

Capstone Project Phase A

**AttendEase**

**Attendance Tracking App**

Project Number

**24-2-D-4**

Supervisor: Alexander Keselman

Amr Jarrar (Amr.Jarrar@e.braude.ac.il)

Deborah Dubinker (Deborah.Dubinker@e.braude.ac.il)

# Table of Contents

[Abstract 3](#_Toc176366295)

[Chapter 1 –](#_Toc176366296) [Introduction 4](#_Toc176366297)

[1.1 Problem Statement 4](#_Toc176366298)

[1.2 Existing Solutions 4](#_Toc176366299)

[1.3 Proposed Solution 4](#_Toc176366300)

[1.4 Stakeholders and Benefits 4](#_Toc176366301)

[Chapter 2 –](#_Toc176366302) [Background and Related Work 6](#_Toc176366303)

[2.1 Existing Tools 6](#_Toc176366304)

[2.2 Literature Review 6](#_Toc176366305)

[Chapter 3 –](#_Toc176366306) [Expected Achievements 7](#_Toc176366307)

[3.1 Objectives 7](#_Toc176366308)

[3.2 Expected Outcomes 7](#_Toc176366309)

[3.3 Algorithm and System Functionality 7](#_Toc176366310)

[3.4 Success Criteria 8](#_Toc176366311)

[3.5 Challenges 8](#_Toc176366312)

[Chapter 4 –](#_Toc176366313) [Engineering Process 9](#_Toc176366314)

[**4.1 Process 9**](#_Toc176366315)

[4.1.1 Development Approach 9](#_Toc176366316)

[4.1.2 Development Stages 9](#_Toc176366317)

[4.1.3 Constraints 9](#_Toc176366318)

[**4.2 Product 10**](#_Toc176366319)

[4.2.1 Algorithms 10](#_Toc176366320)

[4.2.2 Models 10](#_Toc176366321)

[4.2.3 Implementation 10](#_Toc176366322)

[4.2.4 Data Structure 11](#_Toc176366323)

[4.2.5 User Interface 11](#_Toc176366324)

[4.2.6 Use Case Diagram 11](#_Toc176366325)

[4.2.7 Class Diagram 12](#_Toc176366326)

[4.2.8 Activity Diagram 13](#_Toc176366327)

[4.2.8 Functional Requirements 14](#_Toc176366328)

[4.2.9 Non-Functional Requirements 15](#_Toc176366329)

[Chapter 5 –](#_Toc176366330) [Testing Plan 16](#_Toc176366331)

[5.1 Testing Strategy 16](#_Toc176366332)

[5.2 Unit Testing 16](#_Toc176366333)

[5.3 Integration Testing 18](#_Toc176366334)

[5.4 System Testing 18](#_Toc176366335)

[5.5 Manual Testing 19](#_Toc176366336)

# Abstract

This document presents the Capstone Project Phase A for "AttendEase," an advanced attendance tracking app designed to revolutionize attendance management in educational institutions. The app leverages facial recognition technology integrated with OpenAI capabilities to provide an efficient, accurate, and user-friendly solution, utilizing client-server architecture to facilitate real-time data handling and user interaction. Traditional attendance methods are often time-consuming, prone to errors, and easily manipulated. AttendEase addresses these issues by automating the process, reducing manual effort, and enhancing data reliability. The system includes intuitive interfaces for teachers, students, and administrators, ensuring seamless interaction and effective attendance monitoring. The project aims to deliver a scalable, secure, and efficient attendance tracking system that benefits students, teachers, and administrators.

# Chapter 1

# Introduction

## 1.1 Problem Statement

In many educational institutions, tracking student attendance is a critical task that ensures students are attending their classes and helps in maintaining accurate records for academic and administrative purposes. Traditional methods of attendance, such as calling out names or passing around sign-in sheets, are time-consuming, prone to errors, and can easily be manipulated. This creates a need for a more efficient, accurate, and reliable attendance tracking system.

## 1.2 Existing Solutions

Several solutions exist for automating attendance tracking, including:

1. **Biometric Systems:** These systems use fingerprint or iris recognition. While accurate, they require physical contact or close proximity, which may not be ideal in all situations, especially considering hygiene concerns.
2. **RFID Systems:** Students carry RFID cards that are scanned upon entering the classroom. However, students can forget their cards or give them to others to falsely mark attendance.
3. **Manual Entry Apps:** Teachers use apps to manually enter attendance. This reduces paper usage but still requires manual input and can be time-consuming.

## 1.3 Proposed Solution

We propose developing an attendance tracking app that uses facial recognition technology integrated with OpenAI's capabilities. The app will leverage a database containing photos and IDs of students. By taking a picture of the class, the app will compare the faces in the picture with the database to accurately and efficiently determine who is present.

## 1.4 Stakeholders and Benefits

1. **Students:** Simplifies the attendance process and ensures their attendance is recorded accurately.
2. **Teachers:** Saves time and reduces the administrative burden of tracking attendance manually.
3. **Administrators:** Provides reliable attendance data for academic records and regulatory compliance.

# Chapter 2

# Background and Related Work

Attendance tracking applications have gained importance in educational institutions, offering enhanced efficiency and accuracy in monitoring student attendance. In this literature survey, we explore existing tools and methods relevant to the development and functionality of our proposed attendance tracking app, which integrates facial recognition technology and OpenAI capabilities.

## 2.1 Existing Tools

1. **Iris Recognition Systems** – **Iris ID**: A leading provider of iris recognition technology, used for secure access in various institutions. More information can be found [here](https://www.irisid.com/).
2. **Manual Entry Apps** – **TeacherKit**: An app that helps teachers track attendance, grades, and behavior. More information can be found [here](https://www.teacherkit.net/).

## 2.2 Literature Review

1. **Review of Biometric Systems for Attendance**:

**Ross, A., Nandakumar, K., & Jain, A. K. (2006). "Handbook of Multibiometrics."** This book chapter reviews various biometric systems, including those used for attendance tracking, and discusses their advantages and limitations.

1. **Biometric Systems**:

**Jain, A. K., & Ross, A. (2008). "Introduction to Biometrics."** This comprehensive overview discusses various biometric systems, including fingerprint and iris recognition, and their applications in attendance tracking.

1. **Facial Recognition Technology**:

**Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). "Face Recognition: A Literature Survey."** This survey covers the development and application of facial recognition technologies, including their use in educational settings for automated attendance.

The proposed attendance tracking app aims to leverage facial recognition technology and integrate OpenAI's advanced capabilities to address the limitations of existing attendance tracking methods. This comprehensive approach will enhance the efficiency, accuracy, and user-friendliness of the attendance tracking process, benefiting all stakeholders involved. By automating attendance tracking and providing AI-driven insights, the system will help educational institutions manage attendance more effectively and improve overall student engagement.

# Chapter 3

# Expected Achievements

## 3.1 Objectives

In this project, we aim to develop an advanced attendance tracking system that leverages facial recognition technology and integrates OpenAI capabilities to enhance the accuracy and efficiency of the attendance process in educational institutions.

## 3.2 Expected Outcomes

1. **Automated Attendance System:** A functional application that can automatically capture and record student attendance by recognizing faces from a classroom photograph.
2. **User-Friendly Interface:** A user-friendly interface for teachers to easily take pictures, review attendance, and generate reports.
3. **Database Integration:** A robust database system that securely stores student photos and IDs.
4. **Real-Time Processing:** Real-time processing capabilities to quickly match captured faces with stored records.
5. **Error Reduction:** Significant reduction in attendance tracking errors compared to traditional methods.
6. **Scalability:** The system should be scalable to handle large numbers of students and multiple classes.

## 3.3 Algorithm and System Functionality

1. **Face Detection and Recognition Algorithm:** Utilizes pre-trained models for face detection and recognition, with potential fine-tuning for increased accuracy in the specific educational environment.
2. **Integration with OpenAI:** Employs OpenAI’s natural language processing capabilities to provide intelligent insights and automate administrative tasks related to attendance tracking.
3. **Data Security Measures:** Ensures that all student data is encrypted and securely stored to maintain privacy and comply with data protection regulations.

## 3.4 Success Criteria

1. **Accuracy:** The system should achieve an accuracy rate of at least 95% in recognizing student faces.
2. **Efficiency:** The time taken to process and record attendance should be less than 5 minutes per class.
3. **User Satisfaction:** Positive feedback from teachers and administrators regarding ease of use and effectiveness.
4. **Scalability:** The system should seamlessly handle up to 500 students per institution without performance degradation.\

## 3.5 Challenges

1. **Facial Recognition Accuracy:** Achieving consistent accuracy across diverse student populations and varying classroom conditions (e.g., lighting, camera angles, occlusions) could be challenging. Fine-tuning pre-trained models to work effectively in different environments is necessary but time-consuming.
2. **Data Privacy and Compliance:** Handling student biometric data (facial images) involves strict data protection and privacy considerations. Ensuring compliance with regulations requires robust encryption and secure data management practices.
3. **Scalability and Performance:** The system must remain efficient while processing large volumes of data. Handling real-time face matching for hundreds of students across multiple classes could stress the database and processing resources.

# Chapter 4

# Engineering Process

## 4.1 Process

### 4.1.1 Development Approach

1. **Agile Methodology:** Adopting an agile development process to allow for iterative improvements and flexibility in response to feedback.
2. **Incremental Development:** Breaking down the project into manageable modules and developing each module incrementally.

### 4.1.2 Development Stages

1. **Requirements Gathering:** Collecting detailed requirements from stakeholders to understand the specific needs and constraints.
2. **System Design:** Creating detailed system architecture and design documents.
3. **Prototype Development:** Developing an initial prototype to validate the core functionality.
4. **Iterative Improvement:** Continuously improving the prototype based on user feedback.
5. **Final Implementation:** Completing the development of all features and integrating them into a cohesive system.

### 4.1.3 Constraints

1. **Resource Limitations:** Managing development within the constraints of time and budget.
2. **Technical Limitations:** Addressing any limitations posed by the technology stack or existing infrastructure.

## 4.2 Product

### 4.2.1 Algorithms

1. **Face Detection Algorithm:** Utilizes convolutional neural networks (CNN) for detecting faces in images. The algorithm processes the images and identifies faces within those images.
2. **Face Recognition Algorithm:** Employs a deep learning model trained on a large dataset of faces to match detected faces with stored records in the student database. This algorithm ensures accurate identification of students by comparing the detected faces with the pre-stored facial data.

### 4.2.2 Models

1. **Pre-Trained Models:** Leveraging pre-trained models such as VGG-Face or FaceNet for face recognition. These models are initially used to bootstrap the facial recognition process, providing a solid foundation of accuracy and reliability.
2. **Custom Models:** Fine-tuning pre-trained models with specific datasets to improve accuracy in the educational context. This involves adjusting the models to better handle the unique lighting conditions, camera angles, and diverse student appearances encountered in the classroom environment.

### 4.2.3 Implementation

1. **Frontend Implementation**

The frontend of the system will be developed using **React**, a popular JavaScript library known for building dynamic user interfaces. This choice ensures that the user interfaces for teachers, administrators, and students are responsive, interactive, and easy to use.

1. **Backend Implementation**

The backend will be implemented using **Next.js**, a powerful React framework that supports server-side rendering and static site generation. This will allow for efficient data handling and faster load times, enhancing the overall user experience.

1. **Database Implementation**

The database will be hosted on **Supabase**, a scalable open-source alternative to Firebase. Supabase will manage the student database, class records, and attendance logs, providing real-time updates and secure data storage.

### 4.2.4 Data Structure

1. **Student Database:** A secure database storing student IDs, photos, and attendance records.
2. **Class Records:** A relational database structure to manage class schedules and attendance logs.

### 4.2.5 User Interface

1. **Teacher Interface:** An intuitive interface for teachers to capture class photos, review attendance, and generate reports.
2. **Administrator Interface:** A comprehensive dashboard for administrators to manage student data, monitor attendance patterns, and access reports.
3. **Student Interface:** Allows students to view their attendance records, send messages to teachers, and review feedback.

### 4.2.6 Use Case Diagram

The Use Case Diagram illustrates the interactions between users (actors) and the system, highlighting the functionalities available to each actor

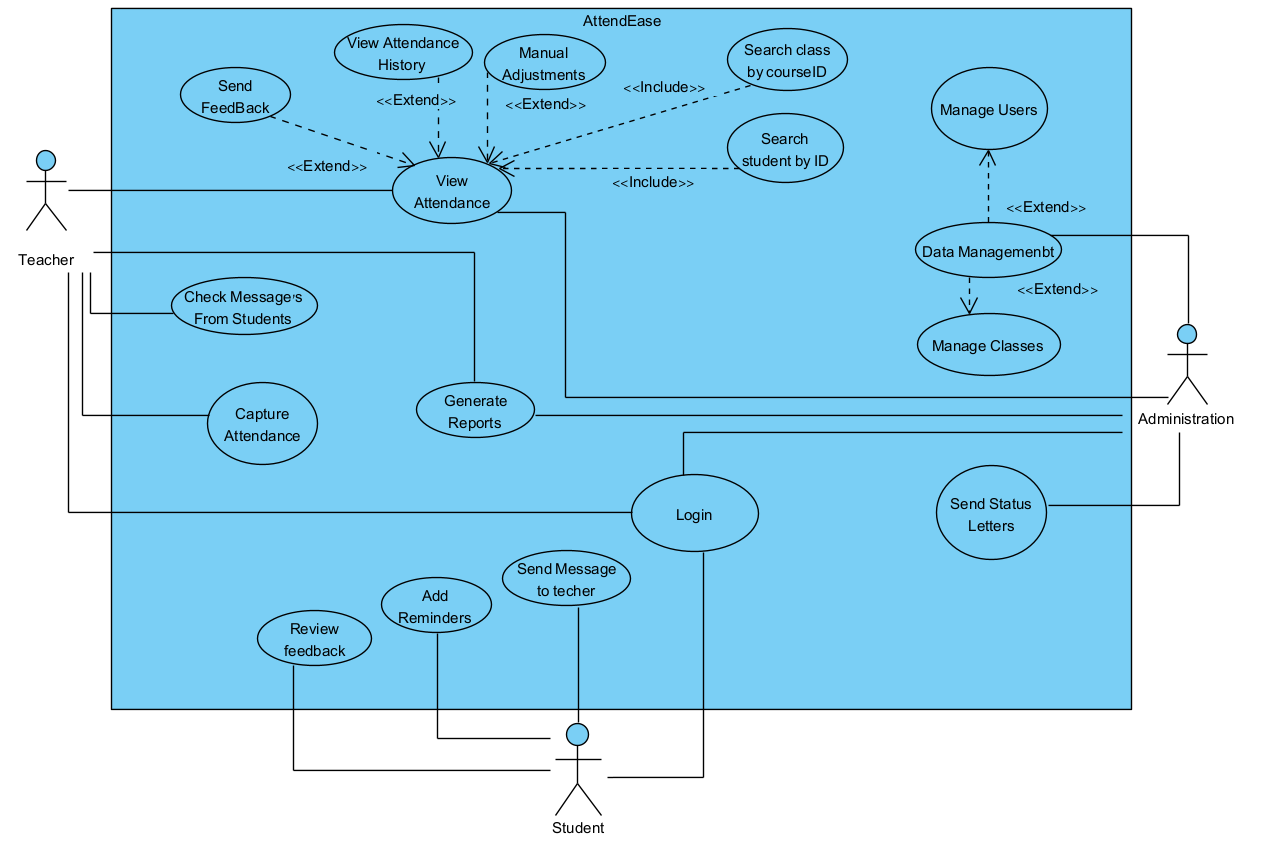


Figure 1: Use Case Diagram of AttendEase

### 4.2.7 Class Diagram

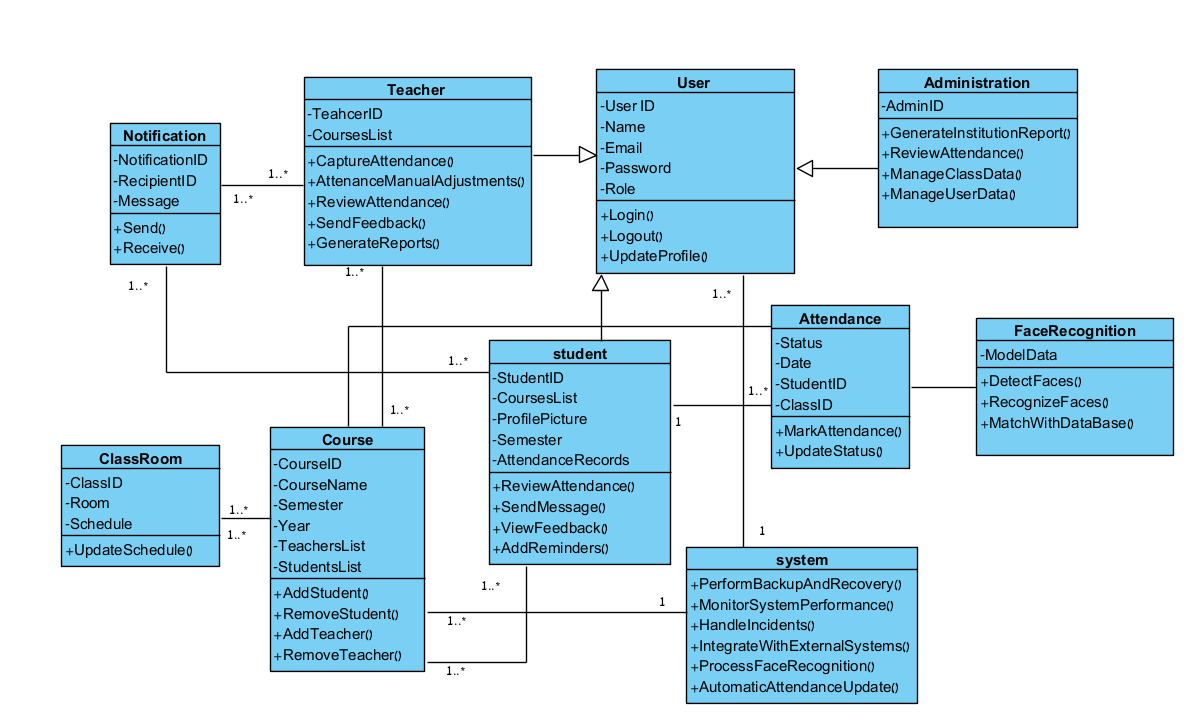
The Class Diagram outlines the system's structure, showing classes, attributes, methods, and relationships.

Figure 2: Class Diagram of AttendEase

### 4.2.8 Activity Diagram

The Activity Diagram represents the workflow of the attendance capturing process.

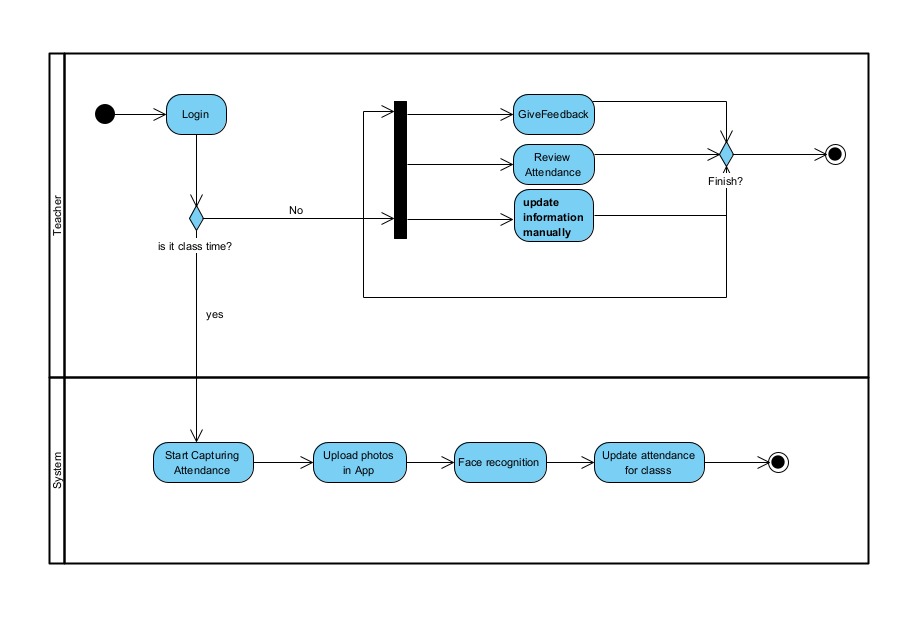


Figure 3: Activity Diagram of AttendEase

### 4.2.8 Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Category** | **Who Can Do the Action** | **Description** |
| **1** | User Authentication | Teachers, Students, Administrators | The system must allow to log in using secure credentials. |
| **2** | Data Management | Administrators | The system must allow to add, update, and delete student C. |
| **3** | Data Management | Teachers | The system must support uploading existing photos if needed. |
| **4** | Face Recognition | System (Automated) | The system must detect faces in the captured class photo. |
| **5** | Face Recognition | System (Automated) | The system must recognize and match detected faces with the student database. |
| **6** | Attendance Recording | System (Automated) | The system must automatically mark attendance based on face recognition results. |
| **7** | Attendance Recording | Teachers | The system must allow to manually adjust attendance records if necessary. |
| **8** | Attendance History and Reports | Students, Teachers, Administrators | The system must provide access to historical attendance records. |
| **9** | Attendance History and Reports | Teachers, Administrators | The system must generate attendance reports for individual students, classes, and overall statistics. |
| **10** | Notifications and Alerts | System (Automated) | The system must notify students of attendance status via email or app notifications. |
| **11** | Notifications and Alerts | System (Automated) | The system must alert teachers if a significant number of students are absent. |
| **12** | Notifications and Alerts | Students, Teachers | The system must allow to send and review messages and feedback. |
| **13** | Backup and Recovery | System (Automated) | The system must automatically back up data to prevent loss. |
| **14** | Backup and Recovery | System (Automated) | There must be a mechanism for recovering data in case of failure. |

### 4.2.9 Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **No.** | **Category** | **Description** |
| **1** | Performance | The system must process and record attendance for a class of up to 50 students within 5 minutes. |
| **2** | Performance | Support concurrent use by multiple teachers and classes without performance degradation. |
| **3** | Scalability | The system must be scalable to handle up to 500 students per institution. |
| **4** | Scalability | The system must be scalable to accommodate future growth in the number of students and classes. |
| **5** | Usability | The user interface must be intuitive and easy to use for teachers, administrators, and students. |
| **6** | Usability | The system must provide clear instructions and feedback to users. |
| **7** | Security | The system must encrypt all sensitive data, including student photos and attendance records. |
| **8** | Security | The system must comply with relevant data protection regulations (e.g., GDPR). |
| **9** | Security | The system must include measures to prevent unauthorized access. |
| **10** | Reliability | The system must have an uptime of at least 99.5%. |
| **11** | Reliability | The system must handle errors gracefully and provide meaningful error messages to users. |
| **12** | Maintainability | The system must be designed with modularity to facilitate updates and maintenance. |
| **13** | Maintainability | The codebase must be well-documented to ensure ease of maintenance. |
| **14** | Compatibility | The system must be compatible with various devices, including smartphones, tablets, and desktops. |
| **15** | Compatibility | The system must support common web browsers (e.g., Chrome, Firefox, Safari). |

# Chapter 5

# Testing Plan

## 5.1 Testing Strategy

The testing strategy for AttendEase is designed to ensure that the system meets all functional and non-functional requirements. The testing process will follow a systematic approach, beginning with unit testing, followed by integration testing, system testing, and concluding with manual testing. Each phase is intended to identify and address potential issues, ensuring the system's reliability, scalability, and usability.

**Testing Phases:**

1. **Unit Testing**: Testing individual components to ensure they work as intended.
2. **Integration Testing**: Verifying that different modules work together seamlessly.
3. **System Testing**: Ensuring the entire system functions as a cohesive whole.
4. **Manual Testing**: Simulating real-world usage scenarios to uncover issues that automated tests may miss.

## 5.2 Unit Testing

Unit testing will focus on verifying the functionality of individual components and methods within the system. The goal is to catch and resolve issues at the earliest stage of development, ensuring that each component performs as expected in isolation.

|  |  |  |
| --- | --- | --- |
| **Component** | **Test Case** | **Expected Result** |
| User Login | Login with valid details | Login successful, user navigates to the main dashboard |
| User Login | Login with invalid details | Login failed, error message displayed |
| User Profile | Update user profile with valid details | Profile updates successful, changes reflected in the database |
| User Profile | Update profile with invalid details | Update failed, error message displayed |
| View Reports (Student) | View attendance history | Attendance history displayed correctly |
| View Reports (Teacher) | View class attendance | Attendance records displayed correctly |
| View Reports (Administrator) | View overall attendance | Overall attendance data displayed correctly |
| Generate Reports (Administrator) | Generate institution report | Institution report generated successfully |
| Data Management (Administrator) | Manage user data with valid user ID | User data updated successfully |
| Data Management (Administrator) | Manage user data with invalid user ID | Update failed, error message displayed |
| Data Management (Administrator) | Manage class data with valid class ID | Class data updated successfully |
| Data Management (Administrator) | Manage class data with invalid class ID | Update failed, error message displayed |
| User Registration (Administrator) | Register a new user with valid details | Registration successful, user data stored in database |
| User Registration (Administrator) | Register a new user with invalid details | Registration failed, error message displayed |
| Users Management (Administrator) | Remove user with valid user ID | Student removed successfully from class |
| Users Management (Administrator) | Remove user with invalid user ID | Removal failed, error message displayed |
| Attendance Record Management | Mark attendance with valid details | Attendance marked successfully, records updated in the database |
| Attendance Record Management | Mark attendance with invalid details | Attendance marking failed, error message displayed |
| Attendance Record Management | Update attendance with valid details | Attendance updated successfully |
| Attendance Record Management | Update attendance with invalid details | Attendance update failed, error message displayed |
| Capture Attendance | Capture attendance with valid class ID | Attendance captured successfully, records updated in the database |
| Capture Attendance | Capture attendance with invalid class ID | Attendance capture failed, error message displayed |
| Generate Reports | Generate report for valid class ID | Report generated successfully |
| Generate Reports | Generate report with invalid class ID | Report generation failed, error message displayed |
| Face Recognition | Process face recognition with valid photo | Face recognized successfully, attendance records updated |
| Face Recognition | Process face recognition with invalid photo | Face recognition failed, error message displayed |

## 5.3 Integration Testing

Integration testing focuses on verifying the interactions between different modules of the system. The goal is to ensure that the modules, when combined, function as intended.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Modules Involved** | **Expected Result** |
| User Login and Dashboard | User Authentication, Dashboard | Successful login should redirect the user to the correct dashboard based on their role (teacher, student, administrator). |
| Attendance Capture and Face Recognition | Face Detection, Face Recognition, Database | Successfully captured attendance should trigger the face recognition module, which updates attendance records in the database. |
| Report Generation | Attendance Records, Reporting Module | Generating a report should correctly retrieve and display data from the attendance records. |
| Data Management and Notification System | Database, Notification Module | Updates in student data should trigger appropriate notifications to students. |
| User Interface Interaction | UI, Backend APIs | All user interface actions (e.g., button clicks, form submissions) should correctly interact with backend APIs and reflect changes in the UI. |

## 5.4 System Testing

System testing ensures that the entire system meets the specified requirements and functions as a complete, integrated application.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Description** | **Expected Result** |
| Full System Workflow | Simulate a complete attendance capture process from login to report generation | The system should successfully handle the entire workflow without errors, accurately capturing attendance and generating reports. |
| Performance Testing | Test system performance under maximum load (e.g., 500 students across multiple classes) | The system should remain responsive and complete tasks within the performance criteria specified in the non-functional requirements. |
| Security Testing | Test system security features, including login, data encryption, and access control | The system should enforce all security measures, preventing unauthorized access and securely handling all data transactions. |
| Data Recovery Testing | Simulate a system crash and test data recovery procedures | The system should successfully recover data from backups without loss. |
| Cross-Platform Compatibility | Test system on different devices (desktop, tablet, smartphone) and browsers | The system should function correctly across all tested platforms and browsers, maintaining usability and performance. |

## 5.5 Manual Testing

Manual testing simulates real-world user scenarios to identify issues that automated tests might miss. This testing is crucial for ensuring that the user interface is intuitive and that the system behaves as expected under various conditions.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Scenario** | **Expected Result** |
| Teacher Interface Navigation | Teacher navigates through the dashboard, captures attendance, and generates reports | All actions should be smooth, with clear feedback and no unexpected errors or delays. |
| Student Profile Update | A student attempts to update their profile information | The system should allow updates, show confirmation, and reflect changes in the database. |
| Error Handling | Enter incorrect data in forms (e.g., invalid class ID during attendance capture) | The system should display meaningful error messages and guide the user to correct the input. |
| User Satisfaction Survey | Collect feedback from users after completing common tasks (e.g., capturing attendance, viewing reports) | Feedback should indicate ease of use, with any issues reported for further refinement. |