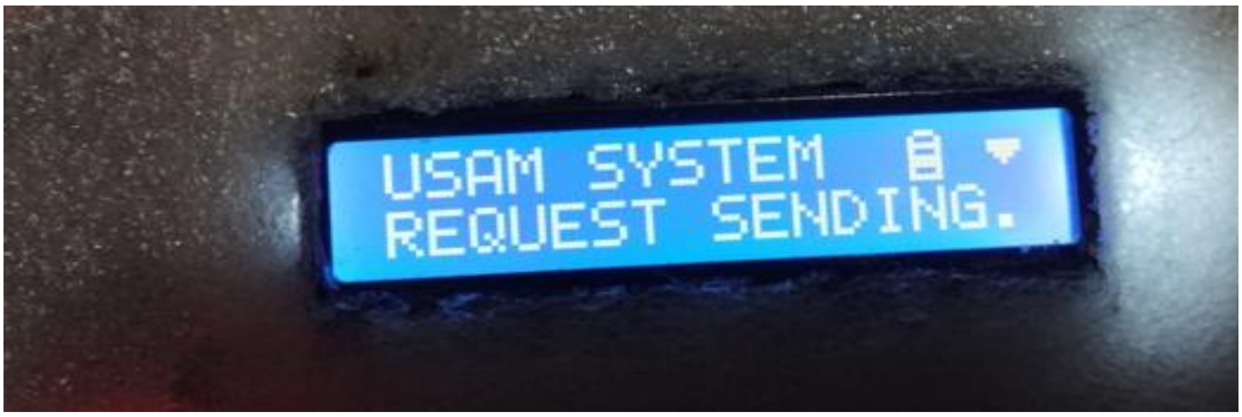


Figure 5.8: HTTP Client and Server Communication (Source code: page 30)

5.2.5 Display and Feedback Mechanisms

The LiquidCrystal_I2C display and buzzer provide real-time feedback to users, showing system status and notifications. This section explains how the system sends data to the LCD for display and triggers the buzzer for audio feedback. These mechanisms are crucial for ensuring user interaction and system transparency.

CHAPTER 6. SOFTWARE IMPLEMENTATION

6.1 Introduction to Laravel Framework

The Laravel framework, a powerful and elegant PHP-based web application development framework. Laravel simplifies the process of building robust web applications by providing tools for routing, templating, and managing databases. Its modern, developer-friendly syntax makes it an excellent choice for building dynamic, scalable websites and applications.

6.1.1 Setting up Laravel

The setup process for Laravel involves installing the framework, configuring a local development environment, and initializing a new project. This section details the installation of Composer, the PHP dependency manager used to install Laravel, and setting up local servers like Apache or Nginx. It also covers the creation of a new Laravel project, configuration of environment variables, and setting up database connections to ensure a smooth development workflow.

6.1.2 Configuring Routes and Controllers

Routing and controllers form the backbone of any Laravel application. This section explains how to define routes that map to specific controller actions, enabling the application to respond to HTTP requests. It covers the setup of controllers that handle logic for processing requests and returning views or data, and includes examples of creating routes for simple pages and RESTful endpoints to manage data.

(Source code: page 30)

6.2 Front-End Design

Laravel's Blade templating engine allows for a smooth integration between back-end logic and front-end design. This section covers the principles of designing a clean, intuitive, and responsive user interface, ensuring that users can easily interact with the application. It also touches on the importance of front-end frameworks like Bootstrap or Tailwind CSS to style Laravel applications.

(Source code: page 30)

6.2.1 Creating Blade Templates

Blade is Laravel’s powerful templating engine, allowing developers to structure and extend HTML layouts easily. This section describes how to create and manage Blade templates, including setting up master layouts, incorporating dynamic content with Blade directives, and building reusable components. It also explains how to pass data from controllers to Blade templates, ensuring a seamless flow between back-end and front-end.

(Source code: page 30)

6.2.2 Designing User-Friendly Interfaces

Designing a user-friendly interface is essential for enhancing user experience. This section explores the principles of good UI/UX design, such as simplicity, consistency, and responsiveness. It provides guidelines on how to design forms, tables, navigation menus, and interactive elements, making the web application visually appealing and easy to use across different devices and screen sizes.

6.3 Backend Integration with SQL

In this section, we explore the integration of the Laravel application with a database using SQL. Efficient database design and interaction are crucial for storing and managing data in a web application. Laravel provides powerful tools, such as Eloquent ORM and migrations, to streamline the process of connecting to a database, designing schemas, and performing operations like querying and updating data.

6.3.1 Database Design and Schema

The Database Design and Schema focuses on designing a robust database schema that reflects the application's data structure. It covers key concepts like defining tables, relationships between entities (e.g., one-to-many, many-to-many), and normalizing data to ensure efficient data storage and retrieval. It also discusses the use of Laravel's migration system to create and modify database schemas in a controlled and versioned manner.

6.3.2 Implementing Models and Migrations

Models in Laravel represent the entities in your database and allow for interaction with tables through the Eloquent ORM. This section explains how to define models that map to database tables, and how to use Laravel migrations to create and update tables without manually writing SQL. It also highlights best practices for implementing relationships, validations, and accessing data through models.

(Source code: page 30)

6.3.3 CRUD Operations

Create, Read, Update, and Delete (CRUD) operations are the basic actions performed on database records. This section delves into how to implement these operations using Laravel’s Eloquent ORM. It covers defining routes and controllers to manage these operations and providing examples of creating new records, retrieving data, updating existing records, and deleting unwanted data.

(Source code: page 30)

6.4 API Development

APIs (Application Programming Interfaces) allow external systems or clients to interact with your application. This section explains how to build and expose RESTful APIs in Laravel, enabling data storage and retrieval from external applications, mobile apps, or other services. It covers best practices for API design, security, and optimization.

6.4.1 Creating APIs for Data Storage and Retrieval

This subsection focuses on setting up routes, controllers, and methods to handle HTTP requests for storing and retrieving data through APIs. It explains how to structure a RESTful API in Laravel and how to send and receive data in JSON format, which is a common standard for API communication. Example implementations include creating APIs for user authentication, data submission, and fetching records from the database.

(Source code: page 30)

6.4.2 Securing API Endpoints

Security is a critical aspect of API development. This section describes how to secure API endpoints using Laravel’s built-in authentication and authorization features. It covers techniques like token-based authentication (using Laravel Passport or Sanctum), rate limiting to prevent abuse, and implementing proper validation and error handling for incoming requests.

CHAPTER 7. USER INTERFACE

7.1 Home Page

The Home Page provides general information about the university, including department details, the university's mission and goals, faculty profiles, course offerings, and the current class schedule. This frontend view is designed to give visitors an overview of the university's key information

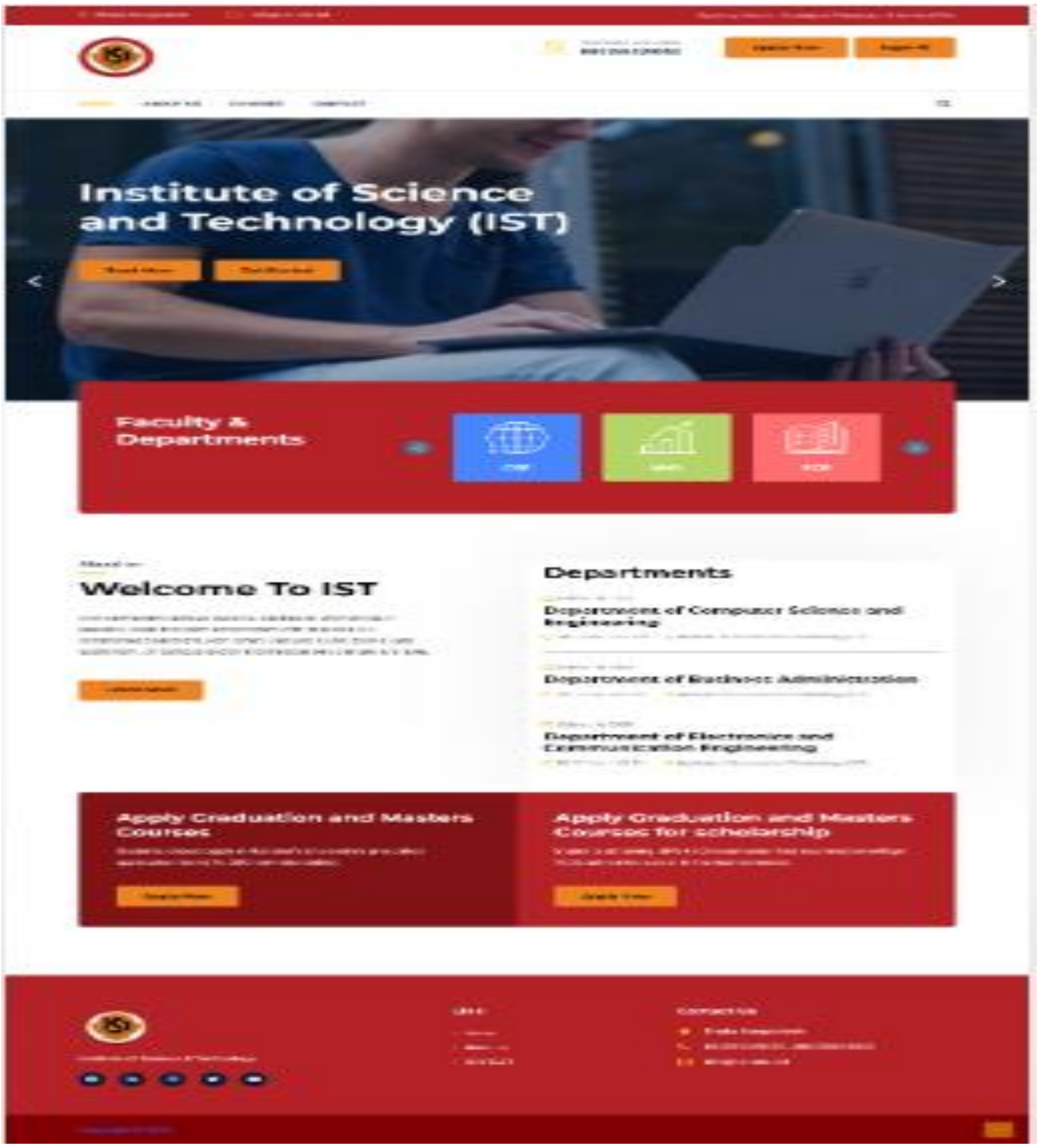

Institute of Science and Technology (IST)
Faculty & Departments
Welcome To IST
Departments
Department of Computer Science and Engineering
Department of Business Administration
Department of Electronics and Communication Engineering
Apply Graduation and Masters Courses
Apply Graduation and Masters Courses for scholarship

Figure 7.1: Home Page Interface

7.2 Login Page

The Login Page is where the admin enters credentials to securely access the system. Once logged in, the admin is redirected to the Dashboard, which provides access to all management functionalities. The login page is designed to ensure security and prevent unauthorized access.

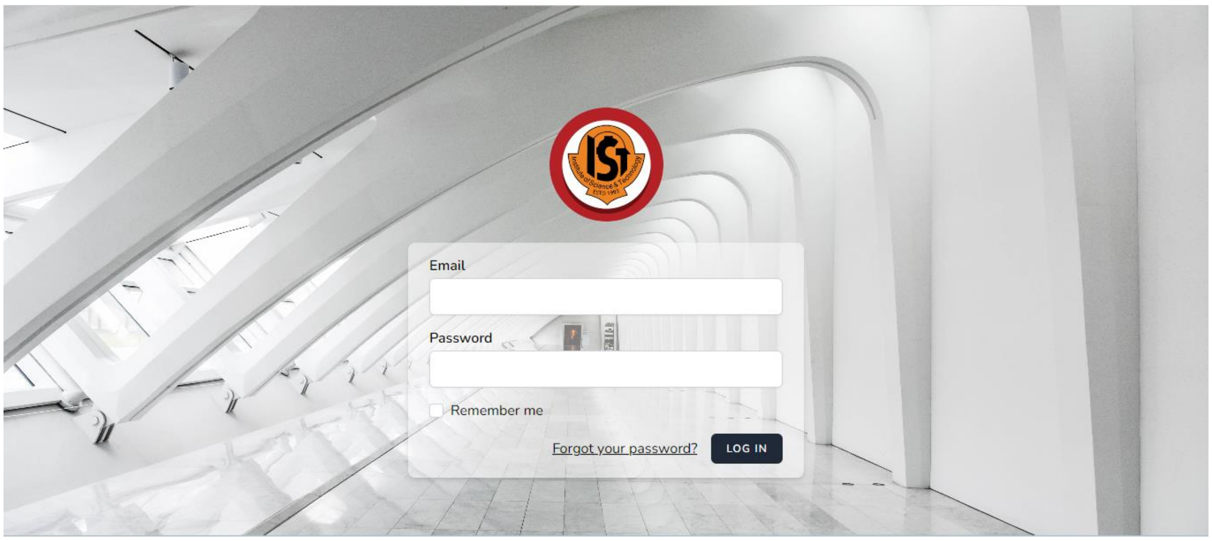


Figure 7.2: Login Page Interface

7.3 Dashboard

The Dashboard serves as the control center for all administrative tasks. From here, the Admin can manage departments, semesters, sessions, students, attendance, and website settings. This centralized hub allows the Admin to oversee the entire system efficiently.

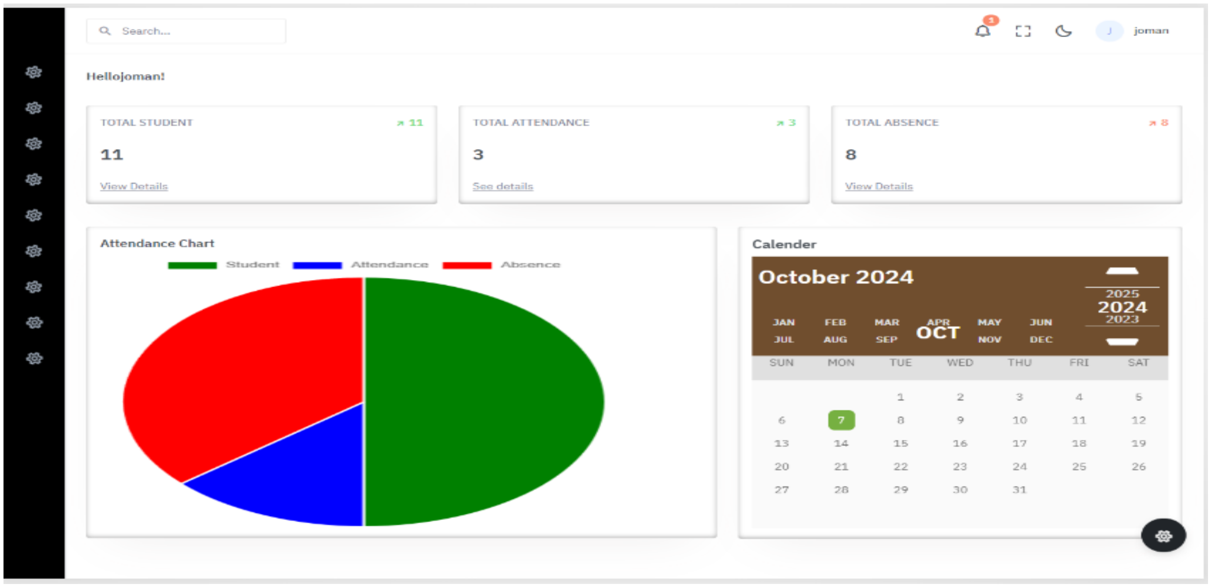
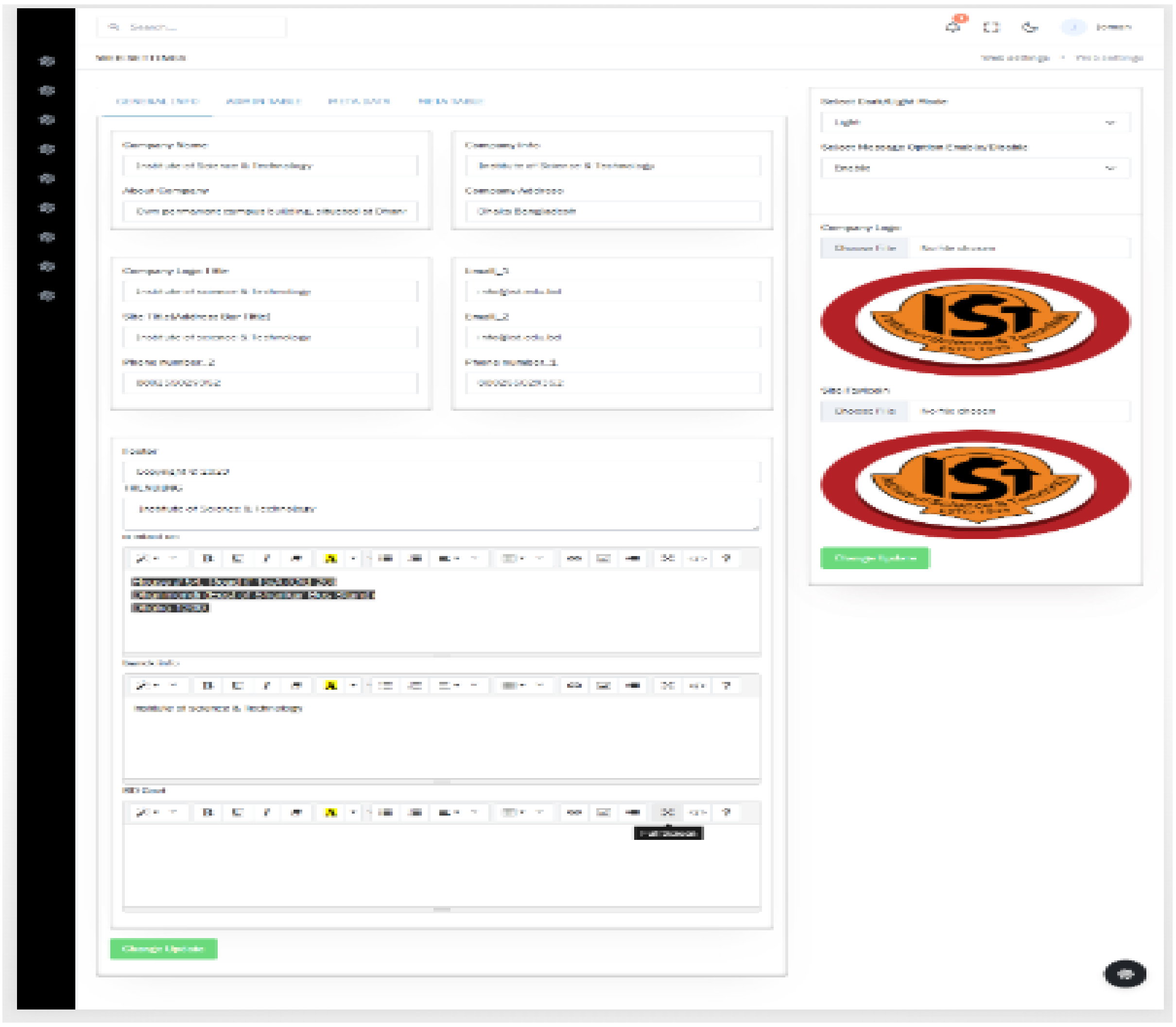


Figure 7.3: Dashboard Interface

7.3.1 General Website Settings

The General Website Settings menu allows the Admin to update the content and layout of the website. This includes modifying text, images, and other visual elements displayed on the Home Page and other sections visible to users, ensuring the content remains up to date.



Search...

GENERAL INFO | [illegible] | [illegible] | [illegible]

Company Name: Institute of Science & Technology

Company Info: Institute of Science & Technology

About Company: [illegible] permanent campus building, situated at Dhaka

Company Address: Dhaka Bangladesh

Company Logo Title: Institute of science & Technology

Email_1: info@ist.edu.bd

Site Title(Address Bar Title): Institute of science & Technology

Email_2: info@ist.edu.bd

Phone number_2: [illegible]

Phone number_1: [illegible]

Footer: Copyright [illegible]

[illegible]: Institute of Science & Technology

[illegible]: [illegible]

[illegible]: Institute of science & Technology

[illegible]

Select Dark/Light Mode: Light

Select Message Option Enable/Disable: Disable

Company Logo: Choose File | No file chosen

Site Favicon: Choose File | No file chosen

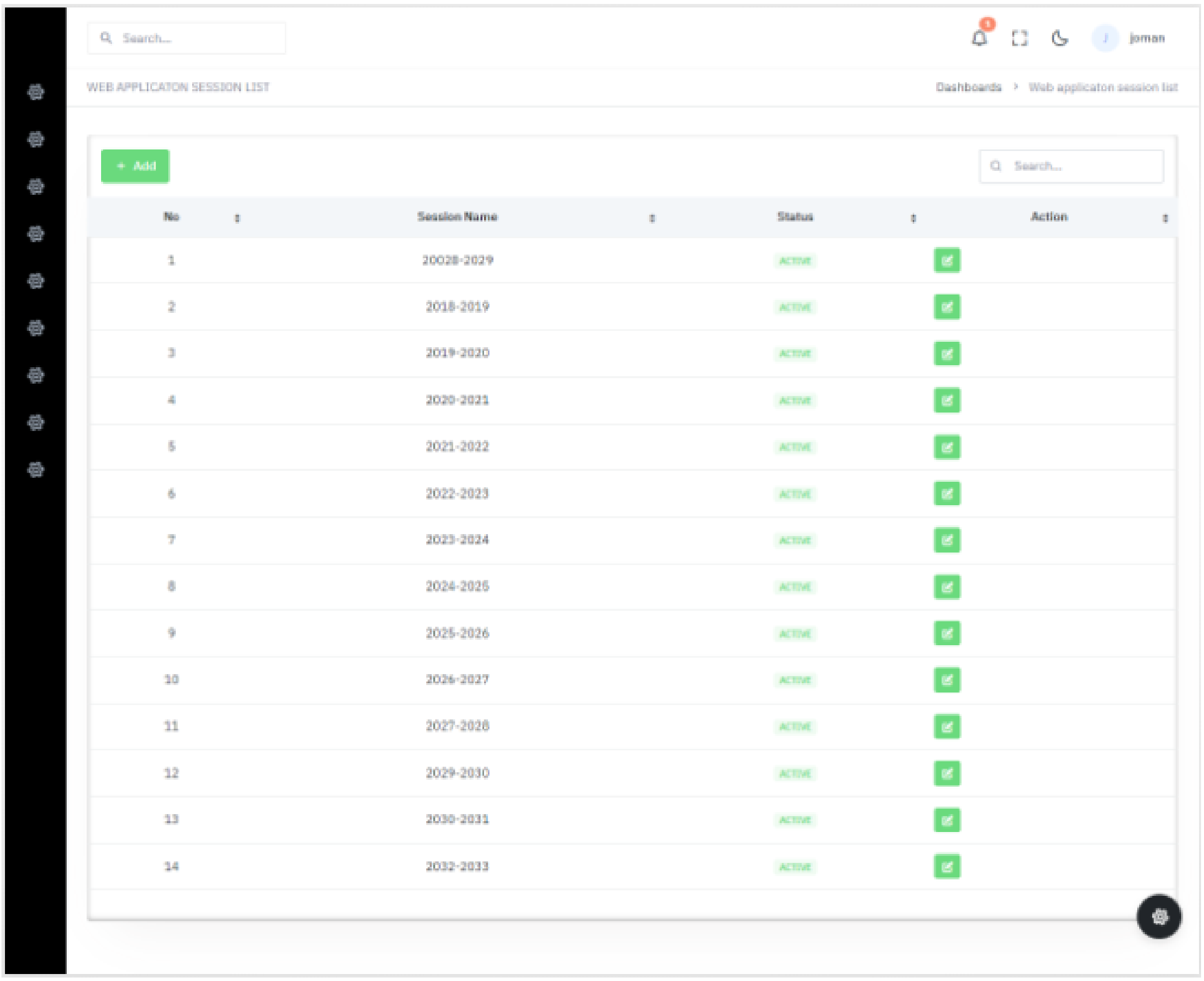
Figure 7.4: General Website Settings Interface

7.3.2 Semester Management

The Semester Management feature enables the Admin to create, update, and manage semesters. It ensures that course schedules and academic activities are aligned with the correct semester. This feature plays a crucial role in keeping academic records organized.

7.3.3 Session Management

This section allows the Admin to manage academic sessions, which typically represent academic years. The Admin can add, edit, or delete session details to ensure that all records are in sync with the current academic structure.



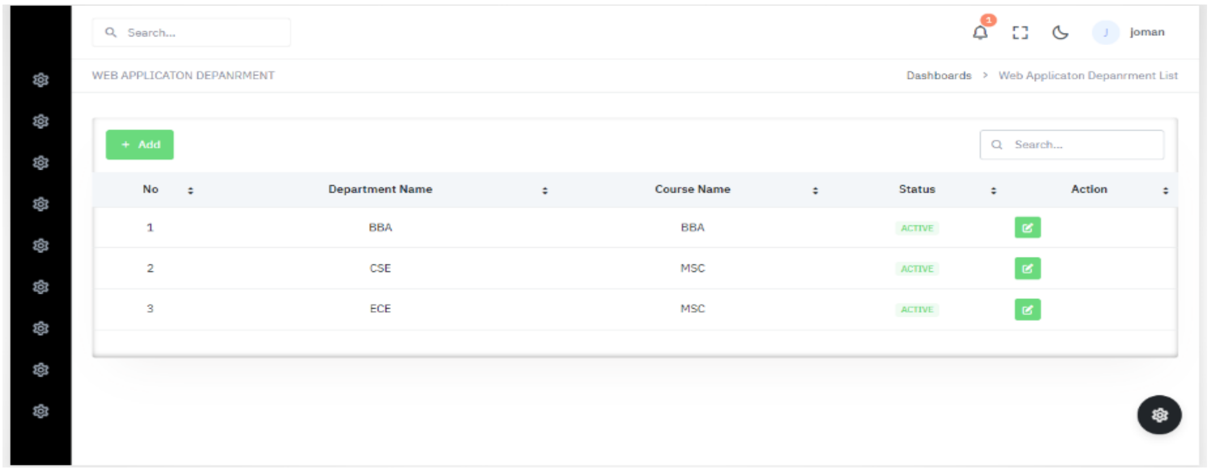
Search...
joman
WEB APPLICATON SESSION LIST
Dashboards › Web applicaton session list
+ Add
Search...

No	Session Name	Status	Action
1	20028-2029	ACTIVE	✓
2	2018-2019	ACTIVE	✓
3	2019-2020	ACTIVE	✓
4	2020-2021	ACTIVE	✓
5	2021-2022	ACTIVE	✓
6	2022-2023	ACTIVE	✓
7	2023-2024	ACTIVE	✓
8	2024-2025	ACTIVE	✓
9	2025-2026	ACTIVE	✓
10	2026-2027	ACTIVE	✓
11	2027-2028	ACTIVE	✓
12	2029-2030	ACTIVE	✓
13	2030-2031	ACTIVE	✓
14	2032-2033	ACTIVE	✓

Figure 7.5: Session Management Interface

7.3.4 Department Management

Department Management enables the Admin to create and maintain department records within the university. The Admin can add new departments, update existing ones, or remove inactive departments, ensuring that the system reflects the current organizational structure.

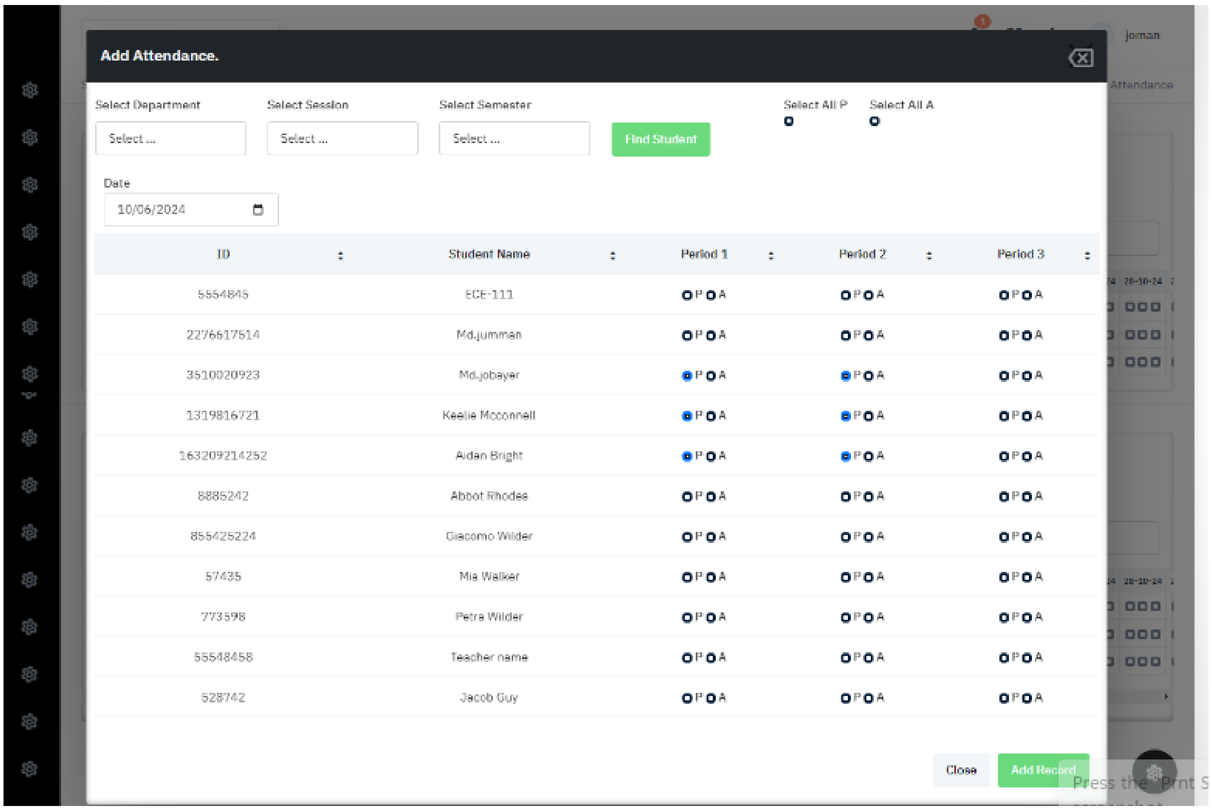

Search...
joman
WEB APPLICATON DEPANRMENT
Dashboards > Web Applicaton Depanrment List
+ Add
Search...

No	Department Name	Course Name	Status	Action
1	BBA	BBA	ACTIVE	
2	CSE	MSC	ACTIVE	
3	ECE	MSC	ACTIVE	

Figure 7.6: Department Management Interface

7.3.5 Student and Attendance Management

In this section, the Admin can manage student profiles and attendance data. This includes adding or updating student information, marking attendance, and ensuring that all attendance records are properly maintained and accessible when needed.

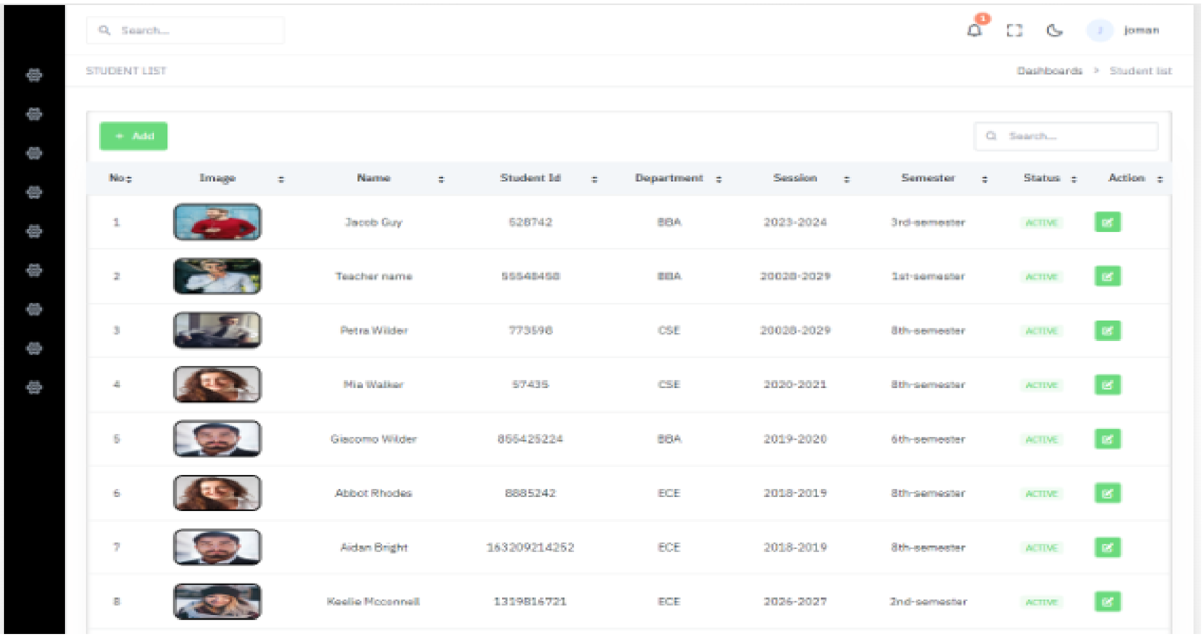


Date
10/06/2024

ID	Student Name	Period 1	Period 2	Period 3
5554845	ECE-111	P A	P A	P A
2276617514	Md.jumman	P A	P A	P A
3510020923	Md.jobayer	[✓ P] A	[✓ P] A	P A
1319816721	Keelie Mcconnell	[✓ P] A	[✓ P] A	P A
163209214252	Aidan Bright	[✓ P] A	[✓ P] A	P A
8885242	Abbot Rhodes	P A	P A	P A
855425224	Giacomo Wilder	P A	P A	P A
57435	Mia Walker	P A	P A	P A
773598	Petra Wilder	P A	P A	P A
55548458	Teacher name	P A	P A	P A
528742	Jacob Guy	P A	P A	P A

7.3.6 Student Addition

The Student Addition feature allows the Admin to register new students by entering their details, such as student ID, department, and contact information. This ensures that new students are added to the system and are accounted for in the attendance records.


Search...

joman

STUDENT LIST

Dashboards > Student list

+ Add

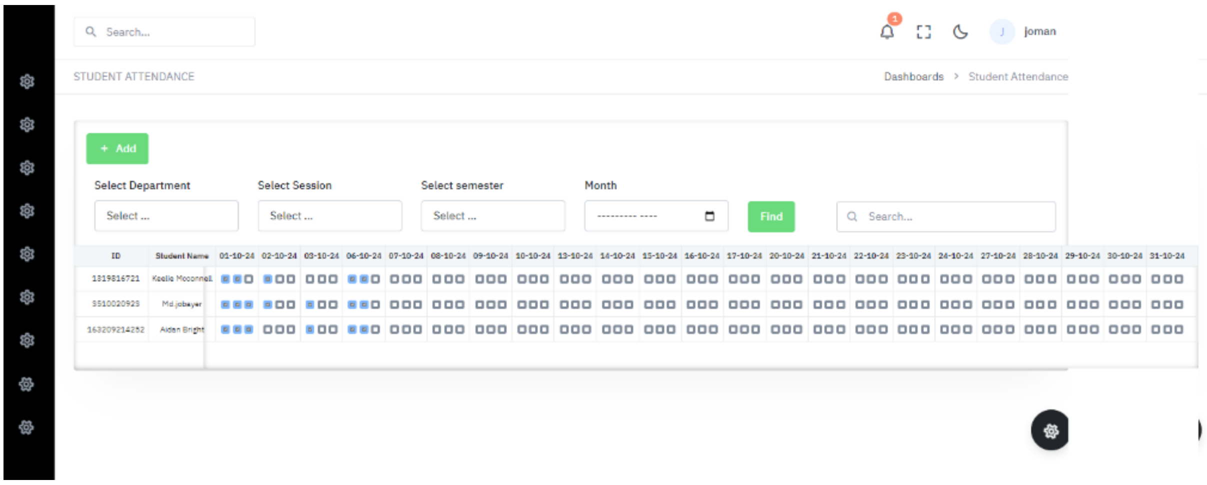
Search...

No	Image	Name	Student Id	Department	Session	Semester	Status	Action
1		Jacob Guy	528742	BBA	2023-2024	3rd-semester	ACTIVE	
2		Teacher name	55548458	BBA	20028-2029	1st-semester	ACTIVE	
3		Petra Wilder	773598	CSE	20028-2029	8th-semester	ACTIVE	
4		Mia Walker	57435	CSE	2020-2021	8th-semester	ACTIVE	
5		Giacomo Wilder	855425224	BBA	2019-2020	6th-semester	ACTIVE	
6		Abbot Rhodes	8885242	ECE	2018-2019	8th-semester	ACTIVE	
7		Aidan Bright	163209214252	ECE	2018-2019	8th-semester	ACTIVE	
8		Keelie Mcconnell	1319816721	ECE	2026-2027	2nd-semester	ACTIVE	

Figure 7.8: Student Addition Interface

7.3.7 Attendance Addition and Reports

In this section, the Admin can manually input attendance data and generate reports. These reports can be filtered by student, class, or department, providing a clear view of attendance trends and helping with attendance tracking and analysis.


Search...

joman

STUDENT ATTENDANCE

Dashboards > Student Attendance

+ Add

Select Department: Select ...

Select Session: Select ...

Select semester: Select ...

Month: --------- ----

Find

Search...

ID	Student Name	01-10-24	02-10-24	03-10-24	06-10-24	07-10-24	08-10-24	09-10-24	10-10-24	13-10-24	14-10-24	15-10-24	16-10-24	17-10-24	20-10-24	21-10-24	22-10-24	23-10-24	24-10-24	27-10-24	28-10-24	29-10-24	30-10-24	31-10-24
1319816721	Keelie Mcconnell	☑☑☐	☑☐☐	☐☐☐	☑☑☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐
3510020923	Md.jobayer	☑☑☑	☑☐☐	☑☐☐	☑☑☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐
163209214252	Aidan Bright	☑☑☑	☐☐☐	☑☐☐	☑☑☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐	☐☐☐

Figure 7.9: Attendance Addition and Reports Interface

CHAPTER 8. CONCLUSION AND FUTURE SCOPE

8.1 Summary of Work

In this project, a comprehensive university attendance management system was developed, integrating both hardware and software components. The system is designed to streamline the process of recording and managing student attendance using RFID technology. The software, built using Laravel, MySQL, HTML, CSS, Bootstrap, and JavaScript, provides a user-friendly interface that allows the admin to manage departments, students, attendance, and other university-related operations. On the hardware side, the project includes an ESP module, RFID scanner, and an LCD display, which work together to scan and send attendance data over the internet via APIs. Overall, the system successfully automates attendance management, improving efficiency and accuracy.

8.2 Challenges Faced

Several challenges were encountered during the development of this project:

- **Software Development**: Building the user interface and integrating it with the hardware components required a significant amount of effort. Ensuring that the software was responsive, secure, and user-friendly while also communicating with the hardware posed technical difficulties.
- **RFID and ESP Integration**: One of the key challenges was establishing reliable communication between the RFID scanner and the ESP module. This involved sending scanned RFID data over the internet using APIs.
- **Displaying Status on LCD**: Another challenge was displaying real-time status updates on the LCD, such as successful scans or errors.
- **SPI Communication**: Implementing reliable SPI communication between the RFID scanner, LCD, and ESP module was a complex task that required precise synchronization of data transmission.

Overcoming these challenges took considerable effort and required fine-tuning both the hardware and software components to ensure smooth operation.

8.3 Future Enhancements

In the future, there are several enhancements that can be made to improve the system:

- **Improved Device Performance**: Developing a faster, more reliable device capable of scanning RFID tags and potentially integrating fingerprint scanning and keypad-based input options would enhance the system's versatility.
- **Offline Functionality**: Adding the ability to store data locally when the internet is not available would ensure the system can still function in offline scenarios, with data synced once connectivity is restored.
- **Control Panel Integration**: A more advanced control panel could be added to provide detailed administrative control over various system functions.
- **Single-Board or PCB Integration**: Consolidating the entire system onto a single board or PCB would improve the device’s portability and ease of deployment.
- **Enhanced Software Features**: Additional software modules could be developed to extend the system’s functionality, covering more aspects of institutional management, such as academic reporting, grading, or student analytics. The user interface can also be further optimized for better performance and user experience.
-

REFERENCES

Documentary Reference:

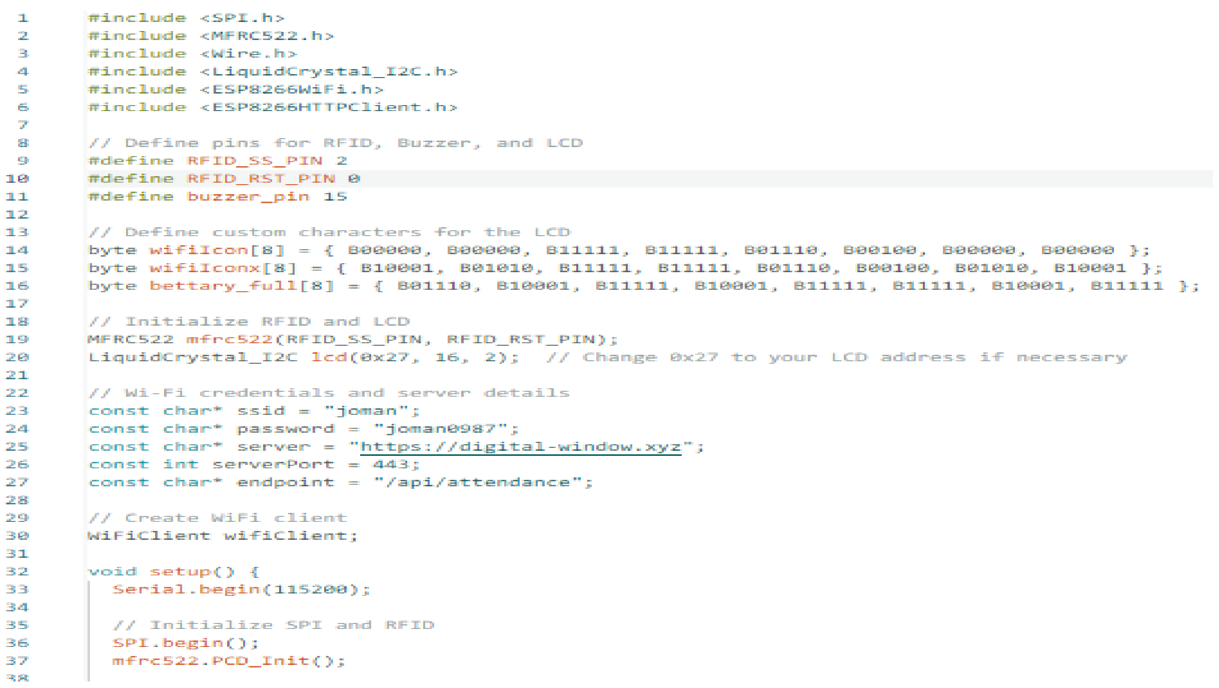
[1] J. Smith and A. Johnson, "The Impact of Smart Card Attendance Systems on Educational Institutions," Journal of Educational Technology, vol. 15, no. 2, pp. 45-58, 2020.

[2] Q. Zhang, L. Wang, and X. Li, "Design and Implementation of a Cost-Effective Smart Card Attendance System," Journal of Information Technology in Education, vol. 20, no. 1, pp. 78-91, 2022.

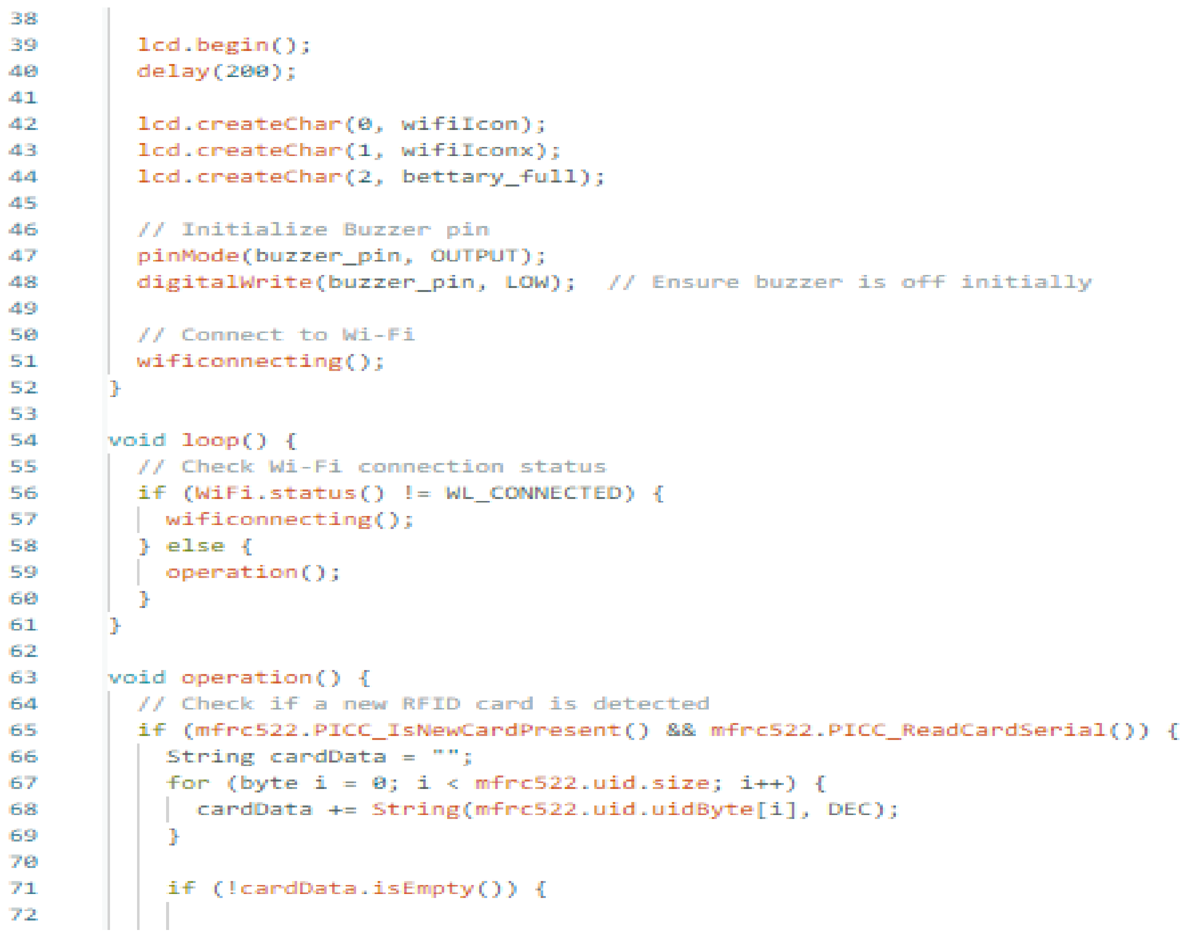
[3] S. Chen, H. Liu, and Q. Wang, "Enhancing Efficiency and User Experience through a Cost-Effective Smart Card Attendance System," Journal of Educational Technology, vol. 16, no. 3, pp. 112-125, 2023.

[4] "Laravel Documentation," Laravel, [Online]. Available: https://laravel.com/docs/.

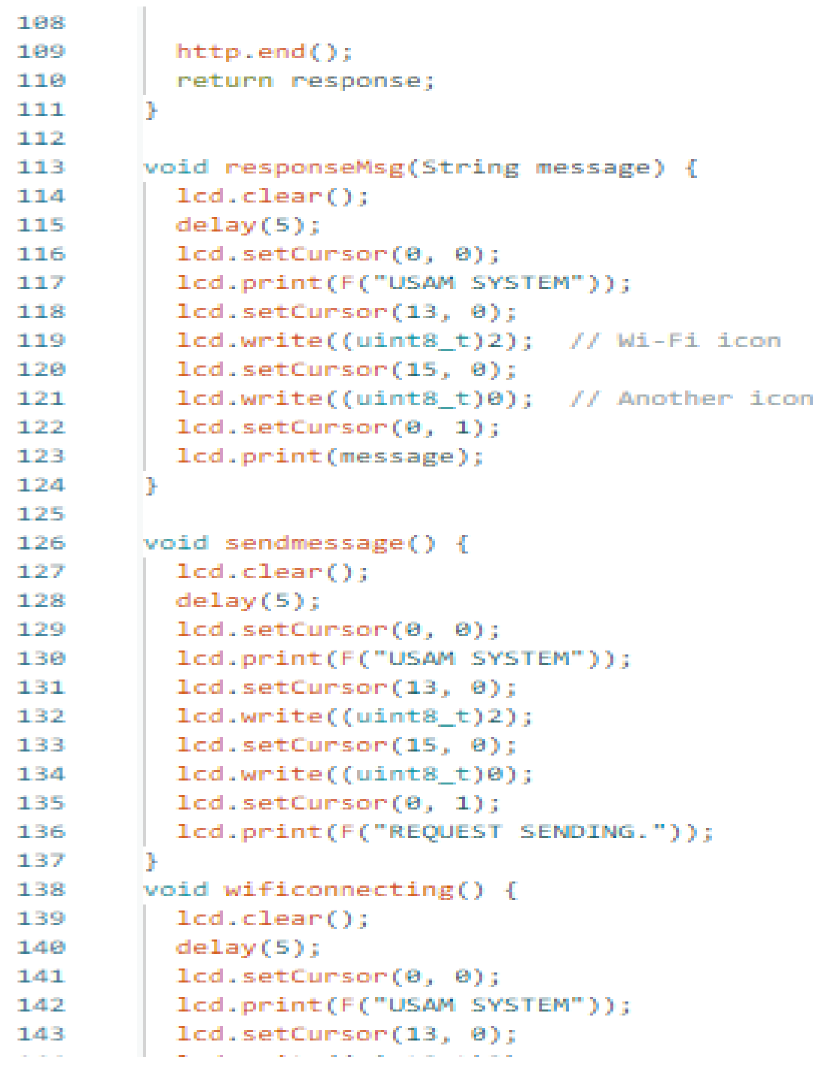
Appendix



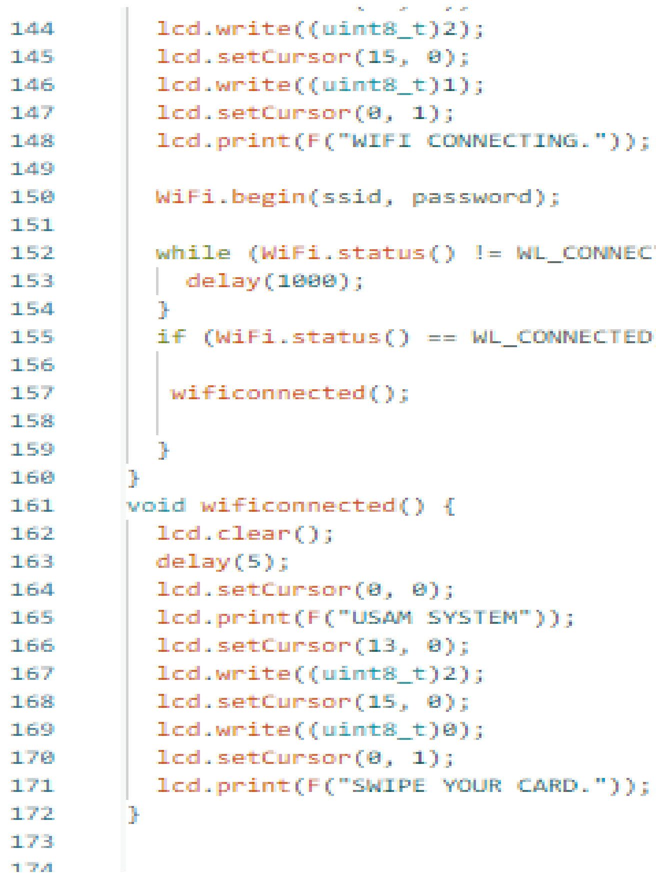
```
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>

// Define pins for RFID, Buzzer, and LCD
#define RFID_SS_PIN 2
#define RFID_RST_PIN 0
#define buzzer_pin 15

// Define custom characters for the LCD
byte wifiIcon[8] = { B00000, B00000, B11111, B11111, B01110, B00100, B00000, B00000 };
byte wifiIconx[8] = { B10001, B01010, B11111, B11111, B01110, B00100, B01010, B10001 };
byte bettary_full[8] = { B01110, B10001, B11111, B10001, B11111, B11111, B10001, B11111 };

// Initialize RFID and LCD
MFRC522 mfrc522(RFID_SS_PIN, RFID_RST_PIN);
LiquidCrystal_I2C lcd(0x27, 16, 2);  // Change 0x27 to your LCD address if necessary

// Wi-Fi credentials and server details
const char* ssid = "joman";
const char* password = "joman0987";
const char* server = "https://digital-window.xyz";
const int serverPort = 443;
const char* endpoint = "/api/attendance";

// Create WiFi client
WiFiClient wifiClient;

void setup() {
  Serial.begin(115200);

  // Initialize SPI and RFID
  SPI.begin();
  mfrc522.PCD_Init();
```



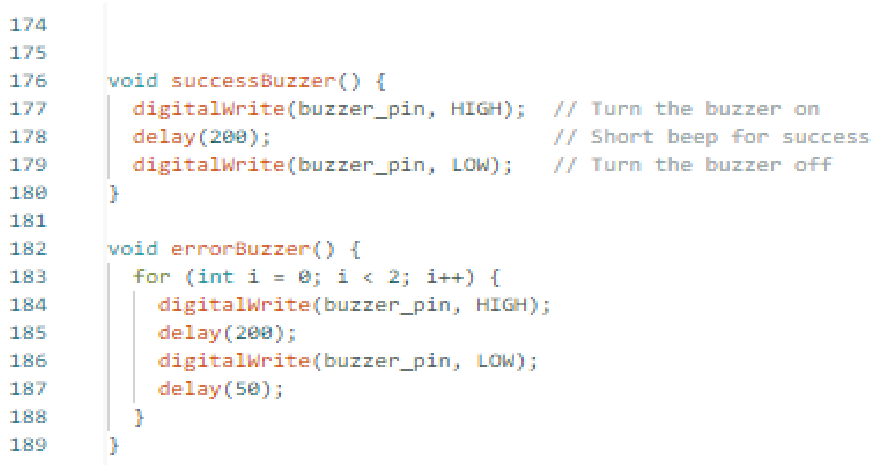
```

  lcd.begin();
  delay(200);

  lcd.createChar(0, wifiIcon);
  lcd.createChar(1, wifiIconx);
  lcd.createChar(2, bettary_full);

  // Initialize Buzzer pin
  pinMode(buzzer_pin, OUTPUT);
  digitalWrite(buzzer_pin, LOW);  // Ensure buzzer is off initially

  // Connect to Wi-Fi
  wificonnecting();
}

void loop() {
  // Check Wi-Fi connection status
  if (WiFi.status() != WL_CONNECTED) {
    wificonnecting();
  } else {
    operation();
  }
}

void operation() {
  // Check if a new RFID card is detected
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    String cardData = "";
    for (byte i = 0; i < mfrc522.uid.size; i++) {
      cardData += String(mfrc522.uid.uidByte[i], DEC);
    }

    if (!cardData.isEmpty()) {

```



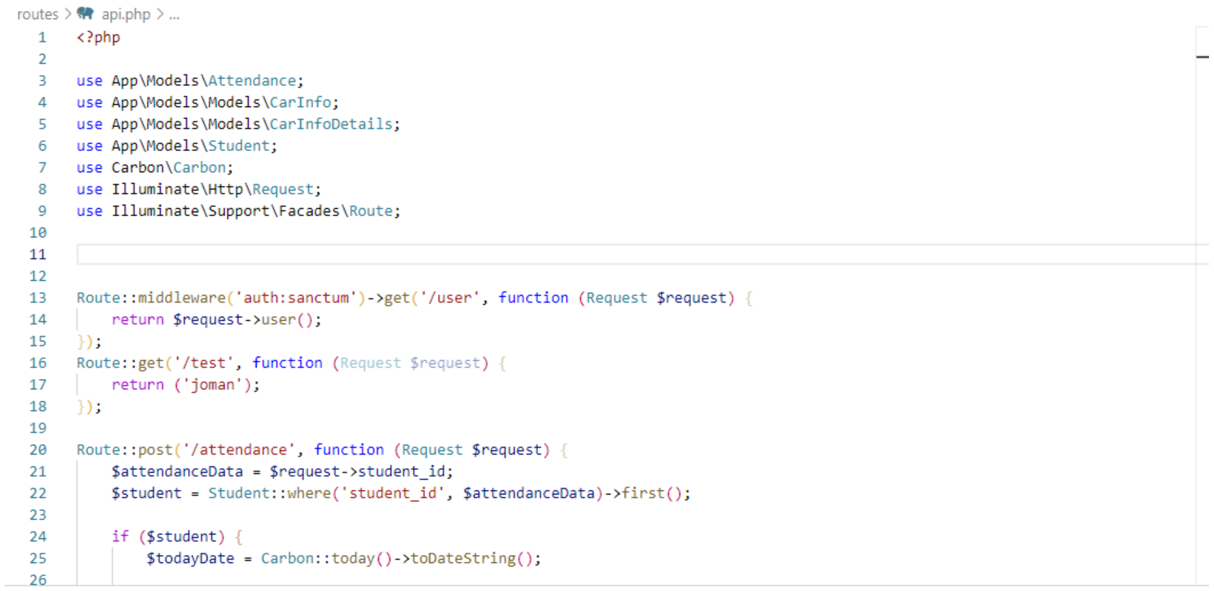
```

  http.end();
  return response;
}

void responseMsg(String message) {
  lcd.clear();
  delay(5);
  lcd.setCursor(0, 0);
  lcd.print(F("USAM SYSTEM"));
  lcd.setCursor(13, 0);
  lcd.write((uint8_t)2);  // Wi-Fi icon
  lcd.setCursor(15, 0);
  lcd.write((uint8_t)0);  // Another icon
  lcd.setCursor(0, 1);
  lcd.print(message);
}

void sendmessage() {
  lcd.clear();
  delay(5);
  lcd.setCursor(0, 0);
  lcd.print(F("USAM SYSTEM"));
  lcd.setCursor(13, 0);
  lcd.write((uint8_t)2);
  lcd.setCursor(15, 0);
  lcd.write((uint8_t)0);
  lcd.setCursor(0, 1);
  lcd.print(F("REQUEST SENDING."));
}
void wificonnecting() {
  lcd.clear();
  delay(5);
  lcd.setCursor(0, 0);
  lcd.print(F("USAM SYSTEM"));
  lcd.setCursor(13, 0);
```



```
  lcd.write((uint8_t)2);
  lcd.setCursor(15, 0);
  lcd.write((uint8_t)1);
  lcd.setCursor(0, 1);
  lcd.print(F("WIFI CONNECTING."));

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
  }
  if (WiFi.status() == WL_CONNECTED) {

   wificonnected();

  }
}
void wificonnected() {
  lcd.clear();
  delay(5);
  lcd.setCursor(0, 0);
  lcd.print(F("USAM SYSTEM"));
  lcd.setCursor(13, 0);
  lcd.write((uint8_t)2);
  lcd.setCursor(15, 0);
  lcd.write((uint8_t)0);
  lcd.setCursor(0, 1);
  lcd.print(F("SWIPE YOUR CARD."));
}

```

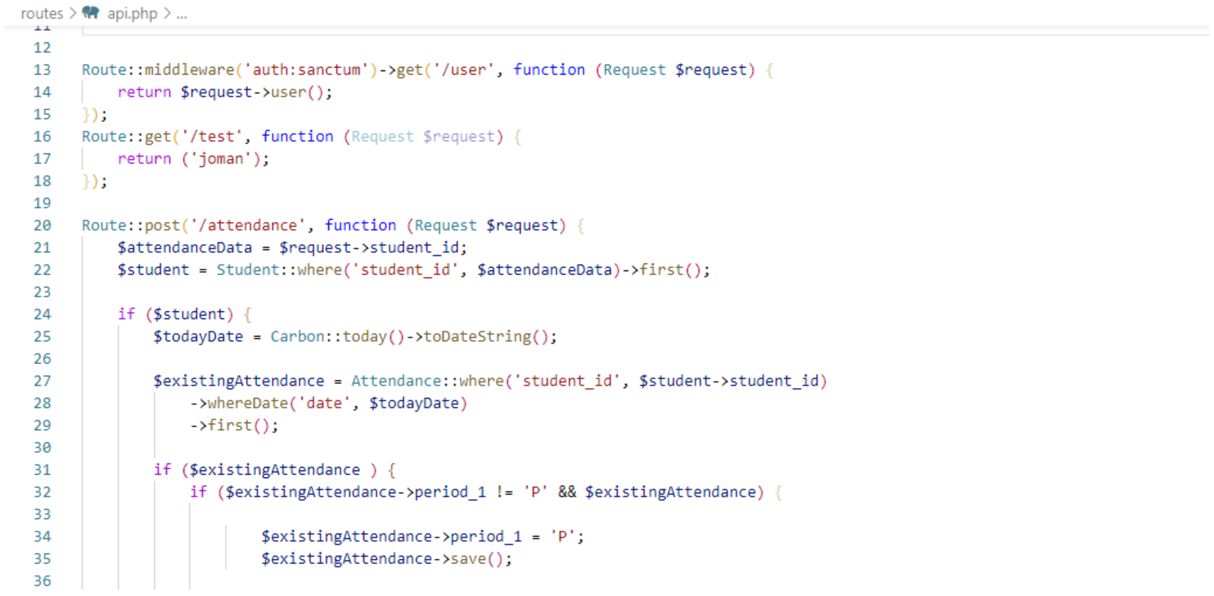


```
void successBuzzer() {
  digitalWrite(buzzer_pin, HIGH);  // Turn the buzzer on
  delay(200);                      // Short beep for success
  digitalWrite(buzzer_pin, LOW);   // Turn the buzzer off
}

void errorBuzzer() {
  for (int i = 0; i < 2; i++) {
    digitalWrite(buzzer_pin, HIGH);
    delay(200);
    digitalWrite(buzzer_pin, LOW);
    delay(50);
  }
}
```



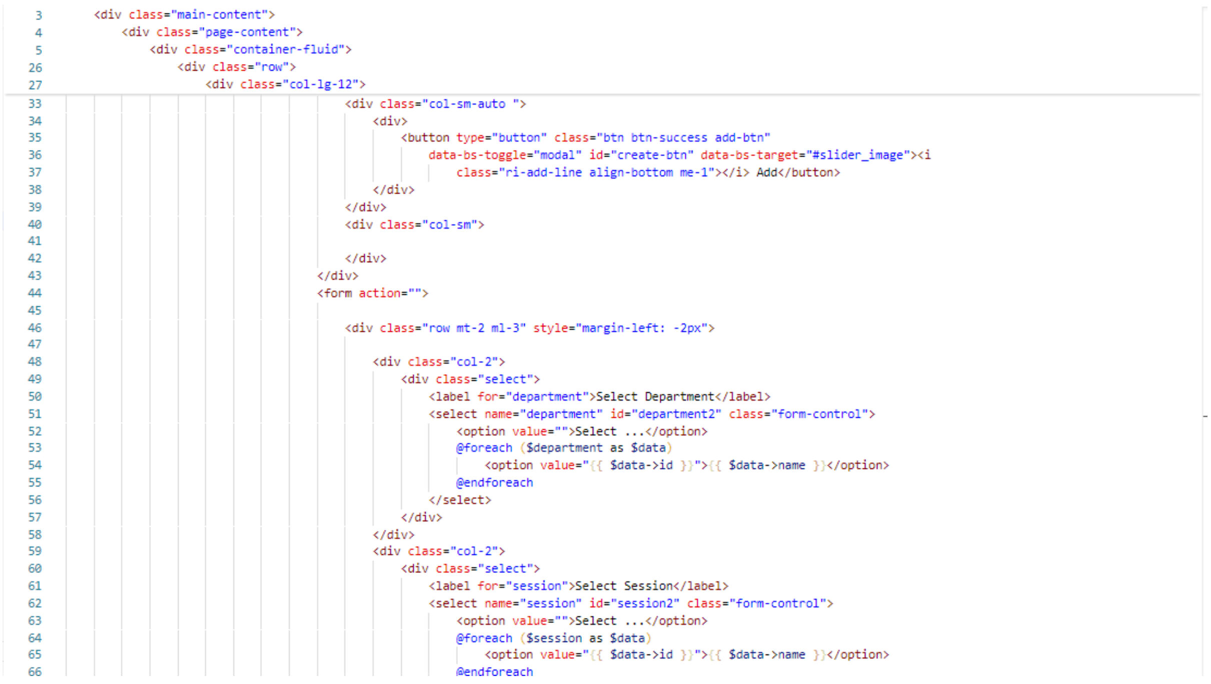
routes > api.php > ...

```php
<?php

use App\Models\Attendance;
use App\Models\Models\CarInfo;
use App\Models\Models\CarInfoDetails;
use App\Models\Student;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;



Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
Route::get('/test', function (Request $request) {
    return ('joman');
});

Route::post('/attendance', function (Request $request) {
    $attendanceData = $request->student_id;
    $student = Student::where('student_id', $attendanceData)->first();

    if ($student) {
        $todayDate = Carbon::today()->toDateString();
```

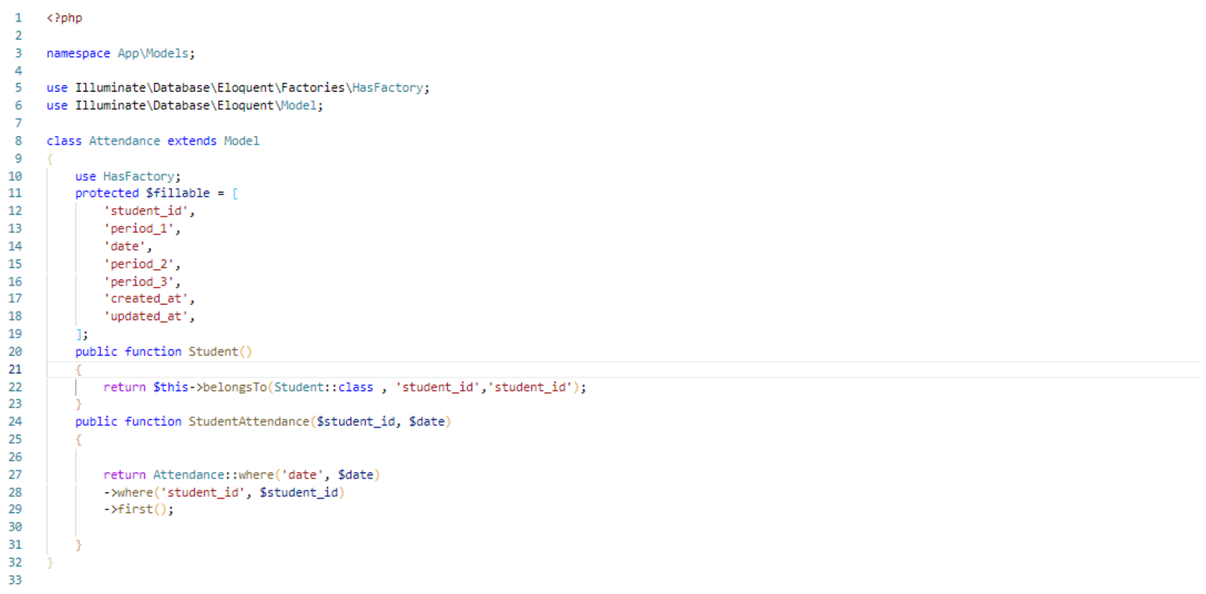


routes > api.php > ...

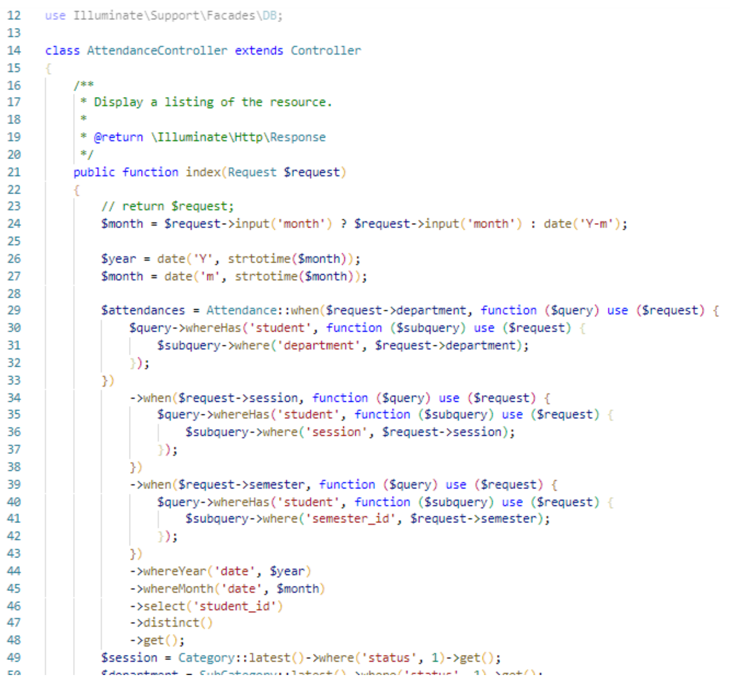
```php

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});
Route::get('/test', function (Request $request) {
    return ('joman');
});

Route::post('/attendance', function (Request $request) {
    $attendanceData = $request->student_id;
    $student = Student::where('student_id', $attendanceData)->first();

    if ($student) {
        $todayDate = Carbon::today()->toDateString();

        $existingAttendance = Attendance::where('student_id', $student->student_id)
            ->whereDate('date', $todayDate)
            ->first();

        if ($existingAttendance ) {
            if ($existingAttendance->period_1 != 'P' && $existingAttendance) {

                $existingAttendance->period_1 = 'P';
                $existingAttendance->save();

```



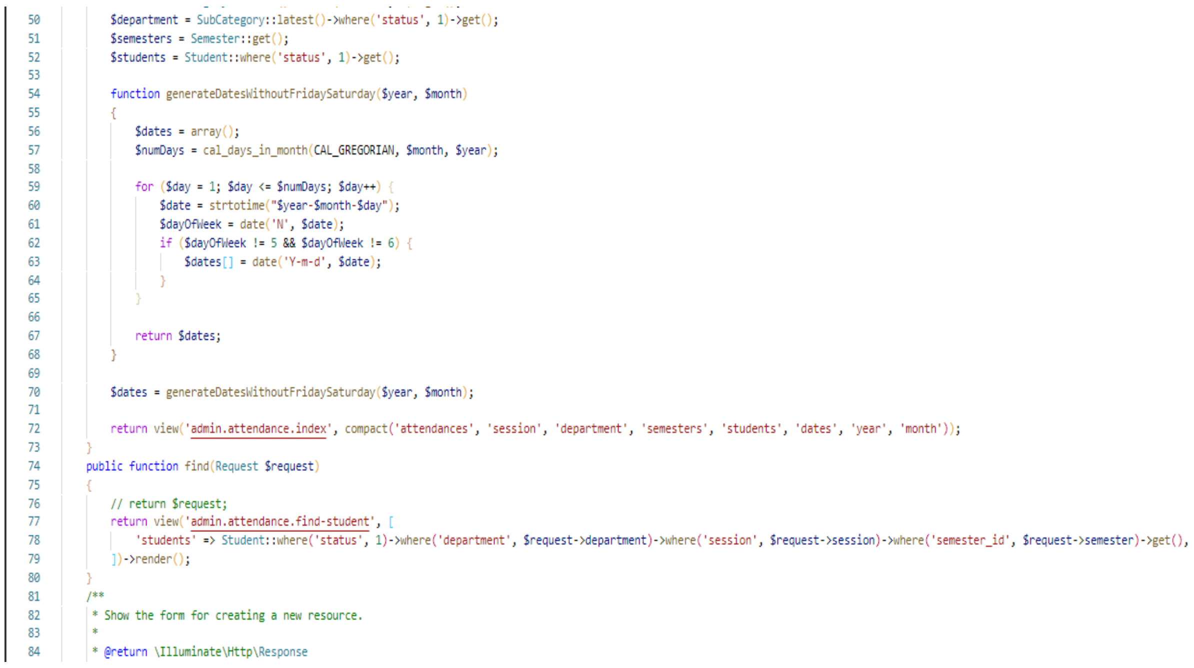
```
    <div class="main-content">
        <div class="page-content">
            <div class="container-fluid">
                <div class="row">
                    <div class="col-lg-12">
                                        <div class="col-sm-auto ">
                                            <div>
                                                <button type="button" class="btn btn-success add-btn"
                                                    data-bs-toggle="modal" id="create-btn" data-bs-target="#slider_image"><i
                                                        class="ri-add-line align-bottom me-1"></i> Add</button>
                                            </div>
                                        </div>
                                        <div class="col-sm">

                                        </div>
                                    </div>
                                    <form action="">

                                        <div class="row mt-2 ml-3" style="margin-left: -2px">

                                            <div class="col-2">
                                                <div class="select">
                                                    <label for="department">Select Department</label>
                                                    <select name="department" id="department2" class="form-control">
                                                        <option value="">Select ...</option>
                                                        @foreach ($department as $data)
                                                            <option value="{{ $data->id }}">{{ $data->name }}</option>
                                                        @endforeach
                                                    </select>
                                                </div>
                                            </div>
                                            <div class="col-2">
                                                <div class="select">
                                                    <label for="session">Select Session</label>
                                                    <select name="session" id="session2" class="form-control">
                                                        <option value="">Select ...</option>
                                                        @foreach ($session as $data)
                                                            <option value="{{ $data->id }}">{{ $data->name }}</option>
                                                        @endforeach
```



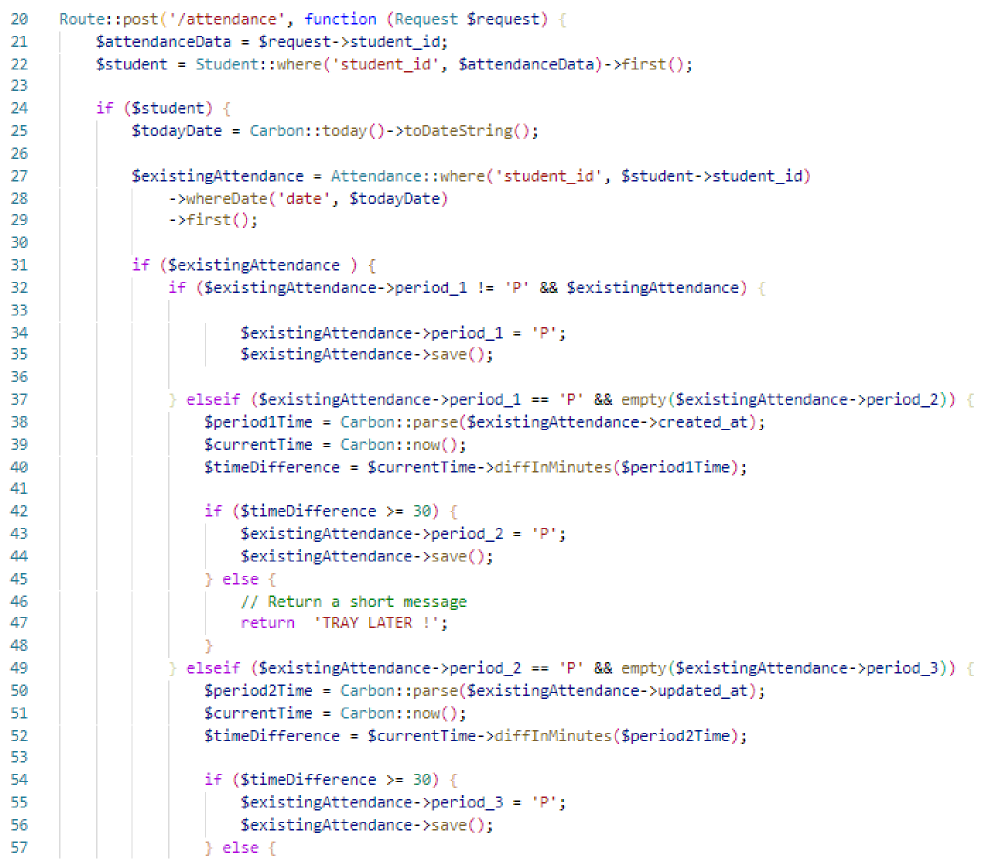
```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Attendance extends Model
{
    use HasFactory;
    protected $fillable = [
        'student_id',
        'period_1',
        'date',
        'period_2',
        'period_3',
        'created_at',
        'updated_at',
    ];
    public function Student()
    {
        return $this->belongsTo(Student::class , 'student_id','student_id');
    }
    public function StudentAttendance($student_id, $date)
    {

        return Attendance::where('date', $date)
        ->where('student_id', $student_id)
        ->first();

    }
}

```



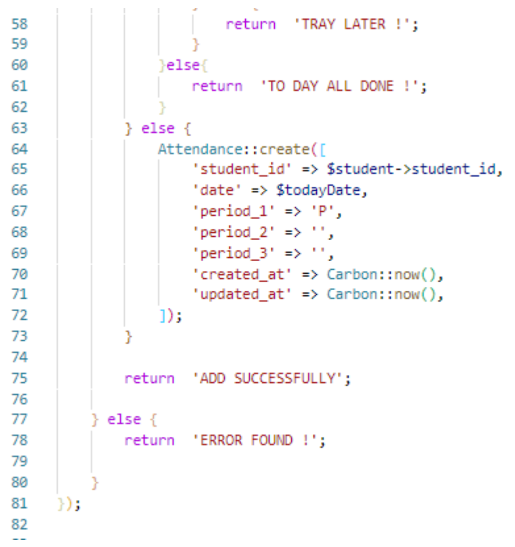
```php
use Illuminate\Support\Facades\DB;

class AttendanceController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index(Request $request)
    {
        // return $request;
        $month = $request->input('month') ? $request->input('month') : date('Y-m');

        $year = date('Y', strtotime($month));
        $month = date('m', strtotime($month));

        $attendances = Attendance::when($request->department, function ($query) use ($request) {
            $query->whereHas('student', function ($subquery) use ($request) {
                $subquery->where('department', $request->department);
            });
        })
            ->when($request->session, function ($query) use ($request) {
                $query->whereHas('student', function ($subquery) use ($request) {
                    $subquery->where('session', $request->session);
                });
            })
            ->when($request->semester, function ($query) use ($request) {
                $query->whereHas('student', function ($subquery) use ($request) {
                    $subquery->where('semester_id', $request->semester);
                });
            })
            ->whereYear('date', $year)
            ->whereMonth('date', $month)
            ->select('student_id')
            ->distinct()
            ->get();
        $session = Category::latest()->where('status', 1)->get();
```



```php
        $department = SubCategory::latest()->where('status', 1)->get();
        $semesters = Semester::get();
        $students = Student::where('status', 1)->get();

        function generateDatesWithoutFridaySaturday($year, $month)
        {
            $dates = array();
            $numDays = cal_days_in_month(CAL_GREGORIAN, $month, $year);

            for ($day = 1; $day <= $numDays; $day++) {
                $date = strtotime("$year-$month-$day");
                $dayOfWeek = date('N', $date);
                if ($dayOfWeek != 5 && $dayOfWeek != 6) {
                    $dates[] = date('Y-m-d', $date);
                }
            }

            return $dates;
        }

        $dates = generateDatesWithoutFridaySaturday($year, $month);

        return view('admin.attendance.index', compact('attendances', 'session', 'department', 'semesters', 'students', 'dates', 'year', 'month'));
    }
    public function find(Request $request)
    {
        // return $request;
        return view('admin.attendance.find-student', [
            'students' => Student::where('status', 1)->where('department', $request->department)->where('session', $request->session)->where('semester_id', $request->semester)->get(),
        ])->render();
    }
    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
```



```php
Route::post('/attendance', function (Request $request) {
    $attendanceData = $request->student_id;
    $student = Student::where('student_id', $attendanceData)->first();

    if ($student) {
        $todayDate = Carbon::today()->toDateString();

        $existingAttendance = Attendance::where('student_id', $student->student_id)
            ->whereDate('date', $todayDate)
            ->first();

        if ($existingAttendance ) {
            if ($existingAttendance->period_1 != 'P' && $existingAttendance) {

                    $existingAttendance->period_1 = 'P';
                    $existingAttendance->save();

            } elseif ($existingAttendance->period_1 == 'P' && empty($existingAttendance->period_2)) {
                $period1Time = Carbon::parse($existingAttendance->created_at);
                $currentTime = Carbon::now();
                $timeDifference = $currentTime->diffInMinutes($period1Time);

                if ($timeDifference >= 30) {
                    $existingAttendance->period_2 = 'P';
                    $existingAttendance->save();
                } else {
                    // Return a short message
                    return  'TRAY LATER !';
                }
            } elseif ($existingAttendance->period_2 == 'P' && empty($existingAttendance->period_3)) {
                $period2Time = Carbon::parse($existingAttendance->updated_at);
                $currentTime = Carbon::now();
                $timeDifference = $currentTime->diffInMinutes($period2Time);

                if ($timeDifference >= 30) {
                    $existingAttendance->period_3 = 'P';
                    $existingAttendance->save();
                } else {
```



```php
                    return  'TRAY LATER !';
                }
            }else{
                return  'TO DAY ALL DONE !';
            }
        } else {
            Attendance::create([
                'student_id' => $student->student_id,
                'date' => $todayDate,
                'period_1' => 'P',
                'period_2' => '',
                'period_3' => '',
                'created_at' => Carbon::now(),
                'updated_at' => Carbon::now(),
            ]);
        }

        return  'ADD SUCCESSFULLY';

    } else {
        return  'ERROR FOUND !';

    }
});
```