**A PROJECT REPORT ON**

**“FACE ATTENDANCE SYSTEM”**

**Submitted to the Mangalore University**

**in partial fulfilment of the requirement**

**for the award of the Degree of**

**BACHELOR OF COMPUTER APPLICATION**

**Submitted by**

**PUNEETH N AIL**

**Register Number:** **U05SD21S0065**

**Under the valuable guidance of**

**MRS. SHRUTHI K**

****

**SHRI DHARMASTHALA MANJUNATESHWARA COLLEGE OF BUSINESS MANAGEMENT**

**MANGALURU – 575003**

**2023–2024**

**Shri Dharmasthala Manjunatheshwara College of**

**Business Management, Mangaluru - 575003**

**(Affiliated to Mangalore University)**

****

**CERTIFICATE**

This is to certify that the project report entitles **“FACE ATTENDANCE SYSTEM”** is an authenticated record of the project work carried out by **Mr. Puneeth N Ail** bearing Register Number **U05SD21S0065** in partial fulfilment of the requirement for the award of **Bachelor’s Degree in Computer Application of Mangalore University** under the guidance and supervision during the year 2023-2024.

**Forwarded to Principal for Approval**

Place: Mangaluru (Mrs. Shruthi K)

Date: June 1, 2024 Project guide

**Approved and forwarded to Mangalore University**

Place: Mangaluru (Prof. Aruna P Kamath)

Date: June 1, 2024 Principal

**Signature of the Examiners:**

**1………………………………… 2……………………………….**

**DECLARATION**

I hereby declare that the project report titled as **“FACE ATTENDANCE SYSTEM”** has been prepared by me during the year 2023– 2024 under the valuable guidance and supervision of **Mrs. Shruthi K**, Assistant Professor and project guide, SDM College of Business Management, Mangaluru, in partial fulfilment of the requirement for the award of degree in Bachelor of Computer Application from Mangalore University for the academic year 2023-2024.

I also declare that this project is the result of my own effort and has not been submitted to any other University for the award of any degree or diploma.

Place: Mangaluru **…………………………**

Date: June 1, 2024 Mr. Puneeth N Ail

Reg No : U05SD21S0065

**ACKNOWLEDGEMENT**

Whatever I am able to put forward in terms of outwork, is only due to few people from whom I have learnt. Thus, it is a great pleasure in mentioning their names.

Working on the live project was very interesting and really enhanced our knowledge.

I would like to express our sincere gratitude to our beloved **Principal Mrs. Aruna P Kamath** without whose permission I would not be able to do our project and for taking keen interest for the students of BCA in providing useful guidelines and giving all the necessary facilities.

I extend thanks to our guide **Mrs. Shruthi K,** Assistant Professor of Computer Application Department for her valuable guidance and constant encouragement, which helped me successfully completing our project.

Finally, I extend thanks to our parents and friends who were directly or indirectly involved in the completion of our project.

Place: Mangaluru **…………………………**

Date: June 1, 2024 Mr. Puneeth N Ail

Reg No : U05SD21S0065

**INDEX**

**Chapter 1: Synopsis (Page 1 - 4)**

1.1 Title of the Project

1.2 Introduction

1.2.1 Objective and scope of the project

1.3 Project Category

1.4 Tools / Category

1.5 Hardware and Software Requirements

1.5.1 Hardware requirements

1.5.2 Software requirements

1.6 Languages used

1.7 Module Description

1.8 Limitation of the project

1.9 Future scope of the project

**Chapter 2: Database Design (Page 5 - 6)**

2.1 Introduction

2.2 Purpose

2.3 Scope

2.4 Database Tables

**Chapter 3: Implementation (Page 7 - 31)**

3.1 Introduction

3.2 Code

**Chapter 4: Snapshots (Page 32 – 36)**

Contents

**Chapter 5: Testing (Page 37 – 41)**

5.1 Introduction

5.2 Objective of testing

5.3 Testing Methods

5.4 Testing Steps

5.4.1 Unit Testing:

5.4.2 Integration Testing:

5.4.3 Validation Testing:

5.4.4 Output Testing:

5.4.5 User acceptance Testing:

5.5 Test cases:

5.5.1 Testing for login

5.5.2 Testing for taking attendance

5.5.3 Testing for Attendance tracker sheet

5.5.4 Testing for adding student

5.5.5 Testing for student deletion

5.5.6 Testing for forget password

**Conclusion (Page 42)**

**Bibliography / References (Page 43)**

**Chapter 1**

**SYNOPSIS**

**1.1 Title of the Project**

**FACE ATTENDANCE SYSTEM**

**1.2 Introduction**

A face attendance model is a technological system that utilizes facial recognition technology to track and record attendance automatically. This system captures and analyzes facial features to identify individuals and mark their attendance, eliminating the need for manual sign-ins or traditional card-based systems. By leveraging advanced machine learning algorithms and computer vision techniques, the face attendance model offers a user-friendly, secure, and efficient solution for attendance management. This technology can be deployed in various environments such as schools, offices, conferences, and events to ensure accurate and real-time attendance tracking.

**1.2.1 Objectives and scope of the Project.**

The objective of the face attendance model is to streamline the attendance tracking process, make it more convenient for users to mark their presence, improve the efficiency and accuracy of attendance management, and enhance security and compliance. The face attendance model offers several benefits to both the organization and its users. For example, users can have their attendance automatically recorded through facial recognition, which saves time and effort compared to traditional methods like manual sign-ins or card swipes.

**1.3 Project category:**

“Web Application Development and Backend Systems" with a focus on "Automated Attendance Management using Python and Facial Recognition Technology"

**1.4 Tools / Platform**

* **Visual Studio Code**

One of the key features of VS Code is its extensive marketplace of extensions, which allows developers to customize and extend the functionality of the editor. There are thousands of extensions available, covering everything from syntax highlighting and code formatting to version control and testing frameworks.

* **PYTHON**

Python is a popular high-level programming language used for a wide range of applications, including web development, data analysis, artificial intelligence, and scientific computing. It was created by Guido van Rossum and first released in 1991. Python's design philosophy emphasizes code readability and simplicity, which makes it an excellent choice for both beginners and experienced developers. Python is known for its clear and concise syntax, which mimics natural language, making it easy to learn and write. As an interpreted language, Python executes code line by line, facilitating quick testing and debugging. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

* **FLASK**

Flask is a popular micro web framework for Python, used for developing web applications. It was created by Armin Ronacher and released in 2010. Flask is designed to be simple and lightweight, providing the essential tools to build web applications without imposing extensive dependencies or complexities. Flask follows a minimalist approach, giving developers the flexibility to choose the components they need and integrate third-party libraries as required. It is based on the WSGI (Web Server Gateway Interface) protocol and Jinja2 template engine, ensuring robust performance and a clean separation of logic and presentation.

* **XAMPP**

XAMPP is a free and open-source software package that provides a complete web server solution for developers. It includes several components that are necessary to run a web server, including the Apache web server, the MySQL database server. One of the main advantages of XAMPP is its simplicity and ease of use. XAMPP also includes several useful tools and features, such as phpMyAdmin for managing databases, and FileZilla for FTP file transfers. Another advantage of XAMPP is its flexibility. Overall, XAMPP is a powerful and versatile web server solution that continues to be widely used by developers around the world

* **My SQL**

MySQL is the world’s most widely used open-source Relational Database Management System (RDBMS) that runs as a server providing multi-user access to a number of databases. The MySQL development project has made it source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is a popular choice of database for use in web applications, and is a central development of the widely used LAMP open-source web application software stack..

**1.5.1 Hardware Requirements**

* RAM : 8 GB or above
* Hard disk : 10 GB or above
* Processor : 2.4 GHZ or above

**1.5.2 Software Requirements**

* Packages: Flask, cv2, Subprocess
* IDE: Visual Studio Code
* Database : My SQL
* Languages : HTML, CSS, Python , Java Script
* Server: Flask, Xamp

**1.6 Languages used**

* Python
* Html
* CSS
* Javascript

**1.7 Module description**

**Login:** Users can enter the website using username and password.

**Admin DashBoard**: In this page admin can add user and also reset password of the user

**Home**: In this page, all the operations are present. You can choose which one you want to do at that moment.

**Take Attendance**: This module is used to take attendance using webcam.

**View Attendance**: This module is used to view attendance.

**Add Students**: This model is used to add faces and train faces and create encodings of and used to Recognize faces and mark attendance.

**Remove Students** : This module is used to delete the data of a particular student.

**1.8 Limitation of the project**

* Flask is a package used in this project which not fit for production

**1.9 Future scope and further enhancement of the project.**

* Enhance security and accuracy by integrating the attendance system with biometric authentication methods such as fingerprint or iris scanning.
* Develop a mobile application version of the attendance system, allowing users to mark their attendance remotely using their smartphones.
* Implement machine learning algorithms to analyze attendance patterns and predict future attendance, helping administrators anticipate staffing needs and optimize resources

**Chapter 2:**

**DATABASE DESIGN**

**2.1 Introduction:**

Database description describes all the databases used in the software to store all the records. The database in turn is further described in detail giving all the fields used with their data type, constraints available to them and description. Constraints include primary key, foreign key, etc., which allow the entities to be uniquely identified

In this database description we describe all databases which are used to store all the records of the Face Attendance System.

**2.2 Purpose**:

* Database description describes the entire database used in the software to store all the records.
* Database design is the process of producing a detailed data model of a database. Database design is a collection of related data.
* This document describes standards to use when designing and developing the database.

**2.3 Scope:**

A good database is one that is simple to understand and well planned. The database doesn’t have redundant tables. One can use ER Diagram (Entity Relationship Diagrams) in order to make a good database. This database design is used to understand the software hotel KOT and billing

* Organize the system into modules
* Organize sub-modules for each module
* Allocate tasks to processors
* Choose an approach to manage data store
* Handle access to global resources
* Choose implementation logic.

**2.4 Database Tables:**

**Table Name**: USER

**Description:** used for user login

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Length** | **Constraints** |
| username | Varchar | 45 | Primary Key |
| password | Varchar | 45 | Not Null |

**Table Name**: admin

**Description:** used for user login

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Length** | **Constraints** |
| adminusername | Varchar | 45 | Primary Key |
| adminpassword | Varchar | 45 | Not Null |

**Table Name**: studentdetails

**Description:** used for user Student Info

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Length** | **Constraints** |
| roll\_no | Integer | 10 | Primary key |
| name | Varchar | 45 | Not Null |
| class | Varchar | 45 | Not Null |

**Table Name**: attendance

**Description:** used for mark attendance

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | **Data Type** | **Length** | **Constraints** |
| roll\_no | Integer | 10 | Not null |
| name | Varchar | 45 | Not Null |
| Date | Varchar | 45 | Not Null |
| Time | Varchar | 45 | Not Null |
| class | Varchar | 45 | Not Null |

**CHAPTER - 3**

**IMPLEMENTATION**

**3.1 Introduction:**

Implementation refers to the process of executing a plan or design to bring a project to

fruition. In the context of software development, implementation involves translating the

requirements and design into actual code and systems that perform the desired tasks. This phase

includes:

**1. Coding:** Writing the program code using a specific programming language and

adhering to the design specifications.

**2. Integration:** Combining various components and modules into a functioning system.

**3. Testing:** Conducting various tests to ensure that the software works correctly and meets the requirements.

**4. Deployment:** Installing the software in the intended environment for use by the endusers.

**5. Documentation:** Creating user manuals, technical documentation, and other materials to support the software.

**6. Maintenance:** Ongoing support and updates to fix bugs, improve performance, and add new features.

In summary, implementation is the stage where theoretical plans are put into action to create a working system or product.

**3.2 Code:**

**app.py**

from flask import Flask, render\_template, request, flash, redirect, url\_for, session, jsonify

import subprocess

from mysql.connector import connection, Error

from datetime import datetime, date

import pandas as pd

import time

# from werkzeug.security import generate\_password\_hash, check\_password\_hash

app = Flask(\_\_name\_\_)

app.secret\_key = 'your\_secret\_key'

@app.route('/')

def index():

    if 'username' not in session or session.get('role') != 'user':

         return redirect(url\_for('login'))

    return render\_template('index.html')

@app.route('/login', methods=['GET', 'POST'])

def login():

    if request.method == 'POST':

        username = request.form['username']

        password = request.form['password']

        role = request.form['role']

        connection = create\_connection()

        if connection is None:

            flash('Database connection failed', 'error')

            return redirect(url\_for('login'))

        try:

            cursor = connection.cursor(dictionary=True)

            if role == 'admin':

                query = "SELECT \* FROM admin WHERE adminusername = %s AND adminpassword = %s"

            else:

                query = "SELECT \* FROM users WHERE username = %s AND password = %s"

            cursor.execute(query, (username, password))

            user = cursor.fetchone()

            if user:

                session['username'] = username

                session['role'] = role

                if role == 'admin':

                    return redirect(url\_for('admin\_dashboard'))

                else:

                    return redirect(url\_for('index'))

            else:

                flash('Invalid credentials', 'error')

        except Error as e:

            flash(f"Database error: {e}", 'error')

        finally:

            cursor.close()

            connection.close()

        return redirect(url\_for('login'))

    return render\_template('login.html')

@app.route('/logout')

def logout():

    session.pop('username', None)

    session.pop('role', None)

    return redirect(url\_for('login'))

@app.route('/admin\_dashboard')

def admin\_dashboard():

    if 'username' not in session or session.get('role') != 'admin':

        return redirect(url\_for('login'))

    return render\_template('admin\_dashboard.html')

def create\_connection():

    """ create a database connection to the MySQL database """

    try:

        conn = connection.MySQLConnection(user='root', password='', host='127.0.0.1', database='attendance')

        if conn.is\_connected():

            print("Connection to MySQL DB successful")

    except Error as e:

        print(f"The error '{e}' occurred")

    return conn

@app.route('/add\_user', methods=['POST'])

def add\_user():

    if 'username' not in session or session.get('role') != 'admin':

        return redirect(url\_for('login'))

    new\_username = request.form['new\_username']

    new\_password = request.form['new\_password']

    connection = create\_connection()

    if connection is None:

        flash('Database connection failed', 'error')

        return redirect(url\_for('admin\_dashboard'))

    try:

        cursor = connection.cursor()

        # Check if user already exists in both user and admin tables

        query = "SELECT \* FROM users WHERE username = %s UNION SELECT \* FROM admin WHERE adminusername = %s"

        cursor.execute(query, (new\_username, new\_username))

        user = cursor.fetchone()

        if user:

            flash('User already exists', 'error')

            session['scroll\_position'] = request.args.get('scroll\_position', type=int) or 0

            return redirect(url\_for('admin\_dashboard', scroll\_position=session['scroll\_position']))

        else:

            insert\_query = "INSERT INTO users (username, password) VALUES (%s, %s)"

            cursor.execute(insert\_query, (new\_username, new\_password))

            connection.commit()

            flash('User added successfully', 'success')

    except Error as e:

        flash(f"Database error: {e}", 'error')

    finally:

        cursor.close()

        connection.close()

    flash(f"User '{new\_username}' created successfully!", 'success')

    # return redirect(url\_for('admin\_dashboard'))

    # Redirect to the same position on the page

    session['scroll\_position'] = request.args.get('scroll\_position', type=int) or 0

    return redirect(url\_for('admin\_dashboard', scroll\_position=session['scroll\_position']))

@app.route('/reset\_password', methods=['POST'])

def reset\_password():

    if 'username' not in session or session.get('role') != 'admin':

        return redirect(url\_for('login'))

    reset\_username = request.form['reset\_username']

    new\_password = request.form['reset\_password']

    connection = create\_connection()

    if connection is None:

        flash('Database connection failed', 'error')

        return redirect(url\_for('admin\_dashboard'))

    try:

        cursor = connection.cursor()

        # Check if the user exists

        cursor.execute("SELECT \* FROM users WHERE username = %s", (reset\_username,))

        user = cursor.fetchone()

        if user is None:

            flash(f"User '{reset\_username}' not found", 'error')

        else:

            # Update the user's password

            query = "UPDATE users SET password = %s WHERE username = %s"

            cursor.execute(query, (new\_password, reset\_username))

            connection.commit()

            flash('Password reset successfully', 'success')

    except Error as e:

        flash(f"Database error: {e}", 'error')

    finally:

        cursor.close()

        connection.close()

    # Redirect to the same position on the page

    session['scroll\_position'] = request.args.get('scroll\_position', type=int) or 0

    return redirect(url\_for('admin\_dashboard', scroll\_position=session['scroll\_position']))

@app.route('/backhome')

def backhome():

    return render\_template('index.html', selected\_date='', no\_data=False)

@app.route('/attendance\_view')

def attendance\_view():

    return render\_template('attendance.html', selected\_date='', no\_data=False)

@app.route('/takeattendance')

def takeattendance():

    subprocess.run(["python", "attendance\_taker.py"])

    return render\_template('index.html', no\_data=False)

@app.route('/add\_student')

def add\_student():

    subprocess.run(["python", "get\_faces\_from\_camera\_tkinter.py"])

    return render\_template('index.html', no\_data=False)

@app.route('/delete\_student', methods=['POST'])

def delete\_student():

    rollno = request.form.get('roll\_no')

    name = request.form.get('sname')

    subprocess.run(["python", "delete\_face.py", rollno, name])

    flash("Student Deleted Successfully","success")

    return render\_template('remove.html', no\_data=False, name='', roll\_no='')

@app.route('/get\_attendance\_data')

def get\_attendance\_data():

    conn = create\_connection()

    data = attendance\_details()

    print(data)

    formatted\_data = [(row[0], str(row[1]), str(row[2]), row[3]) for row in data]

    data\_df = pd.DataFrame(formatted\_data, columns=['name', 'time', 'date', 'roll\_no'])

    cursor = conn.cursor()

    cursor.execute("SELECT count(name) as count FROM studentdetails")

    total\_student = cursor.fetchone()[0]

    stcursor = conn.cursor()

    stcursor.execute("SELECT name, roll\_no, COUNT(\*) AS attendance FROM attendance GROUP BY name, roll\_no;")

    all\_day = stcursor.fetchall()

    cursorcount = conn.cursor()

    cursorcount.execute("SELECT COUNT(DISTINCT date) AS unique\_dates\_count FROM attendance")

    total\_att = cursorcount.fetchone()[0]

    conn.close()

    total\_records = total\_student

    present\_count = len(data\_df)

    absent\_count = total\_records - present\_count

    overall = {

        'present': present\_count,

        'absent': absent\_count

    }

    student\_attendance = pd.DataFrame(all\_day, columns=['name', 'roll\_no', 'attendance']).groupby('name')['attendance'].mean() \* 100 / total\_att

    student\_names = student\_attendance.index.tolist()

    student\_percentages = student\_attendance.values.tolist()

    students = {

        'names': student\_names,

        'attendance': student\_percentages

    }

    return jsonify({'overall': overall, 'students': students})

@app.route('/get\_today\_attendance\_data')

def get\_today\_attendance\_data():

    attendance\_data = attendance\_details()

    print(attendance\_data)

    formatted\_data = [(row[0], str(row[1]), str(row[2]), row[3]) for row in attendance\_data]

    today\_attendance = {

        'students': {

            'names': [row[0] for row in formatted\_data],

            'roll\_no': [row[3] for row in formatted\_data]

        }

    }

    return jsonify(today\_attendance)

@app.route('/search\_rollno')

def search\_rollno():

    return render\_template('remove.html', rollno='', name='', no\_data=False)

@app.route('/search\_rollno', methods=['POST'])

def search\_student():

    rollno = request.form.get('rollno')

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT name FROM studentdetails WHERE roll\_no = %s", (rollno,))

    name = cursor.fetchone()

    conn.close()

    if not name:

        return render\_template('remove.html', roll\_no='', name='', no\_data=True)

    return render\_template('remove.html', roll\_no=rollno, name=name[0], no\_data=False)

@app.route('/attendance', methods=['POST'])

def attendance():

    selected\_date = request.form.get('selected\_date')

    selected\_date\_obj = datetime.strptime(selected\_date, '%Y-%m-%d')

    formatted\_date = selected\_date\_obj.strftime('%Y-%m-%d')

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT roll\_no, name, time FROM attendance WHERE date = %s", (formatted\_date,))

    attendance\_data = cursor.fetchall()

    conn.close()

    if not attendance\_data:

        return render\_template('attendance.html', selected\_date=selected\_date, no\_data=True)

    return render\_template('attendance.html', selected\_date=selected\_date, attendance\_data=attendance\_data)

def attendance\_details():

    formatted\_date = datetime.today().strftime('%Y-%m-%d')

    conn = create\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT name, time, date, roll\_no FROM attendance WHERE date = %s", (formatted\_date,))

    attendance\_data = cursor.fetchall()

    conn.close()

    return attendance\_data

@app.route('/classes')

def view\_classes():

    conn = create\_connection()

    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT class, COUNT(\*) as total\_students FROM studentdetails GROUP BY class")

    class\_summaries = cursor.fetchall()

    cursor.close()

    conn.close()

    return render\_template('classes.html', class\_summaries=class\_summaries)

@app.route('/class/<class\_name>')

def view\_class\_details(class\_name):

    conn = create\_connection()

    cursor = conn.cursor(dictionary=True)

    query = """

    SELECT

        s.name,

        s.roll\_no,

        IFNULL((COUNT(a.date) / (SELECT COUNT(DISTINCT date) FROM attendance) \* 100), 0) AS attendance

    FROM

        studentdetails s

    LEFT JOIN

        attendance a ON s.roll\_no = a.roll\_no

    WHERE

        s.class = %s

    GROUP BY

        s.name, s.roll\_no

    """

    cursor.execute(query, (class\_name,))

    class\_details = cursor.fetchall()

    cursor.close()

    conn.close()

    return jsonify(class\_details)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

    <script src="{{ url\_for('static', filename='javascript/functions.js') }}"></script>

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <title>Student Attendance System</title>

    <style>

        .charts-container {

            display: flex;

            justify-content: space-around;

            align-items: center;

            flex-wrap: wrap;

        }

        .chart-wrapper {

            flex: 1;

            margin: 20px;

            min-width: 300px;

        }

    </style>

</head>

<body>

    <div class="sidebar">

        <div class="sidebar-header">

            <h2>SDM</h2>

        </div>

        <ul>

            <li><a href="/">Home</a></li>

            <li><a href="/attendance\_view">Attendance</a></li>

            <!-- <li><a href="#">Students</a></li> -->

            <li><a href="/classes">Classes</a></li>

            <li><a href="/remove">Remove Student</a></li>

            <!-- <li><a href="#">Settings</a></li> -->

        </ul>

    </div>

    <div class="main-content">

        <header>

            <h1>Student Attendance System</h1>

            <a href="/logout" class="logout-button">Logout</a>

        </header>

        <div class="content">

            <div class="card card1" onclick="TakeAttendance()">

                <h1>Take Attendance</h1>

                <button class="btn">Take</button>

            </div>

            <div class="card card2" onclick="viewAttendance()">

                <h1>View Attendance</h1>

                <button class="btn">View</button>

            </div>

            <div class="card card3" onclick="Addstudent()">

                <h1>Add Student</h1>

                <button class="btn">Add</button>

            </div>

            <div class="card card4" onclick="delete\_student()">

                <h1>Remove Student</h1>

                <button class="btn">Remove</button>

            </div>

        </div>

        <div class="analytics charts-container">

            <div class="chart-wrapper">

                <canvas id="todayAttendancePieChart" width="600" height="250"></canvas>

            </div>

            <div class="chart-wrapper">

                <canvas id="overallAttendanceChart" width="400" height="200"></canvas>

            </div>

        </div>

        <div id="loading-screen">

            Loading, please wait...

        </div>

    </div>

    <script>

        Promise.all([

            fetch('/get\_today\_attendance\_data').then(response => response.json()),

            fetch('/get\_attendance\_data').then(response => response.json())

        ]).then(([todayData, overallData]) => {

            // Alert message for successful data retrieval

            // alert('Attendance data fetched successfully.');

            // Extract student names and roll numbers for today's attendance

            const todayStudents = todayData.students.names;

            const todayRollNos = todayData.students.roll\_no;

            // Extract student names and attendance percentages for overall attendance

            const overallStudents = overallData.students.names;

            const overallPercentages = overallData.students.attendance;

            // Calculate today's attendance percentage

            const presentStudentsCount = todayRollNos.length;

            const totalStudentsCount = overallStudents.length; // Use the length of overallStudents to get the total number of students

            const absentStudentsCount = totalStudentsCount - presentStudentsCount;

            // Create a pie chart for today's attendance

            const todayPieCtx = document.getElementById('todayAttendancePieChart').getContext('2d');

            const todayPieChart = new Chart(todayPieCtx, {

                type: 'pie',

                data: {

                    labels: ['Present', 'Absent'],

                    datasets: [{

                        label: 'Today\'s Attendance',

                        data: [presentStudentsCount, absentStudentsCount],

                        backgroundColor: ['rgba(75, 192, 192, 0.2)', 'rgba(255, 99, 132, 0.2)'],

                        borderColor: ['rgba(75, 192, 192, 1)', 'rgba(255, 99, 132, 1)'],

                        borderWidth: 1

                    }]

                },

                options: {

                    responsive: false,

                    plugins: {

                        title: {

                            display: true,

                            text: 'Today\'s Attendance'

                        }

                    }

                }

            });

            // Create a bar chart for overall attendance

            const overallCtx = document.getElementById('overallAttendanceChart').getContext('2d');

            const overallChart = new Chart(overallCtx, {

                type: 'bar',

                data: {

                    labels: overallStudents,

                    datasets: [{

                        label: 'Overall Attendance Percentage',

                        data: overallPercentages,

                        backgroundColor: 'rgba(75, 192, 192, 0.2)',

                        borderColor: 'rgba(75, 192, 192, 1)',

                        borderWidth: 1

                    }]

                },

                options: {

                    scales: {

                        y: {

                            beginAtZero: true,

                            title: {

                                display: true,

                                text: 'Attendance Percentage'

                            }

                        }

                    }

                }

            });

        }).catch(error => {

            // Alert message for error in data retrieval

            alert('Error fetching attendance data: ' + error.message);

        });

    </script>

</body>

</html>

**admin\_dashboard.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

<link rel="stylesheet" href="{{ url\_for('static', filename='css/admindashboard.css') }}">

<title>Admin Dashboard</title>

</head>

<body>

<div class="sidebar">

<div class="sidebar-header">

<h2>SDM</h2>

</div>

<ul>

<li><a href="#" onclick="toggleForm('add-user-form')">Add User</a></li>

<li><a href="#" onclick="toggleForm('password-reset-form')">Password Reset</a></li>

</ul>

</div>

<div class="main-content">

<header>

<h1>Admin Dashboard</h1>

<a href="/logout" class="logout-button">Logout</a>

</header>

<div class="container">

{% with messages = get\_flashed\_messages(with\_categories=true) %}

{% if messages %}

<ul class="flashes">

{% for category, message in messages %}

<li class="{{ category }}">{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

<div id="add-user-form" class="form-container">

<h2>Add User</h2>

<!-- Add User Form -->

<form action="{{ url\_for('add\_user') }}" method="post">

<label for="new\_username">New Username:</label>

<input type="text" id="new\_username" name="new\_username" required><br>

<label for="new\_password">New Password:</label>

<input type="password" id="new\_password" name="new\_password" required><br>

<button type="submit">Add User</button>

</form>

</div>

<div id="password-reset-form" class="form-container">

<!-- Forget Password Form -->

<h2>Password Reset</h2>

<form action="{{ url\_for('reset\_password') }}" method="post">

<label for="reset\_username">Username:</label>

<input type="text" id="reset\_username" name="reset\_username" required><br>

<label for="reset\_password">New Password:</label>

<input type="password" id="reset\_password" name="reset\_password" required><br>

<button type="submit">Reset Password</button>

</form>

</div>

</div>

</div>

<script>

function toggleForm(formId) {

const forms = document.querySelectorAll('.form-container');

forms.forEach(form => {

if (form.id === formId) {

form.style.display = 'block';

} else {

form.style.display = 'none';

}

});

// Hide the other form

const otherFormId = (formId === 'add-user-form') ? 'password-reset-form' : 'add-user-form';

document.getElementById(otherFormId).style.display = 'none';

}

</script>

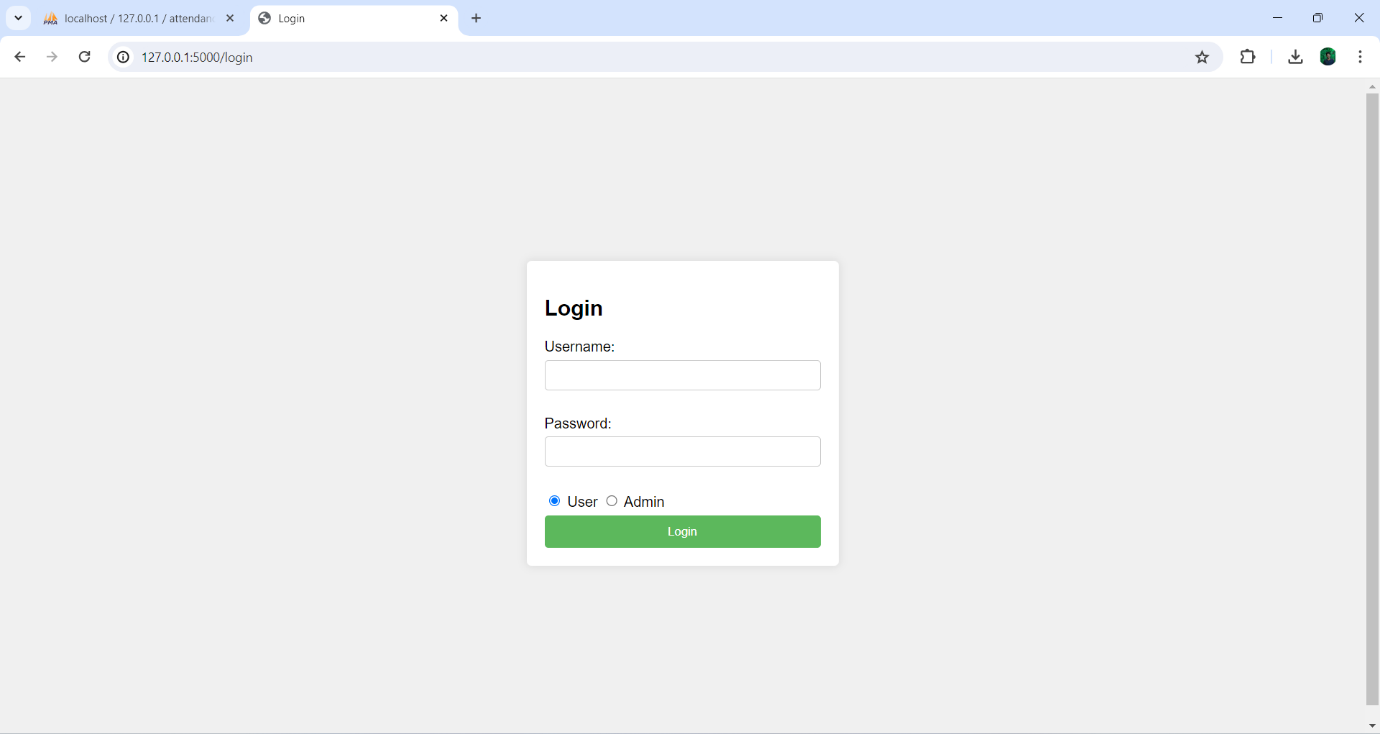
</body>

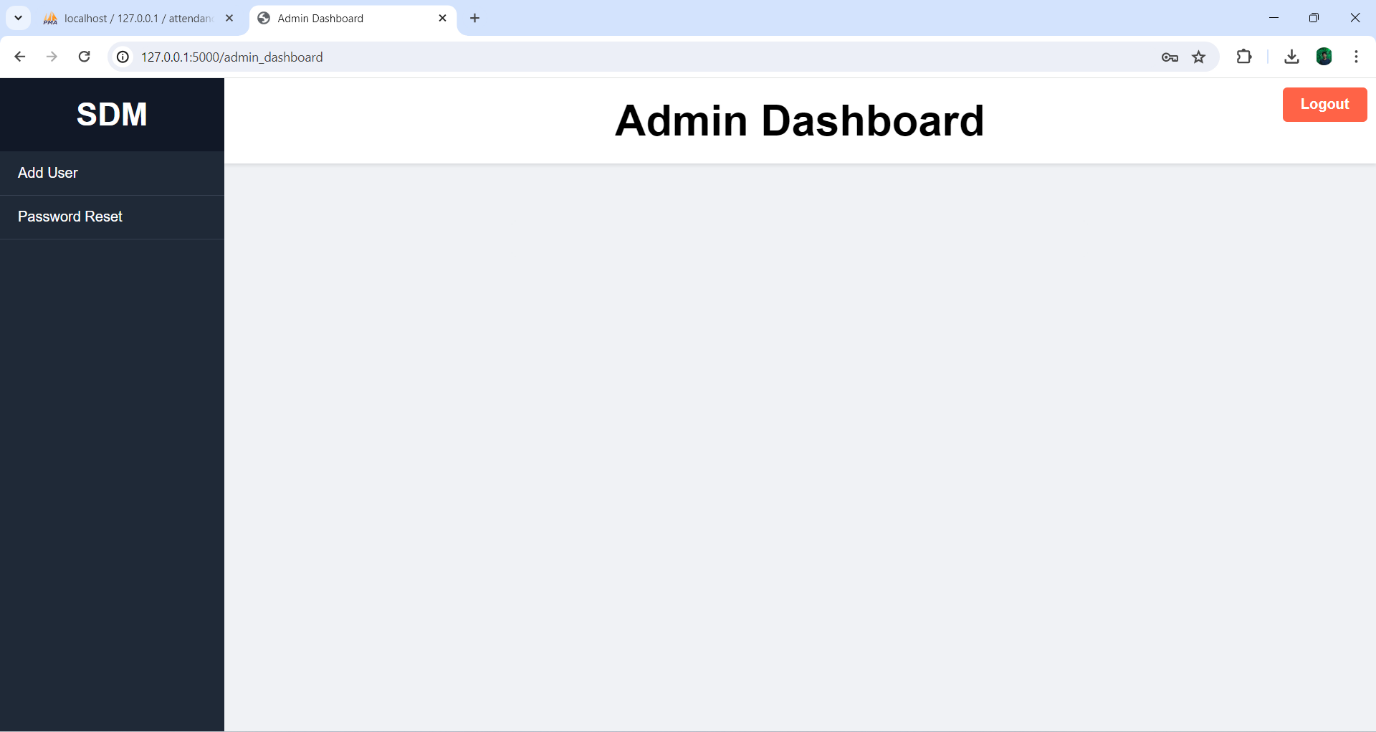
</html>

**CHAPTER - 4**

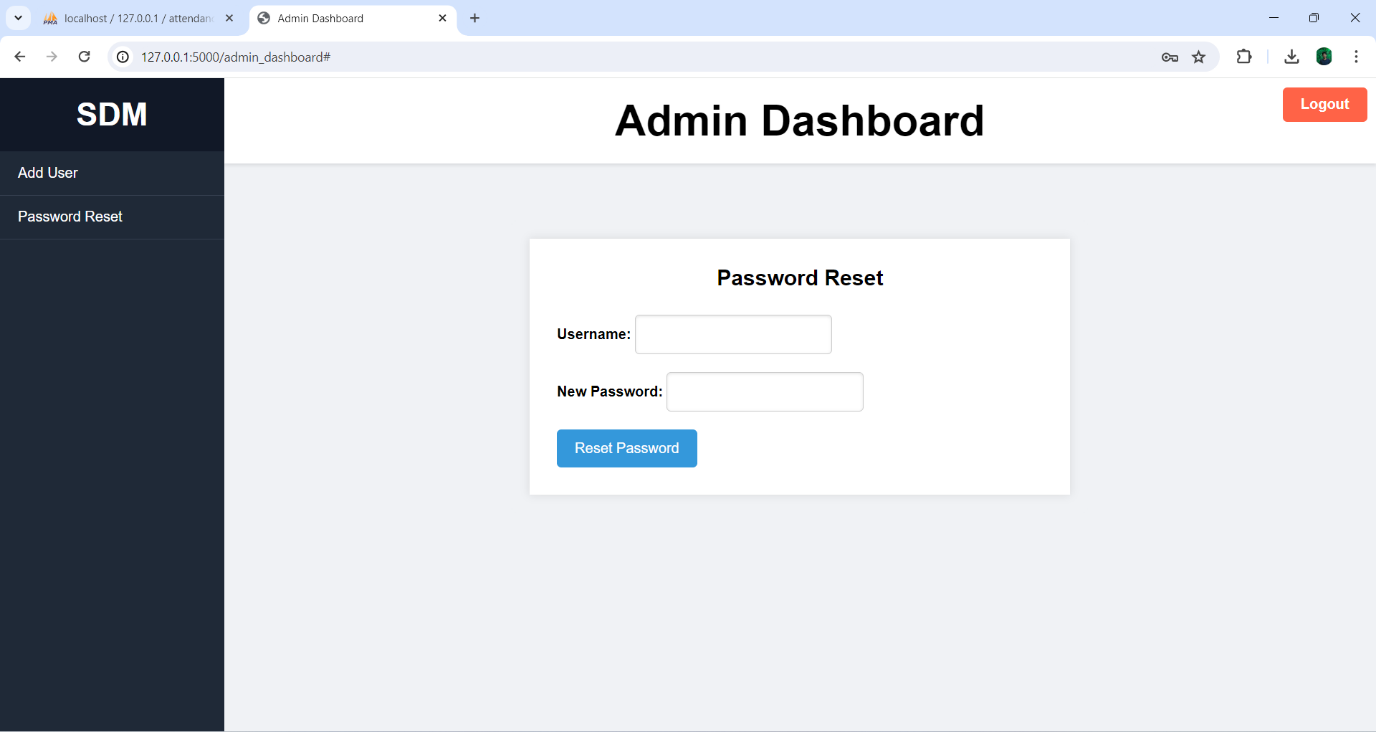
**SNAPSHOTS**

**Login page:**

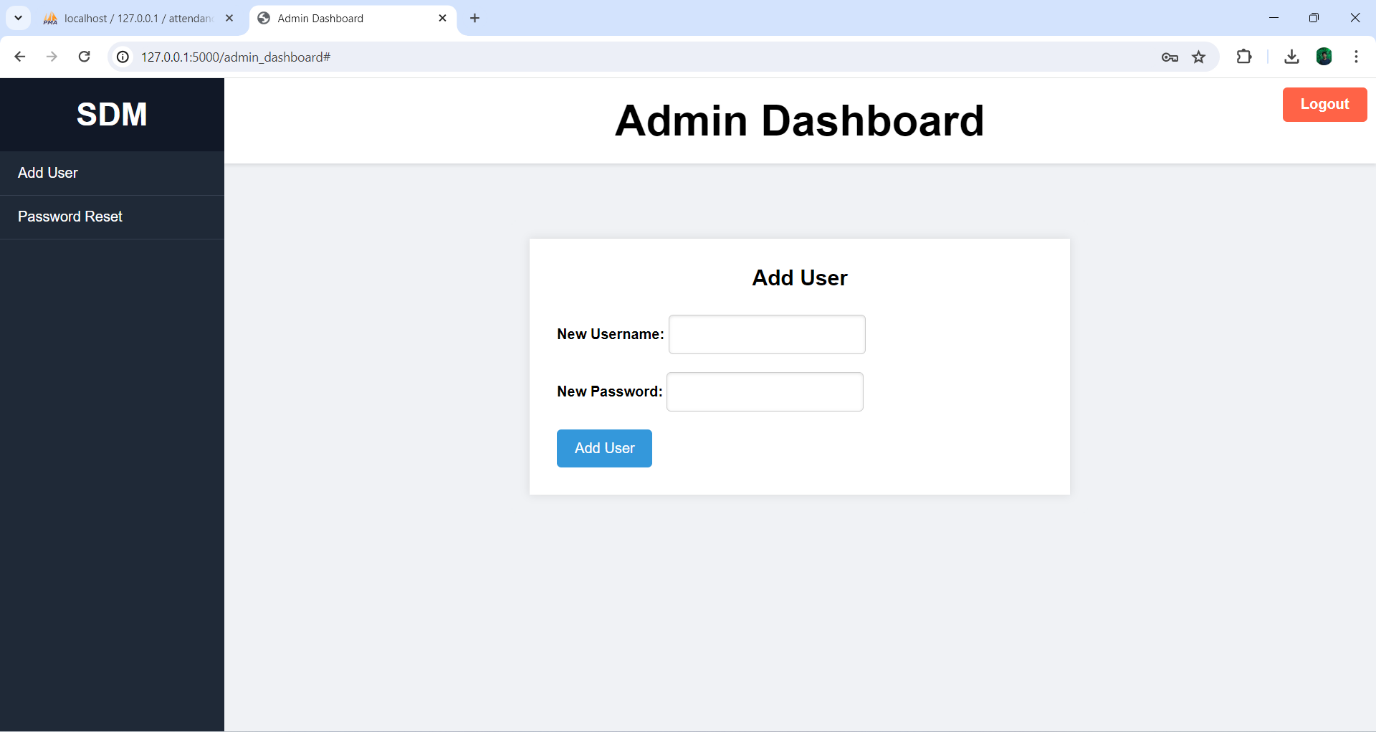


**Admin Dashboard:**

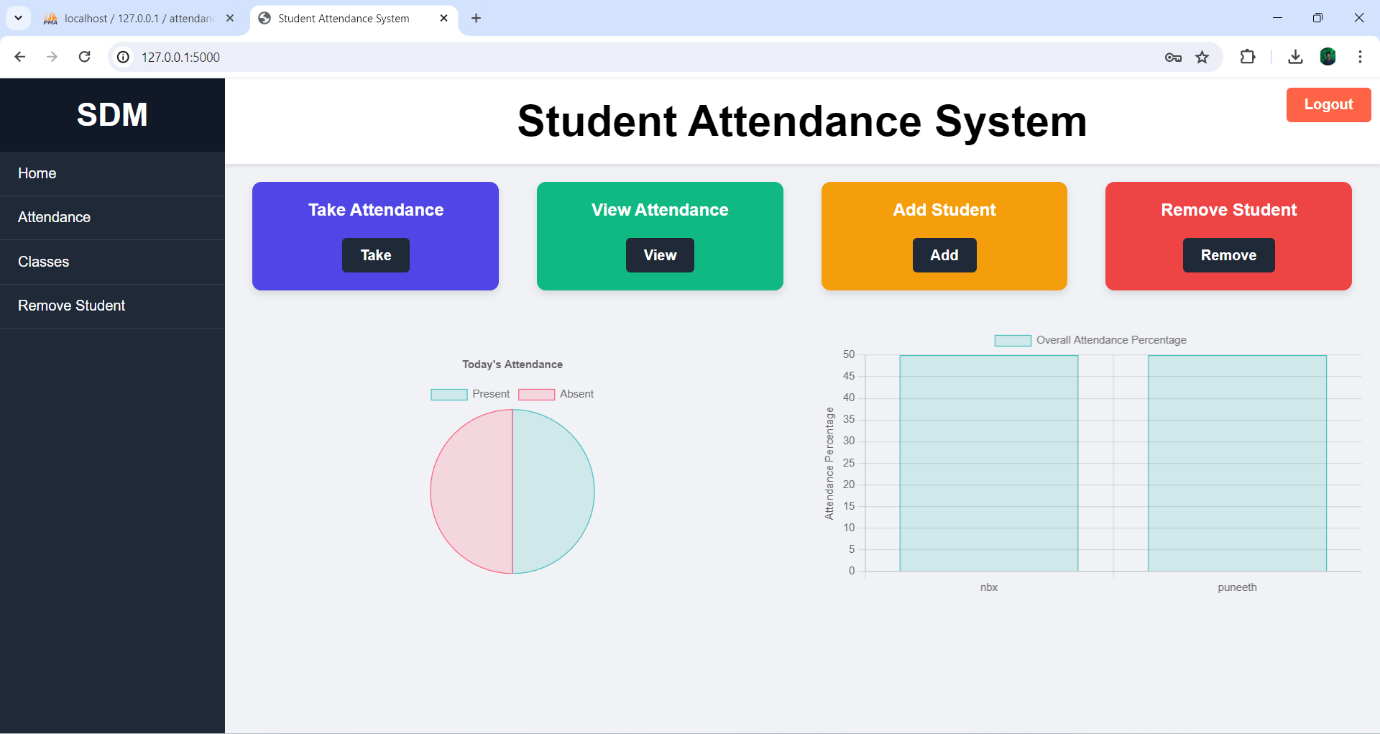
**User Password reset:**

****

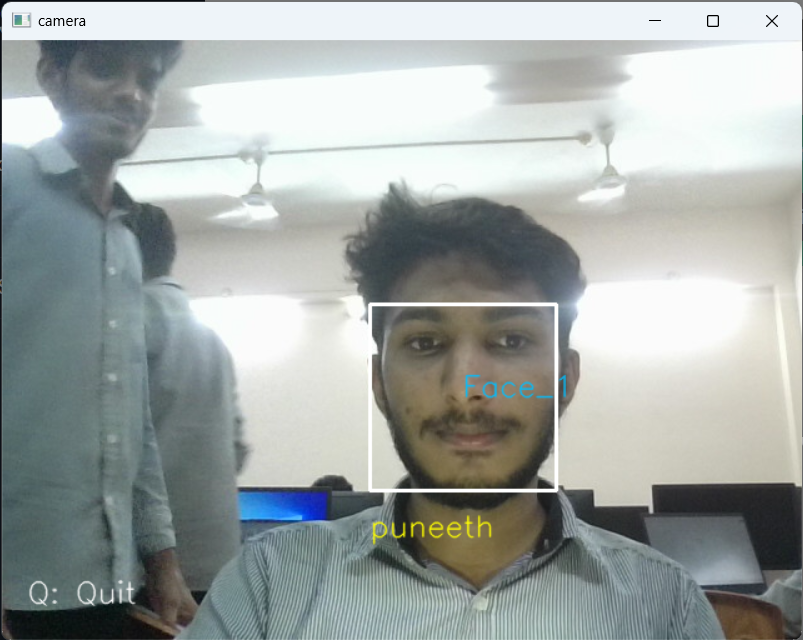
**Add Admin:**

****

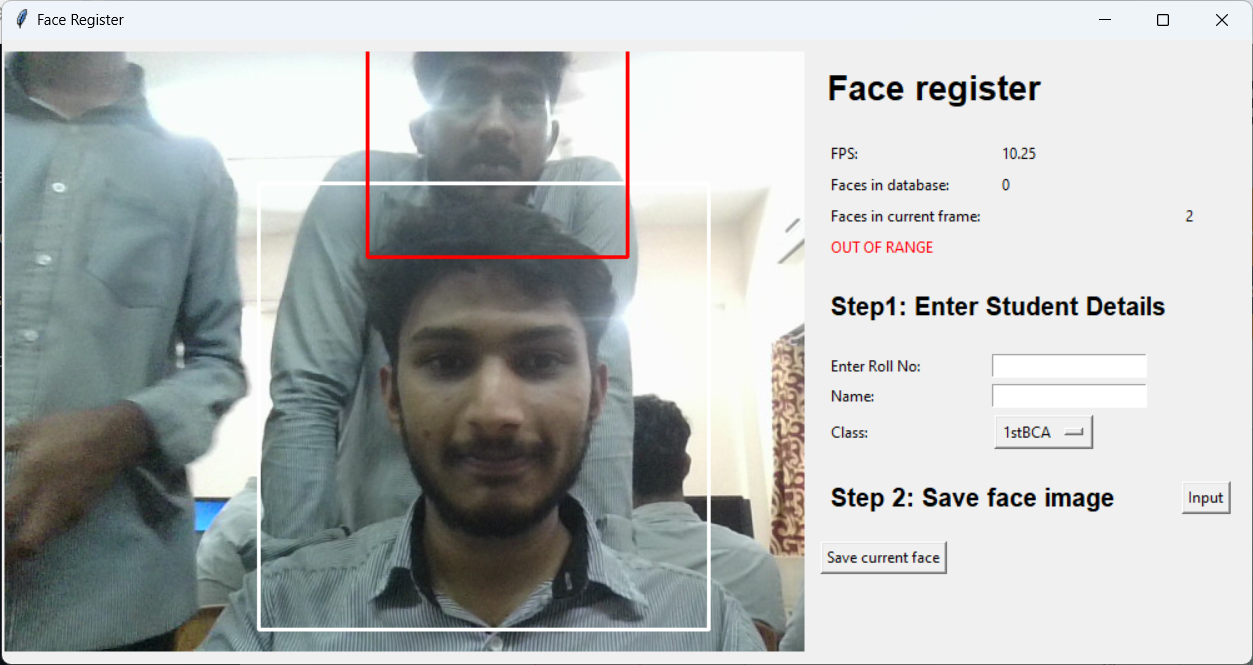
**Home:**



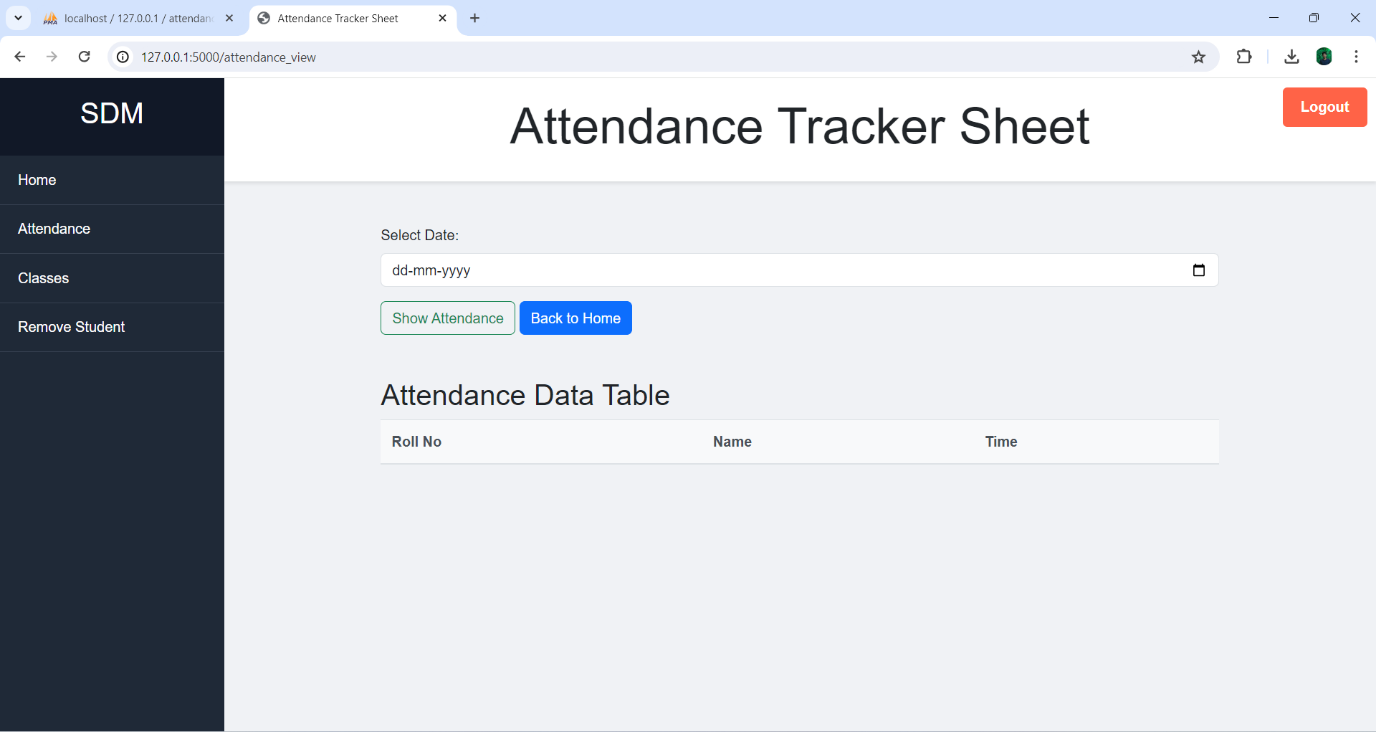
**Take attendance:**

****

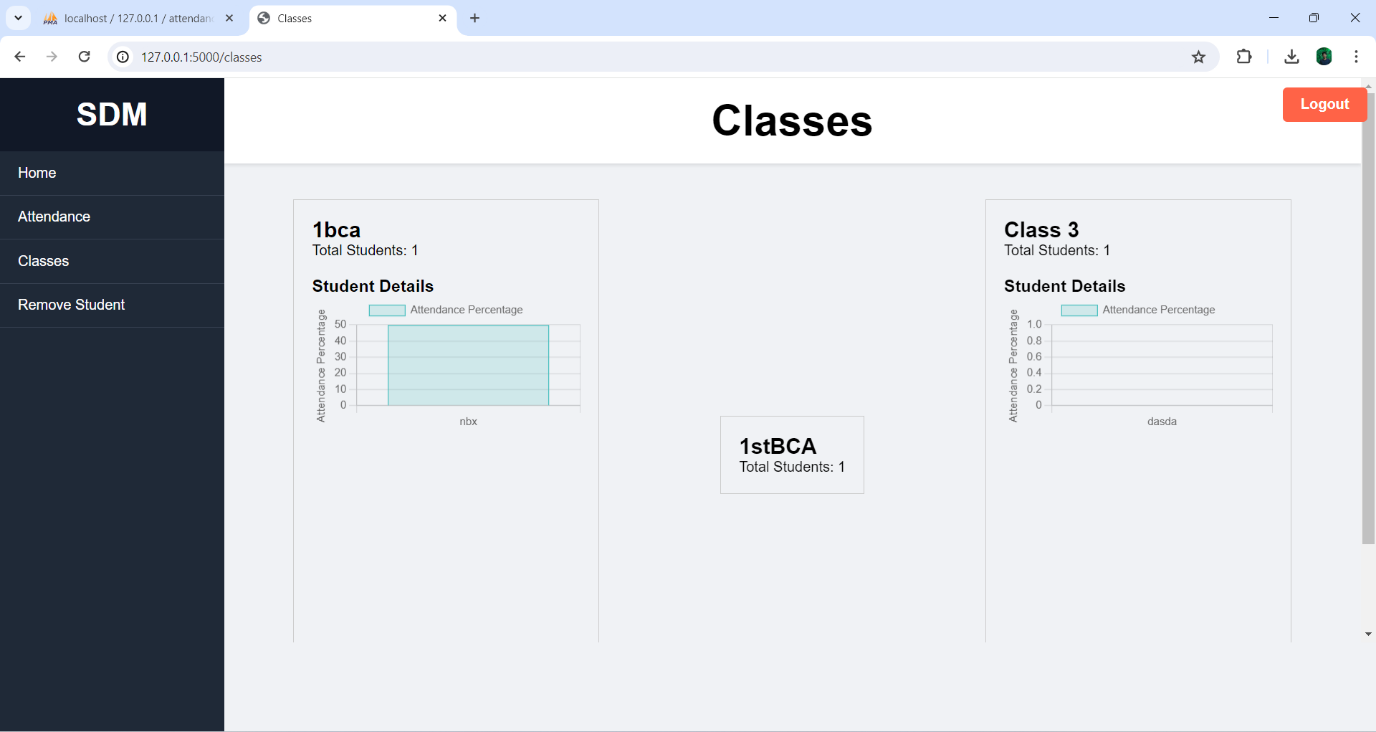
**Add Student:**

****

**Attendance Sheet:**

****

**Class wise logs:**

****

**CHAPTER - 5**

**TESTING**

**5.1 Introduction:**

Software testing is an investigation conducted to provide stack holders with information about the quality of the product or service under test. Testing has been defined as the process of analysing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item.

Software testing is the process used to assess the quality of computer software. It involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they should. It is oriented to ‘detection’.

**Software testing has three main purposes:**

* The verification process confirms that the software meets its technical specifications. A “specification” is a description of a function in terms of a measurable output value given a specific input value under specific preconditions.
* The validation process confirms that the software meets the business requirements.
* A defect is a variance between the expected and actual result. The defect’s ultimate source may be traced to a fault introduced in the specification, design, or development phases. Not all the defects will necessarily result in failures.

**There are two types of software testing:**

* **Black Box Testing**: Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.
* **White box testing**: This testing is based on knowledge of the internal logic of an application’s code. Also known as glass box testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths and conditions.

A test case is a software testing document, which consists of event, action, input, output, expected result and actual result. It can occasionally be a series of steps but one with expected results or expected outcome. A test case should also contain a place for the actual result.

White box testing is applicable at the unit, integration and system levels of the software testing process.

**5.2 Objective of testing:**

* Finding defects which may get created by the programmer while developing the software.
* Gaining confidence in and providing information about the level of quality.
* To prevent defects.
* To make sure that the end results meet the business and user requirements.
* To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specification.

**5.3 Testing Methods:**

System testing is the stage of implementation. This is to check whether system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. The candidate system is subject to a variety of tests: online response, volume, stress, recovery, security, and usability tests. A series of tests are performed for the proposed system is ready for user acceptance testing.

**5.4 Testing Steps:**

**5.4.1 Unit Testing:**

Unit testing focuses efforts on the smallest unit of software design this is known as module testing or white box testing, the modules are tested separately. The test is carried out during the programming stage itself. In this step, each module is found to be working satisfactorily as regards to the expected output from the module.

**5.4.2** **Integration Testing:**

In integration testing the different units of the system are integrated together to form the complete system. This type of testing checks the system to ensure that it is doing what it's supposed to do the testing of an integrated system can be carried out top-down, bottom-up or Big-Bang. In this type of testing some parts are tested with white box testing and some with black box testing techniques. This type of testing plays a very important role in increasing systems productivity. We have checked the system by using integration testing techniques.

**5.4.3 Validation Testing:**

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists:

* The function or performance characteristics confirm to specification and are accepted.
* A deviation from specification is uncovered and a deficiency list is created.
* Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

**5.4.4 Output testing:**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct. The format was designed in the system design time according to the user needs. For the hard copy also, the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

**5.4.5** **User acceptance testing:**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required.

This is done regarding the following point:

* Input screen design.
* Output screen design.
* Online message should be guide to the user.
* Format of reports and other outputs.

**5.5** **Test cases:**

**5.5.1 Testing for login:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Mandatory fields left empty | Mandatory fields cannot be left empty | Should not be blank |
| 2 | Click Login | Entering details | Redirects to respected page |

**5.5.2 Testing taking attendance:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Unknown Student | Face not registered | Appropriate error message |
| 2 | Known Student | Attendance marked | Attendance Marked |
| 3 | Attendance already marked | Attendance can marked only once a day | Appropriate error message |

**5.5.3 Testing for Attendance Tracker Sheet:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Date not selected | Should enter the date | Appropriate error message |
| 2 | Date Entered | Date exist in DB | Attendance accepted |

**5.5.4 Testing for adding student:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Mandatory fields left empty | Mandatory fields cannot be left blank | Appropriate error message |
| 2 | Duplicate entry | Duplicate entry not allowed | Appropriate error message |
| 3 | Input without any faults | Valid Input | Input accepted |
| 4 | Face not in range | Face not recognized | Appropriate error message |

**5.5.5 Testing for student deletion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Deletion attempted when no entries present | Entries if not present cannot be deleted | Appropriate error message |
| 2 | Deletion without any errors | Valid deletion | Entries deleted |

**5.5.6 Testing for Forget password:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Input** | **Test Description** | **Output** |
| 1 | Mandatory fields left empty | Mandatory fields cannot be left empty | Should not be blank |

**CONCLUSION**

In conclusion, "Face Attendance System" represents a modern and efficient solution for managing attendance, offering a comprehensive range of features and functionalities. By leveraging advanced digital tools and a user-friendly interface, the platform aims to streamline the attendance tracking process, connecting educators and administrators in a seamless manner. The system's robust database design ensures data integrity, supports complex reporting, and enhances user experience, making it a reliable and convenient choice for educational institutions and businesses alike.

Moving forward, " Face Attendance System " is committed to continuous improvement and innovation, seeking to enhance its features and expand its capabilities to meet the evolving needs of its users. Through its dedication to providing excellent customer service and maintaining a strong online presence, the platform aims to establish itself as a trusted and preferred solution for attendance management, contributing to the efficiency and effectiveness of organizational operations in the digital age.

**Bibliography / References:**

1. <https://youtu.be/z_dbnYHAQYg?si=BheD4llBj3bfoize>

2. <https://youtu.be/qeHXHphI9cg?si=P5_zMfj-lFrqBrnt>

3. For SQL: <https://youtu.be/A0fm26PKlwk?si=Sc9CSzg-Ehe_GbV->

4. GIT Hub Repo: <https://github.com/Arijit1080/Face-Recognition-Based-Attendance-System>

5. <https://github.com/puneethail/faceattendance_model>

6. <https://github.com/nithinbhandary>

7. Python Crash Course: A Hands-On, Project-Based Introduction to Programming

8. Head-First Python: A Brain-Friendly Guide

**TEXTBOOKS:**

1. Learning PHP, MySQL and CSS: A Step-by-Step Guide to Creating Dynamic Websites – by Robin Nixon

2. Python Crash Course: A Hands-On, Project-Based Introduction to Programming

3. Head-First Python: A Brain-Friendly Guide

4. PHP & MySQL Web Development – by Luke Welling & Laura Thompson