

**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**FACE DETECTION ATTENDANCE SYSTEM USING EUCLIDEAN DISTANCE**

**A PROJECT REPORT**

**Submitted To**

**Department of Computer Application**

**Aadim National College**

***In partial fulfillment of the requirements for the Bachelors in Computer Application***

Submitted By:

Nigam Rai (105902063)

Jestha 2081

6th Semester

Under the supervision of

**Er. Prem Subed****i**

**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Aadim National College**

# SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **NIGAM RAI** entitled **“FACE DETECTION ATTENDANCE SYSTEM BY USING EUCLIDEAN DISTANCE ALGORITHM”** in partial fulfillment of the requirements for the degree of Bachelor of Computer Application is recommended for the final evaluation.

………………

**SIGNATURE**

**Er. Prem Subedi**

**Supervisor**

**Lecturer**

Aadim National College

Chabahil, Chuchepati, Kathmandu





**Tribhuvan University**

**Faculty of Humanities and Social Science**

**Aadim National College**

# LETTER OF APPROVAL

This is to certify that this project prepared by **NIGAM RAI** entitled **“FACE DETECTION ATTENDANCE SYSTEM BY USING EUCLIDEAN DISTANCE ALGORITHM”** in partial fulfillment of the requirements for the degree of Bachelor's in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| ………………………….  **Er. Prem Subedi**  **Supervisor**  **Lecturer**  Aadim National College  Chabahil, Chuchepati, Kathmandu | ….……………………….  **Yogendra Bahadur Mahata**  **Coordinator**  Aadim National College  Chabahil, Chuchepati, Kathmandu |
| ……………………………  **Internal Examiner** | .………………………….  **External Examiner** |

# Abstract

Manual attendance systems are often inefficient and error-prone. They require significant human effort and are susceptible to issues like proxy attendance, incorrect data entry, and lost records. In academic institutions and workplaces where attendance tracking is crucial, these limitations can lead to administrative burdens and inaccurate reporting. The motivation for this project was to automate the attendance process, eliminate tampering, and improve reliability and efficiency through real-time face detection.

The objective was to create a system that could recognize faces and mark attendance automatically, helping reduce human involvement, prevent proxy attendance, and allow real-time tracking of records. The system also intended to provide easy access to attendance history and reports for both users and administrators through a user-friendly platform.

React Native was used for the mobile frontend due to its smooth experience on Android devices. The backend was developed using Node.js and Express.js, while Python powered the face recognition component using machine learning libraries such as face-recognition. MongoDB, a NoSQL database, was used to store attendance data.

The final system successfully detected and recognized faces, marked attendance automatically, and maintained secure records. It met its goals by simplifying the process, reducing manual effort, and offering a reliable tool for real-world use.

***Keywords:*** *Traditional Attendance Systems, Automated Face Detection, Proxy Attendance, Real-Time Tracking, Attendance History, React Native, Node.js, Express.js, API, Python, Face-Recognition, MongoDB, NoSQL, Database*

# Acknowledgements

I would like to express my heartfelt gratitude to Tribhuvan University (TU) and Aadim National College for providing me with the opportunity and resources to undertake this project. The support and environment offered by the institution have played a vital role in the successful completion of this work.

I am deeply thankful to my supervisor, Er. Prem Subedi, for his continuous guidance, valuable feedback, and unwavering encouragement throughout the project. His insights and expertise have been instrumental in shaping the direction and outcome of this work.

I would also like to extend my sincere thanks to the Head of Department, Er. Prem Subedi, for his support and for facilitating a productive learning environment.

My sincere appreciation goes to all the IT staff for their cooperation and for providing me with sufficient information and assistance during the course of the project.

Lastly, I am immensely obliged to my friends for their uplifting inspiration, encouraging guidance, and kind support, which helped me remain focused and motivated throughout the completion of this project.

Sincerely,

Nigam Rai (105902065)

**Table of Contents**

[SUPERVISOR'S RECOMMENDATION ii](#_Toc205470791)

[SUPERVISOR'S RECOMMENDATION iii](#_Toc205470792)

[LETTER OF APPROVAL iv](#_Toc205470793)

[Abstract v](#_Toc205470794)

[Acknowledgements vi](#_Toc205470795)

[List of Figures ix](#_Toc205470796)

[List of Tables x](#_Toc205470797)

[Chapter 1: Introduction 1](#_Toc205470798)

[1.1 Introduction 1](#_Toc205470799)

[1.2 Problem Statement 1](#_Toc205470800)

[1.3 Objectives 1](#_Toc205470801)

[1.4 Scope and Limitation 2](#_Toc205470802)

[1.4.1 Scope 2](#_Toc205470803)

[1.4.2 Limitations 2](#_Toc205470804)

[1.5 Development Methodology 2](#_Toc205470805)

[1.6 Report Organization 3](#_Toc205470806)

[Chapter 2: Background Study and Literature Review 4](#_Toc205470807)

[2.1 Background Study 4](#_Toc205470808)

[2.2 Literature Review 4](#_Toc205470809)

[Chapter 3: System Analysis and Design 6](#_Toc205470810)

[3.1 System Analysis 6](#_Toc205470811)

[3.1.1 Requirement Analysis 6](#_Toc205470812)

[3.1.2 Feasibility Analysis 8](#_Toc205470813)

[3.1.3 Object Modelling: Object & Class Diagram 9](#_Toc205470814)

[3.1.4 Dynamic Modelling: State & Sequence Diagram 11](#_Toc205470815)

[3.1.5 Process Modelling: Activity Diagram 12](#_Toc205470816)

[3.2 System Design 14](#_Toc205470817)

[3.2.1 Refinement of Classes and Object 14](#_Toc205470818)

[3.2.2 Component Diagram 15](#_Toc205470819)

[3.2.3 Deployment Diagram 16](#_Toc205470820)

[3.3 Algorithm Details 16](#_Toc205470821)

[Chapter 4: Implementation and Testing 19](#_Toc205470822)

[4.1 Implementation 19](#_Toc205470823)

[4.1.1 Tools Used 19](#_Toc205470824)

[4.1.2 Implementation Details of Modules 19](#_Toc205470825)

[4.2 Testing 22](#_Toc205470826)

[4.2.1 Test Cases for Unit Testing 22](#_Toc205470827)

[4.2.2 Test Cases for System Testing 25](#_Toc205470828)

[Chapter 5: Conclusion and Future Recommendations 26](#_Toc205470829)

[5.1 Conclusion 26](#_Toc205470830)

[5.2 Lesson Learnt/Outcome 26](#_Toc205470831)

[5.3 Future Recommendations 26](#_Toc205470832)

[References 27](#_Toc205470833)

# List of Figures

[Figure 1.1 CRISP-DM Methodology 2](#_Toc205470834)

[Figure 3.1 Use Case Diagram 6](#_Toc205470835)

[Figure 3.2 Class Diagram 9](#_Toc205470836)

[Figure 3.3 Object Diagram 10](#_Toc205470837)

[Figure 3.4 State Diagram 11](#_Toc205470838)

[Figure 3.5 Sequence Diagram 12](#_Toc205470839)

[Figure 3.6 User Activity Diagram 12](#_Toc205470840)

[Figure 3.7 Admin Activity Diagram 13](#_Toc205470841)

[Figure 3.8 Refinement of Classes 14](#_Toc205470842)

[Figure 3.9 Refinement of Objects 15](#_Toc205470843)

[Figure 3.10 Component Diagram 15](#_Toc205470844)

[Figure 3.11 Deployment Diagram 16](#_Toc205470845)

# List of Tables

[Table 3.1 Functional Requirements 7](#_Toc205470851)

[Table 3.2 Non-Functional Requirements 7](#_Toc205470852)

[Table 4. 1 Tools Used 19](#_Toc205470853)

[Table 4.2 Test Cases for User Login 22](#_Toc205470854)

[Table 4.3 Test Cases for Face Matching 22](#_Toc205470855)

[Table 4.4 Test Cases for Attendance Record 23](#_Toc205470856)

[Table 4.5 Test Cases for Integration & API Testing 25](#_Toc205470857)

# Chapter 1: Introduction

## Introduction

A Face Detection Attendance System is a biometric system that uses face detection technology to for easy and simple attendance marking. Unlike the traditional ways that use manual signtaures, or fingerprint scanners, this system marks attendance based on the basis of individual's unique facial features for a fast and contact way of marking attendance.

By using artificial intellingence and computer webcam, the system captures and processes face photographs in real-time and verifies the person. This helps to remove proxy attedance and human error, making attendance management more reliable and faster.

Face detection attendance systems are generally used in offices, colleges, hospitals etc where accurate attendance management is needed.

## Problem Statement

Traditional attendance systems are often susceptible to manipulation, particularly in cases of proxy attendance or buddy punching, where one individual marks the attendance on behalf of another. Furthermore, systems based on RFID cards or fingerprint scanners require some degree of physical contact, which raises concerns related to hygiene and long-term wear-and-tear. These drawbacks become especially problematic in high-traffic environments over extended periods.

Such limitations give rise to a range of challenges in educational institutions, workplaces, and other organizational settings—namely, inconsistent attendance data, increased administrative workload, and weakened security measures. Moreover, managing attendance for large groups can become a logistically complex task, ultimately affecting overall efficiency and productivity.

To overcome these issues, the Face Detection Attendance System presents a contactless, automated solution designed to eliminate the need for human intervention. Unlike traditional systems, it prevents fraudulent attendance and reduces the potential for hygiene-related concerns. By leveraging AI-based facial recognition technology, the system ensures high accuracy, supports real-time processing, and offers seamless integration into existing infrastructures. This makes it a superior alternative to conventional methods, providing a secure, scalable, and efficient approach to attendance management.

## Objectives

* To develop a web-based face recognition system that allows individuals to mark their attendance automatically by scanning their face without any manual input.
* To allow users and the admins to view attendance record.

## Scope and Limitation

### Scope

* The project will capture and verify to mark attendance without manual intervention.
* The project will store the attendance and allowed to be viewed by user and admin

### Limitations

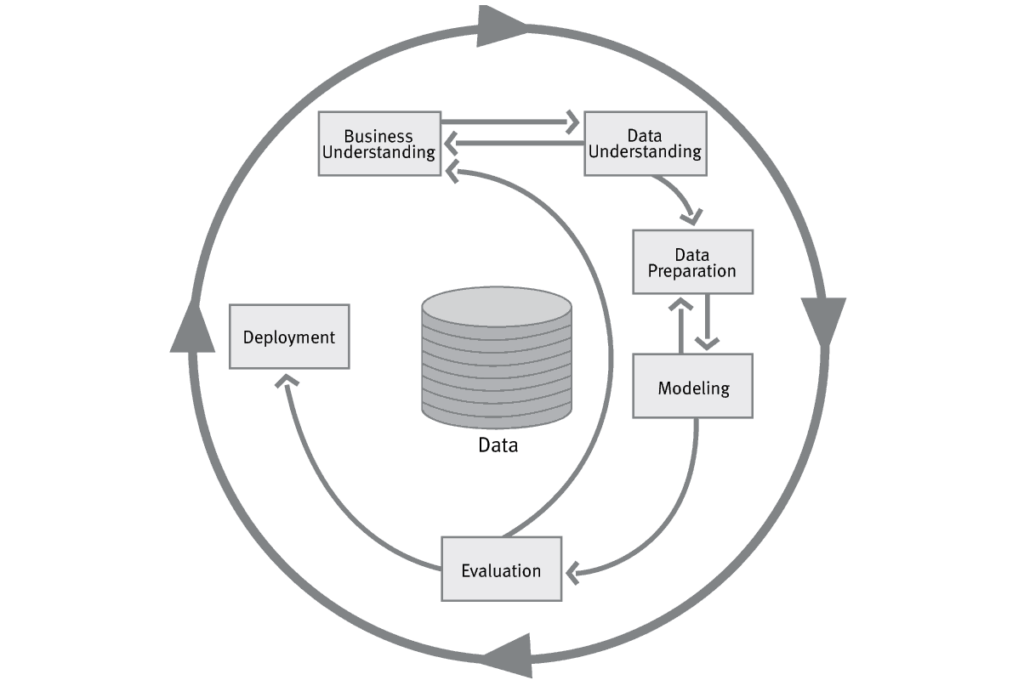
* The system's accuracy may decrease under poor lighting conditions, partial facial occlusion (e.g., masks, hats), or significant changes in facial appearance.
* The facial recognition model may not function optimally on low-end devices with limited camera or processing capabilities.
* Initial training and enrollment of users into the system must be performed manually to ensure reliable face recognition.

## Development Methodology

For the methodology of the proposed system, CRISP-DM methodology is being used.

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. It is a robust and well-proven methodology. We do not claim any ownership over it. We did not invent it. We are however evangelists of its powerful practicality, its flexibility and its usefulness when using analytics to solve thorny business issues. It is the golden thread than runs through almost every client engagement. The CRISP-DM model is shown on the right.

This model is an idealised sequence of events. In practice many of the tasks can be performed in a different order and it will often be necessary to backtrack to previous tasks and repeat certain actions. The model does not try to capture all possible routes through the data mining process.



**Figure 1.1 CRISP-DM Methodology**

## Report Organization

The first part of the report contains the summarized introduction of the whole report. It includes the overview, scope and limitation, problem statement and objectives of this project.

The second chapter includes background study i.e., description of fundamental theories, general concepts and terminology related to the project. It also includes the literature review i.e., review of the similar projects, research and theories done by other researchers.

The third chapter includes the system analysis and design phase in which the report of functional and non-functional requirements of the project is stated using use case and system diagrams. It also includes the feasibility study about the system which explains whether the system development process is affordable and within the knowledge range of the developers. It shows the technical, operational and economic feasibility of the project development phase. The explanation of the designing of the system is also done in this chapter. It includes data modeling and process modeling which is explained by using ER diagram and Data Flow Diagram. The architectural design, database design and the user interface design are also listed in this chapter.

# Chapter 2: Background Study and Literature Review

## 2.1 Background Study

Attendance recording is a fundamental operation in various atmospheres such as schools, universities, corporate offices, and events. Traditionally, this task is accomplished by using manual methods like attendance sheets and punch cards or by semi-automated methods such as using RFID cards and biometric fingerprint scanners.

Such methods, though performing the task to a large extent, have many restrictions:

* Manual systems are prone to instances of proxy attendance, forgery, and human error. It becomes very cumbersome and inefficient to keep large volumes of paper-based attendance records and even to analyze them.
* Biometric systems, such as fingerprint scanners, though offering stronger security, demand physical contact, thus posing health concerns, especially in a crisis situation such as the COVID-19 pandemic. They also fail to work effectively if a user's finger is dirty, wet, or injured.
* Card-based systems such as RFID or ID cards are easy to lose and allow the holder to share or misuse the card; thus, there are reliability and security concerns.

In recent years, the realms of artificial intelligence (AI) and computer vision have opened new possibilities for modern and contactless identity verification systems. Within this spectrum, face detection and recognition have grown popular as being non-invasive, fast, and relatively accurate.

## 2.2 Literature Review

Face recognition-based attendance systems have emerged as an efficient alternative to traditional methods of recording attendance, which are often manual, time-consuming, and vulnerable to manipulation such as proxy marking. Conventional systems like RFID and biometric fingerprint scanners, though improvements over manual systems, still require physical contact or close proximity, which can be limiting in large or dynamic environments [1].

The introduction of automated face detection and recognition technologies has significantly improved the automation of attendance systems. The Viola-Jones algorithm was one of the first widely used methods for real-time face detection, using Haar-like features and a cascade of classifiers to detect frontal faces efficiently [2]. Despite its speed, it is sensitive to lighting, pose variations, and partial occlusions.

With advancements in computer vision, more robust face detection approaches such as Histogram of Oriented Gradients (HOG) and deep learning-based Convolutional Neural Networks (CNNs) have been adopted. Dalal and Triggs demonstrated the effectiveness of HOG for human detection, which laid the groundwork for facial feature extraction [3]. Later, Schroff et al. introduced FaceNet, a deep CNN that directly learns a mapping from face images to a compact Euclidean space where distances correspond to a measure of facial similarity [4].

Libraries like Dlib have made these techniques more accessible by providing pre-trained models for facial landmark detection and recognition, which have been widely used in real-time attendance systems [5]. In educational settings, machine learning-based facial recognition has shown high accuracy in automating attendance logging while preventing proxy attendance [6].

Mobile platforms such as React Native have enabled the development of cross-platform applications that work seamlessly on Android and iOS. Combined with a Node.js and Express.js backend, and a MongoDB NoSQL database for storage, developers have created end-to-end attendance systems capable of real-time operation and remote access to data [7].

However, issues such as facial occlusion (e.g., due to masks), low lighting conditions, and privacy concerns remain significant challenges. Future systems must continue to improve in robustness while addressing ethical concerns related to facial data collection and storage [8].

# Chapter 3: System Analysis and Design

## 3.1 System Analysis

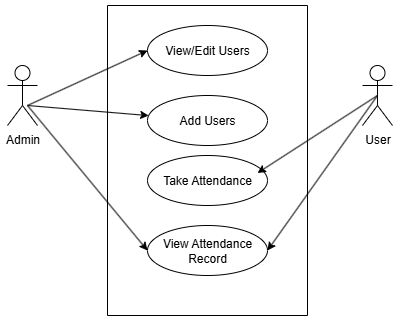
This chapter discusses about the proposed model of the system, the methodology used in building the system, tools and techniques, requirement analysis, requirement specification. It also talks about the functional requirements of the system, the system design. The application architecture of the system will also be shown in this chapter, the use case diagram, data design, activity diagrams, dataflow diagram, control flow diagram, entity-relationship diagram (ERD) and user interface design will all be shown in this chapter.

### 3.1.1 Requirement Analysis

The following section presents the complete set of functional requirements of PARG system.

#### Functional Requirements

Following figure shows the requirements:



**Figure 3.1 Use Case Diagram**

**Table 3.1 Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Description** | **Type** |
| R-101 | The web application will contain user interface. | Functional |
| R-102 | The web application will require internet to operate. | Configuration |
| R-103 | The web applications will allow navigating menus. | Functional |
| R-104 | The web applications will allow admin to manage (add, update, view, delete) users and attendances | Functional |
| R-105 | The web application will allow users to take attendance and view attendance records | Functional |

#### Non-functional Requirements

Some of the contents of non-functional requirements are shown table below.

**Table 3.2 Non-Functional Requirements**

|  |  |  |
| --- | --- | --- |
| **Req. No.** | **Description** | **Type** |
| NR-101 | The web application shall ensure sensitive information is secure | Security |
| NR-102 | The web application will be user friendly | Usability |
| NR-103 | Unauthorized usersvwill not be able to access the system. | Security |
| NR-104 | The web application shall run well on desktop and mobile devices | Configuration |

### 3.1.2 Feasibility Analysis

Some important feasibility studies are mentioned below:

#### Technical Feasibility

It is technically feasible as I already have hardware and software required for development of a software. Also, I have technical knowledge on how the project is made through programming language like JavaScript and python.

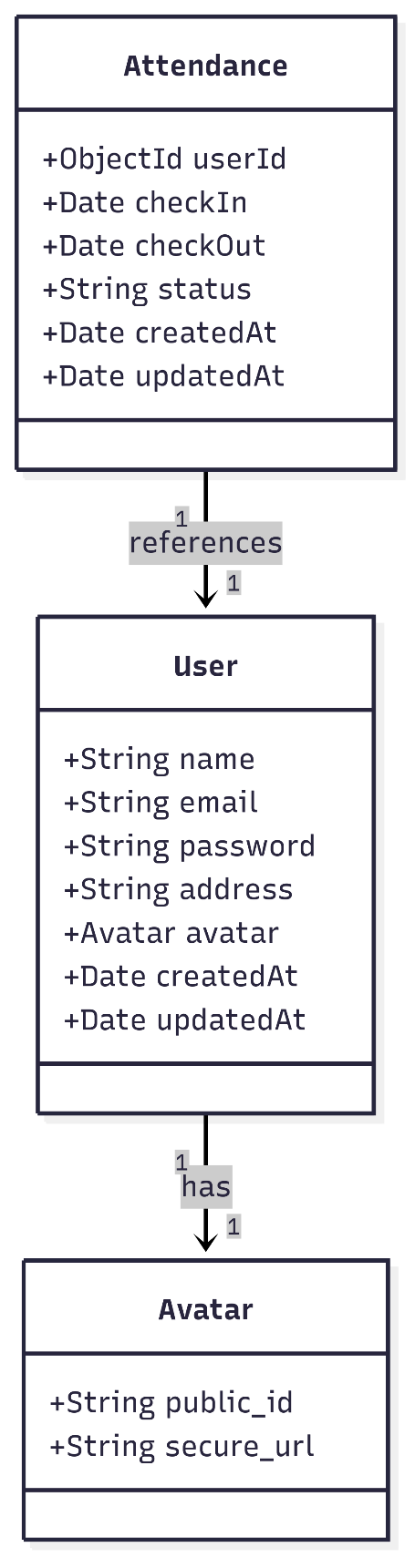
#### Operational Feasibility

It is operationally feasible as I am making this system by removing the threats and weakness of existing non manageable system which is reliable for the users.

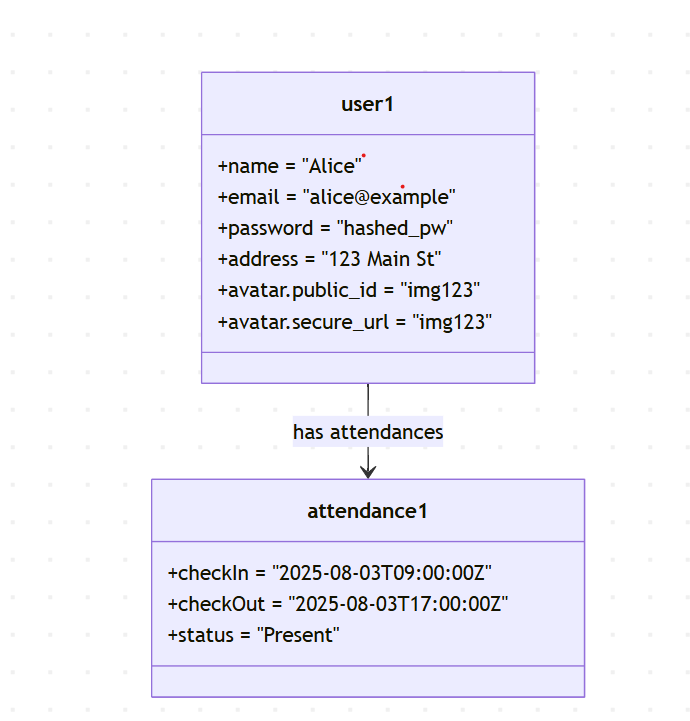
#### Economic Feasibility

The system does not require extra software and hardware. So, there is no recurring cost than just the internet connection.

### 3.1.3 Object Modelling: Object & Class Diagram

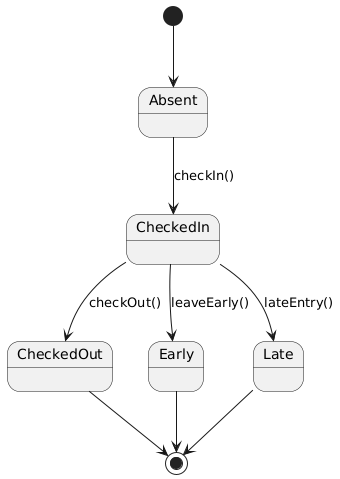


**Figure 3.2 Class Diagram**

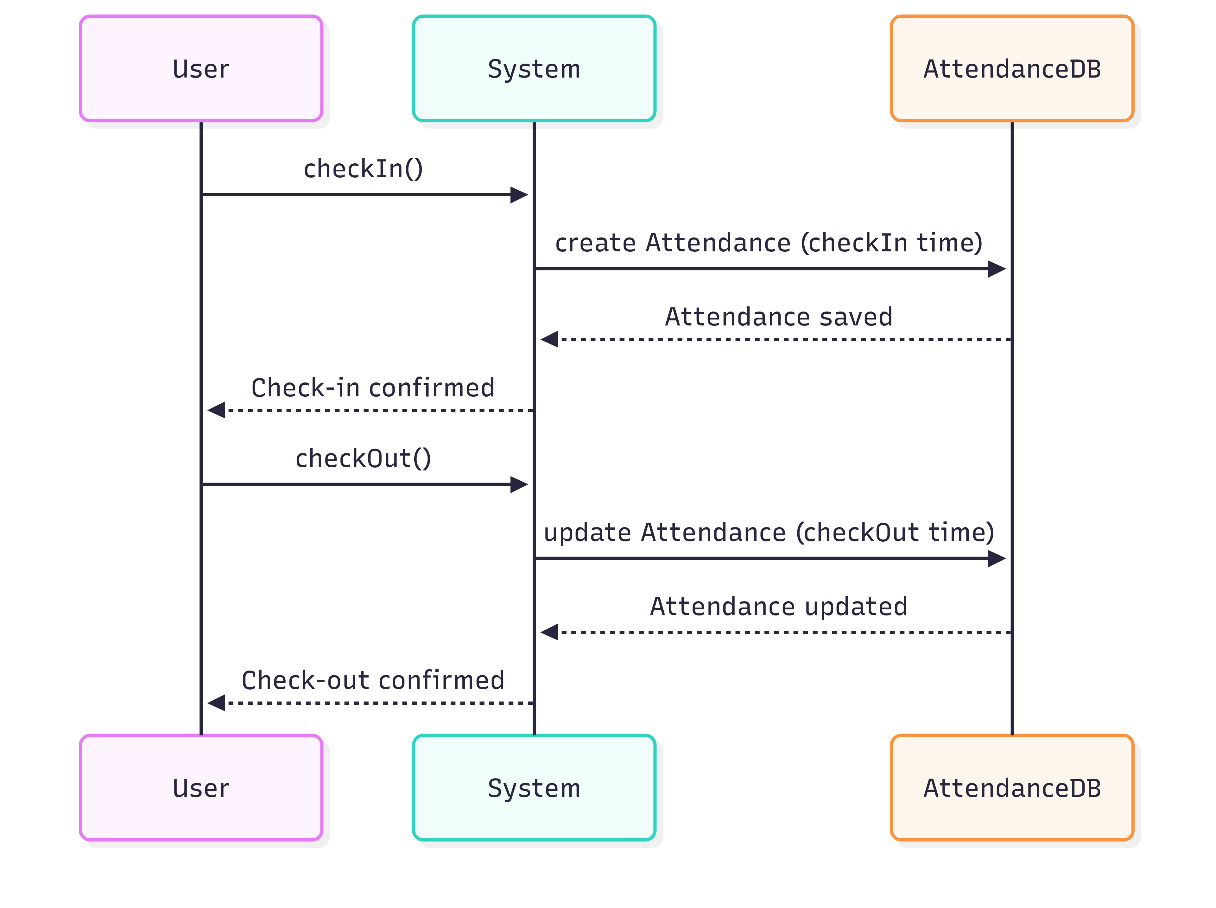


**Figure 3.3 Object Diagram**

### 3.1.4 Dynamic Modelling: State & Sequence Diagram

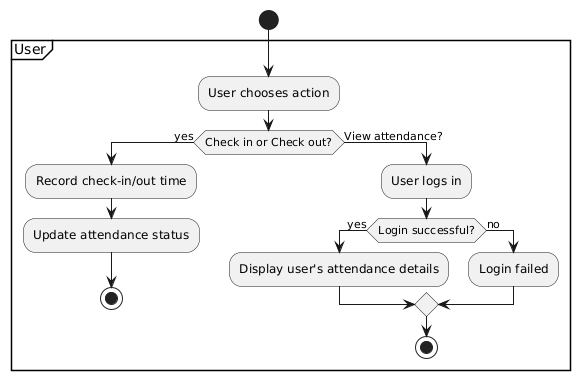


**Figure 3.4 State Diagram**

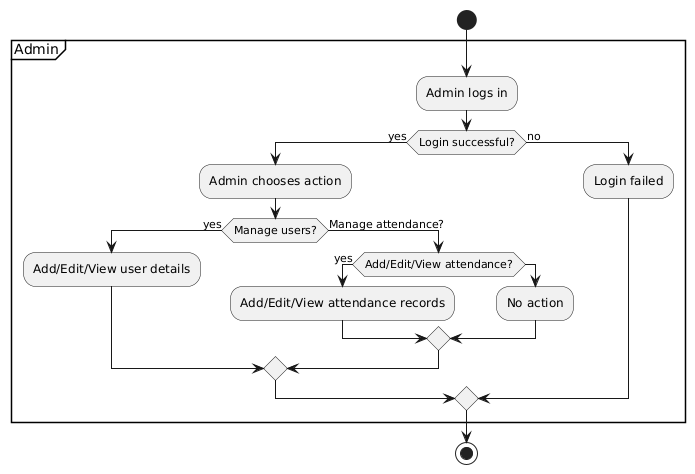


**Figure 3.5 Sequence Diagram**

### 3.1.5 Process Modelling: Activity Diagram



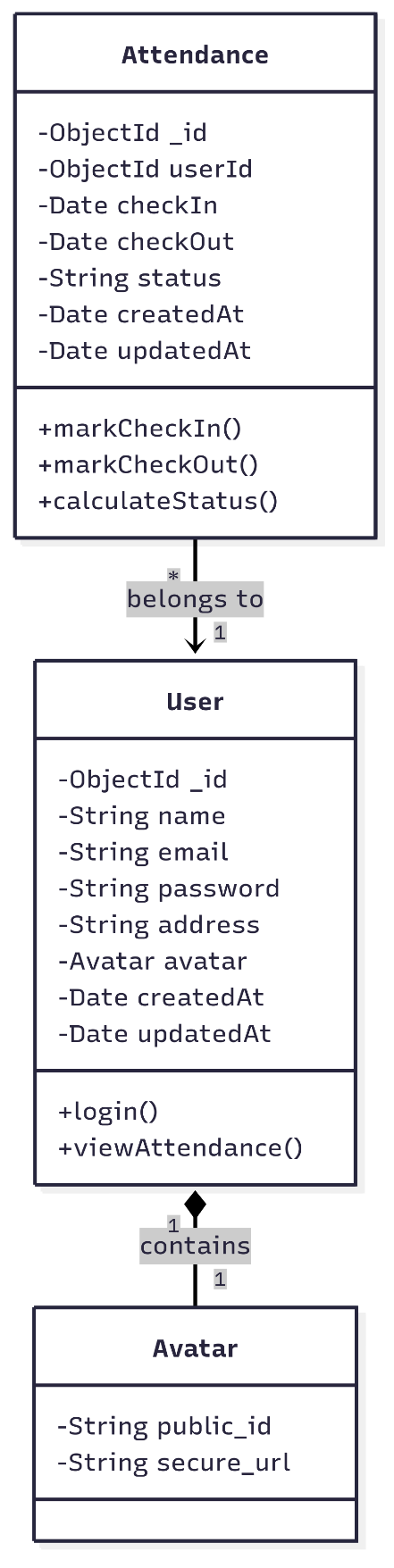
**Figure 3.6 User Activity Diagram**



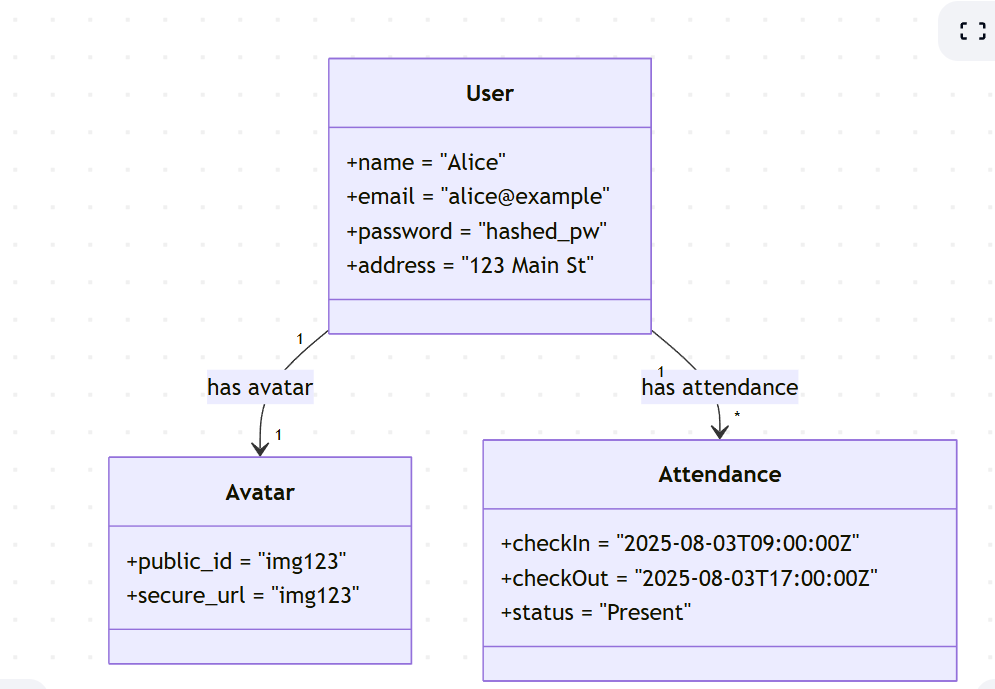
**Figure 3.7 Admin Activity Diagram**

## 3.2 System Design

### 3.2.1 Refinement of Classes and Object

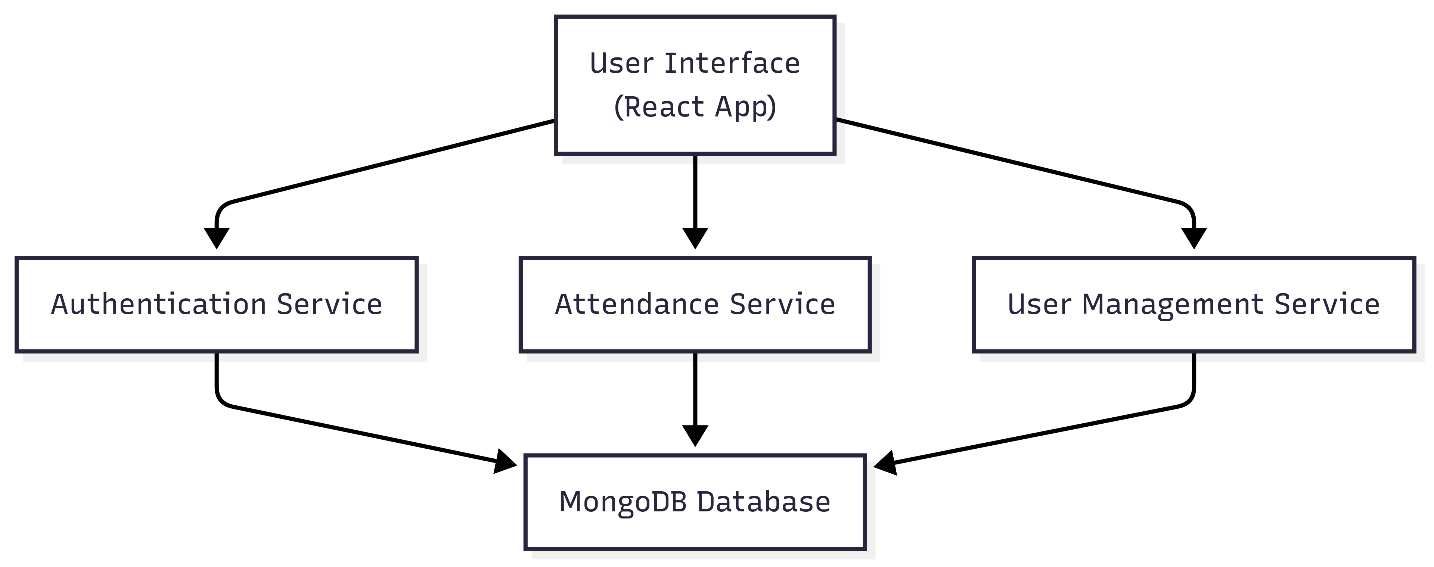


**Figure 3.8 Refinement of Classes**



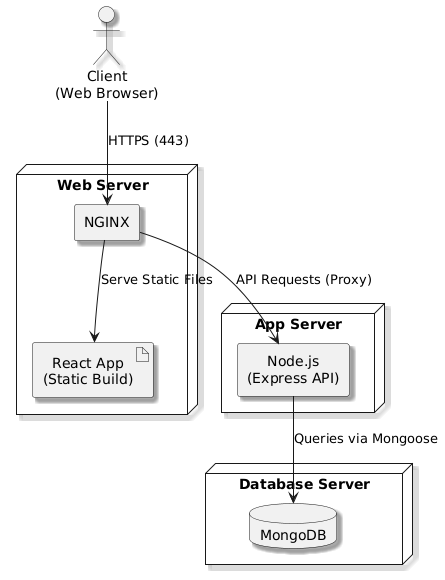
**Figure 3.9 Refinement of Objects**

### 3.2.2 Component Diagram



**Figure 3.10 Component Diagram**

### 3.2.3 Deployment Diagram



**Figure 3.11 Deployment Diagram**

## 3.3 Algorithm Details

In a **face detection attendance system**, once a face is detected and extracted from an image, the next crucial step is **face matching** — determining whether the detected face matches a known (enrolled) face in the database. One of the most common techniques for this is **Euclidean distance**.

**How It Works**

**1. Face Embedding Generation:**

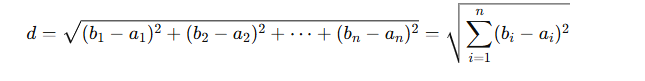
First, each face image is passed through a pre-trained face recognition model (e.g., FaceNet, Dlib, DeepFace) to generate a face embedding — a high-dimensional vector (typically 128 or 512 values) that uniquely represents facial features.

Example:

Face A → [0.23, -0.12, 0.98, ..., 0.45]

Face B → [0.21, -0.10, 0.97, ..., 0.44]

**2. Calculating Euclidean Distance**:  
The **Euclidean distance** between two embedding vectors is computed using the formula:



**3. Matching Decision**:

A **threshold value** is defined (e.g., 0.6). If the distance between the two face embeddings is **less than or equal to the threshold**, it is considered a match (i.e., same person). If it's greater, it's a mismatch.

Example:

* + Distance = 0.45 → Match
  + Distance = 0.82 → No Match

**Why Euclidean Distance?**

* It is simple and computationally efficient.
* Works well when embeddings are **L2-normalized** (unit-length vectors).
* It allows fast comparison across many faces, which is crucial in real-time attendance systems.

**Use in Attendance System**

In the system:

* The user's face is captured at the time of attendance.
* The system generates an embedding for the live face and compares it to stored embeddings in the database.
* If a match is found (distance ≤ threshold), attendance is marked.

This process ensures **accuracy**, **speed**, and **automation** without requiring manual verification.

**Code in project**

    def euclidean\_distance(a, b):  
        # Compute Euclidean distance between two 1D arrays (vectors)  
        # For each pair of elements (x, y) in a and b, compute (x - y) \*\* 2  
        # Sum all squared differences  
        # Take the square root of the sum to get the Euclidean distance  
        return sum((x - y) \*\* 2 for x, y in zip(a, b)) \*\* 0.5

 for i, uploaded\_encoding in enumerate(uploaded\_face\_encodings):  
        try:  
            # Compute Euclidean distances to each reference encoding  
            distances = [euclidean\_distance(uploaded\_encoding, ref\_enc) for ref\_enc in reference\_encodings]  
            # Find the minimum distance and its index  
            min\_distance = min(distances)  
            min\_index = distances.index(min\_distance)  
  
            # print(f"Face distances for face {i+1} in {uploaded\_image\_path}: {distances}")  
  
            if min\_distance < threshold:  
                # Match found  
                filename = os.path.basename(reference\_image\_paths[min\_index])  
                user\_id = filename.split('\_')[1].split('.')[0]  # Extract user ID from filename  
                # print(f"Match found for user: {user\_id}")  
                print(user\_id)  
                sys.exit(0)  
            else:  
                print(f"Face {i+1} is not a match (min distance: {min\_distance:.4f})")  
        except Exception as e:  
            print(f"Error processing uploaded image: {e}")  
            sys.exit(1)

# Chapter 4: Implementation and Testing

## 4.1 Implementation

The aim of this chapter is to document the process of development of the main features. It gives a detailed breakdown of the problems encountered and how they were resolved. It also goes through the test plan and test report of the project to ensure all the functionalities are functioning properly. This chapter is where the project is going to be implemented. However, the software and hardware components used in the implementation of this project will be analyzed below.

### 4.1.1 Tools Used

**Table 4. 1 Tools Used**

|  |  |
| --- | --- |
| **Tool / Technology** | **Purpose / Description** |
| **React / React Native** | JavaScript library for building reusable, dynamic UI components for web and mobile apps. |
| **Node.js (with Express)** | Server-side runtime for handling APIs, routing, authentication, and real-time features. |
| **MongoDB** | NoSQL database for flexible, scalable storage of user data, messages, and attendance. |
| **Visual Studio Code** | Lightweight code editor for writing and debugging in multiple programming languages. |
| **Git / GitHub** | Version control and collaboration tools for code tracking, branching, and team work. |
| **Microsoft Word** | Used for writing structured documentation like reports and system specs. |
| **Draw.io** | Diagramming tool for system architecture, use case diagrams, and UI mockups. |
| **Mermaid.js** | Markdown-based tool for creating flowcharts and diagrams in documentation. |

### 4.1.2 Implementation Details of Modules

**Backend Modules**

**1. Attendance Model (**[**Attendance.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Purpose: Defines the MongoDB schema for attendance records.

Fields:

[userId](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Reference to the User, required.

[checkIn](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Date/time of check-in.

[checkOut](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Date/time of check-out.

[status](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Attendance status (Present, Absent, Late, Early).

[timestamps](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html): Automatically adds [createdAt](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and updatedAt.

**2. User Model (**[**User.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Purpose: Defines the MongoDB schema for users.

Fields: Name, email, password, address, avatar (with [public\_id](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and [secure\_url](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)), and timestamps.

**3. Attendance Controller (**[**attendanceController.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Functions:

getAttendanceByUser:

Fetches all attendance records for a user, sorts by date, and returns formatted data for the frontend.

recognizeFace:

Handles attendance marking (check-in/check-out) via face recognition.

Receives an image and action (checkin/checkout).

Spawns a Python process ([face\_match.py](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)) to match the face.

If a user is recognized, updates or creates the attendance record for today.

Handles error cases (no face, already checked in/out, etc.).

**4. Auth Controller (**[**authController.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Function:

login:

Authenticates a user by email and password.

Returns a JWT token and user info on success.

**5. Routes (**[**attendaceRoutes.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**,**[**authRoutes.js**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Attendance Routes:

POST /api/v1/attendance: Uploads an image and marks attendance.

GET /api/v1/attendance/:userId: Fetches attendance records for a user.

Auth Routes:

POST /api/v1/auth/login: User login.

**6. Python Face Matching (**[**face\_match.py**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html))

Procedures:

download\_image(url, filename): Downloads an image from a URL.

Main Flow:

Parses user data and downloads reference images.

Encodes faces from reference and uploaded images.

Compares encodings using Euclidean distance.

Prints recognized user ID or error messages.

**Frontend Modules**

**1. Attendance Page (**[**Attendance.tsx**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Component:

Uses webcam to capture an image.

Lets user select "Check In" or "Check Out".

Sends image and action to backend.

Displays result and toast notifications.

**2. Attendance Record Page (**[**AttendanceRecord.tsx**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Component:

Fetches and displays the user's attendance records in a styled table.

Shows date, check-in/out times, and statuses.

**3. Login Page (**[**Login.tsx**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Component:

Handles user login.

Stores JWT and user info in local storage on success.

**4. API Helpers (**[**attendanceApi.ts**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**,**[**authApi.ts**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Functions:

fetchAttendanceByUser: Fetches attendance records for a user.

addAttendance: (not used in current UI) Adds a new attendance record.

login: Authenticates a user and returns token/user info.

**5. Axios Instance (**[**axiosInstance.ts**](vscode-file://vscode-app/c:/Users/tanji/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)**

Purpose:

Configures Axios with base URL and attaches JWT token to requests.

**6. Layout and UI Components**

HomeLayout: Wraps pages with navbar and footer.

Navbar: Shows navigation and user info.

Footer: Simple copyright.

## 4.2 Testing

### 4.2.1 Test Cases for Unit Testing

**Table 4.2 Test Cases for User Login**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case** | **Test Input** | **Expected Result** | **Outcome** |
| 1 | Valid user login | Email:ashish@gmail.com, Password: Ashish@123 | User logged in, token created | Pass |
| 2 | Login with wrong password | Email: [ram@example.com](mailto:ram@example.com), Password: Ram@123 | Error message: "Invalid credentials" | Pass |
| 3 | User logout | Click logout button | Token expired, clear local storage, redirected to landing page | Pass |

**Table 4.3 Test Cases for Face Matching**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case** | **Test Input** | **Expected Result** | **Outcome** |
| 1 | Match known face (registered user) | Upload a clear face image of registered user | Attendance marked and confirmation message shown | Pass |
| 2 | Match unknown face (unregistered user) | Upload a face image not in database | Error: "Face not recognized" | Pass |
| 3 | Match partially covered face | Upload face image with mask or partial obstruction | Error or warning message depending on model confidence | Pass |
| 4 | Match with multiple faces in image | Upload an image with two faces | Error: "Multiple faces detected. Please try again." | Pass |
| 5 | Match using low-light image | Upload dim or poorly lit face image | Error: "Face not clearly visible" | Pass |
| 6 | Match with server/API error | Simulate backend failure during match | Error: "Face recognition service unavailable" | Pass |
| 7 | Attempt match without uploading image | Click "Submit" without selecting file | Error: "Please upload a face image" | Pass |
| 8 | Match face and store attendance (logged in) | Upload valid image as authenticated user | Attendance saved to database | Pass |
| 9 | Match face and store attendance (public user) | Upload valid image as guest (if allowed) | Attendance stored or redirected to login | Pass |
| 10 | Upload invalid file format | Upload .txt or non-image file | Error: "Invalid file format. Please upload an image." | Pass |

**Table 4.4 Test Cases for Attendance Record**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case** | **Test Input** | **Expected Result** | **Outcome** |
| 1 | View attendance records (admin) | Login as admin and navigate to 'View Records' | Table displays all user attendance data | Pass |
| 2 | View attendance records (user) | Login as registered user and navigate to 'My Attendance' | Table displays only the user's own records | Pass |
| 3 | Access attendance without login | Navigate to 'My Attendance' without logging in | Redirect to login or access denied | Pass |
| 4 | Filter records by date | Select a date range and apply filter | Records matching the date range are displayed | Pass |
| 5 | Filter records by user (admin) | Admin selects a specific user from dropdown | Only selected userâ€™s records are shown | Pass |
| 6 | Sort records by date | Click 'Date' column header to sort | Records sorted in ascending or descending order | Pass |
| 7 | Export attendance to CSV (admin) | Click 'Export CSV' button | CSV file of all attendance records downloaded | Pass |
| 8 | Search attendance by name | Enter partial or full name in search bar | Matching records displayed dynamically | Pass |
| 9 | Attempt to delete record (unauthorized user) | User clicks delete on their own record | Error: 'Unauthorized action' | Pass |
| 10 | Backend failure while fetching records | Simulate server/API error | Error: 'Unable to fetch records' | Pass |

### 4.2.2 Test Cases for System Testing

**Table 4.5 Test Cases for Integration & API Testing**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SN** | **Test Case** | **Test Input** | **Expected Result** | **Outcome** |
| 1 | Database connection failure | Simulate DB unavailability | Appropriate error messages, graceful degradation | Pass |
| 2 | Euclidena Distance algorithm accuracy | Change facial structures | Matches face | Pass |

# Chapter 5: Conclusion and Future Recommendations

## 5.1 Conclusion

The development of the face detection attendance system successfully demonstrated how modern technologies can automate and enhance traditional attendance methods. By utilizing facial recognition for identity verification, the system minimized the need for manual intervention, reduced the chances of proxy attendance, and improved the overall accuracy and reliability of attendance tracking. The integration of technologies like Python for face recognition, React Native for the user interface, Node.js for backend services, and MongoDB for data storage allowed the creation of a scalable and responsive system. The project effectively met its objectives by providing a solution that is not only efficient but also practical for use in educational institutions and organizations.

## 5.2 Lesson Learnt/Outcome

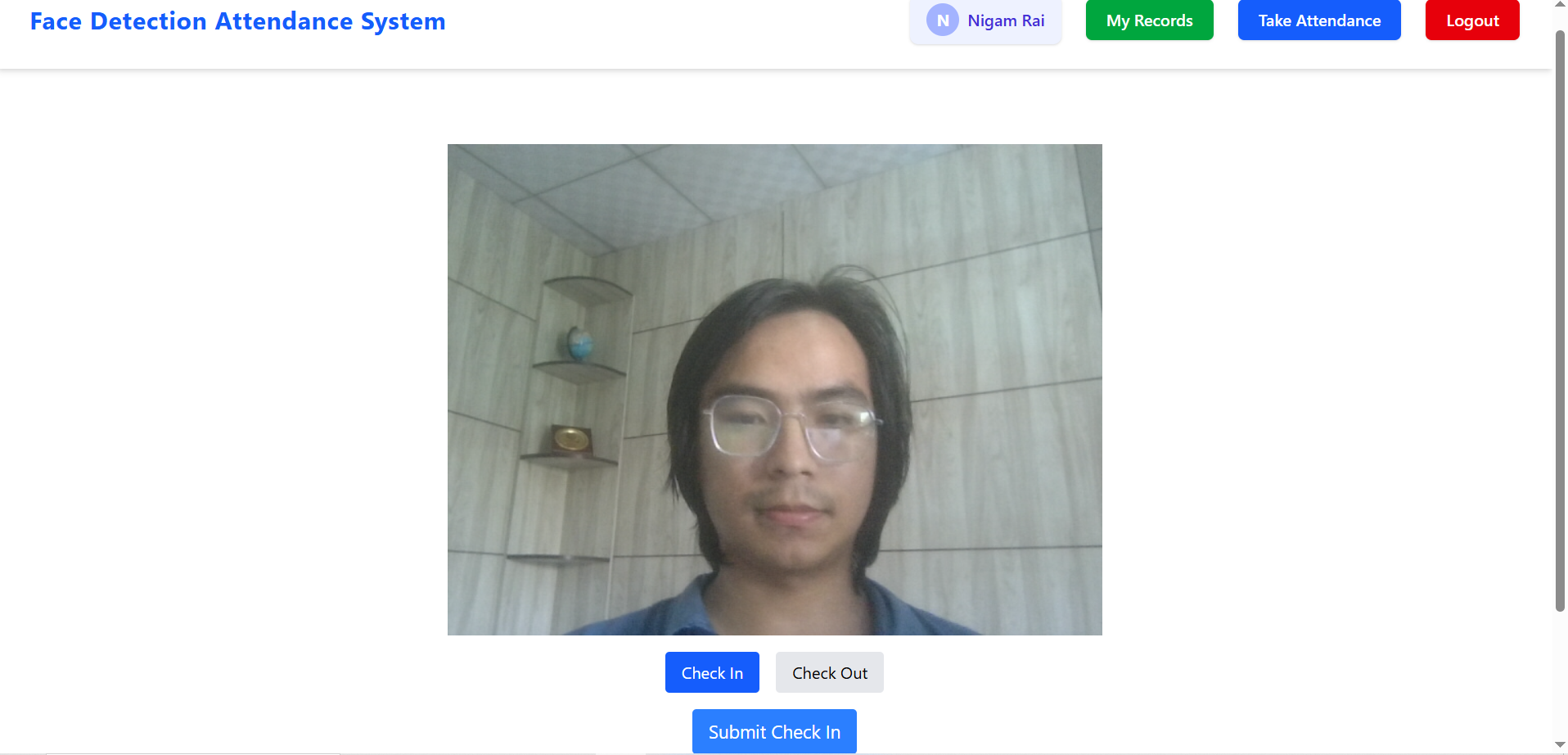
While developing the application, I learned a lot about creating a modular system, which made the code easier to manage and test. I also realized how important it is to organize data efficiently, especially when using mongodb to handle databases. Connecting the front-end (React) and back-end (Node JS) using REST APIs helped me understand how to sync data smoothly between them. I also gained experience in handling errors and validating user input, which improved the system's reliability.

## 5.3 Future Recommendations

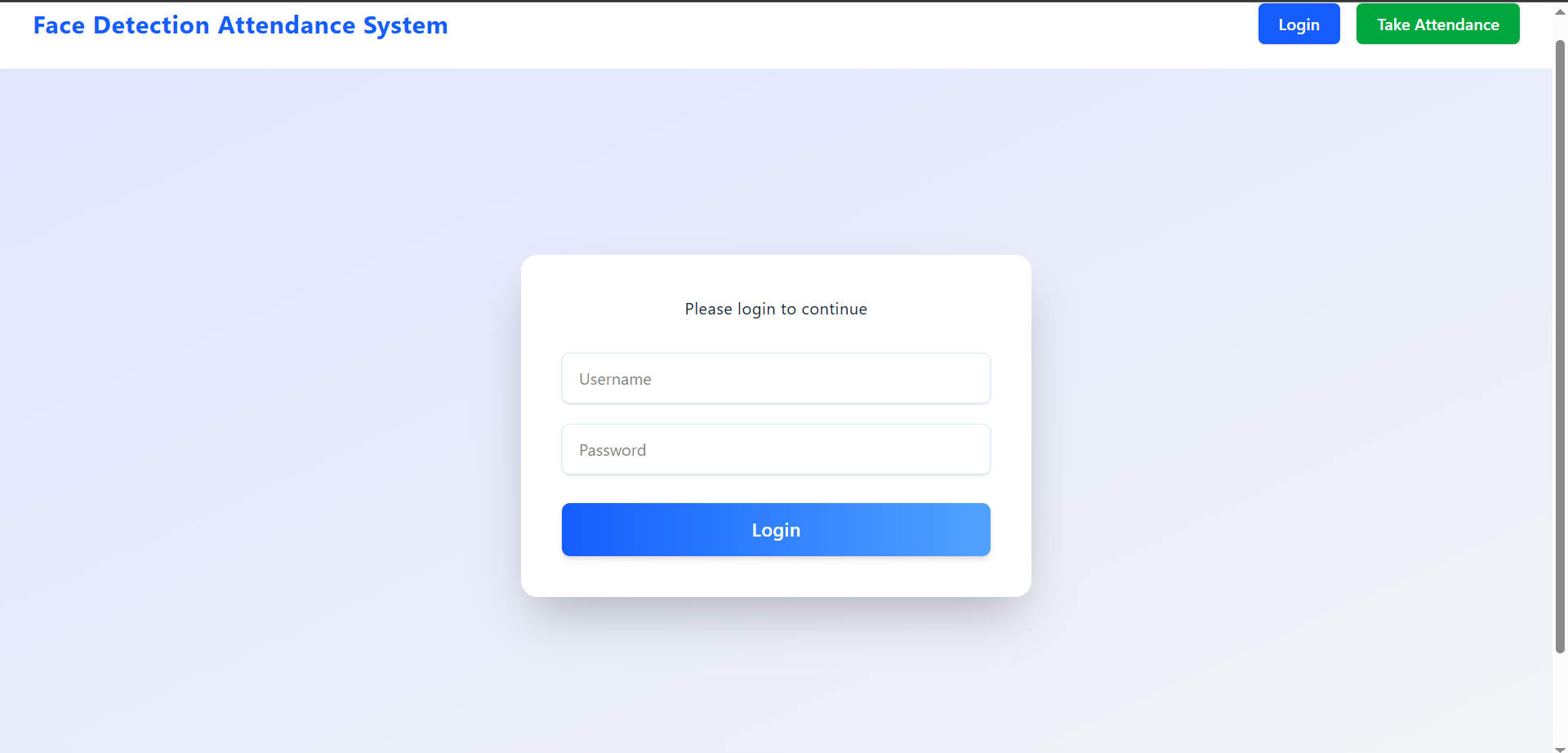
To further improve the system, several enhancements can be considered. The addition of **liveness detection** could help prevent spoofing using photos or videos. Integrating **push notifications** would alert users of successful or missed attendance in real time. Developing a **web-based admin panel** would make it easier to monitor records and manage users. Furthermore, incorporating **cloud storage** can ensure data security and scalability. Expanding compatibility to **iOS platforms** and integrating with **institutional management systems** would also improve usability and adoption in diverse environments. Continuous training of the recognition model with more diverse datasets will enhance accuracy across a wider range of users.

# Appendices

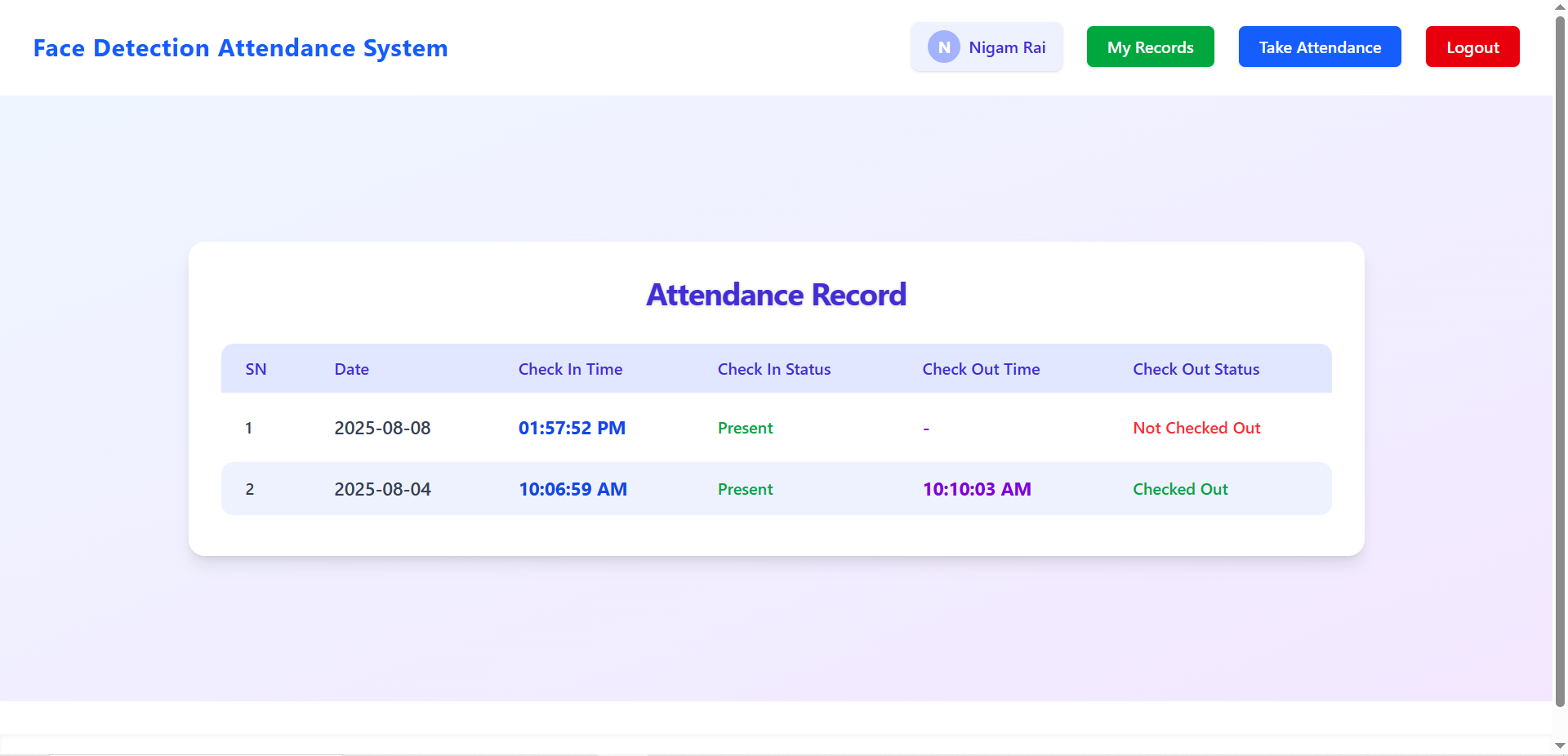
**Screen Shots**

****

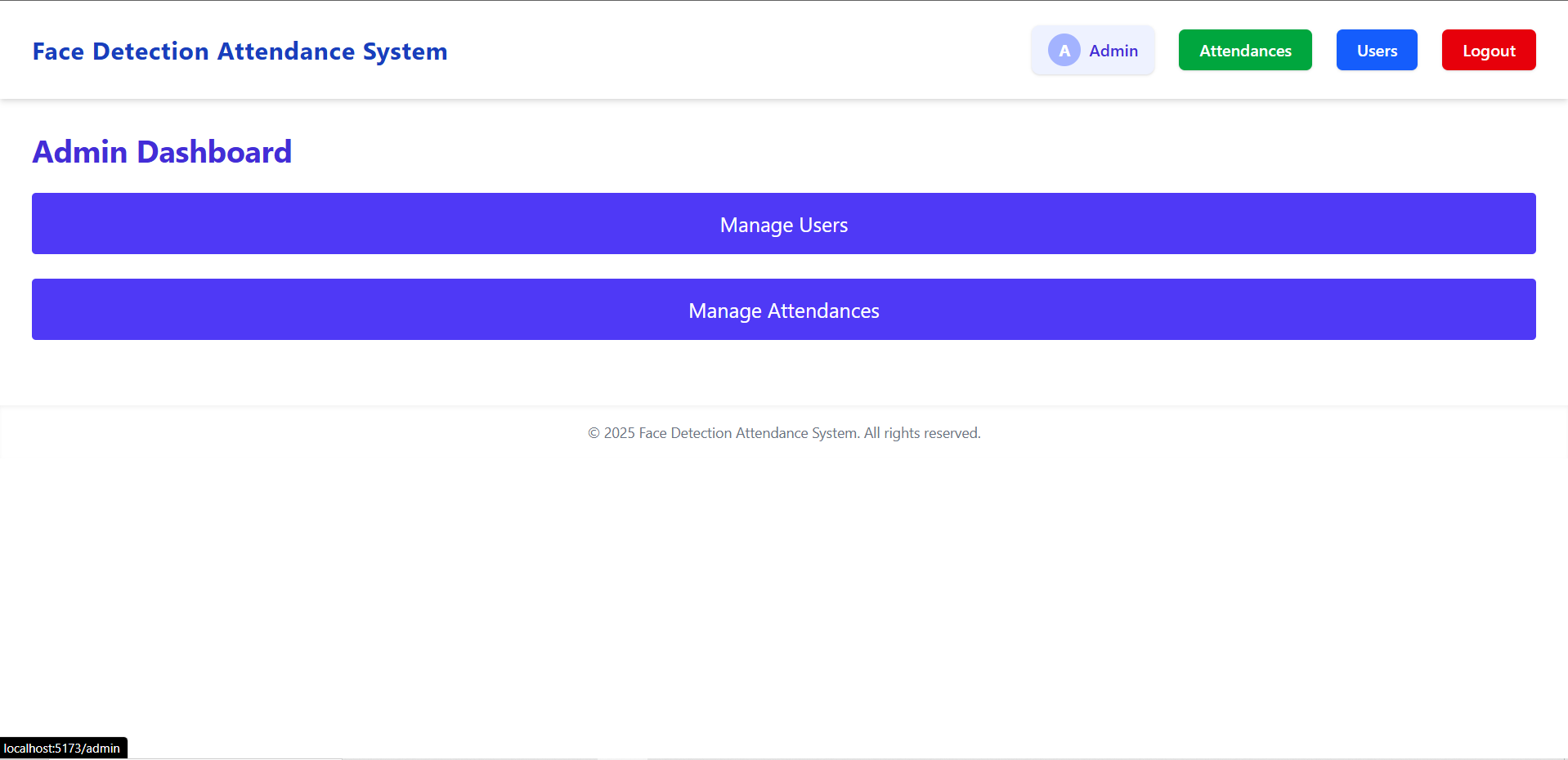
**Take Attendance Page**

****

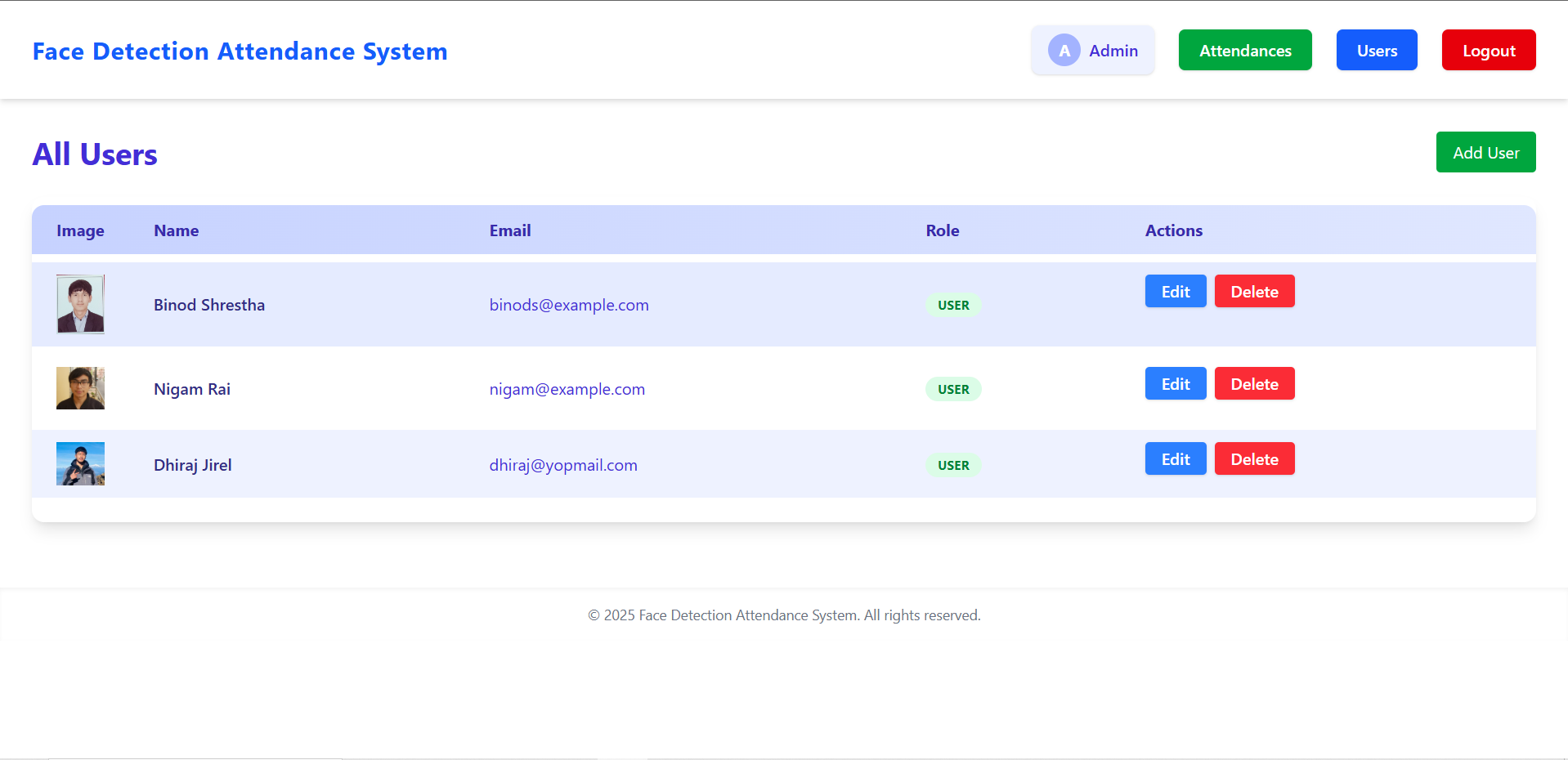
**Login Page**

****

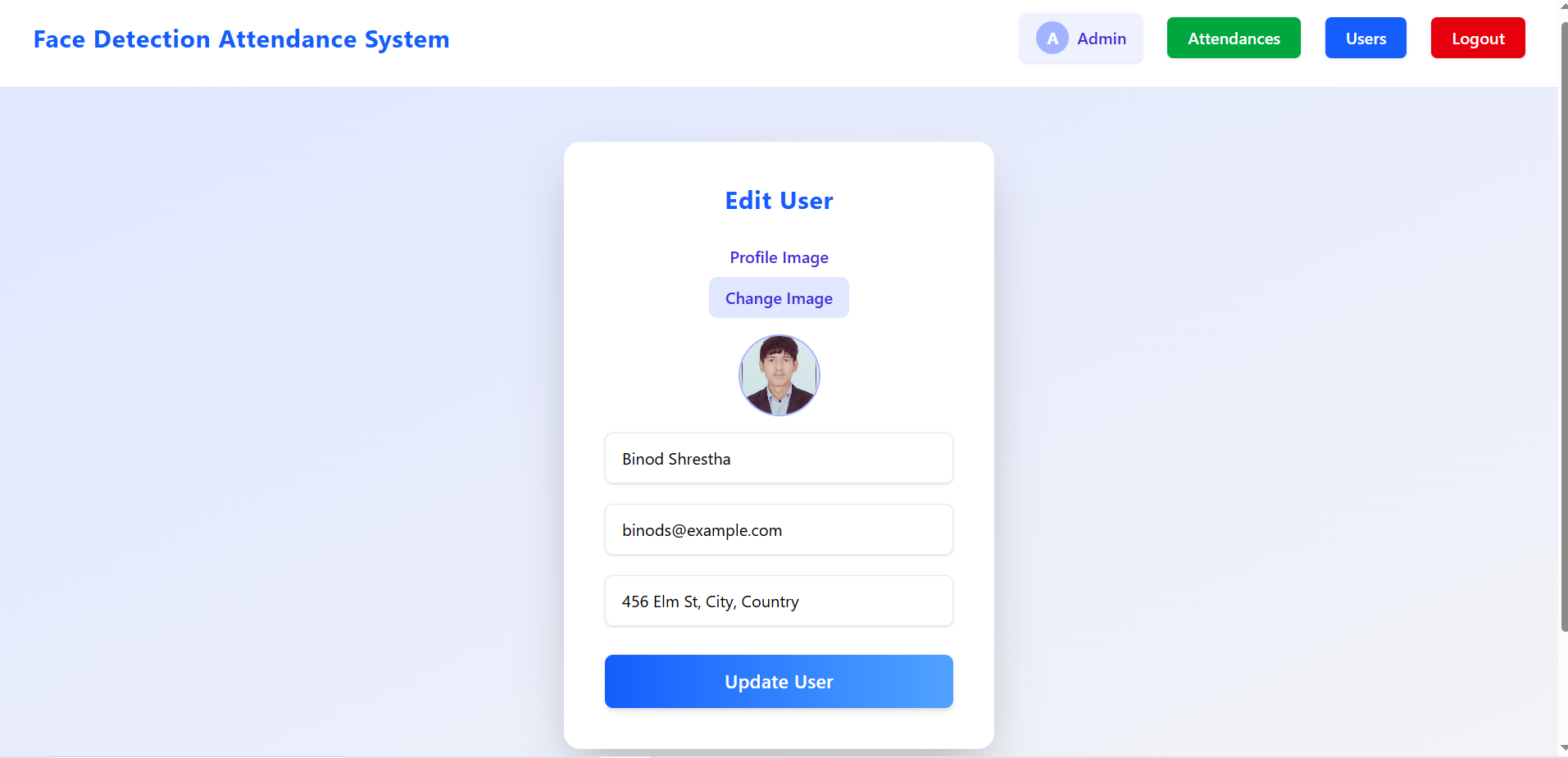
**User Attendance Page**

****

**Admin Dashboard Page**

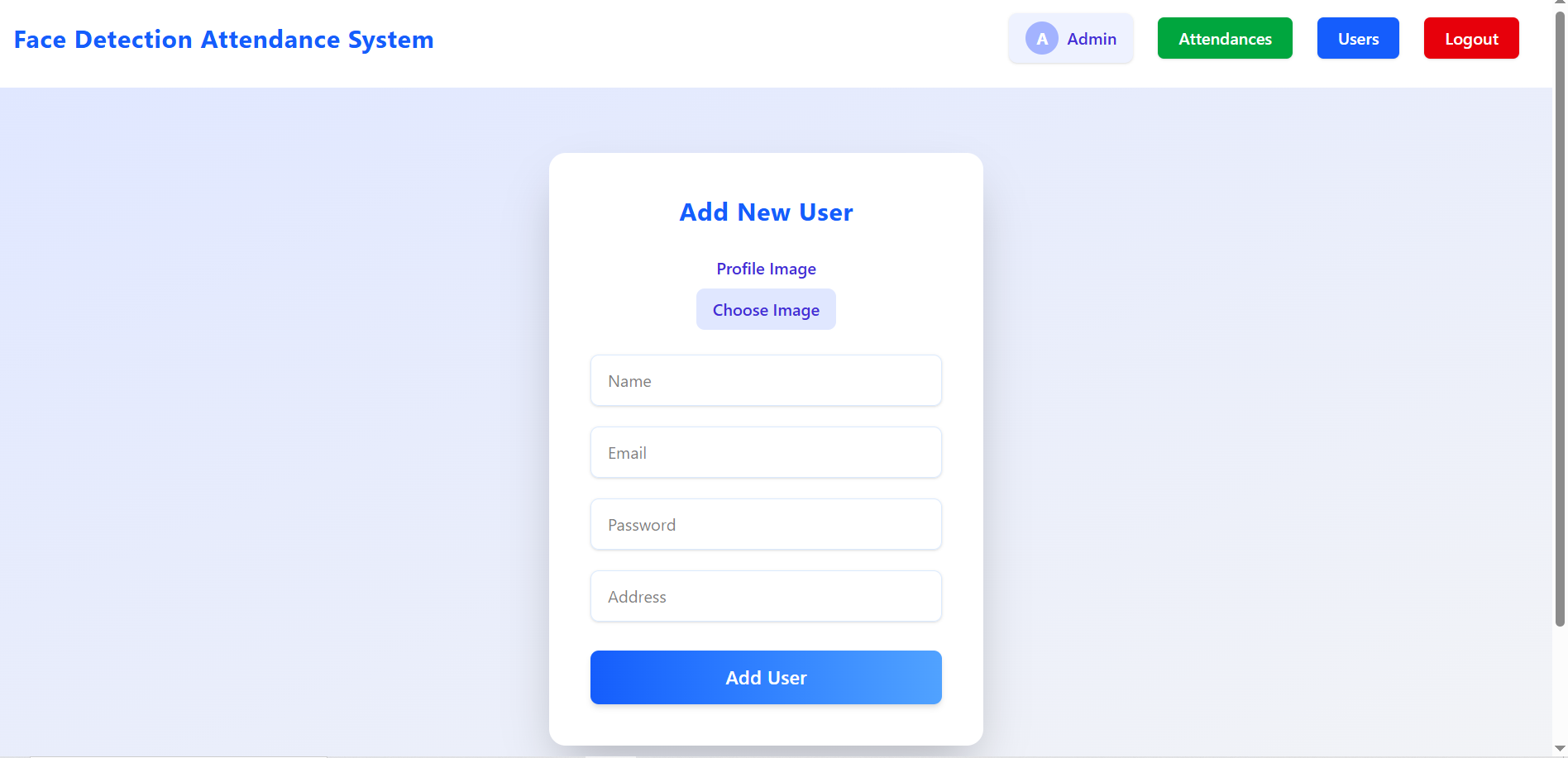
****

**List of Admin Users Page**

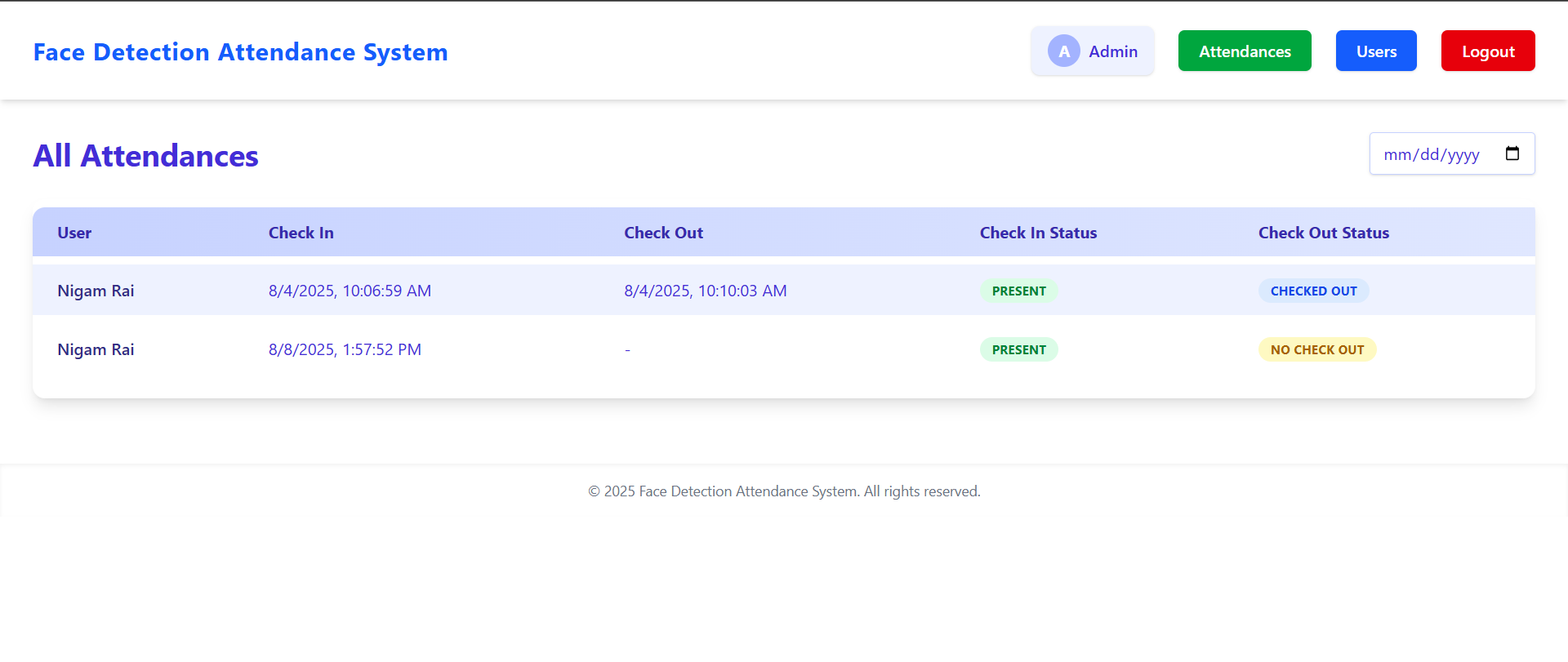
****

**Admin User Edit Page**

**Source Code**

****

**Add User Page**

****

**List of Attendances Page**

# References

[1] A. K. Jain, A. Ross, and S. Prabhakar, “An introduction to biometric recognition,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, Jan. 2004.

[2] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[3] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA, USA, 2005, pp. 886–893.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 815–823.

[5] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[6] S. Sharma, P. Choudhary, and R. Singh, “Face recognition based attendance system using machine learning,” *International Journal of Computer Applications*, vol. 179, no. 26, pp. 20–24, 2018.

[7] M. Al-Shabi, M. Y. Mashor, and A. M. A. Rahman, “Mobile-based student attendance management system using face recognition,” *Journal of Engineering and Applied Sciences*, vol. 14, no. 22, pp. 8602–8608, 2019.

[8] S. Z. Li and A. K. Jain (Eds.), *Handbook of Face Recognition*, 2nd ed., Springer, 2011.