Capstone management portal   
Sign in as – Student,Teacher,Admin

Email ,password

Don’t hv a account? Sign up

Background-PES Logo

Registering as

Student-First Name,Last Name,SRN,Email,Department,password

Teacher-Name,Email,Department,Expertise,password

Admin-Name,email,password

Student

Personal info as button on right side,on pressing it

Should display team name,members,guide name

Display-Exam\_id,Exam date,Exam type,timings,venue

Display ISA1, ISA2,ESA marks – with sem . display grade taking total of marks

Project – Name,description,progress

Document-file path,upload date,version

Schedule appointments- Date,description Display status upon received from guide

How will they form a team?

Teacher

Personal info as button on right side ,on pressing it

Projects-Display list of projects….on down arrow-- > display Team name,team members,view project progress, View document of project submitted by team with version number

Display part of panel(panel id) – panel members,give feedback on selecting team name,give marks for ISA1,ISA2,ESA based on exam\_id

View appointments-give permission as available or not available

Admin

Display personal info as button on right hand side and opening it

Schedule exams with Exam\_id,Exam date,Exam type,Timings and venue which will reach all students

Able to view all Teachers -their personal info and team\_name and project\_name handling

Able to view all Students-their personal info,team\_name and project\_name,their marks in all exam\_type based on exam\_id,

Login as other faculty

Project\_id not getting stored in document,so document table is not coming

CSS- LOGIN sign in

Admin-Show top 10 students

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

import mysql.connector

from mysql.connector import Error

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

from werkzeug.utils import secure\_filename

from datetime import datetime

import os

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

app.secret\_key = 'A@1sdfgHjk2$%3Asdf'

app.config['UPLOAD\_FOLDER'] = os.path.join(os.getcwd(), 'uploads')

app.config['SECRET\_KEY'] = 'A@1sdfgHjk2$%3Asdf'

MYSQL\_HOST='localhost'

MYSQL\_USER='root'

MYSQL\_PASSWORD='varshini@2004'

MYSQL\_DB='dbms'

connection=mysql.connector.connect(

    host=MYSQL\_HOST,

    user=MYSQL\_USER,

    password=MYSQL\_PASSWORD,

    database=MYSQL\_DB

)

# Database configuration

"""db\_config = {

    'user': 'root',

    'password': 'varshini@2004',

    'host': 'localhost',

    'port': 33060,

    'database': 'dbms',

}"""

# Function to create a database connection

'''def get\_db\_connection():

    connection = None

    try:

        connection = mysql.connector.connect(\*\*db\_config)

    except Error as e:

        print(f"Error: {e}")

    return connection'''

# Routes

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/register\_choice')

def register\_choice():

    return render\_template('register\_choice.html')

# Registration Routes

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

def register\_student():

    if request.method == 'POST':

        srn = request.form['srn']

        first\_name = request.form['first\_name']

        last\_name = request.form['last\_name']

        department = request.form['department']

        email = request.form['email']

        password = generate\_password\_hash(request.form['password'], method='pbkdf2:sha256')

        cursor = connection.cursor()

        try:

            insert\_query = """

            INSERT INTO STUDENT\_DETAILS (srn, first\_name, last\_name, department, email, password)

            VALUES (%s, %s, %s, %s, %s, %s)

            """

            cursor.execute(insert\_query, (srn, first\_name, last\_name, department, email, password))

            connection.commit()

            flash("Student registered successfully! Please log in.", "success")

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

        except Error as e:

            print(f"Error: {e}")

            flash("An error occurred while registering. Please try again.", "danger")

        finally:

            cursor.close()

    return render\_template('register\_student.html')

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

def register\_teacher():

    if request.method == 'POST':

        faculty\_id = request.form['Faculty\_ID']  # Capture Faculty\_ID

        faculty\_name = request.form['Faculty\_Name']

        department = request.form['Department']

        expertise = request.form['Expertise']

        email = request.form['Email']

        password = generate\_password\_hash(request.form['Password'], method='pbkdf2:sha256')

        cursor = connection.cursor()

        try:

            # SQL query to insert new teacher record with Faculty\_ID

            insert\_query = """

            INSERT INTO FACULTY\_DETAILS (Faculty\_ID, Faculty\_Name, Department, Expertise, Email, Password)

            VALUES (%s, %s, %s, %s, %s, %s)

            """

            cursor.execute(insert\_query, (faculty\_id, faculty\_name, department, expertise, email, password))

            connection.commit()

            flash("Teacher registered successfully! Please log in.", "success")

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

        except Error as e:

            print(f"Error: {e}")

            flash("An error occurred while registering. Please try again.", "danger")

        finally:

            cursor.close()

    return render\_template('register\_teacher.html')

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

def register\_admin():

    if request.method == 'POST':

        name = request.form['name']

        email = request.form['email']

        password = generate\_password\_hash(request.form['password'], method='pbkdf2:sha256')

        cursor = connection.cursor()

        try:

            # Insert new admin details into ADMIN\_DETAILS

            insert\_query = "INSERT INTO ADMIN\_DETAILS (name, email, password) VALUES (%s, %s, %s)"

            cursor.execute(insert\_query, (name, email, password))

            connection.commit()

            flash("Admin registered successfully! Please log in.", "success")

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

        except Error as e:

            print(f"Error: {e}")

            flash("An error occurred during registration. Please try again.", "danger")

        finally:

            cursor.close()  # Ensure cursor is closed

    return render\_template('register\_admin.html')

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

def login():

    if request.method == 'POST':

        role = request.form['role']

        email = request.form['email']

        password = request.form['password']

        # Connect to the database

        connection = mysql.connector.connect(

            host=MYSQL\_HOST,

            user=MYSQL\_USER,

            password=MYSQL\_PASSWORD,

            database=MYSQL\_DB

        )

        cursor = None

        try:

            cursor = connection.cursor()

            # Set the query and password index based on the role

            if role == 'student':

                query = "SELECT \* FROM STUDENT\_DETAILS WHERE email = %s"

                password\_index = 6

            elif role == 'teacher':

                query = "SELECT \* FROM FACULTY\_DETAILS WHERE email = %s"

                password\_index = 5

            elif role == 'admin':

                query = "SELECT \* FROM ADMIN\_DETAILS WHERE email = %s"

                password\_index = 3

            else:

                flash("Invalid role selected.", "danger")

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

            # Execute query to fetch user details

            cursor.execute(query, (email,))

            user = cursor.fetchone()

            if user:

                # Verify the password using the role-specific password index

                if check\_password\_hash(user[password\_index], password):

                    session['user\_role'] = role

                    session['user\_id'] = user[0]  # Assuming user[0] is the unique user identifier (e.g., SRN)

                    # Redirect based on user role

                    if role == 'student':

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

                    elif role == 'teacher':

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

                    elif role == 'admin':

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

                else:

                    flash("Invalid password. Please try again.", "danger")

            else:

                flash("No user found with that email. Please check your credentials.", "danger")

        except mysql.connector.Error as e:

            print(f"Database error: {e}")

            flash("An error occurred while accessing the database. Please try again.", "danger")

        finally:

            if cursor:

                cursor.close()

            if connection:

                connection.close()

    return render\_template('login.html')

# Function to execute a query and fetch results

def execute\_query(query, params=()):

    connection = mysql.connector.connect(

        host=MYSQL\_HOST,

        user=MYSQL\_USER,

        password=MYSQL\_PASSWORD,

        database=MYSQL\_DB

    )

    cursor = connection.cursor(dictionary=True)

    cursor.execute(query, params)

    result = cursor.fetchall()

    cursor.close()

    connection.close()

    return result

# Function to fetch a single record

def fetch\_one(query, params=()):

    connection = mysql.connector.connect(

        host=MYSQL\_HOST,

        user=MYSQL\_USER,

        password=MYSQL\_PASSWORD,

        database=MYSQL\_DB

    )

    cursor = connection.cursor(dictionary=True)

    cursor.execute(query, params)

    result = cursor.fetchone()

    cursor.close()

    connection.close()

    return result

def execute\_update(query, params=()):

    connection = mysql.connector.connect(

        host=MYSQL\_HOST,

        user=MYSQL\_USER,

        password=MYSQL\_PASSWORD,

        database=MYSQL\_DB

    )

    cursor = connection.cursor()

    cursor.execute(query, params)

    connection.commit()

    cursor.close()

    connection.close()

@app.route('/dashboard/student')

def student\_dashboard():

    if 'user\_role' in session and session['user\_role'] == 'student':

        student\_id = session['user\_id']  # This should be the SRN from the session

        try:

            # Fetch student details based on the SRN (student\_id)

            student\_query = "SELECT \* FROM STUDENT WHERE SRN = %s"

            student = fetch\_one(student\_query, (student\_id,))

            if student:

                # Fetch team details based on Team\_ID

                team\_query = "SELECT \* FROM TEAM WHERE Team\_ID = %s"

                team = fetch\_one(team\_query, (student['Team\_ID'],))

                if team:

                    # Fetch all team members with the same Team\_ID

                    team\_members\_query = "SELECT \* FROM STUDENT WHERE Team\_ID = %s"

                    team\_members = execute\_query(team\_members\_query, (student['Team\_ID'],))

                    # Fetch project details using the Project\_ID from the TEAM table

                    project\_query = "SELECT \* FROM PROJECT WHERE Project\_ID = %s"

                    project = fetch\_one(project\_query, (team['Project\_ID'],))

                    # Fetch guide details using Guide\_ID from the TEAM table

                    guide\_query = "SELECT \* FROM GUIDE WHERE Guide\_ID = %s"

                    guide = fetch\_one(guide\_query, (team['Guide\_ID'],))

                    # Fetch exams with details including calculated total and grade

                    exams\_query = """

                    SELECT EXAM.Exam\_ID,

                    EXAM.Exam\_Date,

                    EXAM.ISA1,

                    EXAM.ISA2,

                    EXAM.ESA,

                    (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) AS total,

                    CASE

                        WHEN (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) > 90 THEN 'S'

                        WHEN (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) BETWEEN 81 AND 90 THEN 'A'

                        WHEN (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) BETWEEN 71 AND 80 THEN 'B'

                        WHEN (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) BETWEEN 61 AND 70 THEN 'C'

                        WHEN (EXAM.ISA1 + EXAM.ISA2 + EXAM.ESA) BETWEEN 51 AND 60 THEN 'D'

                        ELSE 'F'

                    END AS grade,

                    EXAMGRADE.Marks,

                    EXAMDETAILS.Exam\_Type,

                    EXAMDETAILS.Timings,

                    EXAMDETAILS.Venue,

                    EXAMDETAILS.Semester

                    FROM EXAM

                    JOIN EXAMGRADE ON EXAM.Exam\_ID = EXAMGRADE.Exam\_ID

                    JOIN EXAM\_DETAILS AS EXAMDETAILS ON EXAM.Exam\_ID = EXAMDETAILS.Exam\_ID

                    WHERE EXAMGRADE.SRN = %s

                    """

                    exams = execute\_query(exams\_query, (student\_id,))

                    # Fetch other dashboard information such as documents and appointments

                    documents\_query = "SELECT \* FROM DOCUMENT WHERE Project\_ID = %s"

                    documents = execute\_query(documents\_query, (project['Project\_ID'],))

                    appointments\_query = "SELECT \* FROM APPOINTMENT WHERE Team\_ID = %s"

                    appointments = execute\_query(appointments\_query, (team['Team\_ID'],))

                    # Render the dashboard with all the fetched data

                    return render\_template('student\_dashboard.html',

                                        student=student,

                                        team=team,

                                        guide=guide,

                                        team\_members=team\_members,

                                        project=project,

                                        exams=exams,

                                        documents=documents,

                                        appointments=appointments)

                else:

                    flash("Team details not found.", "danger")

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

            else:

                flash("Student details not found.", "danger")

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

        except Error as e:

            print(f"Database error: {e}")

            flash("An error occurred while accessing the database. Please try again.", "danger")

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

    else:

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

@app.route('/schedule\_appointment', methods=['GET', 'POST'])

def schedule\_appointment():

    if request.method == 'POST':

        # Get form data

        appointment\_date = request.form['date']

        description = request.form['description']

        # Get Team\_ID and Guide\_ID from session or other logic (assuming they are stored in session)

        team\_id = session.get('Team\_ID')  # Assuming the team ID is stored in the session

        guide\_id = session.get('Guide\_ID')  # Assuming the guide ID is stored in the session

        # Convert the appointment date to a datetime object

        appointment\_date = datetime.strptime(appointment\_date, '%Y-%m-%d').date()

        # Prepare the SQL query to insert the new appointment

        insert\_query = """

            INSERT INTO APPOINTMENT (Appointment\_Date, Statuss, Descript, Team\_ID, Guide\_ID)

            VALUES (%s, %s, %s, %s, %s)

        """

        params = (appointment\_date, "Pending", description, team\_id, guide\_id)

        try:

            # Execute the query to insert the new appointment

            execute\_update(insert\_query, params)

            flash("Appointment scheduled successfully!", "success")

            return redirect(url\_for('student\_dashboard'))  # Redirect to dashboard or appointment list page

        except Error as e:

            flash(f"An error occurred while scheduling the appointment: {e}", "danger")

    # If GET request, render the appointment form

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

ALLOWED\_EXTENSIONS = {'pdf', 'doc', 'docx', 'jpg', 'png', 'txt'}

# Function to check file extensions

def allowed\_file(filename):

    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

@app.route('/upload\_document', methods=['POST'])

def upload\_document():

    # Check if a file is part of the request

    if 'document' not in request.files:

        flash('No file part', 'danger')

        return redirect(request.url)

    file = request.files['document']

    # If the user does not select a file, the browser submits an empty part without a filename

    if file.filename == '':

        flash('No selected file', 'danger')

        return redirect(request.url)

    # If the file is allowed, save it to the upload folder

    if file and allowed\_file(file.filename):

        filename = secure\_filename(file.filename)

        filepath = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

        file.save(filepath)

        # Get the additional form data (upload\_date, version, and project\_id)

        upload\_date = request.form['Upload\_Date']

        version = request.form['Versionn']

        project\_id = session.get('Project\_ID') # Assuming this comes from the form

        # Convert upload\_date to a datetime object

        upload\_date = datetime.strptime(upload\_date, '%Y-%m-%d').date()

        # Connect to the MySQL database

        connection = mysql.connector.connect(

        host=MYSQL\_HOST,

        user=MYSQL\_USER,

        password=MYSQL\_PASSWORD,

        database=MYSQL\_DB

        )

        cursor = connection.cursor()

        # Insert document details into the database

        insert\_query = """

        INSERT INTO DOCUMENT (Filepath, Versionn, Upload\_Date, Project\_ID)

        VALUES (%s, %s, %s, %s)

        """

        try:

            cursor.execute(insert\_query, (filepath, version, upload\_date, project\_id))

            connection.commit()

            flash('File uploaded and document details saved successfully', 'success')

        except mysql.connector.Error as err:

            connection.rollback()  # Rollback in case of an error

            flash(f'Error saving document details: {err}', 'danger')

        finally:

            cursor.close()

            connection.close()

        return redirect(url\_for('student\_dashboard'))  # Redirect to the student dashboard or wherever

    flash('Invalid file type. Allowed file types are: pdf, doc, docx, jpg, png, txt', 'danger')

    return redirect(request.url)

@app.route('/dashboard/teacher')

def teacher\_dashboard():

    if 'user\_role' in session and session['user\_role'] == 'teacher':

        teacher\_id = session['user\_id']

        try:

            # Fetch teacher details from GUIDE table

            teacher\_query = "SELECT \* FROM GUIDE WHERE Faculty\_ID = %s"

            teacher = fetch\_one(teacher\_query, (teacher\_id,))

            if teacher:

                # Fetch guide information (this is likely redundant since it's the same query)

                guide = fetch\_one(teacher\_query, (teacher\_id,))

                if guide:

                    # Fetch teams under the teacher based on Team\_ID in the GUIDE table

                    team\_query = """

                        SELECT TEAM.Team\_ID, TEAM.Team\_Name, PROJECT.Project\_Name, PROJECT.Project\_ID

                        FROM TEAM

                        JOIN PROJECT ON TEAM.Project\_ID = PROJECT.Project\_ID

                        WHERE TEAM.Guide\_ID = %s

                    """

                    teams = execute\_query(team\_query, (guide['Guide\_ID'],))

                    # Fetch panels that the teacher is part of

                    panel\_query = """

                        SELECT PANEL.Panel\_ID, PANEL.Project\_ID, PANEL.Feedback, EXAMGRADE.Grade\_ID

                        FROM PANEL

                        JOIN EXAMGRADE ON PANEL.Panel\_ID = EXAMGRADE.Panel\_ID

                        WHERE EXAMGRADE.SRN IN (SELECT SRN FROM STUDENT WHERE Team\_ID IN (SELECT Team\_ID FROM TEAM WHERE Guide\_ID = %s))

                    """

                    panels = execute\_query(panel\_query, (teacher\_id,))

                    # Prepare team details for each team

                    team\_details = []

                    for team in teams:

                        # Fetch team members for each team

                        team\_members\_query = "SELECT \* FROM STUDENT WHERE Team\_ID = %s"

                        team\_members = execute\_query(team\_members\_query, (team['Team\_ID'],))

                        # Fetch project progress and documents for each team

                        project\_query = "SELECT Progress FROM PROJECT WHERE Project\_ID = %s"

                        project = fetch\_one(project\_query, (team['Project\_ID'],))

                        documents\_query = "SELECT Filepath, Versionn FROM DOCUMENT WHERE Project\_ID = %s"

                        documents = execute\_query(documents\_query, (team['Project\_ID'],))

                        # Fetch appointments for each team

                        appointments\_query = """

                            SELECT APPOINTMENT.Appointment\_ID, APPOINTMENT.Appointment\_Date, APPOINTMENT.Statuss, APPOINTMENT.Descript, TEAM.Team\_Name

                            FROM APPOINTMENT

                            JOIN TEAM ON APPOINTMENT.Team\_ID = TEAM.Team\_ID

                            WHERE APPOINTMENT.Team\_ID = %s

                        """

                        appointments = execute\_query(appointments\_query, (team['Team\_ID'],))

                        team\_details.append({

                            'team': team,

                            'members': team\_members,

                            'project\_progress': project['Progress'],

                            'documents': documents,

                            'appointments': appointments

                        })

                    # Render the teacher dashboard with all fetched details

                    return render\_template(

                        'teacher\_dashboard.html',

                        teacher=teacher,

                        guide=guide,  # Pass the guide information

                        team\_details=team\_details,

                        panels=panels

                    )

                else:

                    flash("Guide details not found.", "danger")

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

            else:

                flash("Teacher details not found.", "danger")

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

        except mysql.connector.Error as e:

            print(f"Database error: {e}")

            flash("An error occurred while accessing the database. Please try again.", "danger")

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

    else:

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

@app.route('/update\_appointment\_status', methods=['POST'])

def update\_appointment\_status():

    if 'user\_role' in session and session['user\_role'] == 'teacher':

        appointment\_id = request.form.get('appointment\_id')

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

        try:

            # Update the status of the appointment

            update\_query = "UPDATE APPOINTMENT SET Statuss = %s WHERE Appointment\_ID = %s"

            execute\_query(update\_query, (status, appointment\_id))

            flash("Appointment status updated successfully!", "success")

        except mysql.connector.Error as e:

            print(f"Database error: {e}")

            flash("An error occurred while updating the appointment. Please try again.", "danger")

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

    else:

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

@app.route('/update\_marks\_feedback', methods=['POST'])

def update\_marks\_feedback():

    try:

        exam\_id = request.form['exam\_id']

        panel\_id = request.form['panel\_id']

        project\_id = request.form['project\_id']

        srn = request.form.get(f'SRN\_{exam\_id}')

        isa1 = request.form.get(f'ISA1\_{exam\_id}')

        isa2 = request.form.get(f'ISA2\_{exam\_id}')

        esa = request.form.get(f'ESA\_{exam\_id}')

        feedback = request.form.get(f'feedback\_{exam\_id}')

        # Update the EXAM table with the marks (ISA1, ISA2, ESA)

        update\_exam\_query = """

            UPDATE EXAM SET ISA1 = %s, ISA2 = %s, ESA = %s WHERE Exam\_ID = %s

        """

        execute\_query(update\_exam\_query, (isa1, isa2, esa, exam\_id))

        # Check if there’s already an entry for this SRN in the EXAMGRADE table, and insert/update accordingly

        check\_examgrade\_query = """

            SELECT Grade\_ID FROM EXAMGRADE WHERE Exam\_ID = %s AND SRN = %s

        """

        grade\_entry = fetch\_one(check\_examgrade\_query, (exam\_id, srn))

        if grade\_entry:

            # Update existing grade entry

            update\_examgrade\_query = """

                UPDATE EXAMGRADE SET Marks = %s, Panel\_ID = %s WHERE Exam\_ID = %s AND SRN = %s

            """

            execute\_query(update\_examgrade\_query, (isa1 + isa2 + esa, panel\_id, exam\_id, srn))

        else:

            # Insert new entry if not already present

            insert\_examgrade\_query = """

                INSERT INTO EXAMGRADE (Marks, Exam\_ID, SRN, Panel\_ID)

                VALUES (%s, %s, %s, %s)

            """

            execute\_query(insert\_examgrade\_query, (isa1 + isa2 + esa, exam\_id, srn, panel\_id))

        # Update feedback in PANEL table (or related table)

        update\_feedback\_query = """

            UPDATE PANEL SET Feedback = %s WHERE Panel\_ID = %s

        """

        execute\_query(update\_feedback\_query, (feedback, panel\_id))

        flash("Marks and Feedback updated successfully.", "success")

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

    except Exception as e:

        flash(f"Error: {str(e)}", "danger")

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

# Admin dashboard to view and update GUIDE, STUDENT, TEAM tables

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

def admin\_dashboard():

    if 'user\_role' in session and session['user\_role'] == 'admin':

        admin\_id = session['user\_id']  # Assuming admin's ID is stored in session

        try:

            connection = mysql.connector.connect(

                host=MYSQL\_HOST,

                user=MYSQL\_USER,

                password=MYSQL\_PASSWORD,

                database=MYSQL\_DB

            )

            cursor = connection.cursor(dictionary=True)

            # Fetch admin details

            cursor.execute("SELECT \* FROM ADMIN\_DETAILS WHERE id = %s", (admin\_id,))

            admin = cursor.fetchone()  # Fetch one record for admin details

            # Fetch data from GUIDE, STUDENT, TEAM tables

            cursor.execute("SELECT \* FROM GUIDE")

            guides = cursor.fetchall()

            cursor.execute("SELECT \* FROM STUDENT")

            students = cursor.fetchall()

            cursor.execute("SELECT \* FROM TEAM")

            teams = cursor.fetchall()

            # Ensure that admin details, guides, students, and teams are passed to the template

            return render\_template('admin\_dashboard.html', admin=admin, guides=guides, students=students, teams=teams)

        except mysql.connector.Error as e:

            print(f"Database error: {e}")

            flash("An error occurred while accessing the database.", "danger")

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

        finally:

            if cursor:

                cursor.close()

            if connection:

                connection.close()

    else:

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

@app.route('/update\_guide', methods=['POST'])

def update\_guide():

    # Get form data for the guide

    guide\_id = request.form.get('Guide\_ID')

    first\_name = request.form.get('First\_Name')

    last\_name = request.form.get('Last\_Name')

    department = request.form.get('Department')

    email = request.form.get('Email')

    faculty\_id = request.form.get('Faculty\_ID')

    team\_id = request.form.get('Team\_ID')

    # Insert or update based on Guide\_ID

    query = """REPLACE INTO GUIDE (Guide\_ID, First\_Name, Last\_Name, Department, Email, Faculty\_ID, Team\_ID)

            VALUES (%s, %s, %s, %s, %s, %s, %s)"""

    data = (guide\_id, first\_name, last\_name, department, email, faculty\_id, team\_id)

    execute\_query(query, data)

    flash("Guide information updated successfully.", "success")

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

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

def update\_student():

    # Get form data for the student

    srn = request.form.get('SRN')

    first\_name = request.form.get('First\_Name')

    last\_name = request.form.get('Last\_Name')

    department = request.form.get('Department')

    email = request.form.get('Email')

    team\_id = request.form.get('Team\_ID')

    # Insert or update based on SRN

    query = """REPLACE INTO STUDENT (SRN, First\_Name, Last\_Name, Department, Email, Team\_ID)

            VALUES (%s, %s, %s, %s, %s, %s)"""

    data = (srn, first\_name, last\_name, department, email, team\_id)

    execute\_query(query, data)

    flash("Student information updated successfully.", "success")

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

@app.route('/update\_team', methods=['POST'])

def update\_team():

    # Get form data for the team

    team\_id = request.form.get('Team\_ID')

    team\_name = request.form.get('Team\_Name')

    project\_id = request.form.get('Project\_ID')

    guide\_id = request.form.get('Guide\_ID')

    # Insert or update based on Team\_ID

    query = """REPLACE INTO TEAM (Team\_ID, Team\_Name, Project\_ID, Guide\_ID)

            VALUES (%s, %s, %s, %s)"""

    data = (team\_id, team\_name, project\_id, guide\_id)

    execute\_query(query, data)

    flash("Team information updated successfully.", "success")

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

@app.route('/schedule\_exam', methods=['POST'])

def schedule\_exam():

    if 'user\_role' in session and session['user\_role'] == 'admin':

        try:

            # Retrieve form data

            exam\_id = request.form['exam\_id']

            exam\_date = request.form['exam\_date']

            exam\_type = request.form['exam\_type']

            timings = request.form['timings']

            venue = request.form['venue']

            semester = request.form['semester']

            # Insert data into EXAM table

            insert\_exam\_query = """

                INSERT INTO EXAM (Exam\_ID, Exam\_Date)

                VALUES (%s, %s)

            """

            execute\_query(insert\_exam\_query, (exam\_id, exam\_date))

            # Insert data into EXAM\_DETAILS table

            insert\_exam\_details\_query = """

                INSERT INTO EXAM\_DETAILS (Exam\_ID, Exam\_Type, Timings, Venue, Semester)

                VALUES (%s, %s, %s, %s, %s)

            """

            execute\_query(insert\_exam\_details\_query, (exam\_id, exam\_type, timings, venue, semester))

            flash("Exam scheduled successfully!", "success")

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

        except Exception as e:

            flash(f"An error occurred: {e}", "danger")

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

    else:

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

# Logout Route

@app.route('/logout')

def logout():

    session.clear()

    flash("You have been logged out.", "info")

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

# Run the App

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

    app.run(debug=True)