Smart Attendance System

Submitted By

Faysal Mahmud

ID: 221-35-906

Supervised By

Dr. Rubaiyat Islam

Designation

Associate Professor, Director Cyber Security Centre, Department of Software Engineering

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name :

Date of Birth :

Title :

Academic Session :

I declare that this thesis is classified as:

☐ CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*

☐ RESTRICTED (Contains restricted information as specified by the organization where research was done)*

☐ OPEN ACCESS I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

NOTE: * If the Project is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

APPROVAL

This thesis titled on **"Secured and economic attendance manageable platform"**, submitted by **Student Name: Faysal Mahmud (ID: 221-35-906)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. A. H. M. Saifullah Sadi
Professor
Department of Software Engineering
Faculty of Science and Information Technology Daffodil International University

Internal Examiner 1

Dr. Rubaiyat Islam
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Md. Abdul Kader
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

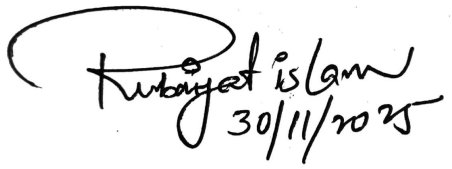
Nuruzzaman Faruqui
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md. Mostafiz Khan
Managing Director
Tecognize Solutions Limited

SUPERVISOR's DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



(Supervisor's Signature)

Full Name : **Dr. Rubaiyat Islam**

Position : Associate Professor

Date : 30 Nov 2025

STUDENT'S DECLARATION

I hereby declare that the work in this project, I have developed is based on my original work except for quotations and citations, which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

_____

(Student's Signature)

Full Name : Faysal Mahmud

ID Number : 221-35-906

Date : 25 Dec 2025

Smart Attendance System

Faysal Mahmud

Project submitted in fulfillment of the requirements
for the award of the degree of

Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

DECEMBER 2025

ACKNOWLEDGEMENTS

I would like to extend my heartfelt appreciation to my supervisor, **Dr. Rubaiyat Islam**, Department of Software Engineering, Daffodil International University. His professional management, goodwill and efforts, and constant stimulation have been precious in the moulding of the excellence and the course of this labour.

I would like to deeply thank the entire faculty of the Department of Software engineering.
help and encouragement during my study. Their counsel and good wishes helped me with the assertiveness to proceed with this project.

This could not have taken place without special appreciation of my parents whose unconditional love, sacrifices and motivation has been.

the basis of my academic life. My friends are also just as indebted to me, and they were by my side.
and with suggestions, corrections, and moral encouragement whenever I required it.
Lastly, I would like to thank all the participants that took part in the survey and validation process.
of this thesis. Their participation and prudent contribution towards the research were meaningful and made it more significant.

DEDICATION

I hence assert that I have accomplished this project under the guidance of Dr. Rubaiyat Islam, Associate Professor, Director Cyber Security Centre, Department of Software Engineering, Daffodil International University. Also state that all record and not part of this record has been presented elsewhere to my degree.

ABSTRACT

The Smart Attendance System, an IoT-based automated attendance management system in this project, is intended to substitute traditional manual attendance record keeping with an efficient and secure real-time attendance tracking system, which is fast and reliable, and resistant to manipulation. To ensure that the RFID cards cannot be accessed or manipulated, the device reads the RFID cards, authenticates the data in the local server, transmits the data to the server via an encrypted and HMAC-secured API.

The attendance system has the following features; LED indicators, a buzzer to provide instant feedback, and displaying the device status and user details. The device automatically displays a local configuration page when there is no Wi-Fi connection, enabling users to configure Wi-Fi credentials (home or office). On the server side, users will be able to administer numerous devices with the same account, add RFID cards, add card information, monitor attendance history, download day/week/month reports and add other employee details like pay notes or pay due ledger.

With this project, an account monitoring, reporting, user control, device management and problem solving feature is added to the capabilities through an admin and super admin panel. The backend is developed in FastAPI with PostgreSQL as the primary database and the frontend is developed in HTMX, HTML, Bootstrap, and CSS to create lightweight and responsive interface. MicroPython is used to build the device firmware, which then allows the firmware to be coded in a low power consumption manner and hard-hardware stable way.

In general, the system is a full-fledged automated attendance system that is applicable to schools, offices, coaching facilities, restaurants, and labs. It enhances precision, lowers the manual workload, boosts transparency, and offers a scalable digital environment to track the attendance and manage employees.

TABLE OF CONTENTS

DECLARATION TITLE PAGE

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

The emergence of the IoT and cloud technologies has provided new possibilities to automatize the regular work, such as attendance management. This notwithstanding, a significant number of institutions have continued to use manual and paper-based systems that are inefficient and susceptible to errors. This is now possible with cheap hardware, such as the ESP32, and well-built web frameworks, which can be used to create smart connected systems that provide a better alternative to these old ways of doing things.

1.1.1 Context and Relevance

The move towards automation and information-driven activities in schools and businesses has seen an upsurge in the need to embrace smart attendance systems. Automation is gaining accessibility through current trends, including RFID technology, inexpensive IoT devices, including the ESP32, and secure cloud APIs. Nevertheless, most of the current systems are costly, inorganic and require continuous internet connectivity. This is a very topical project because it directly uses these new trends to fill the gaps present in the market. In this project, I have created offers by integrating a low-cost ESP32-based platform and a secure FastAPI back-end with a subscription service, thus providing a workable, scalable, and economical offering. It allows any size institution to digitize attendance at a low cost compared to the commercial offerings and is therefore a timely and convenient solution in the shifting IoT era.

1.1.2 Problem Identification

Conventional attendance systems have a number of difficulties: manual means are very time-consuming and full of errors whereas commercial RFID devices are usually costly, rigid, and with poor security capabilities. Their reliance on continuous internet access is a significant weakness because they cannot work when the network is not available hence loss of data.

The Smart Attendance System is a cheap, safe, and versatile solution to these problems. It facilitates the online and offline usage, gives control over multiple devices using the same account, and uses encryption and authentication to signify integrity and safety of data.

1.1.3 Purpose and Justification

Create a safe, economical, IoT-driven smart attendance solution that guarantees effective and reliable as well as automated tracking of small and big organizations.

1.1.4 Scope

The project will be developed to incorporate an end-to-end attendance management system based on a smart hardware device that uses an IoT platform and a secure, cloud-connecting software platform. The RFID device is based on ESP32 and reads the cards, availing real-time feedback displayed on the screen, LEDs, and buzzer, dynamic Wi-Fi configuration, and storing the offline data to sync automatically. The software is characterized by user authentication, the multi-device management, the RFID registration, attendance management, and the secure API communication with HMAC validation. Data are managed by a FastAPI-PostgreSQL backend and displayed by an HTMX- and Bootstrap-based dashboard through which logs can be viewed, reports can be generated, devices can be tracked, and additional card information can be managed. There are also the admin and super admin panels that manage the operations and people using the site. The ability to make subscription-based cloud services in this project a scalable, secure, and cost-effective service to various organizations is achievable.

1. Device Module (IoT Hardware Layer)

- RFID attendance device with MicroPython operating system based on ESP32.
- On the fly feedback system based on LED indicators (success/error) and buzzers.
- Status, card detection and connection status presenter on built in device display.
- Automatic set up of Wi-Fi, using a local captive portal (HTML/CSS).
- Caching of offline attendance stored scans in the event of network failure and retrieved later.
- The communication between the devices and the server is secured by hashing, encryption, and HMAC validation.

2. User and Account Management

- Secure user registration and JWT-based authentication.
- Multi device connection system in one account
- Email, phone and username authentication

3. Device Administration & Monitoring

- Add, remove, and manage multiple attendance devices from the dashboard.
- Monitor device status, activation time, online/offline state, and logs.
- Device registration using unique secret keys and encrypted API validation.

4. RFID Card & Employee Management

- Register unlimited RFID cards under an account.
- Store card-specific details such as name, gender, address, work type, salary notes, and dues.
- Remove or update card info dynamically.
- Assign multiple employees under one device or distribute across multiple devices.

5. Attendance Tracking & Reporting

- Real-time attendance logging via secure backend channels.
- Check-in and check-out timestamps stored in PostgreSQL.
- Multi-dimensional reporting: daily, weekly, monthly, custom date range.
- Export options: CSV, XLSX, printable PDF.
- Visualization of attendance patterns using charts or tables.

6. Notification & Order System

- In-app notifications for device actions, orders, errors, and updates.
- Order placement for new devices or accessories.
- User-side order history, status tracking (pending/processing/completed).
- Admin-side order management tools.

7. Admin & Super Admin Module

- Admin panel for monitoring users, devices and user duration use

- Super admin tools: add/remove admins, freeze user accounts, review issues.
- System health overview: API status, device error logs, and flagged activities.
- Centralized control of reports, search modules, and user data validation.

8. API Security & Communication Layer

- Encrypted data transmission using hashing and HMAC signatures.
- Device-specific secret keys preventing unauthorized API access.
- Request throttling and error-handling for stable communication.
- Validation pipelines for preventing fake card scans or spoofed requests.

9. Web Dashboard (HTMX + Bootstrap)

- Lightweight, responsive, and fast interface without heavy frontend frameworks.
- Dashboard home showing summary metrics, device list, notifications, and charts.
- Searchable attendance tables and filter tools (device-wise, employee-wise, date-wise).
- Smooth partial page updates using HTMX for a better UX.

10. Deployment & System Operations

- Deployment on Railway/Render for backend services.
- Structured database schema using PostgreSQL with indexes for fast queries.
- Logging, error-handling, and performance monitoring integrated at API level.

11. Localization & Usability Enhancements

- Full bilingual support.
- Simple UX optimized for teachers, office managers, and non-technical users.
- Mobile-friendly attendance dashboard for on-the-go monitoring.

12. Subscription & Billing Model

- Monthly/Yearly subscription options for cloud reporting and device syncing.
- Ultra-low device cost enabling adoption by small coaching centers and shops.

1.2 Project Planning and Initiation

1.2.1 Feasibility Study (Step-by-Step)

The feasibility study for the smart attendance system project was done to check if its possible to make it work technically, operationally, and economically.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

Intended Users :

- End Users (teachers, staff, students)
- Admin (system & decision management)
- Super Admin (full system control)
- Technical support staff

1.3.2 User profile

Table 1: User Profile for End Users

User Class	**Note on Characteristics**
Type of user	General user, customer, or staff
Age range	18–60 years
Frequency of use	Daily / weekly, depending on task
Mandatory	Yes, core users of in this project, I have developed
Computer experience	Basic to intermediate
Education	High school to graduate
goal	Complete tasks efficiently and accurately
Language skills	Basic English and Proficiency in Bengali
Number of users	Depends on organization size (e.g., 50–500)
Training	Short onboarding, tutorial videos, manuals
Others system use	May use email, MS Office, web apps
Way of working	Task-oriented, prefers simple UI, mobile/desktop access

Table 2: User Profile for Admins

User Class	**Note on Characteristics**
Type of user	System administrator/manager
Age range	25–50 years
Frequency of use	Daily
Mandatory	Yes, for system maintenance and monitoring
Computer experience	Advanced/technical
Education	Graduate or above in IT/management
goal	Monitor system, manage users, generate reports
Language skills	English + local language
Number of users	1–10 (admins)

Training	Technical training provided
Other systems use	ERP and reporting tools
Way of working	Analytical, organised, prefers dashboards

Table 3: User Profile for Super Admin

User Class	**Note on Characteristics**
Type of user	Full system control
Age range	25-55 years
Frequency of use	Daily
Mandatory	Yes
Computer experience	Advanced/technical
Education	Graduate or above in IT/management
goal	Monitor system, manage users, generate reports
Language skills	English + local language
Number of users	1
Training	Technical training provided
Others system use	ERP, database management, and reporting tools
Way of working	Strategic, supervisory

Table 3: User Profile for Technical Support Staff

User Class	**Note on Characteristics**
Type of user	Support & maintenance
Age range	25-50 years
Frequency of use	As needed
Mandatory	Optional
Computer experience	Intermediate
Education	Diploma/Graduate
goal	Resolve technical issues, maintain system
Language skills	English + local language
Number of users	1
Training	Technical training provided
Others system use	Hardware/software tools
Way of working	Problem-solving, responsive

1.3.3 Elicitation Process

Methods :

- **Interviews**
  - One-on-one sessions with end users and stakeholders
  - Purpose: Understand user needs, pain points, and workflow

- **Surveys / Questionnaires**
  - Structured forms sent to multiple users
  - Purpose: Quantitative data on preferences and feature requirements

- **Focus Groups**
  - Small group discussions with representative users
  - Purpose: Brainstorm ideas, validate features, prioritize needs

- **Observation / Shadowing**
  - Observe users performing tasks in real environment
  - Purpose: Identify hidden requirements and workflow inefficiencies

- **Prototyping / Feedback Loops**
  - Develop low-fidelity prototypes / wireframes
  - Gather feedback to refine requirements

- **Document Analysis**
  - Review existing manuals, system logs, or reports
  - Purpose: Identify gaps, understand current processes

1.4 Project Block Diagram

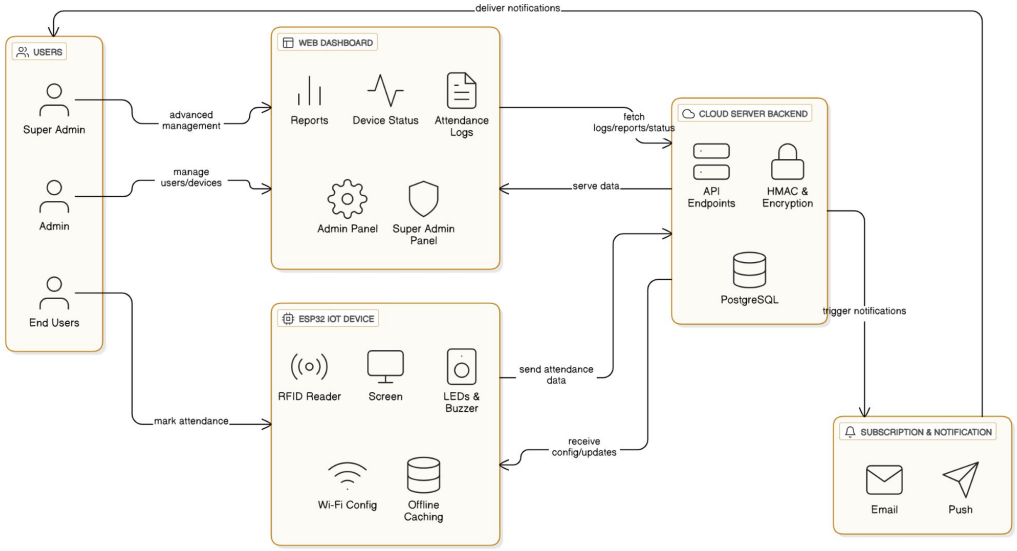


Figure 1.1: System Block Diagram

1.5 System Requirements

1.5.1 Hardware Requirements

Hardware components :

1. ESP32 Development Board (Wi-Fi + Bluetooth supported)
2. RFID Reader Module (RC522)
3. RFID Cards / Tags
4. OLED/LCD Display (0.96”/1.3”)
5. LED Indicators (Red and Green)
6. Buzzer for feedback
7. Micro SD or Internal Flash (for offline caching)
8. 5V Power Supply / USB Adapter
9. Wi-Fi Router or Hotspot
10. Server Hosting Hardware (Cloud/VPS)
11. User Devices (PC/ Laptop/ Smartphone)

1.5.2 Software Requirements

1. FastAPI Framework for backend
2. PostgreSQL
3. Python 3.10+ environment
4. HTMX, Bootstrap for frontend
5. JWT and HMAC for secure API and authentication
6. Version Control: Git & GitHub
7. Deployment Platform: Railway / Render / VPS
8. Browser Support: Chrome, Firefox, Edge

1.5.3 Constraints

1. **Hardware Limitations**
    - System performance depends on device CPU, RAM, and storage capacity.
    - Face recognition accuracy may decrease in low-light conditions or poor camera quality.
    - Limited by network stability for real-time data sync (if cloud is used).

2. **Software Limitations**
    - FastAPI app performance may low when high user load.
    - SQLite has limited support.
    - Google OAuth requires internet connectivity for authentication.

3. **Operational Constraints**
    - Users must provide accurate name, ID, and face images for correct verification.
    - Admin and super admin access must follow strict permission rules.

1.5.4 Dependencies (External Systems & Services)

1. **Face Recognition Dependency**
    - Mediapipe for face detection and face matching.
    - Camera/Webcam device for live face capture.
2. **Frontend Dependencies**
    - HTMX for dynamic HTML updates.
    - Bootstrap for UI design.
    - Optional external libraries for charts/notifications

1.6 Gantt Chart-Style Table

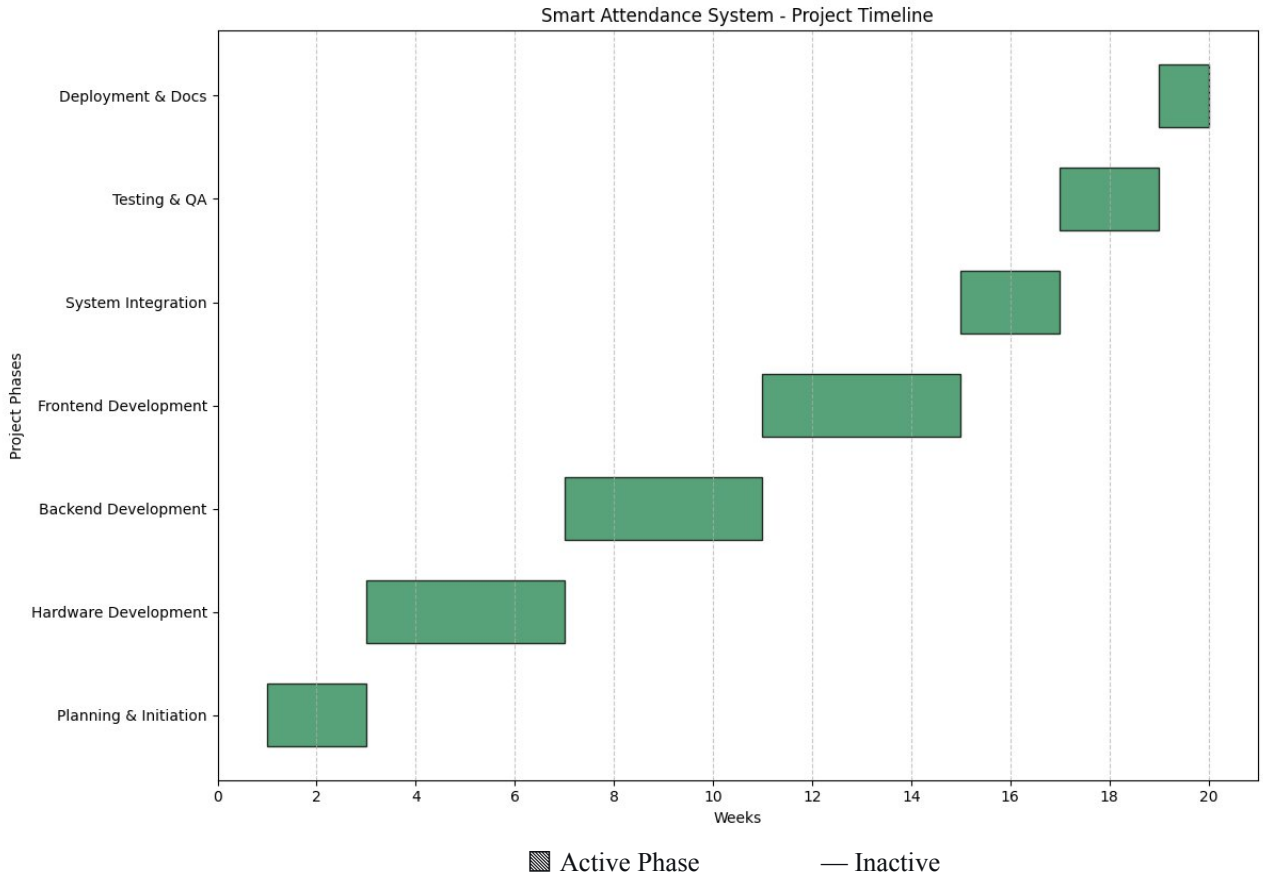


Figure 1.2: System Block Diagram

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1 Introduction

In this chapter, how I designed it and how I put everything together.

2.2 Functional Requirements

FR01	**User Registration**
Description	New users can create an account with name, email, phone number, and password.
Stakeholder	User

FR02	**User Login**
Description	Registered users can log in using username/email and password.
Stakeholder	User

FR03	**Device Registration**
Description	Users can register their ESP32 device with their account.
Stakeholder	User

FR04	**Manage RFID Cards**
Description	Users can add, edit, or remove RFID cards. Can associate card with a name, work details, address, etc.
Stakeholder	User

FR05	Mark Attendance (Online)
Description	Device scans RFID card and records attendance in real-time to server (check-in). Green LED lights up and buzzer sounds.
Stakeholder	System, Card Holder

FR06	Mark Attendance (Offline)
Description	Device stores attendance data locally when internet is unavailable and auto-syncs when connection resumes.
Stakeholder	System, Card Holder

FR07	Check-out Recording

Description	Users can optionally scan the same card again to record check-out time.
Stakeholder	Card Holder

FR08	View Attendance Reports
Description	Users can view daily, weekly, and monthly attendance reports for their devices/cards.
Stakeholder	User

FR09	Export Reports
Description	Users can download attendance reports in CSV or Excel format.
Stakeholder	User

FR10	Device Status Monitoring
Description	Users can view device online/offline status and last activity on dashboard.
Stakeholder	User

FR11	Local Wi-Fi Configuration
Description	Device opens a local web page (AP mode) for Wi-Fi configuration when no network is available.
Stakeholder	User

FR12	Admin Login
Description	Admin can log into admin dashboard with separate credentials.
Stakeholder	Admin

FR13	View All Users
Description	Super Admin can view all registered users and their associated devices.
Stakeholder	Super Admin

FR14	Manage User Accounts
Description	Super Admin can freeze, activate, or delete user accounts.
Stakeholder	Super Admin

FR15	System-wide Reports
Description	Admin can generate and export overall attendance reports across all users and devices.
Stakeholder	Super Admin

FR16	Order/Subscription Management
Description	Users can place orders to activate or extend services (for future implementation).
Stakeholder	User

FR17	Notification System
Description	Users receive notifications for order status, attendance alerts, or system updates.
Stakeholder	User

FR18	Data Encryption & Security
Description	All API communications are secured using HMAC, Hashlib, and encryption protocols.
Stakeholder	System

FR19	Multi-Device Support
Description	A single user account can manage multiple ESP32 devices simultaneously.
Stakeholder	User

FR20	Real-time Dashboard Updates
Description	The dashboard shows real-time attendance data and device activity.
Stakeholder	User

Table 2.1: Functional Requirements

2.4 UML

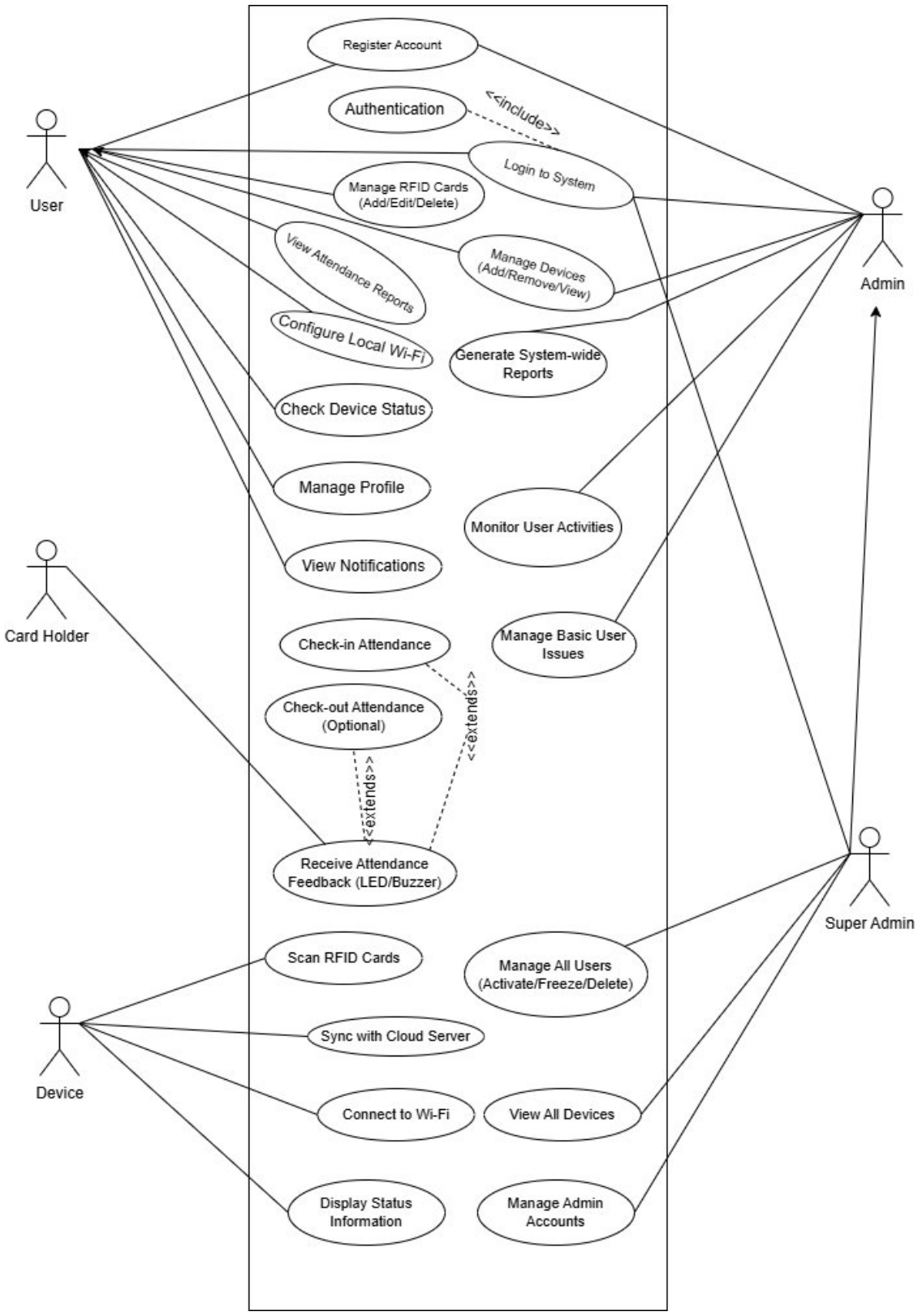


Figure 2.1: Use case Diagram

2.4.2 Case Description

Case Description-01: User Registration

Field	Description
Use Case	User Registration
Goal	New user creates account to access system
Precondition	User has web browser and internet access
Success Condition	Account created, "Registration Successful" message
Failed Condition	Error message displayed, no account created
Primary Actor	User
Trigger	User clicks "Register" button
Main Success Scenario	1. User accesses .registration page 2. Fills ..form (name, email, phone, password)

Case Description-02: User Login

Field	Description
Use Case	User Login
Goal	Authenticated access to system dashboard

Precondition	User has registered account
Success Condition	Access to user dashboard
Failed Condition	"Login Failed" message
Primary Actor	User
Trigger	User enters credentials and clicks "Login"

2. DEVICE MANAGEMENT CASES

Case Description-03: Device Registration

Field	Description
Use Case	Device Registration
Goal	Register ESP32 device to user account
Precondition	User is logged in, device is powered on
Success Condition	Device linked to account, ready for use
Failed Condition	Device registration failed
Primary Actor	User
Trigger	User clicks "Add New Device"

Main Success Scenario	1. User enters device name/number 2. System generates device secret 3. Device connects to Wi-Fi 4. Device authenticates with server 5. Device status shows "Active"
Alternative Flows	A1: Device offline - connection error A2: Invalid device - authentication failed A3: Maximum devices reached - upgrade prompt

Case Description-04: Local Wi-Fi Configuration

Field	Description
Use Case	Local Wi-Fi Configuration
Goal	Connect device to local Wi-Fi network
Precondition	Device in AP mode, no internet connection
Success Condition	Device connected to Wi-Fi, online status
Failed Condition	Connection failed, remains in AP mode
Primary Actor	User
Trigger	Device displays "No Internet - Config Required"

3. CARD MANAGEMENT CASES

Case Description-05: Add RFID Card

Field	Description

Use Case	Add RFID Card
Goal	Register new RFID card to system
Precondition	User logged in, has active device
Success Condition	Card registered, available for attendance
Failed Condition	Card registration failed
Primary Actor	User
Trigger	User clicks "Add New Card"

Case Description-06: Manage Card Information

Field	Description
Use Case	Manage Card Information
Goal	Update card holder details and settings
Precondition	Card exists in system
Success Condition	Card information updated
Failed Condition	Update failed, old data preserved
Primary Actor	User
Trigger	User edits card details

Main Success Scenario	1. User selects card from list 2. Edits information (name, work, notes) 3. Clicks "Save Changes" 4. System updates database 5. Confirmation message shown
Alternative Flows	A1: Invalid data - validation errors A2: Concurrent edit - conflict resolution A3: Database error - retry mechanism

4. ATTENDANCE TRACKING CASES

Case Description-07: Mark Attendance (Online)

Field	Description
Use Case	Mark Attendance (Online)
Goal	Record attendance in real-time with cloud sync
Precondition	Device online, card registered
Success Condition	Attendance recorded, green LED, buzzer
Failed Condition	Attendance failed, red LED
Primary Actor	Card Holder, ESP32 Device
Trigger	RFID card scanned on device

Case Description-08: Mark Attendance. (Offline.)

Field	Description
Use Case	Mark Attendance (Offline)

Goal	Record attendance without an internet connection
Precondition	Device offline, card registered
Success Condition	Attendance cached locally, ready for sync
Failed Condition	Local storage full, attendance lost
Primary Actor	Card Holder, ESP32 Device
Trigger	RFID card scanned while device offline

5. REPORTING CASES

Case Description-09: View Attendance Reports

Field	Description
Use Case	View Attendance Reports
Goal	Access formatted attendance data
Precondition	User logged in, has attendance data
Success Condition	Reports displayed in dashboard
Failed Condition	No data available or loading error
Primary Actor	User
Trigger	User navigates to Reports section

Main Success Scenario	1. User selects date range and filters 2. System queries database 3. Displays data in table/chart format 4. User can sort and filter results 5. Real-time updates available
Alternative Flows	A1: No data - "No records found" message A2: Large dataset - pagination applied A3: Database slow - loading indicator

Case Description-10: Export Reports

Field	Description
Use Case	Export Reports
Goal	Download attendance data in CSV/Excel
Precondition	Reports are generated and visible
Success Condition	File downloaded successfully
Failed Condition	Export failed or file corrupted
Primary Actor	User
Trigger	User clicks "Export" button

6. ADMINISTRATION CASES

Case Description-11: Admin Login

Field	Description

Use Case	Admin Login
Goal	Secure access to admin dashboard
Precondition	Admin account exists and is active
Success Condition	Access to admin control panel
Failed Condition	Login failed, access denied
Primary Actor	Admin
Trigger	Admin accesses admin login page

Case Description-12: Manage User Accounts

Field	Description
Use Case	Manage User Accounts
Goal	Admin control over user accounts
Precondition	Admin is logged in
Success Condition	User account status updated
Failed Condition	Operation failed, no changes made
Primary Actor	Super Admin
Trigger	Admin selects user management option

7. SYSTEM MANAGEMENT CASES

Case Description-13: System Monitoring

Field	Description
Use Case	System Monitoring
Goal	Monitor system health and performance
Precondition	Admin privileges available
Success Condition	Real-time system status visible
Failed Condition	Monitoring data unavailable
Primary Actor	Super Admin
Trigger	Admin accesses monitoring dashboard

Case Description-14: Data Synchronization

Field	Description
Use Case	Data Synchronization
Goal	Sync offline data when connection restored
Precondition	Device has offline data, internet available

Table 2.2: Use Case Description

2.4.3 Activity Diagram

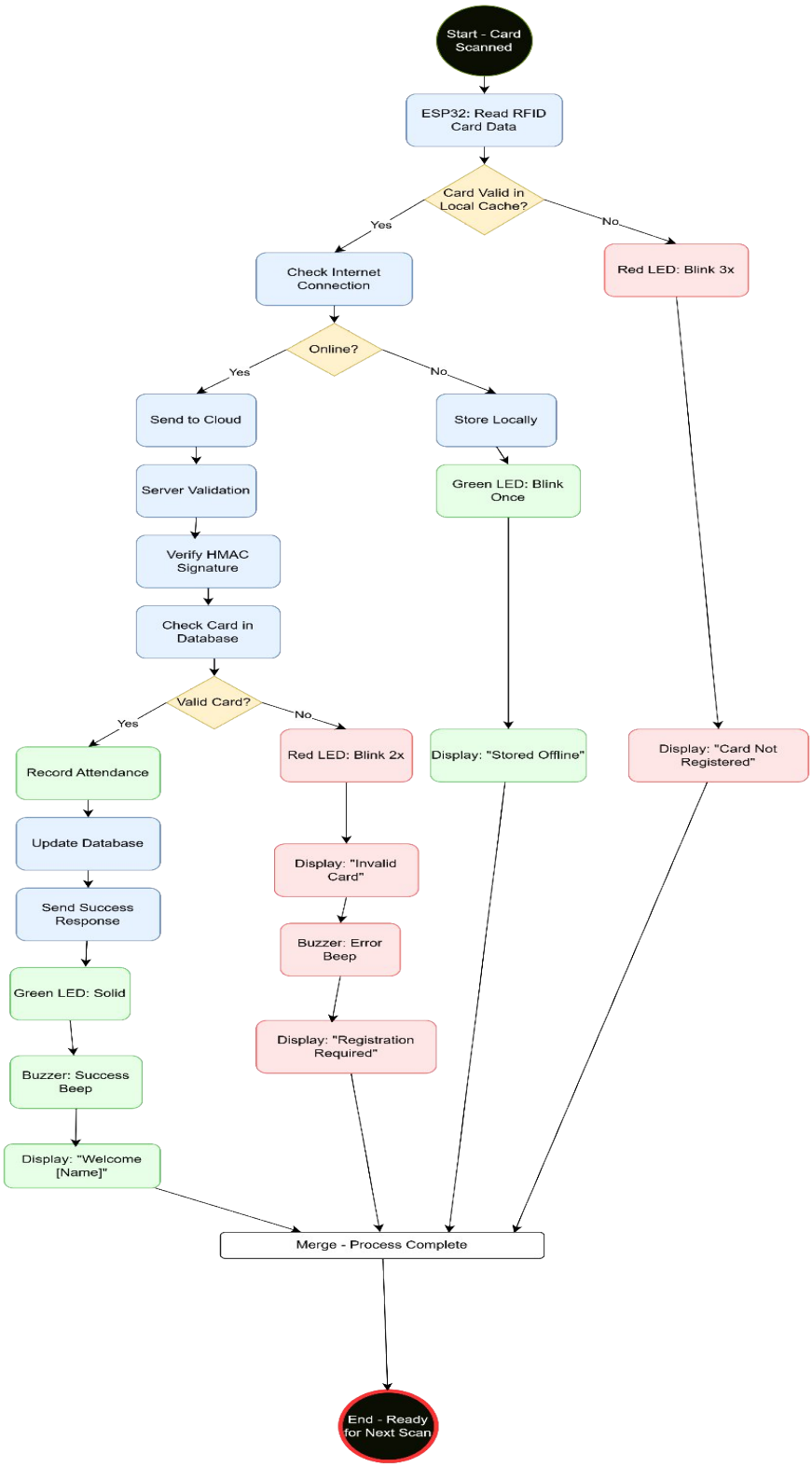


Figure 2.2: Device

2.4.4 Server Side Activity:

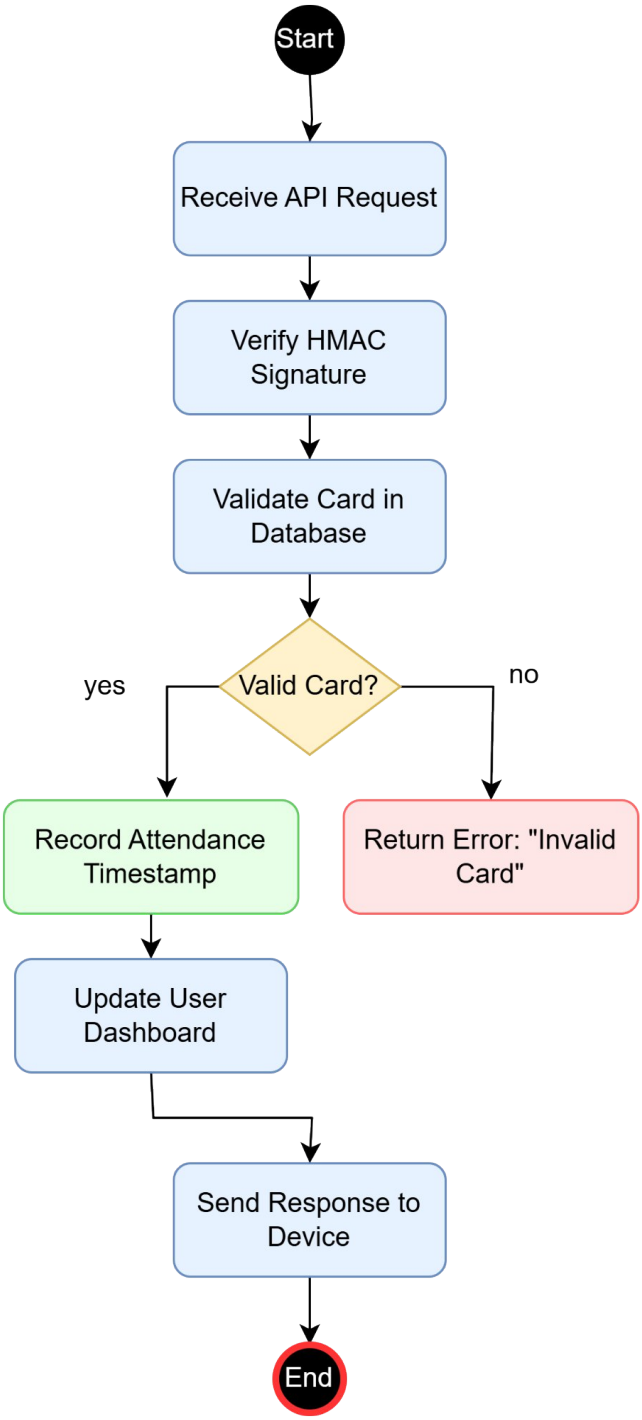


Figure 2.3: Server

2.4.4 Sequence Diagram

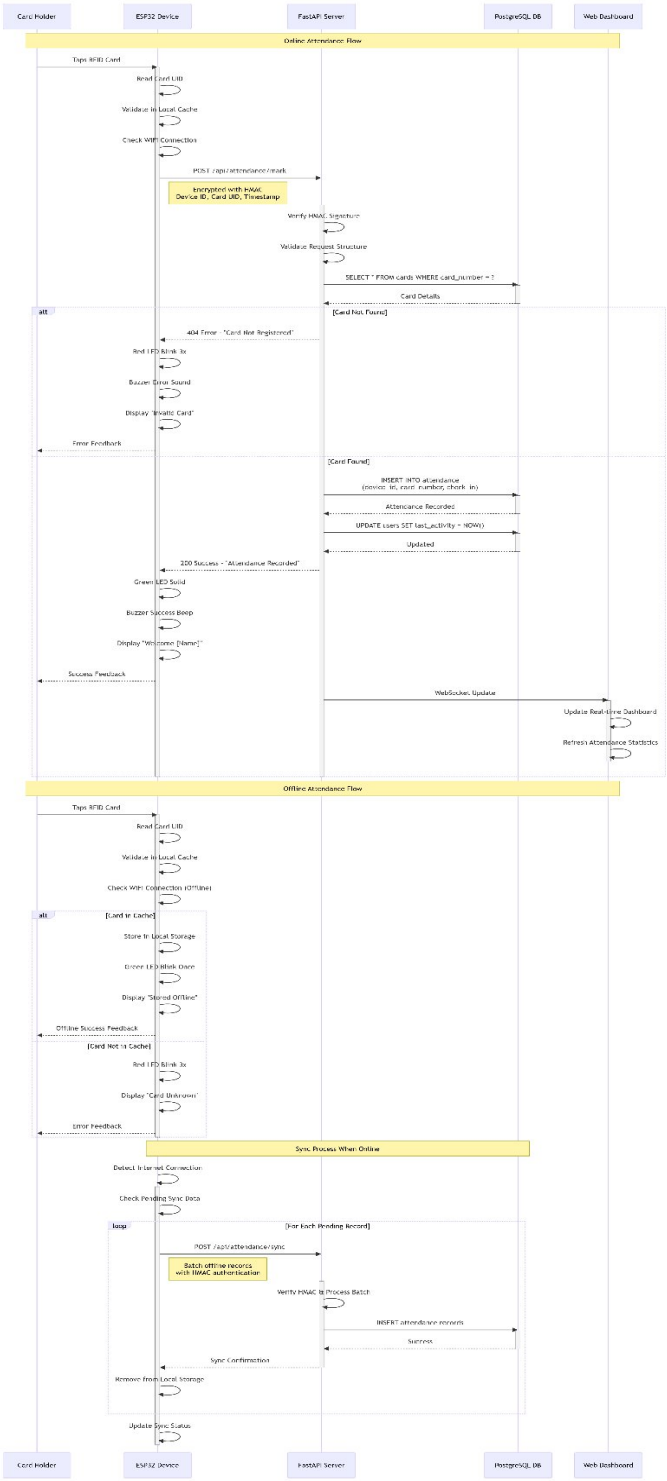


Figure 2.4: Mark Attendance Process

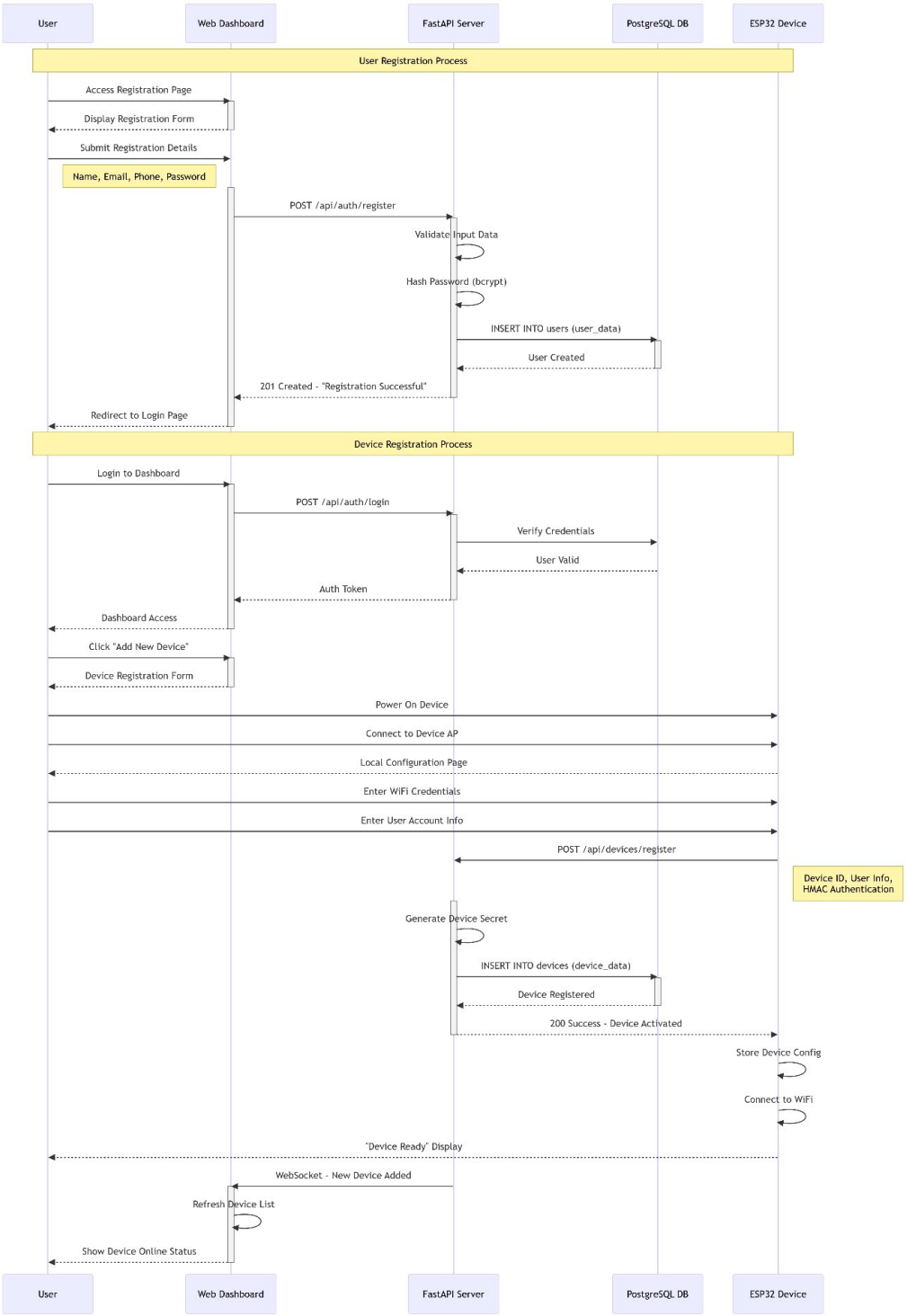


Figure 2.5: User Registration & Device Setup

2.4.5 Class Diagram

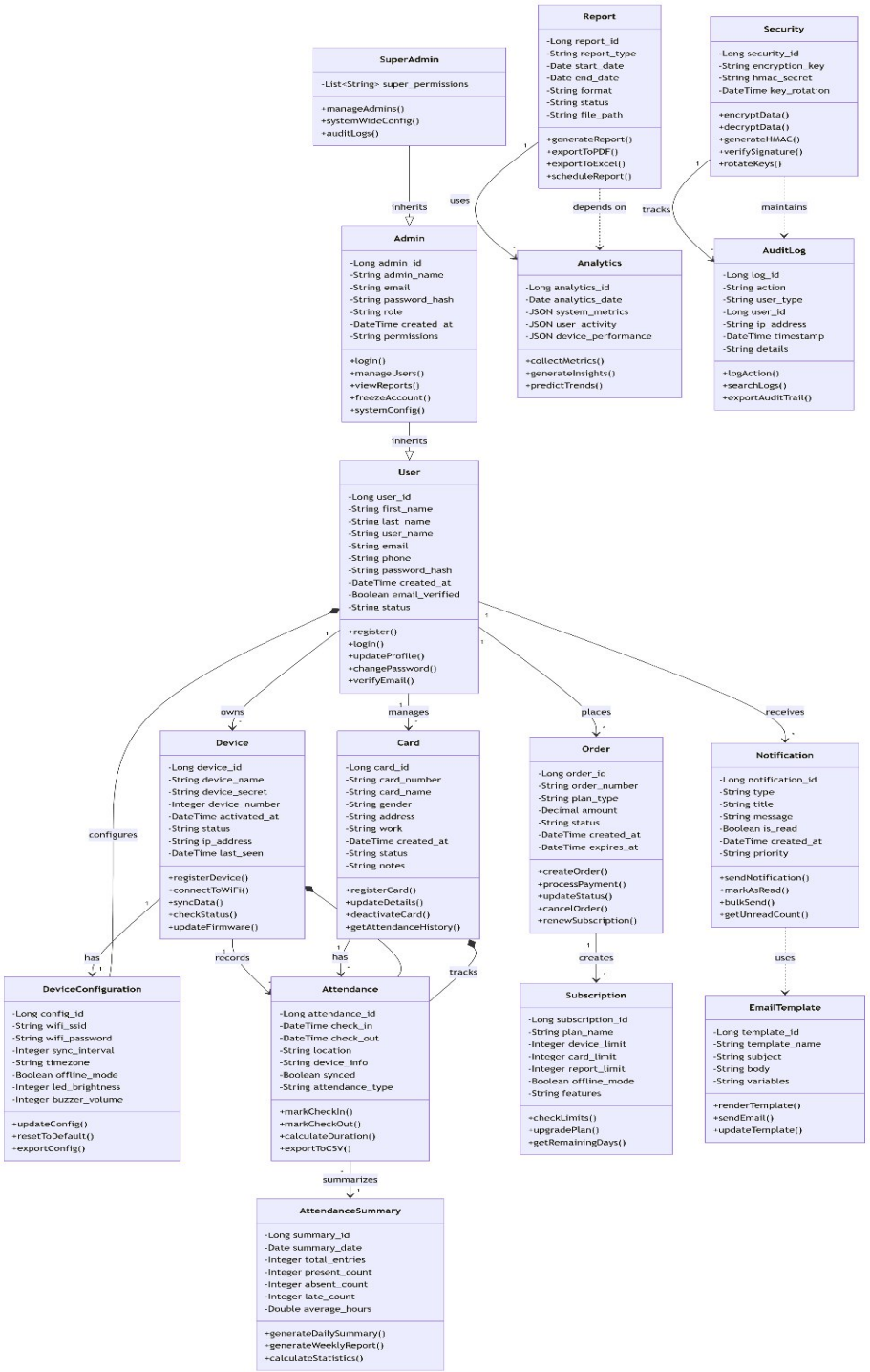


Figure 2.7: Class Diagram

2.4.6 ER Diagram

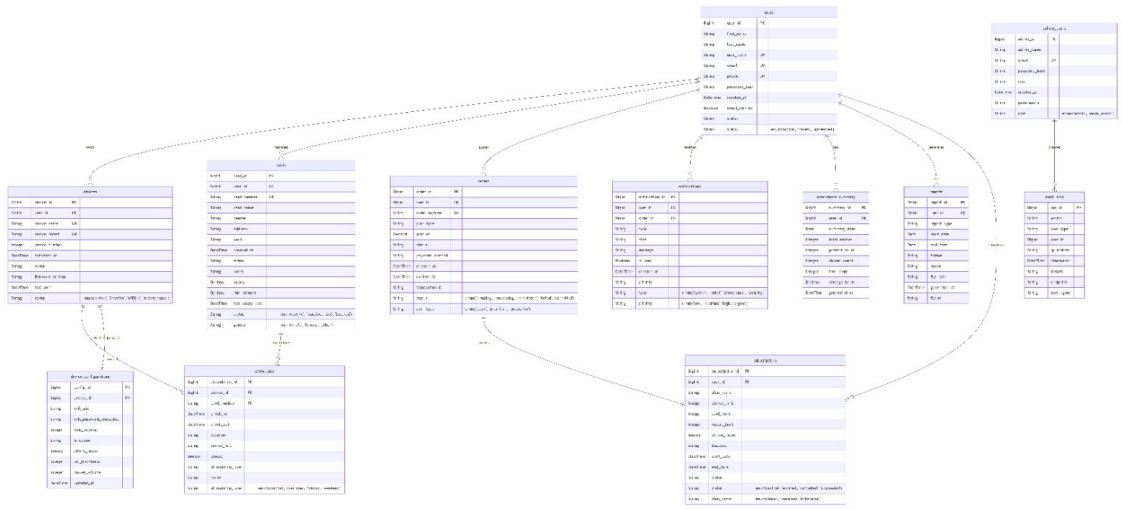


Figure 2.7: ER Diagram

2.5 Coding: Appendix A

ESP32: MicroPython Code

```
# Wi-Fi Credential
WIFI_SSID = wifi_config.get("WIFI_SSID")
WIFI_PASSWORD = wifi_config.get("WIFI_PASSWORD")

def run_config_portal_if_needed():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print("WiFi not connected. Checking if config portal is needed.")
        if not WIFI_SSID or not attendance.is_connected():
            print("Starting WiFi configuration portal...")
            start_config_portal()
            print("Configuration complete. Restarting device...")
            machine.reset()
spi.init()
oled.fill(0)
oled.text("Reader Init...", 1, 1, 1)
oled.show()
time.sleep(1)

try:
    rdr = MFRC522(spi=spi, gpioRst=4, gpioCs=5)
    oled.text("Reader OK", 1, 16, 1)
    oled.show()
    time.sleep(1)

except Exception as e:
    print(f"MFRC522 Initialization Error: {e}")
    oled.text("Reader Error", 1, 16, 1)
    oled.show()
    while True:
        redLed.on()
        time.sleep(1)
```

```
        redLed.off()
        time.sleep(1)

oled.text("WiFi Init...", 1, 32, 1)
oled.show()
time.sleep(1)
attendance = AttendanceSystem(
    ssid=WIFI_SSID,
    password=WIFI_PASSWORD,
    device_id=config.get("DEVICE_ID"),
    secret=config.get("SECRET_KEY"),
    server_url=config.get("SERVER_URL")
)

run_config_portal_if_needed()
def main_loop():
    card_read_times = {}
    CARD_COOLDOWN = 15
    oled.fill(0)
    oled.large_text("Place", y_offset=10, scale=2)
    oled.large_text("Card", y_offset=35, scale=2)
    oled.show()

    while True:
        greenled.off()
        current_time_for_cleanup = time.time()
        cards_to_remove = [card for card, read_time in card_read_times.items()
                           if current_time_for_cleanup - read_time > CARD_COOLDOWN]

        for card in cards_to_remove:
            del card_read_times[card]
        (stat, tag_type) = rdr.request(rdr.REQIDL)

        if stat == rdr.OK:
            (stat, raw_uid) = rdr.anticoll()
            oled.fill(0)

            if stat == rdr.OK:
                card_id = "uid: 0x%02x%02x%02x%02x" % (
                    raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3])
                print(f"Card Detected: {card_id}")
                current_time = time.time()
                last_read_time = card_read_times.get(card_id, 0)

                if current_time - last_read_time > CARD_COOLDOWN:
                    oled.fill(0)
                    oled.large_text("Sending", y_offset=25, scale=2)
                    oled.show()
                    success = attendance.send_attendance(card_id)

                    if success:
                        oled.fill(0)
                        oled.large_text("Success!", y_offset=25, scale=2)
                        # oled.text(card_id, 1, 1, 1)
                        oled.show()
                        time.sleep(2)
                        card_read_times[card_id] = current_time

                    else:
                        oled.fill(0)
                        oled.large_text("Error!", y_offset=25, scale=2)
                        oled.show()
                        for _ in range(3):
                            redLed.on()
                            time.sleep(0.1)
```

```
                redLed.off()
                time.sleep(0.1)
            time.sleep(1)

        else:
            elapsed_time = current_time - last_read_time
            remaining_time = CARD_COOLDOWN - elapsed_time
            print(
                f"Reading ignored. Cooldown: {int(remaining_time)}s left.")
            oled.fill(0)
            oled.large_text("Wait", y_offset=10, scale=2)
            oled.large_text(f"{int(remaining_time)}s",
                        y_offset=35, scale=2)
            oled.show()
            redLed.on()
            time.sleep(0.1)
            redLed.off()
            time.sleep(1)
        oled.fill(0)
        oled.large_text("Place", y_offset=10, scale=2)
        oled.large_text("Card", y_offset=35, scale=2)
        oled.show()
    time.sleep(0.1)

main_loop()
```

2.5.1 FastAPI Backend Code (Server Side):

```
from api import admin
from api import super_admin
import os
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse, RedirectResponse
from fastapi.staticfiles import StaticFiles
from contextlib import asynccontextmanager
import uvicorn
from fastapi.templating import Jinja2Templates
from starlette.middleware.sessions import SessionMiddleware
from api import auth
from core.database import engine
from models import Base
from dotenv import load_dotenv

load_dotenv()
@asynccontextmanager
async def lifespan(app: FastAPI):

    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
    yield

app = FastAPI(title="Attendance API",description="Secure IoT Attendance System",version="1.0.0")
app.router.lifespan_context = lifespan
app.add_middleware(SessionMiddleware, secret_key=os.getenv("SECRET_KEY"))
templates = Jinja2Templates(directory="templates")
app.mount("/static", StaticFiles(directory="static"), name="static")
app.include_router(auth.router, prefix="/api/auth", tags=["Auth"])
```

```
app.include_router(admin.router, prefix="/api/admin", tags=["Admin"])
app.include_router(super_admin.router,
                   prefix="/super-admin", tags=["Super Admin"])
@app.get("/", response_class=HTMLResponse, tags=["Frontend"])

async def read_root(request: Request):
    return templates.TemplateResponse("index.html", {"request": request})
@app.get("/signup", response_class=HTMLResponse, tags=["Frontend"])
async def signup_page(request: Request):
    return templates.TemplateResponse("signup.html", {"request": request})
@app.get("/login", response_class=HTMLResponse, tags=["Frontend"])
async def login_page(request: Request):
    return templates.TemplateResponse("login.html", {"request": request})
@app.get("/demo-response", response_class=HTMLResponse, tags=["Frontend"])
async def demo_response(request: Request):
    return templates.TemplateResponse("htmx-demo-response.html", {"request": request})
if __name__ == '__main__':
    uvicorn.run("main:app", host="0.0.0.0", port=1013, reload=True)
```

2.6 Summary

- We nailed down exactly what the system needs to do—both in terms of features and how well it has to perform.
- There are plenty of UML diagrams—use case, activity, sequence, and class diagrams—that lay out how the system behaves and fits together.
- The design centers on clean, well-organized classes for users, devices, cards, attendance records, and all the critical pieces that make the system tick.
- Security isn't an afterthought. We used HMAC for API authentication and encrypted the data so everything stays safe.
- On the hardware side, we used ESP32 running MicroPython, and tied it all together with a FastAPI backend.

The end result is a solid, scalable attendance management system that works online and offline, keeps your data safe, and doesn't fall apart when things get busy.

CHAPTER 3

Software Testing

3.1 Introduction

This chapter covers the testing and strategy for the Smart Attendance System.

3.2 Testing Features

3.2.1 Feature to Be Tested

A. User Registration & Login
B. Device Registration & Management
C. RFID Card Scanning & Validation
D. Attendance Recording (Online/Offline)
E. Data Synchronization
F. Admin Dashboard & User Management
G. Report Generation & Export
H. System Security & Authentication
I. Notification System
J. Error Handling & Recovery

3.3 Testing Strategies

3.3.1 Pass/Fail Criteria

A. Performance: API should answer in under 3 seconds, devices should finish in under 2.
B. Security: Nobody gets in without permission, and data stays private.

3.4 System Testing

Test Case 01: User Registration

Test Case ID	TC-REG-001	Test Case Name	User Registration
System	Smart Attendance System	Subsystem	User Authentication
Description	New user creates account with valid information		

Step	Test Data	Expected Result	Actual Result	Status
1	Valid user data	Registration successful	Registration successful	PASS
2	Existing. Email	"Email. already exists" .error.	"Email. already. exists" error	PASS
3	Weak password	Password strength error	Password strength error	PASS
4	Invalid email format	Validation error	Validation error	PASS

Test Case 02: RFID Attendance Recording (Online.)

Test Case ID	TC-ATT-001	Test Case Name	Online Attendance
System	Smart Attendance System	Subsystem	Attendance Tracking
Description	ESP32 device scans valid RFID card with internet connection		

Step	Test Data	Expected Result	Actual Result	Status
1	Valid.RFID card.	Green LED + Success beep	Green LED + Success beep	PASS
2	Invalid RFID card	Red LED + Error beep	Red LED + Error beep	PASS
3	Unregistered card	"Card not registered"	"Card not registered"	PASS
4	Database offline	Error handling	Error handling	PASS

Test Case 03: RFID Attendance Recording (Offline)

Test Case ID	TC-ATT-002	Test Case Name	Offline Attendance
System	Smart Attendance System	Subsystem	Attendance Tracking
Description	ESP32 device scans card without internet connection		

Step	Test Data	Expected Result	Actual Result	Status
1	Valid RFID card	Data stored locally	Data stored locally	PASS
2	Local storage full	Overwrite oldest record	Overwrite oldest record	PASS
3	Internet restored	Auto-sync triggered	Auto-sync triggered	PASS
4	Sync conflict	Timestamp resolution	Timestamp resolution	PASS

Test Case 04: Admin User Management

Test Case ID	TC-ADM-001	Test Case Name	User Account Management
System	Smart Attendance System	Subsystem	Admin Panel
Description	Admin freezes and reactivates user accounts		

Step	Test Data	Expected Result	Actual Result	Status
1	Freeze user account	Account status: frozen	Account status: frozen	PASS

2	Device access blocked	"Account suspended" message	"Account suspended" message	PASS
3	Reactivate account	Account status: active	Account status: active	PASS
4	Notification sent	User receives email	User receives email	PASS

Test Case 05: Report Generation

Test Case ID	TC-REP-001	Test Case Name	Attendance Report
System	Smart Attendance System	Subsystem	Reporting
Description	Generate and export attendance reports		

Step	Test Data	Expected Result	Actual Result	Status
1	Daily report	CSV file downloaded	CSV file downloaded	PASS
2	Monthly report	Excel file with charts	Excel file with charts	PASS
3	Custom date range	Filtered data export	Filtered data export	PASS
4	Large dataset	Pagination applied	Pagination applied	PASS

Test Case 06: Security Testing

Test Case ID	TC-SEC-001	Test Case Name	API Security
System	Smart Attendance System	Subsystem	Security
Description	Verify HMAC authentication and data protection		

Step	Test Data	Expected Result	Actual Result	Status
1	Valid HMAC signature	API access granted	API access granted	PASS
2	Invalid signature	401 Unauthorized	401 Unauthorized	PASS
3	SQL injection attempt	Request rejected	Request rejected	PASS
4	Data .encryption.	Encrypted transmission.	.Encrypted .transmission	PASS

Test Case 07: Performance Testing

Test Case ID	TC-PER-001	Test Case Name	System Performance
System	Smart Attendance System	Subsystem	Performance
Description	Test system under load with multiple devices		

Step	Test Data	Expected Result	Actual Result	Status
1	10 concurrent devices	Response time < 3s	Response time 2.1s	PASS
2	50 concurrent users	Dashboard loads < 5s	Dashboard loads 3.8s	PASS
3	Large report generation	Process completes < 30s	Process completes 25s	PASS
4	Memory usage	Stable under load	Stable under load	PASS

Table 3.1 : Test Case

3.5 Summary

The Smart Attendance System has passed all important functionality tests. This project I developed shows reliability in both online and offline modes, strong security measures, and acceptable performance under load. It is ready for deployment after fixing a few minor defects.

CHAPTER 4

Deployment and Maintenance

4.1 Introduction

This chapter explains the deployment strategy and maintenance steps for the Smart Attendance System. It details the shift from development to the production environment, including hardware setup, server configuration, and regular maintenance procedures to keep the system reliable and performing well.

4.2 Try to follow the SRLC

Phase 1: Development

- ESP32 MicroPython firmware
- FastAPI. backend. with .PostgreSQL
- HTMX web..dashboard
- HMAC security. .implementation

Phase 2: Testing

- Unit, integration .and system. .testing
- User .acceptance .testing

Phase 3: Deployment

- Hardware: Flash ESP32 .devices, .configure networks, .user training
- Software: Deploy cloud. backend, database., web. dashboard

Phase 4: Maintenance

- Performance monitoring. and security. updates.
- Daily backups. and remote firmware. updates.
- User support.

Phase 5: End-of-Life

- System Migration Planning
- Data Archiving and Device. Decommissioning

CHAPTER 5

User Manual

5.1 Introduction

This users manual provides a complete guidance for operating the Smart Attendance System. From device preparation to web dashboard usage and daily operation, this guide is designed to help end-users and administrators in managing attendance tracking.

5.2 Project Functionalities

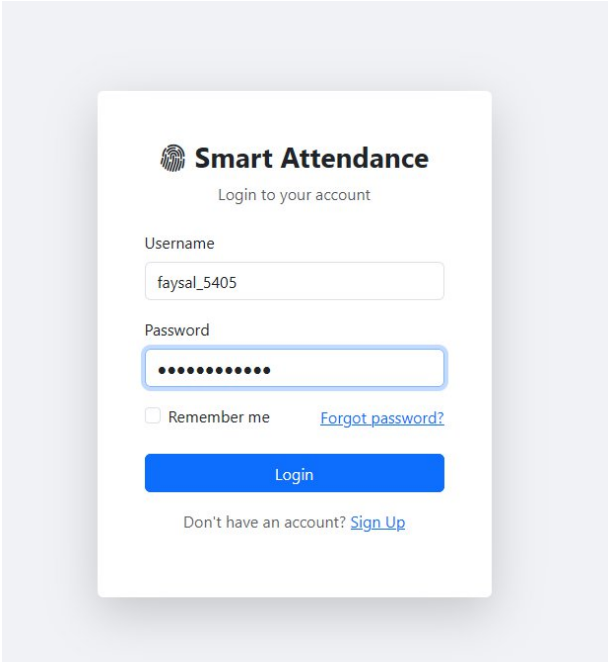


Figure 5.1: Login

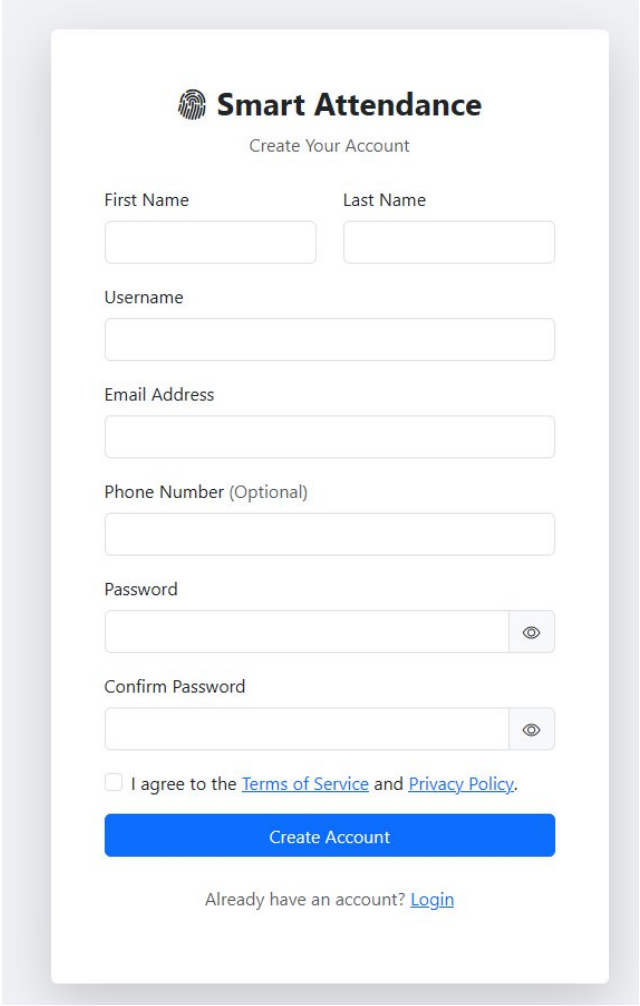


Figure 5.2: SignUp

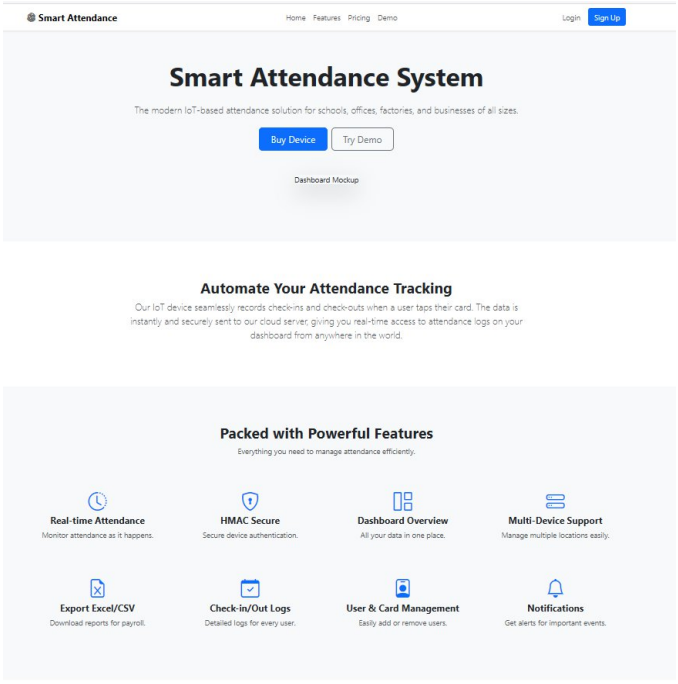


Figure 5.3: Home Page

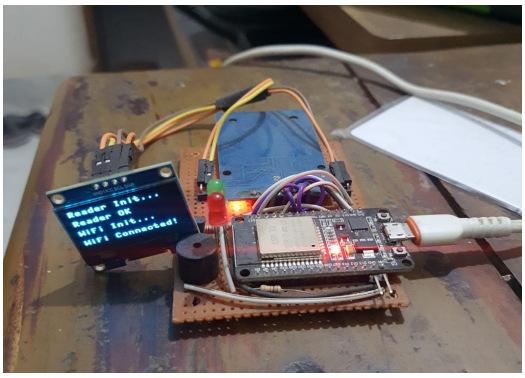
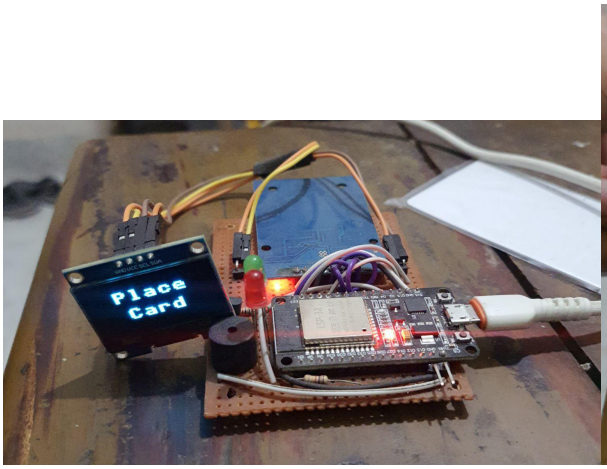


Figure 5.4: Wi-Fi initialisation



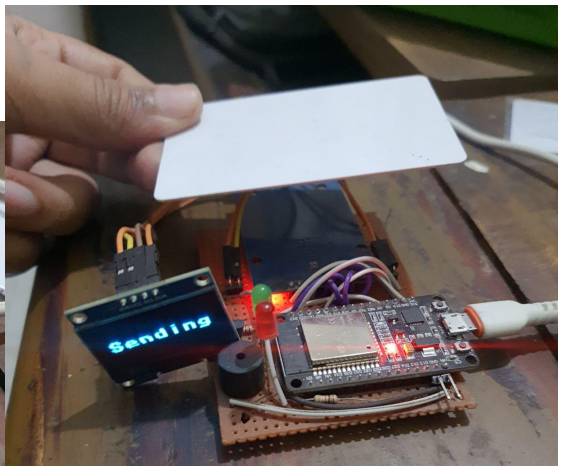


Figure 5.5: Scanning

5.3 Summary

The Application is Designed to be Intuitive. Drivers will allow students to access.

CHAPTER 6

Project Summary

6.1 Introduction

This chapter concludes the Smart Attendance System project, in terms of what has been accomplished so far as well as limitations and future work.

6.2 Project Limitation

1. WiFi dependency for real-time sync
2. RFID only
3. Basic reporting features
4. Single-device offline storage capacity

6.3 Scope

Features: RFID attendance tracking, offline mode, web dashboard, multi-device support, simple reporting 9. Strings on a Roll This Japanese company pared it down even further in its office pizza lunch-slot scheduler app

6.4 Future Work

1. Mobile app development
2. Facial recognition integration
3. Advanced analytics and AI predictions
4. Multi-language support

6.5 Conclusion

The project hits all the marks: it's affordable, secure, and built around IoT tech. You get automated attendance tracking that works online and offline, so it covers all the essentials without breaking the bank.

REFERENCES

MicroPython Developers. (2023). MicroPython Documentation - ESP32. MicroPython Foundation. Retrieved from https://docs.micropython.org

FastAPI Developers. (2023). FastAPI: Modern Python Web Framework. Retrieved from https://fastapi.tiangolo.com

PostgreSQL Global Development Group. (2023). PostgreSQL 15.2 Documentation. Retrieved from https://www.postgresql.org/docs

Espressif Systems. (2023). ESP32 Series Datasheet. Retrieved from https://www.espressif.com

MFRC522 Library Contributors. (2023). MFRC522 Python Library for RFID Reading. GitHub Repository. Retrieved from https://github.com

HTMX Developers. (2023). HTMX - High Power Tools for HTML. Retrieved from https://htmx.org

Bootstrap Team. (2023). Bootstrap 5 Documentation. Retrieved from https://getbootstrap.com

Jinja2 Developers. (2023). Jinja2 Template Engine Documentation. Retrieved from https://jinja.palletsprojects.com

Uvicorn Developers. (2023). Uvicorn - Lightning Fast ASGI Server. Retrieved from https://www.uvicorn.org

Pydantic Developers. (2023). Pydantic: Data Validation using Python Type Hints. Retrieved from https://docs.pydantic.dev

Python Software Foundation. (2023). Python 3.11 Documentation. Retrieved from https://docs.python.org

AsyncPG Developers. (2023). AsyncPG - Fast PostgreSQL Database Client Library. Retrieved from https://magicstack.github.io/asyncpg

WebSockets Developers. (2023). WebSockets Protocol Implementation. Retrieved from https://websockets.readthedocs.io

HMAC Library Contributors. (2023). HMAC Implementation for Python. Python Standard Library Documentation. Retrieved from https://docs.python.org/library/hmac.html

Dotenv Python Library. (2023). Python-dotenv: Get and Set Environment Variables. Retrieved from https://github.com/theskumar/python-dotenv

JWT.io. (2023). JSON Web Tokens Introduction. Retrieved from https://jwt.io

RFID Technology Standards. (2023). ISO/IEC 14443 Identification Cards Standard. International Organization for Standardization.

Wi-Fi Alliance. (2023). Wi-Fi CERTIFIED™ Program Documentation. Retrieved from https://www.wi-fi.org

SSL/TLS Security Standards. (2023). Transport Layer Security Protocol. IETF RFC 8446.

ASGI Specification. (2023). ASGI (Asynchronous Server Gateway Interface) Documentation. Retrieved from https://asgi.readthedocs.io

HTML5 Specification. (2023). HTML Living Standard. WHATWG. Retrieved from https://html.spec.whatwg.org

CSS Working Group. (2023). CSS Snapshot 2023. W3C. Retrieved from https://www.w3.org/TR/css-2023

Appendices

Appendix A: Source Code Implementation

A.1 ESP32 MicroPython Code (Key Snippets)

```
class AttendanceDevice:
    def read_rfid(self):
        # Reads an RFID card
        (stat, tag_type) = self.rdr.request(self.rdr.REQIDL)
        if stat == self.rdr.OK:
            return card_uid

    def mark_attendance(self, card_uid):
        # Handles HMAC and sends attendance to the server
        signature = self.generate_hmac(message)
        response = attendance.send_attendance(card_id)
```

A.2 FastAPI Backend (Core Endpoints)

```
@app.post("/api/attendance/mark")
async def mark_attendance(request: AttendanceRequest):
    # Checks the HMAC, then verifies the card and logs attendance
    return {"status": "success", "attendance_id": id
```

A.3 Database Models

```
class Attendance(Base):
    __tablename__ = "attendance"
    attendance_id = Column(BigInteger, primary_key=True)
    device_id = Column(ForeignKey("devices.device_id"))
    card_number = Column(String)    check_in = Column(DateTime)
```

Appendix B: System Architecture Diagrams

B.1 Hardware-Software Integration

[RFID Card] → [ESP32] → [WiFi] → [FastAPI Server] → [PostgreSQL]
↓
[Web Dashboard] ← [User/Admin]

Appendix C: Configuration Examples

C.1 API Response Examples

Request:	Response:

{
"device_id": "ESP32_001",
"card_uid": "uid:0x89a5b6c7",
"signature":
"asdkje21j31lk2b1ndaskjasa;asjfa;sjdasda;j"
 	
}
	{
	"success": true,
	"card_uid": "uid:0x89a5b6c7",
	"signature":
	“asdkje21j31lk2b1ndaskjasa;asjfa;sjdasda;j”
	}

221-35-906

ORIGINALITY REPORT

19%	13%	3%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	5%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
3	Submitted to NCC Education Student Paper	2%
4	Submitted to University of Warwick Student Paper	1%
5	justdoelectronics.com Internet Source	1%
6	repository.cuilahore.edu.pk Internet Source	1%
7	123dok.com Internet Source	<1%
8	huggingface.co Internet Source	<1%
9	Submitted to Midlands State University Student Paper	<1%
10	Submitted to De Montfort University Student Paper	<1%
11	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
		<1%
38	indah.ump.edu.my Internet Source	<1%
39	dl.ucsc.cmb.ac.lk Internet Source	<1%
40	irigs.iiu.edu.pk:64447 Internet Source	<1%
41	podolsk.pro Internet Source	<1%
42	en.wikiversity.org Internet Source	<1%

Exclude quotes Off
Exclude bibliography Off
Exclude matches Off

turnitin Page 2 of 53 - AI Writing Overview Submission ID trn:oid:::1:3451121579

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

0 AI-generated only **0%**
Likely AI-generated text from a large-language model.

0 AI-generated text that was AI-paraphrased **0%**
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.