Index file

<!DOCTYPE html>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Employee</title>

<!-- Include Tailwind CSS CDN -->

<link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">

<script src="https://cdn.tailwindcss.com"></script>

<script>

tailwind.config = {

theme: {

extend: {

colors: {

clifford: '#da373d',

}

}

}

}

</script>

</head>

<body class="bg-gray-100 flex flex-col min-h-screen">

<!-- Navbar -->

<nav class="bg-blue-100 text-blue-800 p-4">

<div class="flex items-center justify-between">

<!-- Logo on the left -->

<div class="flex items-center">

<a href="index.html"> <img src="https://dhakabankltd.com/wp-content/themes/dhakabankupdate/images/dbl-plc-logo.png" alt="Logo" class="h-8 mr-2"></a>

</div>

<a href="#" class="text-lg font-extrabold">EMPLOYEE MANAGEMENT</a>

<!-- Navigation links on the right -->

<div class="flex items-center">

<a href="index.html" class="mr-4">Home</a>

<a href="about.html" class="mr-4">About</a>

<a href="contact.html" class="mr-4">Contact</a>

</div>

</div>

</nav>

<div class="container mx-auto my-8 p-8 bg-white rounded shadow">

<h2 class="text-2xl font-semibold mb-4">Add Employee</h2>

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

{% if messages %}

<div class="alert bg-red-500 text-white p-4 rounded-lg text-xl">

<ul>

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

</div>

{% endif %}

{% endwith %}

<form action="/save\_employee" method="post" class="mb-8 flex flex-col md:flex-row gap-4">

<div class="flex flex-col mb-4">

<label for="name" class="text-gray-700 font-semibold">Employee Name:</label>

<input type="text" id="name" name="name" placeholder="Employee Name" required class="mt-1 p-2 border rounded">

</div>

<div class="flex flex-col mb-4">

<label for="emp\_id" class="text-gray-700 font-semibold">Employee ID:</label>

<input type="text" id="emp\_id" name="emp\_id" placeholder="Employee ID" required class="mt-1 p-2 border rounded">

</div>

<div class="flex flex-col mb-4">

<label for="designation" class="text-gray-700 font-semibold">Designation:</label>

<input type="text" id="designation" name="designation" placeholder="Designation" required class="mt-1 p-2 border rounded">

</div>

<div class="flex flex-col mb-4">

<label for="email" class="text-gray-700 font-semibold">Email:</label>

<input type="email" id="email" name="email" placeholder="Email" required class="mt-1 p-2 border rounded">

</div>

<div class="flex flex-col mb-4">

<label for="password" class="text-gray-700 font-semibold">Password:</label>

<input type="password" id="password" name="password" placeholder="Password" required class="mt-1 p-2 border rounded">

</div>

<button type="submit" class="bg-blue-400 text-white p-2 rounded self-start md:self-center">Add Employee</button>

</form>

</div>

<h2 class="text-2xl font-semibold mb-4">Employee List</h2>

<table class="w-full border-collapse border border-gray-300 mt-4">

<thead class="bg-blue-300">

<tr>

<th class="border border-blue-400 p-2">SL.</th>

<th class="border border-blue-400 p-2">Name</th>

<th class="border border-blue-400 p-2">Employee ID</th>

<th class="border border-blue-400 p-2">Designation</th>

<th class="border border-blue-400 p-2">Email</th>

<th class="border border-blue-400 p-2">Password</th>

<th class="border border-blue-400 p-2">Actions</th>

</tr>

</thead>

<tbody>

{% for employee in employees %}

<tr>

<td class="border border-gray-300 p-3">{{ employee[0] }}</td>

<td class="border border-gray-300 p-3">

<input id="name\_input\_{{ employee[0] }}" type="text" value="{{ employee[1] }}" class="border rounded p-1">

</td>

<td class="border border-gray-300 p-3">

<input id="emp\_id\_input\_{{ employee[0] }}" type="text" value="{{ employee[2] }}" class="border rounded p-1">

</td>

<td class="border border-gray-300 p-3">

<input id="designation\_input\_{{ employee[0] }}" type="text" value="{{ employee[3] }}" class="border rounded p-1">

</td>

<td class="border border-gray-300 p-3">

<input id="email\_input\_{{ employee[0] }}" type="email" value="{{ employee[4] }}" class="border rounded p-1">

</td>

<td class="border border-gray-300 p-3">

<input id="password\_input\_{{ employee[0] }}" type="password" value="{{ employee[5] }}" class="border rounded p-1">

</td>

<td class="border border-gray-300 p-3">

<button onclick="updateEmployee({{ employee[0] }})" class="text-blue-500 px-2 py-1 rounded bg-blue-100 hover:bg-blue-200">Update</button>

<button onclick="deleteEmployee({{ employee[0] }})" class="text-red-500 px-2 py-1 rounded bg-red-100 hover:bg-red-200">Delete</button>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</table>

</div>

<footer class="bg-gray-200 text-gray-700 text-center py-4">

<p class="text-sm">&copy; 2023 Dhaka Bank. All rights reserved.

Made By <a href="https://www.facebook.com/irinhoque.orchi"><span class="animate-pulse text-blue-500 font-extrabold">Irin Hoque Orchi</span></a>

</p>

</footer>

<script>

function updateEmployee(employeeId) {

var name = document.getElementById('name\_input\_' + employeeId).value;

var emp\_id = document.getElementById('emp\_id\_input\_' + employeeId).value;

var designation = document.getElementById('designation\_input\_' + employeeId).value;

var email = document.getElementById('email\_input\_' + employeeId).value;

var password = document.getElementById('password\_input\_' + employeeId).value;

fetch(`/update\_employee/${employeeId}`, {

method: 'POST',

headers: {'Content-Type': 'application/json'},

body: JSON.stringify({

name: name,

emp\_id: emp\_id,

designation: designation,

email: email,

password: password

}),

})

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

}

function deleteEmployee(employeeId) {

if (confirm('Are you sure you want to delete this employee?')) {

fetch(`/delete\_employee/${employeeId}`, { method: 'POST' })

.then(response => response.json())

.then(data => location.reload())

.catch(error => console.error('Error:', error));

}

}

</script>

</body>

</html>

App.py

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

import mysql.connector

import bcrypt

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

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

# Configure your MySQL database connection

db\_host = 'localhost'

db\_user = 'root'

db\_password = 'tahbib'

db\_name = 'dbms'

'''USE dbms;

CREATE TABLE user (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);

'''

# Add a secret key for session management

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

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

# Create a MySQL connection

connection = mysql.connector.connect(

host=db\_host,

user=db\_user,

password=db\_password,

database=db\_name

)

cursor = connection.cursor()

# Define a function to verify passwords

def verify\_password(user\_password, hashed\_password):

return bcrypt.checkpw(user\_password.encode('utf-8'), hashed\_password.encode('utf-8'))

# Registration Route

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

def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

# Hash the password before storing it in the database

hashed\_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())

# Insert the username and hashed\_password into the 'users' table

cursor.execute('INSERT INTO user (username, password) VALUES (%s, %s)', (username, hashed\_password))

connection.commit()

flash('Registration successful! You can now log in.', 'success')

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

return render\_template('register.html')

# Login Route

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

def login():

if request.method == 'POST':

username = request.form['username']

user\_password = request.form['password']

# Print the SQL query for debugging

print(f"SELECT password FROM user WHERE username = '{username}'")

# Query the 'user' table to retrieve the hashed password for the given username

cursor.execute('SELECT password FROM user WHERE username = %s', (username,))

result = cursor.fetchone()

if result:

hashed\_password\_from\_db = result[0]

if verify\_password(user\_password, hashed\_password\_from\_db):

# Passwords match, login successful

session['username'] = username

flash('Login successful!', 'success')

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

# If username or password is incorrect or user not found

flash('Invalid username or password. Please try again.', 'error')

return render\_template('login.html')

# Dashboard route, where the user goes after successful login

@app.route('/dashboard')

def dashboard():

if 'username' in session:

return render\_template('dashboard.html', username=session['username'])

else:

flash('You must log in to access the dashboard.', 'info')

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

# Logout route

@app.route('/logout')

def logout():

session.pop('username', None)

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

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

# Create the 'vendor\_data' table if it doesn't exist

cursor.execute('''

CREATE TABLE IF NOT EXISTS vendor\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

email VARCHAR(255),

contact VARCHAR(255),

service\_type VARCHAR(255),

vendor\_number VARCHAR(255)

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS user (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

password VARCHAR(255) NOT NULL

);''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS initator\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

project\_number VARCHAR(255),

name VARCHAR(255),

date DATE,

ownership VARCHAR(255),

description VARCHAR(255)

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS materials (

material\_id INT AUTO\_INCREMENT PRIMARY KEY,

disk TEXT,

ram TEXT,

os TEXT,

assigned\_date DATE,

completed\_date DATE

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

emp\_id VARCHAR(255) UNIQUE, -- Add UNIQUE constraint here

designation VARCHAR(255),

email VARCHAR(255),

password VARCHAR(255)

)

''')

#resource\_data

cursor.execute('''

CREATE TABLE IF NOT EXISTS resources (

id INT AUTO\_INCREMENT PRIMARY KEY,

resource\_id VARCHAR(255),

name VARCHAR(255),

team VARCHAR(255),

material VARCHAR(255),

leader\_id VARCHAR(255)

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS planning\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

planning\_id VARCHAR(255),

start\_date DATE,

end\_date DATE,

key\_deliveries TEXT

)

''')

@app.route('/')

def index():

return render\_template('index.html')

# Vendor Routes

@app.route('/vendor')

def vendor\_index():

cursor.execute('SELECT \* FROM vendor\_data')

vendors = cursor.fetchall()

return render\_template('index\_vendor.html', vendors=vendors)

@app.route('/save\_vendor', methods=['POST'])

def save\_vendor():

vendor\_info = {

'name': request.form['name'],

'email': request.form['email'],

'contact': request.form['contact'],

'service\_type': request.form['service\_type'],

'vendor\_number': request.form['vendor\_number']

}

cursor.execute('''

INSERT INTO vendor\_data (name, email, contact, service\_type, vendor\_number)

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

''', (vendor\_info['name'], vendor\_info['email'], vendor\_info['contact'],

vendor\_info['service\_type'], vendor\_info['vendor\_number']))

connection.commit()

return redirect(url\_for('vendor\_index'))

@app.route('/update\_vendor/<int:vendor\_id>', methods=['POST'])

def update\_vendor(vendor\_id):

data = request.get\_json()

cursor.execute('''

UPDATE vendor\_data

SET name = %s, email = %s, contact = %s, service\_type = %s, vendor\_number = %s

WHERE id = %s

''', (data['name'], data['email'], data['contact'], data['service\_type'],

data['vendor\_number'], vendor\_id))

connection.commit()

return jsonify({'message': 'Vendor updated successfully'})

@app.route('/delete\_vendor/<int:id>', methods=['POST'])

def delete\_vendor(id):

cursor.execute('DELETE FROM vendor\_data WHERE id = %s', (id,))

connection.commit()

return jsonify({'message': 'Vendor deleted successfully'})

#project initiator

# Project Initiator Routes

@app.route('/initiator')

def initiator\_index():

cursor.execute('SELECT \* FROM initator\_data')

initiators = cursor.fetchall()

return render\_template('index\_initiator.html', initiators=initiators)

@app.route('/save\_initiator', methods=['POST'])

def save\_initiator():

initiator\_info = {

'project\_number': request.form['project\_number'],

'name': request.form['name'],

'date': request.form['date'],

'ownership': request.form['ownership'],

'description': request.form['description']

}

cursor.execute('''

INSERT INTO initator\_data (project\_number, name, date, ownership, description)

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

''', (initiator\_info['project\_number'], initiator\_info['name'],

initiator\_info['date'], initiator\_info['ownership'],

initiator\_info['description']))

connection.commit()

return redirect(url\_for('initiator\_index'))

@app.route('/update\_initiator/<int:initiator\_id>', methods=['POST'])

def update\_initiator(initiator\_id):

data = request.get\_json()

cursor.execute('''

UPDATE initator\_data

SET project\_number = %s, name = %s, date = %s, ownership = %s, description = %s

WHERE id = %s

''', (data['project\_number'], data['name'], data['date'],

data['ownership'], data['description'], initiator\_id))

connection.commit()

return jsonify({'message': 'Initiator updated successfully'})

@app.route('/delete\_initiator/<int:id>', methods=['POST'])

def delete\_initiator(id):

cursor.execute('DELETE FROM initator\_data WHERE id = %s', (id,))

connection.commit()

return jsonify({'message': 'Initiator deleted successfully'})

#resources

@app.route('/resources')

def resources\_index():

cursor.execute('SELECT \* FROM resources')

resources = cursor.fetchall()

return render\_template('index\_resources.html', resources=resources)

# Save Resource Route

@app.route('/save\_resource', methods=['POST'])

def save\_resource():

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

name = request.form['name']

team = request.form['team']

material = request.form['material']

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

cursor.execute('''

INSERT INTO resources (resource\_id, name, team, material, leader\_id)

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

''', (resource\_id, name, team, material, leader\_id))

connection.commit()

return redirect(url\_for('resources\_index'))

# Update Resource Route

@app.route('/update\_resource/<int:resource\_id>', methods=['POST'])

def update\_resource(resource\_id):

try:

data = request.get\_json()

resource\_id\_updated = data.get('resource\_id')

name = data.get('name')

team = data.get('team')

material = data.get('material')

leader\_id = data.get('leader\_id')

cursor.execute('''

UPDATE resources

SET resource\_id = %s, name = %s, team = %s, material = %s, leader\_id = %s

WHERE id = %s

''', (resource\_id\_updated, name, team, material, leader\_id, resource\_id))

connection.commit()

return jsonify({'message': 'Resource updated successfully'})

except Exception as e:

print(e)

return jsonify({'message': 'An error occurred'}), 500

@app.route('/delete\_resource/<int:resource\_id>', methods=['POST'])

def delete\_resource(resource\_id):

cursor.execute('DELETE FROM resources WHERE id = %s', (resource\_id,))

connection.commit()

return jsonify({'message': 'Resource deleted successfully'})

#Materials

@app.route('/materials')

def materials\_index():

cursor.execute('SELECT \* FROM materials')

materials = cursor.fetchall()

return render\_template('index\_materials.html', materials=materials)

@app.route('/save\_material', methods=['POST'])

def save\_material():

material\_info = {

'disk': request.form['disk'],

'ram': request.form['ram'],

'os': request.form['os'],

'assigned\_date': request.form['assigned\_date'],

'completed\_date': request.form['completed\_date']

}

cursor.execute('''

INSERT INTO materials (disk, ram, os, assigned\_date, completed\_date)

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

''', (material\_info['disk'], material\_info['ram'], material\_info['os'],

material\_info['assigned\_date'], material\_info['completed\_date']))

connection.commit()

return redirect(url\_for('materials\_index'))

@app.route('/update\_material/<int:material\_id>', methods=['POST'])

def update\_material(material\_id):

data = request.get\_json()

try:

cursor.execute('''

UPDATE materials

SET disk = %s, ram = %s, os = %s, assigned\_date = %s, completed\_date = %s

WHERE material\_id = %s

''', (data['disk'], data['ram'], data['os'], data['assigned\_date'], data['completed\_date'], material\_id))

connection.commit()

return jsonify({'message': 'Material updated successfully'})

except Exception as e:

print(e)

return jsonify({'message': 'An error occurred'}), 500

@app.route('/delete\_material/<int:material\_id>', methods=['POST'])

def delete\_material(material\_id):

try:

cursor.execute('DELETE FROM materials WHERE material\_id = %s', (material\_id,))

connection.commit()

return jsonify({'message': 'Material deleted successfully'})

except Exception as e:

print(e)

return jsonify({'message': 'An error occurred'}), 500

#employees

# Employee Routes

@app.route('/employees')

def employees\_index():

cursor.execute('SELECT \* FROM employees')

employees = cursor.fetchall()

return render\_template('index\_employees.html', employees=employees)

@app.route('/save\_employee', methods=['POST'])

def save\_employee():

name = request.form['name']

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

designation = request.form['designation']

email = request.form['email']

password = request.form['password']

try:

cursor.execute('''

INSERT INTO employees (name, emp\_id, designation, email, password)

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

''', (name, emp\_id, designation, email, password))

connection.commit()

return redirect(url\_for('employees\_index'))

except mysql.connector.IntegrityError as e:

# Handle the error when duplicate emp\_id is inserted

flash('Employee with the same Employee ID already exists. Please choose a unique Employee ID.', 'error')

return redirect(url\_for('employees\_index'))

@app.route('/update\_employee/<int:id>', methods=['POST'])

def update\_employee(id):

data = request.get\_json()

cursor.execute('''

UPDATE employees

SET name = %s, emp\_id = %s, designation = %s, email = %s, password = %s

WHERE id = %s

''', (data['name'], data['emp\_id'], data['designation'], data['email'], data['password'], id))

connection.commit()

return jsonify({'message': 'Employee updated successfully'})

@app.route('/delete\_employee/<int:id>', methods=['POST'])

def delete\_employee(id):

cursor.execute('DELETE FROM employees WHERE id = %s', (id,))

connection.commit()

return jsonify({'message': 'Employee deleted successfully'})

#planning

# Add a route to display the Planning List

@app.route('/planning')

def planning\_index():

cursor.execute('SELECT \* FROM planning\_data')

plannings = cursor.fetchall()

return render\_template('index\_planning.html', plannings=plannings)

# Add a route to save Planning

@app.route('/save\_planning', methods=['POST'])

def save\_planning():

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

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

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

key\_deliveries = request.form['key\_deliveries']

cursor.execute('''

INSERT INTO planning\_data (planning\_id, start\_date, end\_date, key\_deliveries)

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

''', (planning\_id, start\_date, end\_date, key\_deliveries))

connection.commit()

return redirect(url\_for('planning\_index'))

# Add a route to update Planning

@app.route('/update\_planning/<int:planning\_id>', methods=['POST'])

def update\_planning(planning\_id):

data = request.get\_json()

try:

cursor.execute('''

UPDATE planning\_data

SET planning\_id = %s, start\_date = %s, end\_date = %s, key\_deliveries = %s

WHERE id = %s

''', (data['new\_planning\_id'], data['start\_date'], data['end\_date'], data['key\_deliveries'], planning\_id))

connection.commit()

return jsonify({'message': 'Planning updated successfully'})

except Exception as e:

print(e)

return jsonify({'message': 'An error occurred'}), 500

# Add a route to delete Planning

@app.route('/delete\_planning/<int:id>', methods=['POST'])

def delete\_planning(id):

cursor.execute('DELETE FROM planning\_data WHERE id = %s', (id,))

connection.commit()

return jsonify({'message': 'Planning deleted successfully'})

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

app.run(debug=True)