

“FACIAL RECOGNITION ATTENDANCE SYSTEM”

SHREE DEVI COLLEGE OF INFORMATION SCIENCE

(Affiliated to Mangalore University)

Ballalbhagh, Mangalore – 575003



A PROJECT REPORT ON
FACIAL RECOGNITION ATTENDANCE SYSTEM
Submitted in Partial fulfillment of requirement for the award of

BACHELOR OF COMPUTER APPLICATIONS
2025-2026

SUBMITTED BY

VIJETHA ACHARYA

[REG NO: U051S225023]

VEEKSHITH AMANNA

[REG NO: U051S225080]

UNDER THE GUIDANCE OF

MR. TRILOKANATH

SHREE DEVI COLLEGE OF INFORMATION SCIENCE

SHREE DEVI COLLEGE OF INFORMATION SCIENCE MANGALORE



CERTIFICATE

This is to certify that the project work entitled as “**FACIAL RECOGNITION ATTENDANCE SYSTEM**” has been successfully carried out by VIJETHA ACHARYA [REG NO: U051S225023] and VEEKSHITH AMANNA [REG NO: U051S225080] students of 3rd year BCA. This dissertation is submitted in partial fulfilment for the award of in Bachelor of Computer Application by Mangalore University during the academic year of 2025-2026.

Internal Guide
(Mr.Trilokanath)

Principal
(Mr.Trilokanath)

Submitted for the Vice-Voice examination held on _____

Internal Examiner :

External Examiner :

DECLARATION

I hereby declare the project entitled “FACIAL RECOGNITION ATTENDANCE SYSTEM” is a genuine work in the partial fulfillment of the requirement for the curriculum of 6thSemester, Bachelor of Computer Application, prescribed by the Mangalore University, Mangalore. It is an authentic project record of work carried out by during the period MARCH 2025 - MAY 2025 under the guidance of Trilokanath.

It is also declared that the matter embodied in the project work is original and has not been submitted by us to any other university wholly or partly.

VIJETHA ACHARYA

[REG NO: U051S225023]

VEEKSHITH AMANNA

[REG NO: U051S225080]

Place: Mangalore

Date :

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any work would be but incomplete without mentioning the people who made it possible, whose constant guidance and encouragement served as beacon light and crowned our efforts with success.

We express our sincere gratitude to H. Triloknath, Principal & HOD , Shree Devi College of Information Science , Mangalore for providing the required facilities and are thankful for his help and instructions rendered to us during our project work, and without whom this project would have been impossible.

Thanks to Shree Devi College for giving us an opportunity to take up this project, as the knowledge gained by working on a project is very valuable.

We would be failing in our duty if we do not thank our parents, our friends and the almighty on which we have relied on for help and encouragement throughout our BCA course and especially during our project work.

VIJETHA ACHARYA

VEEKSHITH AMANNA

TABLE OF CONTENTS

ACKNOWLEDGE			i
ABSTRACT			ii
TABLE OF CONTENTS			iii
1	INTRODUCTION		1
	1.1	Overview	
	1.2	Motivation	
	1.3	Objective	
	1.4	Scope	
	1.5	Existing System	
	1.6	Proposed System	
2	PROBLEM STATEMENT		4
	2.1	Problem Statement	
	2.2	Motivation	
	2.3	Objectives	
3	SYSTEM REQUIREMENT SPECIFICATION		6
	3.1	Functional Requirements	
	3.2	Non-functional Requirements	
	3.3	Hardware Requirements	
	3.4	Software Requirements	
4	SYSTEM DESIGN		10
	4.1	System Design	

		4.1.1	System Architecture	
--	--	--------------	---------------------	--

		4.1.2	Module Design	
	4.2	Detailed Design		
		4.2.1	Activity Diagram	
		4.2.2	Use Case Diagram	
		4.2.3	Scenarios	
5	IMPLEMENTATION			19
6	RESULTS			29
7	CONCLUSIONS			32
8	REFERENCES			33

CHAPTER 1

INTRODUCTION

1.1 Overview

Attendance tracking is a fundamental task in educational institutions, ensuring students' regular participation in academic activities. Traditional methods, such as manual roll calls or signature-based registers, are time-consuming, error-prone, and susceptible to manipulation. To overcome these challenges, modern technology offers automated attendance systems that combine facial recognition with web-based interfaces for efficient and accurate tracking. This project presents a Facial Recognition Attendance System that leverages web development technologies and machine learning to automate attendance marking, ensuring accuracy and saving time.

1.2 Motivation

The primary motivation behind this project stems from the inefficiencies and limitations of conventional attendance systems. Manual methods often consume significant class time and can lead to inaccuracies. Furthermore, the increasing adoption of digital transformation in education inspired the development of a system that aligns with modern technological advancements. Facial recognition offers a non-intrusive and secure way to verify student identity, eliminating the possibility of proxy attendance while streamlining the entire process.

1.3 Objectives

The objectives of this project are:

- To develop a user-friendly web-based platform for managing attendance through facial recognition technology.
- To integrate automated student registration with Google Sheets for efficient data storage and Google Drive for image management.
- To implement a robust authentication system for teachers to access and manage the platform securely.

- To ensure high accuracy and reliability in identifying students based on facial recognition.
- To provide automated record-keeping and attendance reports for better data accessibility and management.

1.4 Scope

The project aims to cater to educational institutions seeking to transition from manual to automated attendance systems. It is designed to support the following functionalities:

- Secure login for teachers to access the system.
- Easy student registration, including the storage of personal details and facial images.
- Real-time facial recognition-based attendance marking using OpenCV.
- Integration with Google Sheets for scalable data storage and Google Drive for image management.
- Automated generation of attendance reports in CSV format for future analysis.

1.5 Existing System

Traditional attendance systems are predominantly manual, requiring teachers to call out names or rely on physical registers. These systems:

- Are time-consuming and inefficient, especially in large classrooms.
- Are prone to human errors, such as incorrect marking or skipped entries.
- Lack security, allowing for manipulation like proxy attendance.
- Do not provide instant data analysis or insights.
- Some automated systems rely on RFID cards or biometric systems, but these require physical interactions or additional hardware, making them less efficient and user-

friendly

1. 6 Proposed System

The proposed Facial Recognition Attendance System overcomes the drawbacks of the existing methods by integrating the following features:

- **Web-Based Platform:** A centralized system accessible through a user-friendly website for teachers.
- **Facial Recognition:** Non-intrusive and accurate identification of students using facial recognition technology.
- **Google Integration:** Data storage and management using Google Sheets and Google Drive for scalability and easy access.
- **Automated Record Keeping:** Attendance data is automatically stored in CSV files, ensuring accuracy and enabling seamless record generation.
- **Security:** A robust login system ensures only authorized teachers can access the platform.
- This system not only reduces the time and effort involved in attendance tracking but also aligns with the growing demand for digital transformation in education

CHAPTER-2

PROBLEM STATEMENT

a. Problem Statement

Managing attendance is a crucial aspect of educational institutions and workplaces, but the traditional methods of attendance tracking—such as manual roll calls or signing attendance sheets—are time-consuming, prone to errors, and susceptible to fraudulent practices like proxy attendance. Furthermore, storing and retrieving attendance data for analysis or reporting becomes cumbersome when handled manually. With advancements in technology, there is an opportunity to streamline the attendance process using modern tools such as facial recognition, which offers a seamless, accurate, and efficient solution.

b. Motivation

- **Inefficiencies in Current Systems:** Traditional attendance systems require significant manual effort, leading to inefficiencies, time loss, and inaccuracies. Automation can resolve these issues.
- **Increasing Use of Technology in Education:** With the growing adoption of digital tools in education, integrating facial recognition aligns with modern trends and prepares institutions for future advancements.
- **Fraud Prevention:** Proxy attendance is a common issue in traditional systems. Using facial recognition ensures that only the right individuals are marked as present.
- **Data Centralization:** Storing data in platforms like Google Sheets and Drive provides centralized, easily accessible, and secure storage for attendance records.
- **Personal Interest and Skill Development:** Building such a project serves as a learning opportunity for understanding web development, Google API integration, and facial recognition technology.

c. Objectives

- **Develop a Secure Login System:** Ensure that only authorized teachers can access the system through a secure login mechanism validated against stored credentials.
- **Implement Automated Student Registration:** Enable teachers to register students seamlessly by storing details like name, department, and photo in a structured Google Sheet, with photos securely stored in Google Drive.
- **Integrate Facial Recognition for Attendance:** Use OpenCV and face recognition libraries to identify and mark student attendance accurately by comparing live camera feeds with pre-registered images.
- **Centralize Data Management:** Store attendance records in a CSV file for easy retrieval, analysis, and reporting.
- **Ensure Scalability and Accessibility:** Design the system to handle growing numbers of users and allow access from any device with internet connectivity.
- **Promote Efficiency and Accuracy:** Minimize the time and effort required for attendance management while ensuring a high degree of accuracy and reliability.

CHAPTER-3

SYSTEM REQUIREMENT SPECIFICATION

3.1 Functional Requirements

The functional requirements for the project are as following:

a. Login System :

- Login Page: Teachers must log in to access the system. The login credentials (email and password) are stored in a JSON file.
- Input: Teacher's email and password.
- Output: Grant access if the credentials match, else show an error message ("Invalid credentials").
- Validation: Only teachers with valid credentials (email and password) can log in. Invalid login attempts result in a denial message.

b. Student Registration form:

- Registration Form: Once logged in, the teacher can access a student registration form. The form collects the following data:
 - Full Name
 - USN (Unique Student Number)
 - Email ID
 - Gender
 - Department
 - Upload Photo

- **Form Submission:** Upon successful form submission, the data, along with the student's photo, is stored in:
 - Google Sheets: For record-keeping and further processing.
 - Google Drive: The student's photo is stored in a publicly accessible, downloadable link.

c. Facial Recognition Attendance:

- **Camera Integration:** The teacher can click the "Mark Attendance" button, which activates the camera using OpenCV.
- **Face Detection:** The system detects faces from the live camera feed, compares them to pre-stored images in the database (Google Sheets), and identifies matching students.
- **Attendance Marking:** If a match is found, the student's attendance is marked as "present" and stored in a CSV file along with the timestamp.
- **Real-Time Processing:** The system continuously monitors the live feed and processes facial recognition for each student.

d. Error Handling:

- The system should handle errors gracefully in case of:
 - Invalid login attempts.
 - No student photo URL available.
 - Face not recognized during attendance marking.
 - Form submission failures.

3.2 Non-functional Requirements

The Non-functional requirements for the project are as following:

a. Performance:

- **Real-Time Face Recognition:** The system should be able to process and identify

faces in real-time with minimal delay (less than 5 seconds per student).

- Scalability: The system should be capable of handling an increasing number of students without significant performance degradation (e.g., handling 1000+ students).

b. Usability:

- The login page should be simple with clear input fields for email and password.
- The student registration form should be simple with fields for required information and an easy file upload option for student photos.
- The attendance page should display a live camera feed and show student names when recognized.

c. Security:

- Login Security: The teacher login system should use basic validation and authentication. The email and password should not be stored in plain text.
- Data Privacy: The student data, including photos, should be stored securely on Google Sheets and Drive, accessible only by authorized teachers.

d. Reliability:

- The system should ensure that student data is consistently saved and updated.
- In case of a network failure, the system should handle timeouts and retry the operation or inform the user of the failure.

e. Availability:

- The system should be available 24/7 for use by authorized teachers. Downtime should be minimal and planned for maintenance.

f. Maintainability:

- The system should be easy to update with new features and bug fixes.
- Code should be well-documented.
- External libraries or services used (e.g., OpenCV, Google Sheets API) should be kept up-to-date.

3.3 Hardware Requirements

The hardware requirements for the project are as following:

a. Teacher's Computer/Device :

- CPU: Intel Core i5 or equivalent (minimum).
- RAM: 4 GB or higher.
- Storage: 100 MB of free space for storing software and logs.
- Operating System: Windows 10/11, macOS, or Linux (depending on the teacher's preference).
- Webcam: Required for capturing live video for facial recognition.

b. Camera for Facial Recognition:

- Resolution: Minimum of 720p (HD) resolution for accurate face detection.
- Frame Rate: At least 30 frames per second (FPS) for smooth video capture.

c. Internet Connection:

- Speed: A stable internet connection with a minimum speed of 1 Mbps for form submissions and Google API calls (e.g., Google Sheets API).

3.4 Software Requirements

The software requirements for the project are as following:

a. Web Technologies Frontend:

- HTML5: For structure and content.
- CSS3: For styling and layout.
- JavaScript: For dynamic client-side interactions (form validation, API requests).

b. Backend Technologies:

- Flask: A lightweight Python framework used to handle backend logic and API calls.
- Python: For facial recognition and attendance marking using OpenCV and other libraries.

c. Third-Party APIs:

- Google Sheets API: To store and retrieve student data in Google Sheets.
- Google Drive API: To upload and store student photos in Google Drive.

d. Libraries/Dependencies:

- Face Recognition Library: Python library for facial recognition (uses dlib under the hood).
- OpenCV: For webcam feed and image processing.

- Requests: To fetch student images from URLs for facial recognition.
- Pillow: For image processing in Python.

e. Cloud Services:

- Google Drive: To store the student images and allow access via a link.
- Google Sheets: To store student data in a tabular format.

CHAPTER-4

SYSTEM DESIGN

4.1 System Design

4.1.1 System Architecture

The System Architecture describes the high-level structure and interaction between components of the system. This project follows a Client-Server Architecture with integration of different web technologies and APIs. Here's a simplified breakdown:

1.Frontend (Client-side) :

Technologies: HTML, CSS, JavaScript (with Flask as a backend framework)

Functionality:

- Provides a user interface (UI) for teachers to log in, register students, and mark attendance.
- Displays the registration form, takes the student data, and sends it to the backend for processing and storage.
- Displays real-time camera feed for facial recognition to mark attendance.

2.Backend (Server-side)

Technologies: Flask (Python framework)

Functionality:

- Handles user authentication and authorization (login page for teachers).
- Processes student registration data (inserting it into Google Sheets, uploading photos to Google Drive).
- Interfaces with OpenCV for facial recognition and compares the captured image with stored student data.

3.Google Sheets and Drive API

Technologies: Google Sheets API, Google Drive API

Functionality:

- Google Sheets: Stores student data (name, USN, email, etc.) and image URLs for the registered students.
- Google Drive: Stores student photos (in the form of downloadable links) uploaded via the registration form.

4. OpenCV and Facial Recognition

Technologies: OpenCV, face_recognition (Python libraries)

Functionality:

- Captures live video feed from the camera.
- Recognizes faces and matches them with the pre-encoded student images stored in Google Sheets.
- Marks attendance by storing the present students' data in a CSV file.

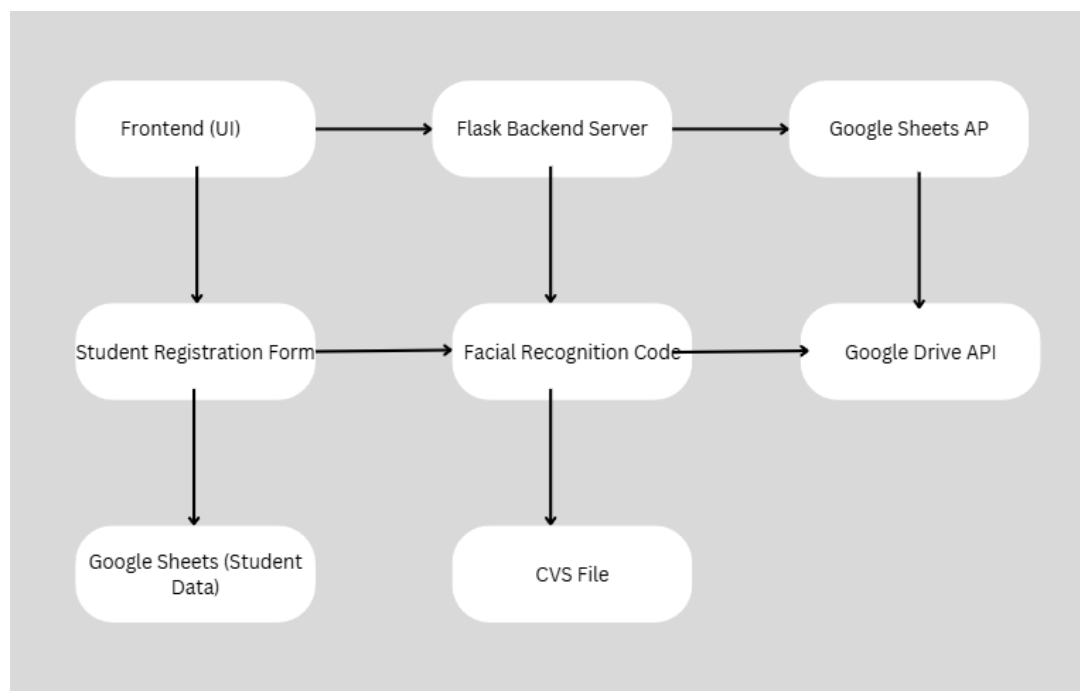


Fig 4.1 System Architecture

4.1.2Module Design

Module 1: Teacher Login and Authentication

Objective: To authenticate teachers before granting access to the system.

Components:

Input: Email ID and password entered by the teacher.

Process: Checks if the provided email and password match the stored details in a JSON file.

Output: If valid credentials are provided, the teacher is granted access to the home page; otherwise, an error message is displayed.

Steps:

The system receives login details from the frontend (HTML form).

The backend compares the details with a JSON file containing authorized teacher credentials.

If a match is found, the teacher is logged in and redirected to the home page; otherwise, an error message is shown.

Module 2: Student Registration

Objective: To allow teachers to register student details and upload photos to Google Sheets and Google Drive.

Components:

Input: Student details (full name, USN, email, gender, department, photo).

Process:

Form data is sent from the frontend to the Flask backend.

The backend sends data to Google Sheets for storage.

The photo is uploaded to Google Drive, and the link is saved in the sheet.

Output: Student details are added to Google Sheets, and the image is stored in Google Drive with a shareable link.

Steps:

A teacher fills in the registration form on the frontend.

The form data is sent via POST request to the Flask backend.

The backend uploads the student's image to Google Drive and stores the image link in Google Sheets, along with other student details.

Module 3: Facial Recognition and Attendance Marking

Objective: To mark attendance based on facial recognition using a camera feed.

Components:

Input: Real-time camera feed capturing the teacher's students.

Process:

Capture the image, encode it, and compare it to the saved student images in Google Sheets.

If a match is found, the student's attendance is marked.

Output: The student is marked as present, and the attendance is recorded in a CSV file.

Steps:

The teacher clicks the "Mark Attendance" button on the homepage.

The camera feed is accessed and processed using OpenCV.

The captured image is compared with encoded images stored in Google Sheets.

The attendance is recorded in a CSV file, noting the student's presence.

Module 4: Google Sheets Integration

Objective: To store and retrieve student data from Google Sheets.

Components:

Input: Student details from the registration form.

Process: The backend communicates with Google Sheets API to insert or retrieve data.

Output: Data is stored in Google Sheets and retrieved for facial recognition matching.

Steps:

Google Sheets stores details like the student's full name, USN, email, gender, department, and photo URL.

The OpenCV facial recognition process retrieves the photos' URLs to compare the captured image with registered students.

Module 5: Google Drive Integration

Objective: To store images (photos) of students in Google Drive.

Components:

Input: Student images uploaded through the registration form.

Process: The images are uploaded to Google Drive, and a shareable URL is generated.

Output: The image URL is stored in Google Sheets for future reference.

Steps:

When the student registers, their photo is uploaded to a specific Google Drive folder.

A link to this photo is stored in Google Sheets, making it accessible for the facial recognition process.

4.2 Detailed Design

4.2.1 Activity Diagram

The Activity Diagram shows the workflow or process of your system from start to finish. Below is a simplified workflow of your Facial Recognition Attendance System:

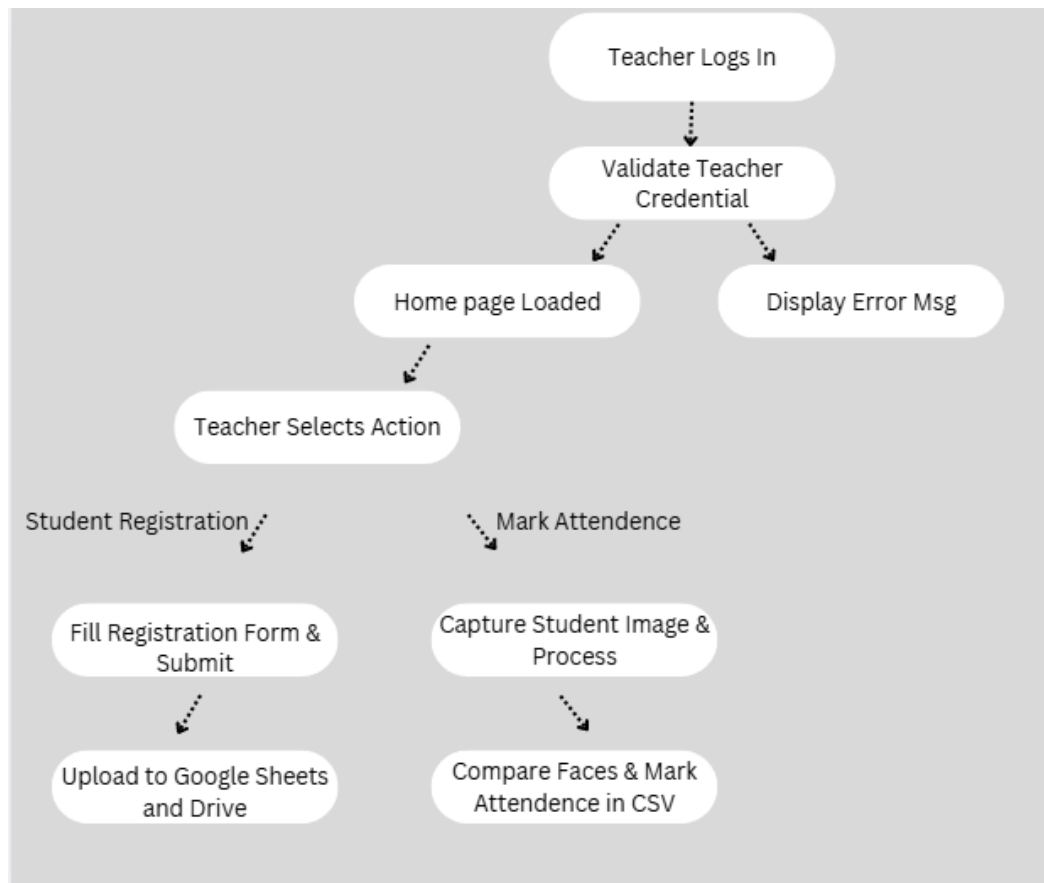


Fig 4.2 Activity Diagram

4.2.2 Use Case Diagram

The Use Case Diagram shows the interactions between the system and its users (actors).

Here is a Use Case Diagram for your Facial Recognition Attendance System:

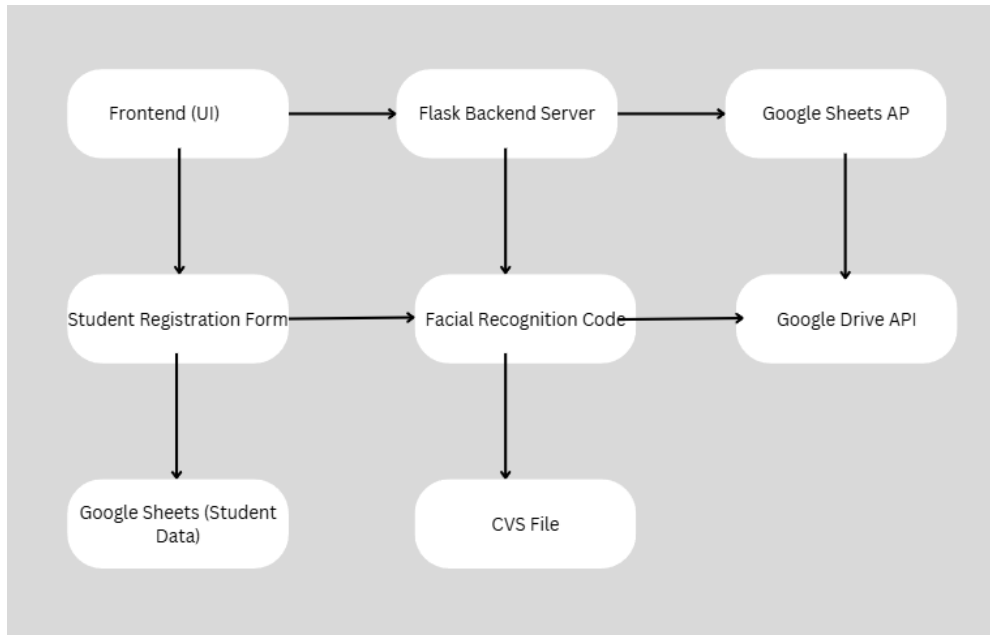


Fig 4.3 Use Case Diagram

4.2.3 Scenarios

The scenarios for the "FACIAL RECOGNITION ATTENDENCE SYSTEM" are as following:

Scenario 1: Teacher Login

Actors: Teacher, System

Description: A teacher enters their email and password to access the system.

Steps:

- The teacher enters the email and password in the login form.
- The system validates the credentials by comparing them with the data stored in the JSON file.
- If valid, the teacher is redirected to the home page. If invalid, an error message is displayed.

Scenario 2: Student Registration

Actors: Teacher, System, Google Sheets, Google Drive

Description: A teacher registers a student with details and an image.

Steps:

- The teacher navigates to the registration page and fills in the student's details (name, USN, email, etc.).
- The teacher uploads a student photo.
- The system uploads the student's photo to Google Drive and retrieves the file URL.
- The student's details, along with the photo URL, are saved to Google Sheets.

Scenario 3: Marking Attendance

Actors: Teacher, System, OpenCV, Google Sheets

Description: The teacher marks attendance based on facial recognition.

Steps:

- The teacher clicks on the Mark Attendance button.
- The camera starts capturing the live video feed.
- The captured image is processed by OpenCV to recognize the student's face.
- The system compares the face with the stored images in Google Sheets.
- If a match is found, the student's attendance is marked as present and saved in the CSV file.

CHAPTER-5

IMPLEMENTATION

Login Page:

The Login Page of our project allows teachers to log in using their email and password, which are stored in a local JSON file. When the teacher enters their credentials and submits the login form, the backend (Flask) validates the entered email and password by comparing them with the stored data in the JSON file. If the credentials match, the teacher is granted access to the Home Page, where they can register students, mark attendance, and perform other tasks. If the credentials are incorrect, an error message is displayed, prompting the teacher to retry. This setup provides a simple, file-based authentication mechanism, suitable for small-scale applications like yours.

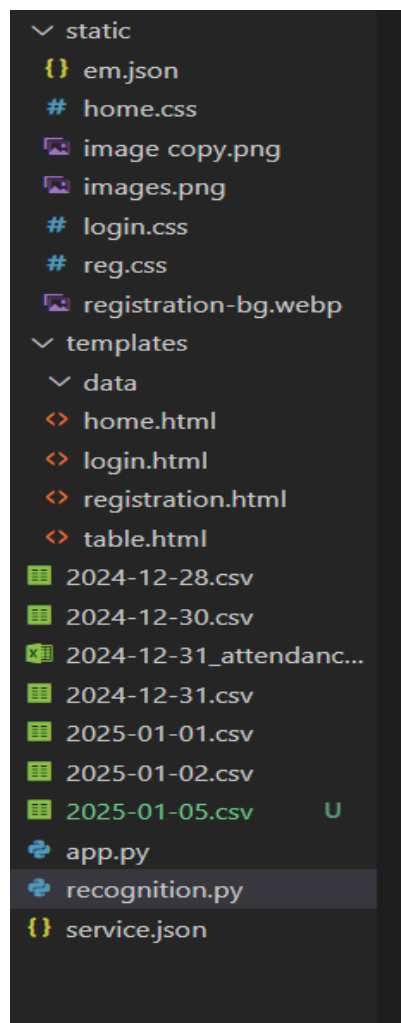


Fig 5.1 Project Structure

```

<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Faculty Login</title>
  <script type="module" src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>
  <script nomodule src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>
  <link rel="stylesheet" href="{{ url_for('static', filename='login.css') }}">
</head>
<script>
  async function validateLogin(event) {
    event.preventDefault(); // Prevent default form submission

    const emailInput = document.getElementById('email');
    const passwordInput = document.getElementById('password');
    const email = emailInput.value.trim();
    const password = passwordInput.value;

    try {
      const response = await fetch('/static/em.json'); // Path to your JSON file
      const allowedCredentials = await response.json();

      if (!Array.isArray(allowedCredentials)) {
        alert("Error: Allowed credentials data is invalid.");
        return false;
      }

      const isValid = allowedCredentials.some(credential =>
        credential.email === email && credential.password === password
      );

      if (!isValid) {
        alert("Access Denied: Invalid email or password.");
        return false;
      }

      window.location.href = '/home'; // Redirect to home page
      return;

      // Submit the form if credentials are valid
      const form = document.querySelector('form');
      form.submit();
    } catch (error) {
      console.error("Error fetching allowed credentials:", error);
      alert("Unable to validate login at this time. Please try again later.");
      return false;
    }
  }
</script>

```

```

async function forgotPassword() {
  const email = prompt("Enter your registered email:");

  if (!email) {
    alert("Email is required to reset password.");
    return;
  }

  try {
    const response = await fetch('/static/em.json');
    const allowedCredentials = await response.json();

    const user = allowedCredentials.find(credential => credential.email === email);

    if (!user) {
      alert("Email not found. Please try again.");
      return;
    }

    const securityQuestion = prompt(`Security Question: ${user.securityQuestion}`);

    if (securityQuestion.toLowerCase() === (user.securityAnswer || '').toLowerCase()) {
      const newPassword = prompt("Enter your new password:");

      if (newPassword) {
        user.password = newPassword;

        // Simulating updating the JSON file
        await fetch('/update_credentials_endpoint', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify(allowedCredentials)
        });

        alert("Password reset successful. Please log in with your new password.");
      } else {
        alert("Password reset cancelled.");
      }
    } else {
      alert("Incorrect answer to the security question.");
    }
  } catch (error) {
    console.error("Error handling forgot password:", error);
    alert("Unable to reset password at this time. Please try again later.");
  }
}
</script>
</head>

```

```

<body>
  <section>
    <div class="form-box">
      <div class="form-value">
        <form onsubmit="validateLogin(event)">
          <h2>Faculty Sign In</h2>
          <div class="inputbox">
            <ion-icon name="mail-outline"></ion-icon>
            <input type="email" id="email" required>
            <label>Email</label>
          </div>
          <div class="inputbox">
            <ion-icon name="lock-closed-outline"></ion-icon>
            <input type="password" id="password" required>
            <label>Password</label>
          </div>
          <div class="forget">
            <label><input type="checkbox">Remember Me</label>
            <a href="#" onclick="forgotPassword()">Forgot Password</a>
          </div>
          <button type="submit">Log In</button>
          <div class="register">
            <p>Don't have an account? <a href="#">Sign Up</a></p>
          </div>
        </form>
      </div>
    </div>
  </section>
</body>

</html>

```

```

static > {} em.json > ...
1
2 [
3   {
4     "email": "preeti.is22@bmsce.ac.in",
5     "password": "preeti",
6     "securityQuestion": "What is your favorite color?",
7     "securityAnswer": "Blue"
8   },
9   {
10    "email": "pragathi.is22@bmsce.ac.in",
11    "password": "pragathi",
12    "securityQuestion": "What is your favorite color?",
13    "securityAnswer": "Blue"
14  },
15  {
16    "email": "rashmitha.is22@bmsce.ac.in",
17    "password": "rash",
18    "securityQuestion": "What is your favorite color?",
19    "securityAnswer": "Blue"
20  },
21  {
22    "email": "rajeshwari.is22@bmsce.ac.in",
23    "password": "rajeshwari",
24    "securityQuestion": "What is your favorite color?",
25    "securityAnswer": "Blue"
26  },
27  {
28    "email": "divyashree.is22@bmsce.ac.in",
29    "password": "divyashree",
30    "securityQuestion": "What is your favorite color?",
31    "securityAnswer": "Blue"
32  }
33 ]
34
35
36
37

```

Home Page:

The Home Page serves as the main dashboard for teachers after they successfully log in. It features three primary options: Student Registration, Mark Attendance, and View Attendance Table. The layout is clean and user-friendly, allowing the teacher to easily navigate through these functionalities. On clicking the respective buttons, the user is redirected to the corresponding pages where they can perform the desired actions. The Home Page acts as the control center for the system, making it easy for the teacher to manage student data and attendance.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Facial Recognition Attendance System</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='home.css') }}">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Ev84W4kVGN5gJGL/FaIDqQ7wQ2vcdIwxfjTHSHBCSR7PBEakC751Ck+u/U6swU2In1vXBSYk94Bh==" crossorigin="an">
</head>
<body>
  <div class="dashboard">
    <aside class="sidebar">
      <h2>Dashboard</h2>
      <nav>
        <ul>
          <li><i class="fa-solid fa-user"></i><a href="{{ url_for('registration') }}"> Student Registration</a></li>
          <li><i class="fa-solid fa-pen"></i><a href="javascript:void(0)" id="mark-attendance"> Mark Attendance</a></li>
          <li><i class="fa-solid fa-circle-info"></i><a href="{{ url_for('table') }}"> Attendance Details</a></li>
        </ul>
      </nav>
    </aside>
    <main class="main-content">
      <h1>Facial Recognition Attendance System</h1>
      <div class="main-buttons">
        <a href="{{ url_for('registration') }}"><button><i class="fa-solid fa-user"></i> Student Registration</button></a>
        <button id="mark"><i class="fa-solid fa-pen"></i> Mark Attendance</button>
        <a href="{{ url_for('table') }}"><button><i class="fa-solid fa-circle-info"></i> Attendance Details</button></a>
      </div>
      <div id="webcam-container" style="display:none">
        <video id="video" width="640" height="480" autoplay</video>
        <button id="capture" onclick="captureFrame()">Capture</button>
      </div>
    </main>
  </div>
  <script>
    document.getElementById("mark").addEventListener("click", () => {
      fetch("/mark", { method: "POST" })
        .then(response => response.json())
        .then(data => {
          alert(data.message);
        })
        .catch(error => {
          alert("Error: " + error);
        });
    });
  </script>
</body>
</html>
```

Registration Page :

The Registration Page allows teachers to add new students into the system. The page features a registration form where the teacher can enter the student's full name, USN (University Serial Number), email, gender, department, and upload a photo of the student. Upon form submission, the data is stored in a Google Sheet, and the image is uploaded to Google Drive as a downloadable link. The page provides visual feedback to confirm that the form was submitted successfully, ensuring a smooth user experience.

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6   <meta http-equiv="X-UA-Compatible" content="ie=edge" />
7   <link rel="stylesheet" href="{{ url_for('static', filename='reg.css') }}">
8   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity="sha512-Evv9K1DRjppisCSHBP4sfy01zqc/NguJIuTfSpx4g6SRkO2WHC7QWcR&rsquo;s51Cr5l&rsquo;sEakr51&rsquo;sCk+w&rsquo;sU6swD2Im&rsquo;sVx8BS&rsquo;sYk9ABhg=&rsquo;" crossorigin="anonymous">
9   <title>Registration Form</title>
10 </head>
11 <body>
12   <a href="{{ url_for('home') }}" class="back-button fa-solid fa-arrow-left"></a>
13   <section class="container">
14     <header>Registration Form</header>
15     <form id="form" class="form" name="submit-to-google-sheet" method="POST" enctype="multipart/form-data" action="https://script.google.com/macros/s/Akfycbm0dJhbglL-c0CsopIDjuY-KSCbwXiIjyIn3938KxmV8fhSHmoasPPMw-X8FDktbx/exec">
16       <div class="input-box">
17         <label>Full Name</label>
18         <input type="text" placeholder="Enter full name" name="fullName" required />
19       </div>
20       <div class="input-box">
21         <label>USN</label>
22         <input type="text" placeholder="Enter USN" name="usn" required />
23       </div>
24       <div class="input-box">
25         <label>Email ID</label>
26         <input type="email" placeholder="Enter college mail ID" name="email" required />
27       </div>
28       <div class="gender-box">
29         <h3>Gender</h3>
30         <div class="gender-option">
31           <div class="gender">
32             <input type="radio" id="check-male" name="gender" value="male" checked />
33             <label for="check-male">Male</label>
34           </div>
35           <div class="gender">
36             <input type="radio" id="check-female" name="gender" value="female" />
37             <label for="check-female">Female</label>
38           </div>
39           <div class="gender">
40             <input type="radio" id="check-other" name="gender" value="prefer not to say" />
41             <label for="check-other">Prefer not to say</label>
42           </div>
43         </div>
44       </div>
45       <div class="input-box">
46         <label>Department</label>
47         <div class="column">
48           <div class="select-box">
49             <select name="department" required>
50               <option hidden="" selected="">Department</option>
51               <option>Computer Science and Engineering</option>
52               <option>Information Science and Engineering</option>
53               <option>Artificial Intelligence</option>
54               <option>Biotechnology</option>

```

```

<form id="form" class="form" name="submit-to-google-sheet" method="POST" enctype="multipart/form-data" action="https://script.google.com/macros/s/AKfycbw0bJhBg1w-cDcspqU0JuY-K9CnwuII3jLn393RkXnuV0fH5moasPMM-K8FDKtbx/exec">
  <div class="input-box">
    </div>
    <button type="submit" id="submitBtn">Submit</button>
  </div>
  <span id="success"></span>
</form>

<script>
  function FileOnChange(p) {
    var file = p.files[0];
    var ImgSrc = URL.createObjectURL(file);
    document.getElementById("imgPreview").src = ImgSrc;
    document.getElementById("imgPreview").style.display = "block"; // Show image preview

    let imgRead = new FileReader();

    imgRead.onload = function(e) {
      var AllData = e.target.result;
      var ImgData = AllData.split("base64,")[1]; // Extract base64 image data
      document.getElementById("imgData").value = ImgData; // Store the base64 image data in a hidden input field
    }

    imgRead.readAsDataURL(file);
  }

  const form = document.getElementById('form');
  const submitBtn = document.getElementById('submitBtn');

  form.addEventListener("submit", (e) => {
    e.preventDefault(); // Prevent default form submission behavior
    submitBtn.innerHTML = "Submitting...";

    const formData = new FormData(form); // Prepare the form data to send

    fetch(form.action, { // Send the form data via a POST request to the Apps Script URL
      method: "POST",
      body: formData,
    })
    .then((res) => res.text())
    .then((response) => {
      submitBtn.innerHTML = "Submit";
      document.getElementById("success").innerHTML = "Form submitted successfully!";
      console.log(response);
      form.reset(); // Reset the form after successful submission
    })
    .catch((error) => {
      submitBtn.innerHTML = "Submit";
      document.getElementById("success").innerHTML = "Error submitting the form. Please try again.";
      console.error("Error:", error);
    });
  });

```


Mark Attendance Page:

The Mark attendance Page(recognition.py) uses the computer's camera to capture the student's face and mark attendance using facial recognition. When a teacher clicks on the Mark Attendance button, the page activates the camera, and the OpenCV library processes the image captured in real-time. It then compares the captured face with the stored images in the system (from the Google Sheet and Drive). If a match is found, the student's attendance is recorded in a CSV file with their name and time of arrival. This page streamlines attendance marking, making it quick and accurate using facial recognition.

```
import os
import pickle
import face_recognition
import cv2
import numpy as np
import csv
from datetime import datetime
from googleapiclient.discovery import build
from google.oauth2.service_account import Credentials
import requests
from io import BytesIO
from PIL import Image

# Authenticate and build the service for Google Sheets API
SCOPES = ['https://www.googleapis.com/auth/spreadsheets.readonly']
SERVICE_ACCOUNT_FILE = r'C:\Users\praga\OneDrive\Desktop\python\service.json' # Use raw string to avoid escape issues

credentials = Credentials.from_service_account_file(
    SERVICE_ACCOUNT_FILE, scopes=SCOPES)
service = build('sheets', 'v4', credentials=credentials)

# Replace with your Google Sheet ID
SPREADSHEET_ID = '1_hawYifDoyZ2HhjY2qaipCo3rNSdYkrzfz1MOMRRpas' # Update with your Google Sheet ID
RANGE_NAME = 'Sheet1!A:F' # Ensure this range includes Name, USN, Email, Gender, Department, Photo URL

# Call the Sheets API to get the data
sheet = service.spreadsheets()
result = sheet.values().get(spreadsheetId=SPREADSHEET_ID, range=RANGE_NAME).execute()
values = result.get('values', [])

if not values:
    print("No data found.")
else:
    known_face_encodings = []
    known_face_names = []
    student_data = {}

    # Fetch data from the sheet
    # Fetch data from the sheet
    for row in values:
        if len(row) >= 6: # Ensure there are enough columns
            name = row[0]
            usn = row[1]
            email = row[2]
            gender = row[3]
            department = row[4]
            photo_url = row[5]

            # Skip empty or invalid photo URLs
            if not photo_url or not photo_url.startswith("https://"):
                continue # Skip processing this row

            # Download the image from the URL
            try:
                response = requests.get(photo_url)
```

```

# Download the image from the URL
try:
    response = requests.get(photo_url)
    response.raise_for_status() # Raise error for bad responses
    img_data = BytesIO(response.content)
    img = Image.open(img_data).convert("RGB") # Ensure RGB format
    img = np.array(img)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_RGB2BGR) # Convert to BGR for OpenCV

# Get the encoding for the image
encoding = face_recognition.face_encodings(img_rgb)
if encoding:
    known_face_encodings.append(encoding[0])
    known_face_names.append(name)
    student_data[name] = {'usn': usn, 'email': email, 'gender': gender, 'department': department}
except Exception:
    # Handle any errors silently to avoid terminal clutter
    continue

# Initialize webcam for face recognition
video_capture = cv2.VideoCapture(0)
if not video_capture.isOpened():
    print("Error: Could not open webcam.")
    exit()

now = datetime.now()
current_date = now.strftime("%Y-%m-%d")

# Create/Append to CSV file
f = open(f"{current_date}.csv", "a", newline="")
lnwriter = csv.writer(f)

students = known_face_names.copy()

while True:
    _, frame = video_capture.read()
    if not _:
        print("Error: Failed to capture frame from webcam.")
        break

    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)

    # Recognize faces
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    for face_encoding in face_encodings:
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        face_distance = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distance)

        name = None
        if matches[best_match_index]:

```

```

5
6     name = None
7     if matches[best_match_index]:
8         name = known_face_names[best_match_index]
9
10    # Attendance
11    if name and name in students:
12        font = cv2.FONT_HERSHEY_SIMPLEX
13        bottomLeftCornerOfText = (10, 100)
14        fontScale = 1.5
15        fontColor = (255, 0, 0)
16        thickness = 3
17        lineType = 2
18
19        cv2.putText(frame, name + " Present", bottomLeftCornerOfText, font, fontScale, fontColor, thickness, lineType)
20        students.remove(name)
21        current_time = datetime.now().strftime("%H-%M-%S")
22        student_info = student_data.get(name, {})
23        lnwriter.writerow([name, student_info.get('usn'), student_info.get('email'), student_info.get('gender'),
24                           student_info.get('department'), current_time])
25
26    cv2.imshow("Attendance", frame)
27    if cv2.waitKey(1) & 0xFF == ord("q"):
28        break
29
30    # Release resources
31    video_capture.release()
32    cv2.destroyAllWindows()
33    f.close()

```

Attendance Status Page:

The Table/HTML Page displays the attendance records of students in a table format, which is populated from the CSV file that is generated during the attendance marking process. The table includes columns such as Student Name, USN, Email, Gender, Department, and Time of Attendance. This page provides an overview of all the students' attendance for the day and allows the teacher to review which students were present. It makes it easier for teachers to monitor and track attendance without having to manually record it, offering a dynamic and organized view of the data.

```
<html lang="en">
<head>
| </style>
</head>
<body>
  <a href="{{ url_for('home') }}" class="back-button"><i class="fa-solid fa-arrow-left"></i></a>
  <header>
    Attendance details
  </header>
  <main>
    <h1>Upload CSV File</h1>
    <p>Select a CSV file to display attendance data</p>
    <input type="file" id="csvFileInput" accept=".csv">

    <table id="csvTable">
      <!-- Table will be populated dynamically -->
    </table>
  </main>
</body>

<script>
const csvFileInput = document.getElementById('csvFileInput');
const csvTable = document.getElementById('csvTable');

csvFileInput.addEventListener('change', (event) => {
  const file = event.target.files[0];

  if (file) {
    const reader = new FileReader();
    reader.onload = (e) => {
      const csvData = e.target.result;

      // Parse CSV data using PapaParse
      Papa.parse(csvData, {
        header: false, // Do not treat the first row as a header
        skipEmptylines: true,
        complete: (result) => {
          displayTable(result.data);
        },
      });
    };
    reader.readAsText(file);
  }
});

function displayTable(data) {
  // Clear existing table content
  csvTable.innerHTML = '';

  if (data.length > 0) {
    // Add fixed headers (Name, Date)
    const headers = ["Name", "USN", "Email", "Gender", "Department", "Date"];
    const thead = document.createElement('thead');
    const headerRow = document.createElement('tr');
```

```

function displayTable(data) {
  // Clear existing table content
  csvTable.innerHTML = '';

  if (data.length > 0) {
    // Add fixed headers (Name, Date)
    const headers = ["Name", "USN", "Email", "Gender", "Department", "Date"];
    const thead = document.createElement('thead');
    const headerRow = document.createElement('tr');

    headers.forEach((header) => {
      const th = document.createElement('th');
      th.textContent = header;
      headerRow.appendChild(th);
    });

    thead.appendChild(headerRow);
    csvTable.appendChild(thead);

    // Create table body
    const tbody = document.createElement('tbody');

    data.forEach((row) => {
      const tr = document.createElement('tr');
      row.forEach((cell) => {
        const td = document.createElement('td');
        td.textContent = cell;
        tr.appendChild(td);
      });
      tbody.appendChild(tr);
    });

    csvTable.appendChild(tbody);
  } else {
    csvTable.innerHTML = '<tr><td colspan="100%" class="no-data">No data available</td></tr>';
  }
}
</script>
</body>
</html>

```

Backend code – Flask:

```

from flask import Flask, render_template, jsonify, request
import subprocess

# Initialize Flask app
app = Flask(__name__)
@app.route('/')
def login():
    return render_template('login.html')
@app.route('/home')
def home():
    return render_template("home.html")

# Route for Student Registration
@app.route('/registration')
def registration():
    return render_template('registration.html')

@app.route("/mark", methods=["POST"])
def mark_attendance():
    try:
        # Run recognition.py as a subprocess
        result = subprocess.run(["python", "recognition.py"], capture_output=True, text=True)

        # Check if the script ran successfully
        if result.returncode == 0:
            return jsonify({"message": "Attendance marked successfully."})
        else:
            return jsonify({"message": f"Error in recognition script: {result.stderr}"}, 500)
    except Exception as e:
        return jsonify({"message": f"An error occurred: {str(e)}"}, 500)

@app.route('/table')
def table():
    return render_template('table.html')

if __name__ == "__main__":
    app.run(debug=True)

```

CHAPTER-6

RESULTS

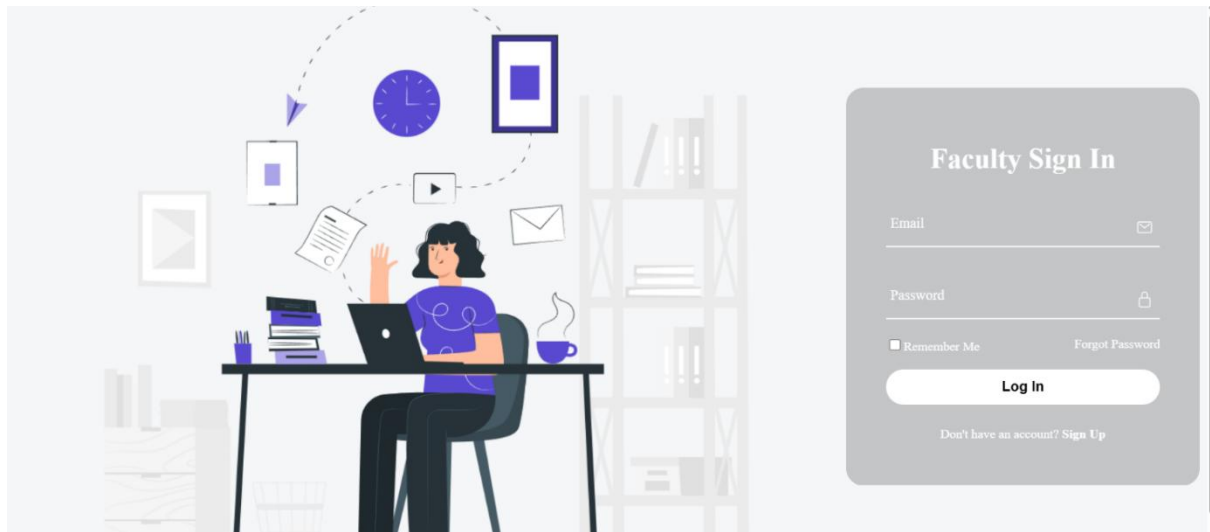


Fig 6.1 The Login Page.

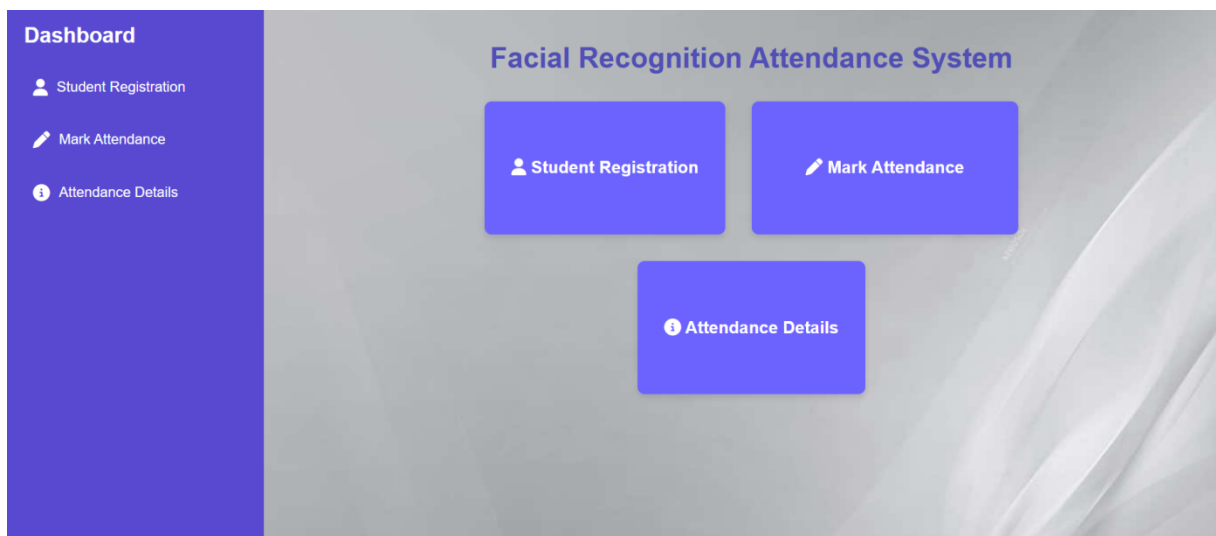
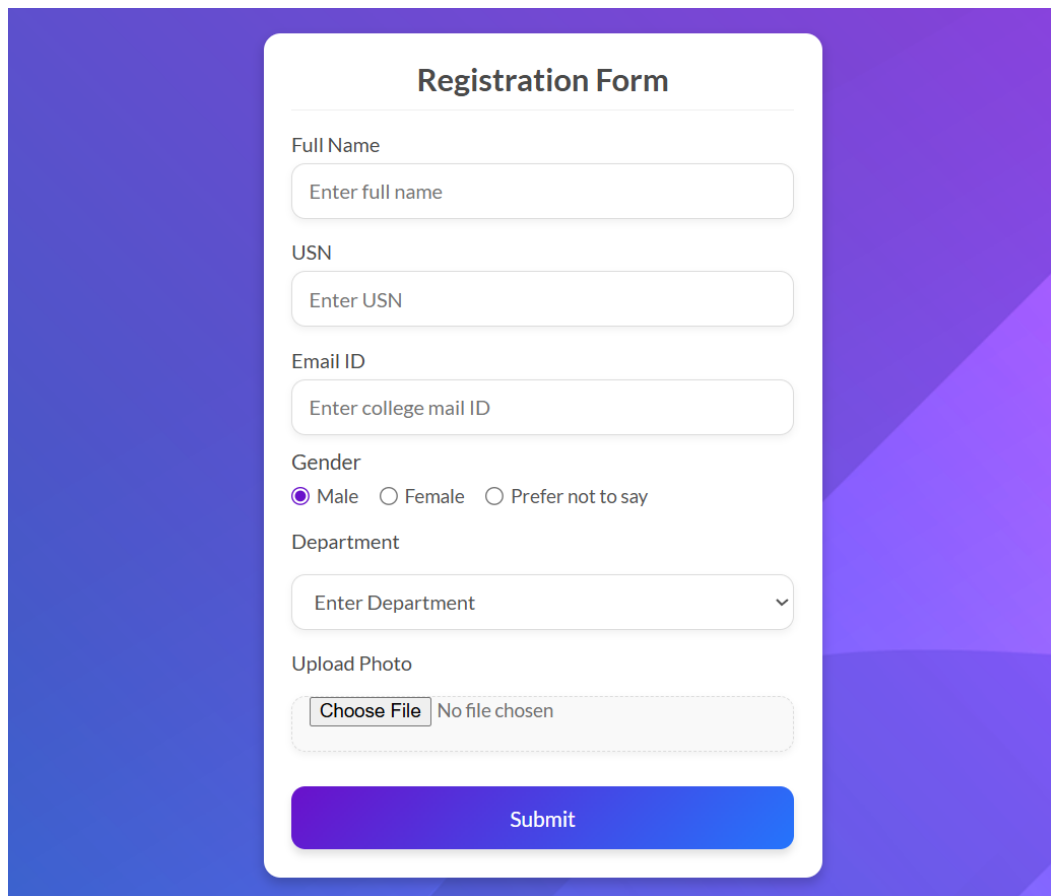


Fig 6.2 The Home Page.



Registration Form

Full Name

USN

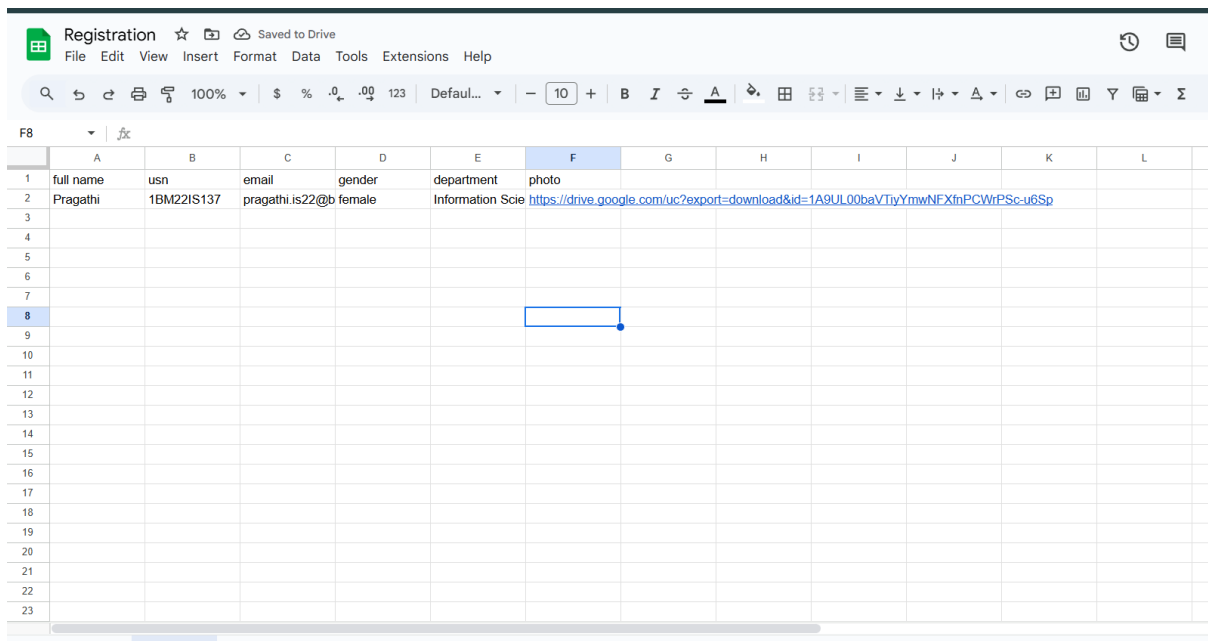
Email ID

Gender
☒ Male ☐ Female ☐ Prefer not to say

Department

Upload Photo
 No file chosen

Fig 6.3 Student Registration Page



Registration ☆ Saved to Drive
 File Edit View Insert Format Data Tools Extensions Help

100% 123 Default... 10 B I A

	A	B	C	D	E	F	G	H	I	J	K	L
1	full name	usn	email	gender	department	photo						
2	Pragathi	1BM22IS137	pragathi.is22@b	female	Information Scie	https://drive.google.com/uc?export=download&id=1A9UL00baVTiyYmwNFXinPCWtPSc-uj6Sp						
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												

Fig 6.4 Csv File where Student Details is stored

Upload CSV File

Select a CSV file to display attendance data

Choose File 2025-01-05.csv

Name	USN	Email	Gender	Department	Date
Pragathi	1BM22IS137	pragathi.is22@bmsce.ac.in	female	Information Science and Engineering	21-37-19

Fig 6.5 Attendance Status Page

CHAPTER-7

CONCLUSION

The Facial Recognition Attendance System provides an efficient, reliable, and automated solution to the challenges faced by traditional attendance management methods. By leveraging advancements in artificial intelligence and computer vision, this system eliminates the need for manual intervention, reduces errors, and ensures a secure and hygienic process for attendance tracking.

The implementation of facial recognition technology offers significant advantages, such as high accuracy, contactless operation, and scalability, making it suitable for use in various organizations, including educational institutions, corporate offices, and events. This project successfully demonstrates how modern technology can be harnessed to solve real-world problems, improving operational efficiency and enhancing user experience.

While the system achieves its primary objectives, such as automating attendance and preventing proxy attendance, it also highlights areas for further improvement, including handling dynamic environments, improving recognition accuracy in diverse lighting conditions, and addressing privacy concerns. Future enhancements could incorporate advanced features like cloud-based storage, integration with existing organizational systems, and multi-factor authentication for added security.

CHAPTER-8

REFERENCES

- R. Habu, S. Motade, S. Kukade, K. Gunale and A. Nair, "Smart Face Recognition Based Attendance System Using ML Algorithm," *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, Greater Noida, India, 2022, pp. 527-532, doi: 10.1109/ICAC3N56670.2022.10074166
- <https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python/>
- <https://www.questjournals.org/jses/papers/Vol5-issue-2/D05021829.pdf>

