Lập trình web cơ bản

Yêu cầu:Lập trình bằng ngôn ngữ tùy chọn (yêu cầu không sử dụng framework có sẵn), sử dụng DB MySQL để xây dựng website quản lý thông tin sinh viên, tài liệu của 1 lớp học có các chức năng như sau:

1.Giao diện đăng nhập Home

Ở đây tôi sẽ thực hiện chương trình bằng ngôn ngữ **Python**

Những thư viện của python và các cấu hình trong Flask được sử dụng để thực hiện chương trình

```
from flask import Flask, request, render_template, redirect, url_for, flash, session,send_file
import mysql.connector
from werkzeug.security import generate_password_hash, check_password_hash
import os
from werkzeug.utils import secure_filename
from unidecode import unidecode
import xml.etree.ElementTree as ET
import requests
import json
import psycopg2
from flask import send_from_directory

app = Flask(__name__)
app.secret_key = 'your_secret_key'
UPLOAD_FOLDER = 'static/uploads'
ALLOWED_EXTENSIONS = {'json', 'csv'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
app.config['UPLOAD_FOLDER'] = os.path.join(os.getcwd(), 'static', 'uploads')
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
```

Những thư viện Python :

1. Flask

- `Flask` : Một micro-framework Python được sử dụng để xây dựng các ứng dụng web.
- `request` : Xử lý các yêu cầu HTTP (GET, POST, etc.).
- `render_template` : Kết xuất các file HTML với dữ liệu động.
- `redirect` : Chuyển hướng người dùng đến một URL khác.
- `url_for` : Tạo URL động cho các endpoint trong ứng dụng.
- `flash` : Hiển thị thông báo tạm thời cho người dùng (thường là lỗi hoặc thành công).
- `session` : Lưu trữ dữ liệu người dùng trong một phiên làm việc.
- `send_file` : Gửi file cho người dùng để tải về.

2. mysql.connector

- Một thư viện Python để kết nối và làm việc với cơ sở dữ liệu MySQL.
- Hỗ trợ truy vấn, thêm, sửa, xóa dữ liệu từ Python.

3. werkzeug.security

- `generate_password_hash` : Mã hóa mật khẩu để lưu trữ an toàn trong cơ sở dữ liệu.
- `check_password_hash` : Kiểm tra mật khẩu do người dùng nhập vào với mật khẩu đã mã hóa trong cơ sở dữ liệu.

4. os

- Một thư viện chuẩn của Python để làm việc với hệ điều hành.
- Dùng để xử lý đường dẫn file, kiểm tra tồn tại file hoặc thư mục, và các thao tác hệ thống khác.

5. werkzeug.utils

- `secure_filename`: Bảo mật tên file khi người dùng upload. Loại bỏ các ký tự không hợp lệ hoặc nguy hiểm khỏi tên file.

6. unidecode

- Loại bỏ dấu tiếng Việt hoặc các ký tự Unicode để chuyển chuỗi thành dạng ASCII.
- Ví dụ: `"Đức Phúc"` sẽ thành `"Duc Phuc"`.

7. xml.etree.ElementTree

- Một module Python để phân tích cú pháp XML.
- Dùng để đọc, sửa đổi, và tạo các file XML.

8. requests

- Một thư viện mạnh mẽ để gửi các yêu cầu HTTP như GET, POST, PUT, DELETE.
- Thường dùng để lấy dữ liệu từ API hoặc tải nội dung từ các trang web.

9. json

- Thư viện chuẩn của Python để làm việc với dữ liệu JSON.
- Hỗ trợ chuyển đổi giữa Python object và chuỗi JSON (serialization/deserialization).

10. psycopg2

- Một thư viện Python để kết nối và thao tác với cơ sở dữ liệu PostgreSQL.
- Tương tự như `mysql.connector`, nhưng dành cho PostgreSQL.

11. flask.send_from_directory

- `send_from_directory`: Cung cấp file từ một thư mục cụ thể.
- Thường dùng để phục vụ file tĩnh như hình ảnh, PDF, hoặc tài liệu cho người dùng.

Những cấu hình trong Flask:

1. app.secret_key

- **Mô tả:**
  Dùng để mã hóa dữ liệu nhạy cảm như session hoặc flash messages. Giá trị này cần được giữ bí mật để đảm bảo tính bảo mật.
- **Ý nghĩa:**
  Đảm bảo an toàn khi lưu thông tin phiên làm việc (session) và gửi các thông báo flash đến client.

2.UPLOAD_FOLDER

- **Mô tả:**
  Đây là một biến chỉ định thư mục nơi các tệp được tải lên (upload) sẽ được lưu trữ.
- **Ý nghĩa:**
  Flask sẽ sử dụng đường dẫn này để xử lý việc lưu file.

3.ALLOWED_EXTENSIONS

- **Mô tả:**
  Đây là tập hợp các loại file được phép upload (trong ví dụ này là `json` và `csv`).
- **Ý nghĩa:**
  Đảm bảo chỉ cho phép các file với phần mở rộng được chỉ định được tải lên, nhằm mục đích bảo mật.

4.app.config['UPLOAD_FOLDER']

- **Mô tả:**
  Thiết lập cấu hình đường dẫn chính thức cho thư mục upload file. Đoạn `os.path.join(os.getcwd(), 'static', 'uploads')` đảm bảo đường dẫn là

tuyệt đối, dựa trên thư mục làm việc hiện tại.

- **Ý nghĩa:**
  Giúp ứng dụng Flask biết chính xác nơi lưu file.

5.app.config['MAX_CONTENT_LENGTH']

- **Mô tả:**
  Giới hạn kích thước tối đa của file upload (ở đây là 16 MB).
- **Ý nghĩa:**
  Đảm bảo rằng không ai có thể tải lên file quá lớn, tránh tình trạng server bị quá tải.

Đăng nhập:

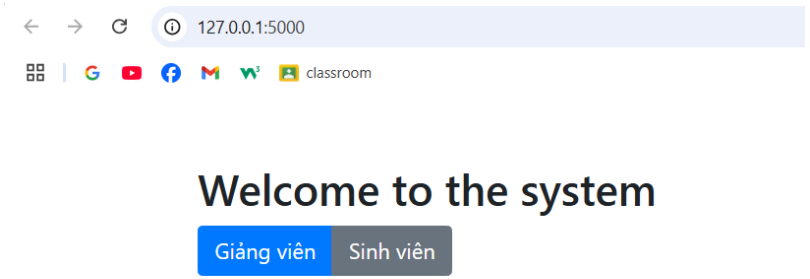
Chúng ta sẽ tạo kết nối với databse và tạo một trang web để lựa chọn đăng nhập giữa giảng viên và sinh viên

```
# Kết nối tới MySQL
def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Phuc00000@",
        database="luutru_thongtin"
    )

@app.route('/')
def home():
    return render_template('index.html')
```

HTML :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h2>Welcome to the system</h2>
        <div class="btn-group">
            <a href="{{ url_for('login_teacher') }}" class="btn btn-primary">Giảng viên</a>
            <a href="{{ url_for('login') }}" class="btn btn-secondary">Sinh viên</a>
        </div>    </div></body>
</html>
```

Khi thực hiện chương trình thì bước đầu tiên sẽ xuất hiện 2 nút để chọn đăng nhập vào bằng tài khoản giảng viên hay tài khoản sinh viên



1.1 Đăng nhập Giảng Viên

Tài khoản của giảng viên sẽ được `admin` cung cập nên chỉ có phần đăng nhập mà không cần đăng ký tài khoản

Đây là cơ sở dữ liệu để lưu tài khoản của giảng viên:

```
CREATE TABLE teachers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    fullname VARCHAR(100) NOT NULL,
    email VARCHAR(100),
    phone VARCHAR(20)
);
```

Code python để thự hiện đăng nhập giáng viên

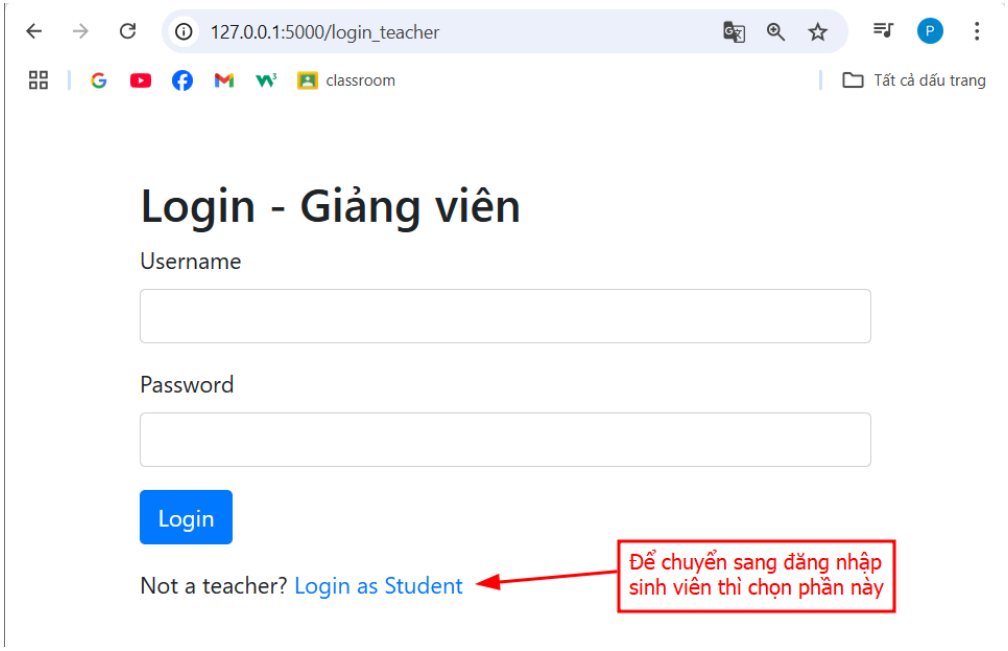
```
# Đăng nhập giảng viên
@app.route('/login_teacher', methods=['GET', 'POST'])
def login_teacher():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Kiểm tra thông tin đăng nhập mặc định
        if username == 'teacher' and password == 'password':  # Logic mẫu, cần điều chỉnh theo thực tế
            session['teacher'] = username  # Lưu thông tin giáo viên vào session
            flash('Login successful!', 'success')
            return redirect(url_for('teacher_dashboard'))

        # Kiểm tra thông tin đăng nhập trong cơ sở dữ liệu
        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)

        cursor.execute("SELECT * FROM teachers WHERE username = %s AND password = %s", (username, password))
        teacher = cursor.fetchone()

        conn.close()

        if teacher:
            session['teacher_id'] = teacher['id']  # Lưu thông tin vào session
            flash('Login successful!', 'success')
            return redirect(url_for('teacher_dashboard'))
        else:
            flash('Invalid credentials. Please try again.', 'danger')

    return render_template('login_teacher.html')
```

HTML để hiển thị sao diện đăng nhập

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login - Giảng viên</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h2>Login - Giảng viên</h2>

        <!-- Form đăng nhập giảng viên -->
        <form action="{{ url_for('login_teacher') }}" method="POST">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control" id="username" name="username" required>
            </div>            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password" name="password" required>
            </div>            <button type="submit" class="btn btn-primary">Login</button>
        </form>
        <!-- Thông báo lỗi nếu có -->
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                <div class="alert alert-danger mt-3">
                    {% for category, message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% endif %}
```

```
        {% endwith %}
        <p class="mt-3">Not a teacher? <a href="{{ url_for('login') }}">Login as Student</a></p>
    </div></body>
</html>
```

Khi thực hiện chương trình sẽ xuất hiện giao diện như thế này:



1.2 Đăng nhập sinh viên

Tải khoản của sinh viên sẽ do sinh viên đăng ký và sẽ được lưu vào cơ sở dữ liệu để thực hiện đăng nhập

```
CREATE TABLE students (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    fullname VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
```

```
    phone VARCHAR(15) NOT NULL
);
```

Code python để thực hiện đăng ký

```
# Đăng ký sinh viên
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        fullname = request.form['fullname']
        email = request.form['email']
        phone = request.form['phone']

        # Tạo kết nối và cursor từ cơ sở dữ liệu
        conn = get_db_connection()
        cursor = conn.cursor()  # Đảm bảo cursor được khởi tạo

        # Mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu        hashed_password = generate_password_hash(password)

        # Kiểm tra nếu tên đăng nhập đã tồn tại
        cursor.execute("SELECT * FROM students WHERE username = %s", (username,))
        user = cursor.fetchone()
        if user:
            flash("Username already exists!", 'danger')
            return render_template('register.html')

        # Thêm thông tin sinh viên vào cơ sở dữ liệu
        cursor.execute("INSERT INTO students (username, password, fullname, email, phone) VALUES (%s, %s, %s, %s, %s)",
                       (username, hashed_password, fullname, email, phone))
        conn.commit()
        flash("Registration successful! Please login.", 'success')
```

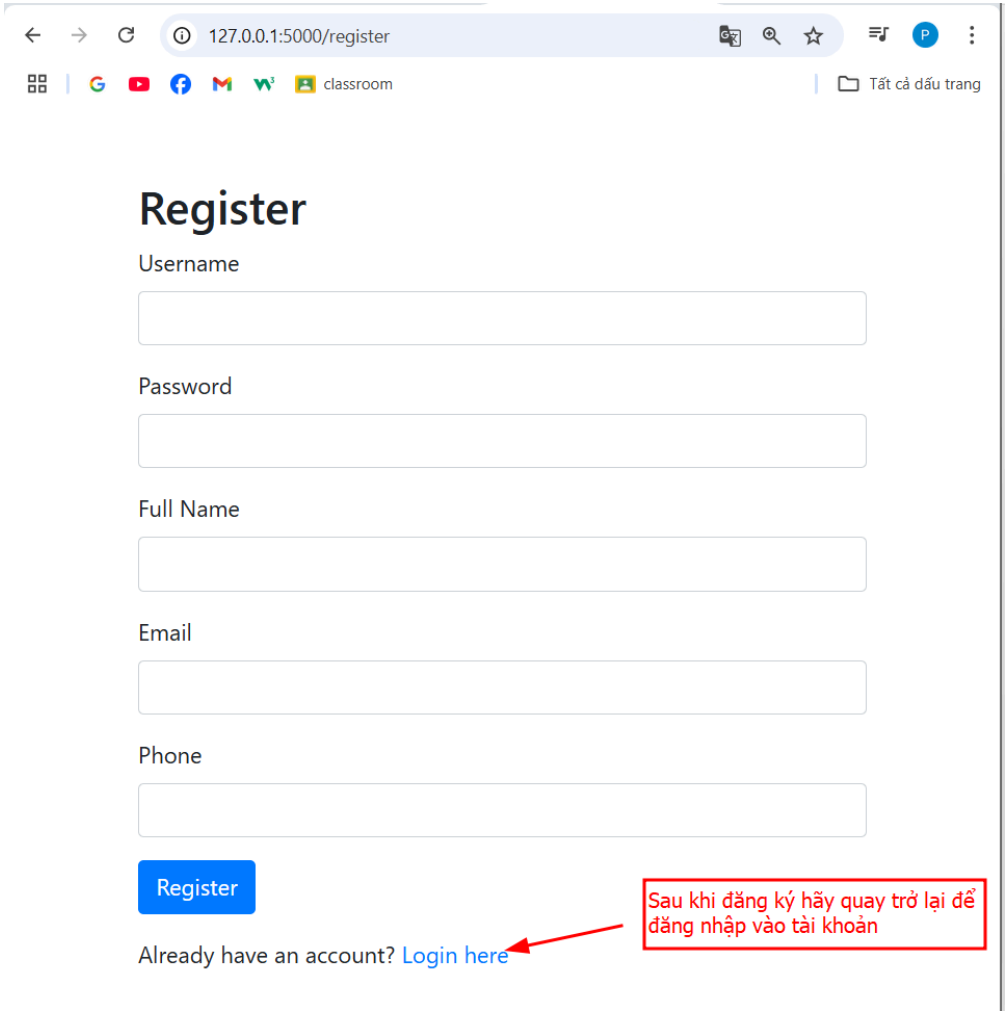
```
        return redirect(url_for('login'))

    return render_template('register.html')
```

HTML đăng ký:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h2>Register</h2>

        <form action="{{ url_for('register') }}" method="POST">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control" id="username" name="username" required>
            </div>            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password" name="password" required>
            </div>            <div class="form-group">
                <label for="fullname">Full Name</label>
                <input type="text" class="form-control" id="fullname" name="fullname" required>
            </div>            <div class="form-group">
                <label for="email">Email</label>
                <input type="email" class="form-control" id="email" name="email" required>
            </div>            <div class="form-group">
                <label for="phone">Phone</label>
```

```
            <input type="text" class="form-control" id="phone" name="phone" required>
        </div>            <button type="submit" class="btn btn-primary">Register</button>
    </form>
    {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
            <div class="alert alert-danger mt-3">
                {% for category, message in messages %}
                    <p>{{ message }}</p>
                {% endfor %}
            </div>
        {% endif %}
    {% endwith %}

    <p class="mt-3">Already have an account? <a href="{{ url_for('login') }}">Login here</a></p>
  </div></body>
</html>
```

Khi thực hiện chương trình thì sinh viên sẽ điền thông tin vào phần đăng ký



Sau khi đăng ký là đến phần đăng nhập cho sinh viên

```
# Đăng nhập sinh viên
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
```

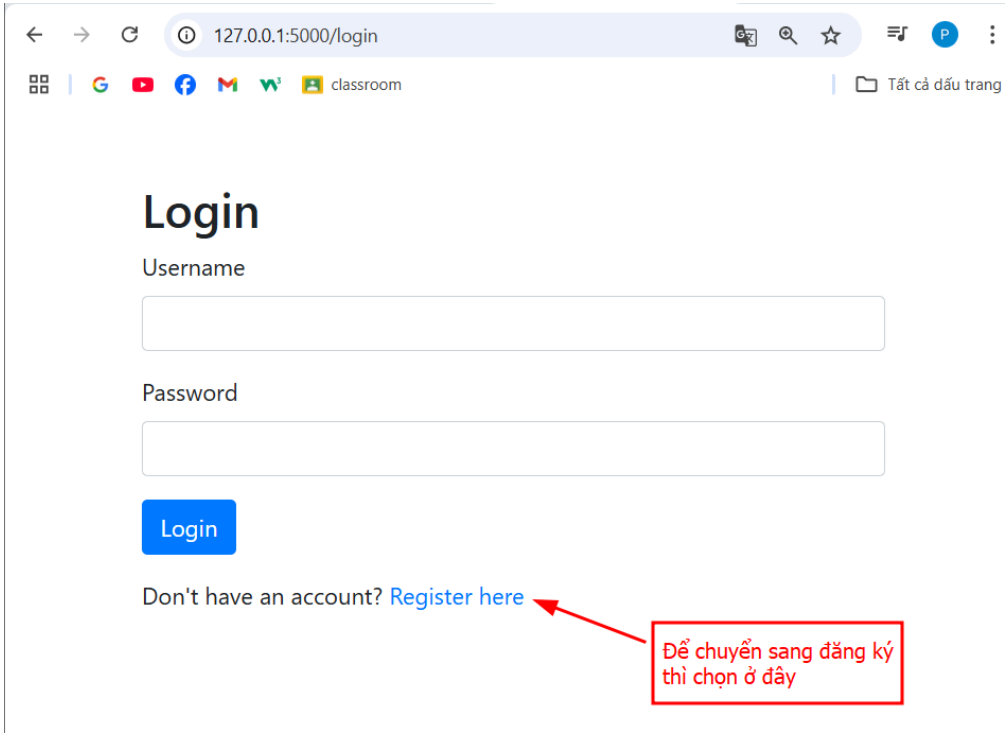
```
        # Tạo kết nối tới cơ sở dữ liệu
        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)  # Đảm bảo kết quả trả về là dictionary

        # Kiểm tra thông tin sinh viên trong cơ sở dữ liệu        cursor.execute("SELECT * FROM students WHERE username = %s", (username,))
        student = cursor.fetchone()

        if student and check_password_hash(student['password'], password):  # Kiểm tra mật khẩu
            # Lưu thông tin sinh viên vào session            session['student_id'] = student['id']
            flash("Login successful!", 'success')

            conn.close()  # Đóng kết nối sau khi hoàn thành
            return redirect(url_for('student_dashboard'))
        else:
            flash("Invalid username or password!", 'danger')
            conn.close()  # Đóng kết nối trong trường hợp thất bại

    return render_template('login.html')
```

HTML để xuất hiện giao diện đăng nhập

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>
<body>
    <div class="container mt-5">
        <h2>Login</h2>
```

```
        <form action="{{ url_for('login') }}" method="POST">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control" id="username" name="username" required>
            </div>            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password" name="password" required>
            </div>            <button type="submit" class="btn btn-primary">Login</button>
        </form>
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                <div class="alert alert-danger mt-3">
                    {% for category, message in messages %}
                        <p>{{ message }}</p>
                    {% endfor %}
                </div>
            {% endif %}
        {% endwith %}

        <p class="mt-3">Don't have an account? <a href="{{ url_for('register') }}">Register here</a></p>
    </div></body>
</html>
```

Khi thực hiện chương trình thì sẽ xuất hiện phần đăng nhập



2.Giao diện

2.1 Giao diện giảng viên : Giáo viên có thể thêm, sửa, xóa các thông tin của sinh viên. Thông tin có các trường cơ bản gồm: tên đăng nhập, mật khẩu, họ tên, email, số điện thoại.

Khi thực hiện đăng nhập vào thì giao diện giảng viên có một bảng sinh viên và giảng viên sẽ được cấp quyền để `Thêm, sửa, xóa` các thông tin của sinh viên

Code python

```
# Thêm sinh viên
@app.route('/add_student', methods=['GET', 'POST'])
def add_student():
    if request.method == 'POST':
```

```
        username = request.form['username']
        password = request.form['password']
        fullname = request.form['fullname']
        email = request.form['email']
        phone = request.form['phone']

        conn = get_db_connection()
        cursor = conn.cursor()

        # Thêm sinh viên vào cơ sở dữ liệu
        try:
            cursor.execute(
                "INSERT INTO students (username, password, fullname, email, phone) VALUES (%s, %s, %s, %s, %s)",
                (username, password, fullname, email, phone)
            )
            conn.commit()
            flash('Student added successfully!', 'success')
            return redirect(url_for('teacher_dashboard'))
        except Exception as e:
            conn.rollback()
            flash(f'Error adding student: {e}', 'danger')
        finally:
            conn.close()

    return render_template('add_student.html')


# Sửa thông tin sinh viên
@app.route('/edit_student/<int:student_id>', methods=['GET', 'POST'])
def edit_student(student_id):
    # Kiểm tra đăng nhập
    if 'student' not in session:
        return redirect(url_for('login'))
```

```
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)  # Sử dụng dictionary=True để trả về dict thay vì tuple

    # Lấy thông tin sinh viên từ CSDL    cursor.execute("SELECT * FROM students WHERE id = %s", (student_id,))
    student = cursor.fetchone()

    if not student:
        flash('Student not found!', 'danger')
        return redirect(url_for('students'))

    if request.method == 'POST':
        # Lấy dữ liệu từ form
        username = request.form['username']
        password = request.form['password']
        fullname = request.form['fullname']
        email = request.form['email']
        phone = request.form['phone']

        try:
            # Xóa các tin nhắn liên quan đến sinh viên trước khi cập nhật
            cursor.execute("""
                DELETE FROM messages              WHERE sender_username = %s OR recipient_username = %s
            """, (student['username'], student['username']))
            connection.commit()

            # Tiến hành cập nhật thông tin sinh viên
            cursor.execute("""
                UPDATE students              SET username = %s, password = %s, fullname = %s, email = %s, phone = %s              WHERE id = %s          """,
(username, password, fullname, email, phone, student_id))
            connection.commit()

            flash('Student information updated successfully!', 'success')
            return redirect(url_for('students'))
        except Exception as e:
```

```
            connection.rollback()
            flash(f'An error occurred: {e}', 'danger')

    return render_template('edit_student.html', student=student)


# Xóa sinh viên
@app.route('/delete_student/<int:student_id>', methods=['POST'])
def delete_student(student_id):
    # Kiểm tra phương thức '_method' từ form
    if request.form.get('_method') == 'DELETE':
        conn = get_db_connection()
        cursor = conn.cursor()

        # Xóa thông tin sinh viên trong bảng students
        cursor.execute("DELETE FROM students WHERE id = %s", (student_id,))
        conn.commit()
        conn.close()

        flash('Student deleted successfully!', 'success')
        return redirect(url_for('teacher_dashboard'))
```

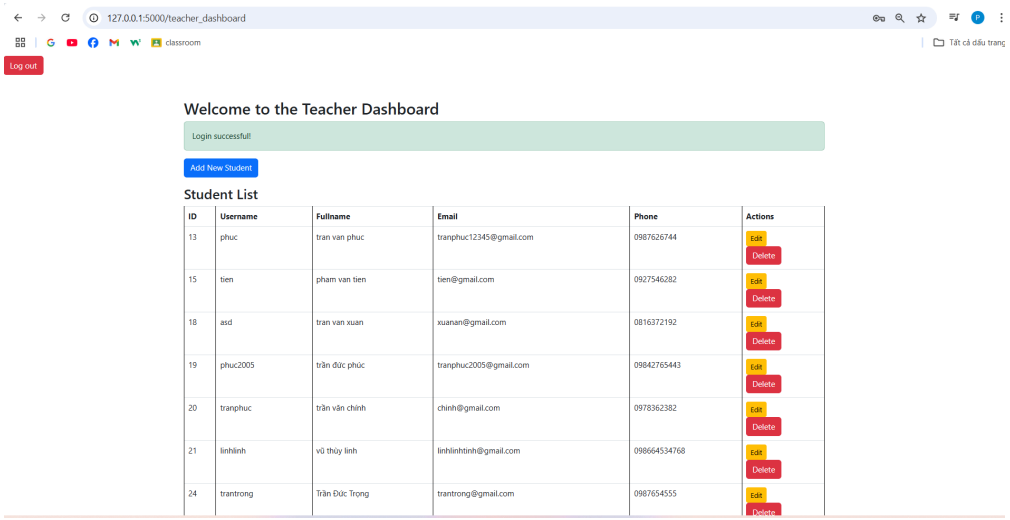
HTML **Thêm, sửa, xóa**

```
# Thêm sinh viên
<body>
    <div class="container mt-5">
        <h1>Add New Student</h1>
        <form action="{{ url_for('add_student') }}" method="POST">
            <div class="mb-3">
                <label for="username" class="form-label">Username</label>
                <input type="text" class="form-control" id="username" name="username" required>
            </div>          <div class="mb-3">
```

```
            <label for="password" class="form-label">Password</label>
            <input type="password" class="form-control" id="password" name="password" required>
        </div>          <div class="mb-3">
            <label for="fullname" class="form-label">Fullname</label>
            <input type="text" class="form-control" id="fullname" name="fullname" required>
        </div>          <div class="mb-3">
            <label for="email" class="form-label">Email</label>
            <input type="email" class="form-control" id="email" name="email" required>
        </div>          <div class="mb-3">
            <label for="phone" class="form-label">Phone</label>
            <input type="text" class="form-control" id="phone" name="phone" required>
        </div>          <button type="submit" class="btn btn-success">Add Student</button>
    </form>
  </div>
</body>

# Sửa sinh viên
<body>
  <div class="container mt-5">
    <h2>Edit Student Information</h2>

    <!-- Flash Messages -->
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        <div>
          {% for category, message in messages %}
            <div class="alert alert-{{ category }}">{{ message }}</div>
          {% endfor %}
        </div>
      {% endif %}
    {% endwith %}

    <!-- Edit Student Form -->
    <form action="/edit_student/{{ student.id }}" method="POST">
```

```
        <div class="mb-3">
            <label for="username" class="form-label">Username</label>
            <input type="text" class="form-control" id="username" name="username" value="{{ student.username }}" required>
        </div>          <div class="mb-3">
            <label for="password" class="form-label">Password</label>
            <input type="password" class="form-control" id="password" name="password" value="{{ student.password }}" required>
        </div>          <div class="mb-3">
            <label for="fullname" class="form-label">Fullname</label>
            <input type="text" class="form-control" id="fullname" name="fullname" value="{{ student.fullname }}" required>
        </div>          <div class="mb-3">
            <label for="email" class="form-label">Email</label>
            <input type="email" class="form-control" id="email" name="email" value="{{ student.email }}" required>
        </div>          <div class="mb-3">
            <label for="phone" class="form-label">Phone</label>
            <input type="text" class="form-control" id="phone" name="phone" value="{{ student.phone }}" required>
        </div>          <button type="submit" class="btn btn-primary">Update Student</button>
        <a href="{{ url_for('students') }}" class="btn btn-secondary">Thoát</a> <!-- Nút Thoát -->
    </form>
  </div>
</body>

# Xóa sinh viên
<form action="{{ url_for('delete_student', student_id=student.id) }}" method="POST">
  <input type="hidden" name="_method" value="DELETE">
  <button type="submit" class="btn btn-danger">Delete</button>
</form>
```

Khi chạy chương trình sẽ trông như thế này



2.2 Giao diện sinh viên: Sinh viên được phép thay đổi các thông tin của mình trừ tên đăng nhập và họ tên.

Khi sinh viên đăng nhập vào thì sẽ hiện lên thông tin của bản thân và sinh viên có thể thay đổi thông tin của bản thân mình

Code Python:

```
# Chỉnh sửa thông tin cá nhân của sinh viên
@app.route('/edit_profile', methods=['GET', 'POST'])
def edit_profile():
    if 'student' not in session:
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)  # Đảm bảo cursor là dictionary

    # Lấy thông tin sinh viên đã đăng nhập    cursor.execute("SELECT * FROM students WHERE username = %s", (session['student'],))
    student = cursor.fetchone()

    if request.method == 'POST':
```

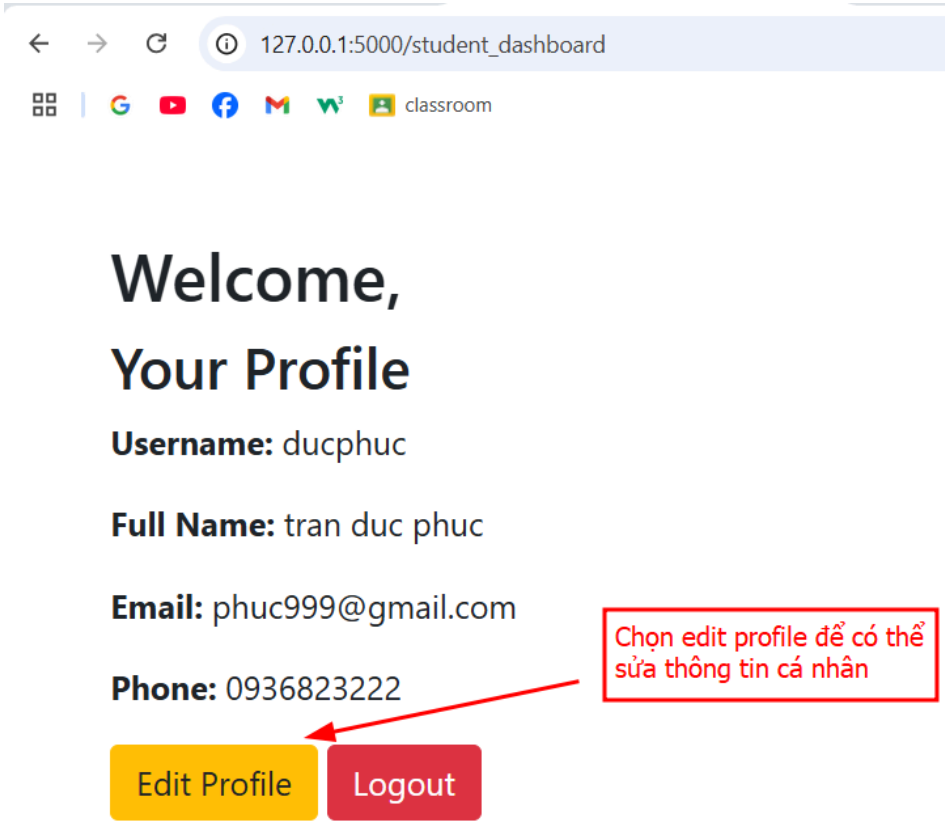
```
        username = request.form['username']
        password = request.form['password']
        fullname = request.form['fullname']
        email = request.form['email']
        phone = request.form['phone']

        # Mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu
        hashed_password = generate_password_hash(password)

        # Cập nhật thông tin sinh viên
        cursor.execute("UPDATE students SET password = %s, fullname = %s, email = %s, phone = %s WHERE username = %s",
                       (hashed_password, fullname, email, phone, session['student']))
        conn.commit()
        flash("Profile updated successfully!", 'success')
        return redirect(url_for('student_dashboard'))

    return render_template('edit_profile.html', student=student)
```

HTML để hiển thị lên thông tin cá nhân

```
<body>
    <div class="container mt-5">
        <h2>Edit Profile</h2>

        <form action="{{ url_for('edit_profile') }}" method="POST">
            <div class="form-group">
                <label for="username">Username</label>
                <input type="text" class="form-control" id="username" name="username" value="{{ student[1] }}" readonly>
            </div>            <div class="form-group">
                <label for="password">Password</label>
                <input type="password" class="form-control" id="password" name="password" value="{{ student[2] }}">
            </div>            <div class="form-group">
                <label for="fullname">Full Name</label>
```

```
            <input type="text" class="form-control" id="fullname" name="fullname" value="{{ student[3] }}" required>
        </div>            <div class="form-group">
            <label for="email">Email</label>
            <input type="email" class="form-control" id="email" name="email" value="{{ student[4] }}" required>
        </div>            <div class="form-group">
            <label for="phone">Phone</label>
            <input type="text" class="form-control" id="phone" name="phone" value="{{ student[5] }}" required>
        </div>            <button type="submit" class="btn btn-primary">Update</button>
    </form>
  </div>
</body>
```

Khi thực hiện kết quả sẽ là:



3. Phần để lại lời nhắn: 1 người dùng bất kỳ đc phép xem danh sách các người dùng trên website và xem thông tin chi tiết của 1 người dùng khác. Tại trang xem thông tin chi tiết của 1 người dùng có mục để lại tin nhắn cho người dùng đó, có thể sửa/xóa tin nhắn đã gửi.

Phần này trong giao diện sinh viên chúng ta sẽ cho hiển thị một bảng sinh viên và có phần nhắn tin để cho sinh viên tương tác với nhanh

MySQL để gửi trữ tin nhắn:

```
CREATE TABLE messages (
    id INT AUTO_INCREMENT PRIMARY KEY,
    sender_username VARCHAR(255),
    recipient_username VARCHAR(255),
    message TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_read BOOLEAN DEFAULT FALSE,  -- Đánh dấu tin nhắn đã đọc hay chưa
    FOREIGN KEY (sender_username) REFERENCES students(username),
    FOREIGN KEY (recipient_username) REFERENCES students(username)ON DELETE SET NULL
);
```

Code Python:

```
@app.route('/student_dashboard', methods=['GET'])
def student_dashboard():
    if 'student' not in session:
        flash("Bạn cần đăng nhập trước khi truy cập bảng điều khiển!", 'warning')
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    try:
        # Lấy thông tin sinh viên đã đăng nhập
        cursor.execute("SELECT * FROM students WHERE username = %s", (session['student'],))
        student = cursor.fetchone()

        if not student:
            flash("Sinh viên không tìm thấy!", 'danger')
            return redirect(url_for('login'))

        # Lấy danh sách sinh viên cùng lớp
```

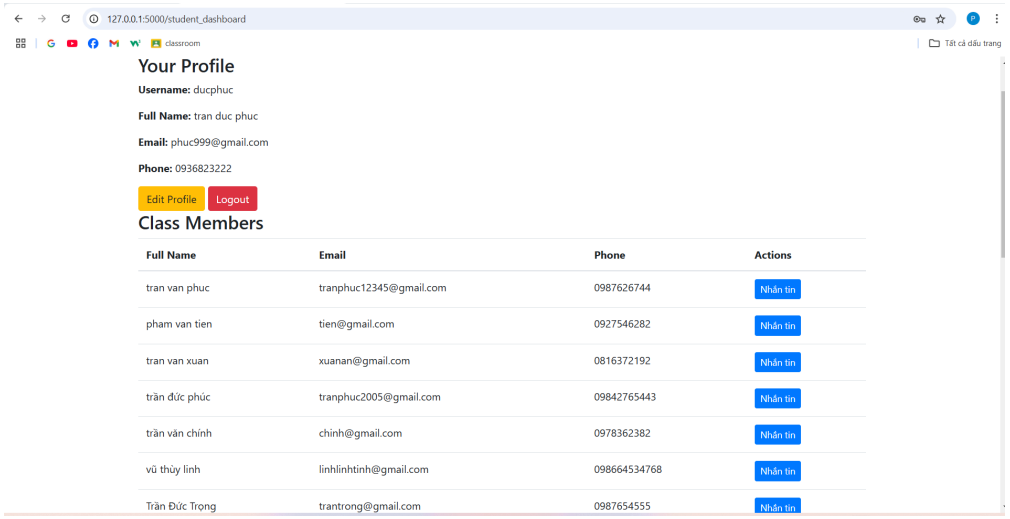
```
        cursor.execute("SELECT fullname, email, phone FROM students WHERE class = %s", (student['class'],))
        students_list = cursor.fetchall()

        return render_template(
            'student_dashboard.html',
            student=student,
            students_list=students_list
        )
    finally:
        cursor.close()
        conn.close()
```

HTML:

```
<!-- Danh sách sinh viên -->
<h3>Class Members</h3>
<table class="table">
    <thead>
        <tr>
            <th>Full Name</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
         {% for s in students_list %}
        <tr>
            <td>{{ s['fullname'] }}</td>  <!-- Thay s[3] thành s['full_name'] -->
            <td>{{ s['email'] }}</td>  <!-- Thay s[4] thành s['email'] -->
            <td>{{ s['phone'] }}</td>  <!-- Thay s[5] thành s['phone'] -->
            <td>
```

```
                <a href="{{ url_for('send_message', recipient_username=s['username']) }}" class="btn btn-primary btn-sm">
                    Nhắn tin
                </a>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>

<!-- Flash Messages -->
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        <div>
            {% for category, message in messages %}
                <div class="alert alert-{{ category }}">{{ message }}</div>
            {% endfor %}
        </div>
    {% endif %}
{% endwith %}
```

Kết quả



Để gửi lại tin nhắn hãy chọn vào phần vào phần nhắn tin, tin nhắn có thể sửa và xóa bởi người gửi

Code Python

```
# Gửi tin nhắn
@app.route('/send_message', methods=['GET', 'POST'])
def send_message():
    if 'student' not in session:
        return redirect(url_for('login'))

    sender_username = session['student']
    recipient_username = request.args.get('recipient_username')

    # Khởi tạo kết nối và cursor
    connection = get_db_connection()
    cursor = connection.cursor()

    # Lấy thông tin người nhận
    cursor.execute("SELECT * FROM students WHERE username = %s", (recipient_username,))
    recipient = cursor.fetchone()  # Lấy thông tin của người nhận
```

```
    if request.method == 'POST':
        message_content = request.form['message_content']
        # Lưu tin nhắn vào cơ sở dữ liệu
        cursor.execute("""
            INSERT INTO messages (sender_username, recipient_username, message)           VALUES (%s, %s, %s)        """, (sender_username,
recipient_username, message_content))
        connection.commit()  # Commit sau khi thực hiện thay đổi trong cơ sở dữ liệu
        flash("Message sent successfully!", 'success')

    # Lấy tất cả tin nhắn giữa người gửi và người nhận
    cursor.execute("""
        SELECT * FROM messages        WHERE (sender_username = %s AND recipient_username = %s)        OR (sender_username = %s AND
recipient_username = %s)        ORDER BY created_at DESC     """, (sender_username, recipient_username, recipient_username, sender_username))

    conversation = cursor.fetchall()

    # Đóng kết nối và cursor sau khi hoàn thành
    cursor.close()
    connection.close()

    # Trả về giao diện gửi tin nhắn với các tin nhắn đã được gửi và thông tin người nhận
    return render_template('send_message.html', recipient=recipient, conversation=conversation)

  # Hiển thị tin nhắn
@app.route('/messages', methods=['GET'])
def messages():
    if 'student' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    cursor = connection.cursor()
    username = session['student']

    # Lấy tin nhắn đã gửi và nhận
```

```
    cursor.execute("SELECT * FROM messages WHERE sender_username = %s OR recipient_username = %s ORDER BY created_at DESC",
                   (username, username))
    messages_list = cursor.fetchall()
    return render_template('messages.html', messages_list=messages_list, username=username)

  # xóa tin nhăn
@app.route('/delete_message/<int:message_id>', methods=['POST'])
def delete_message(message_id):
    if 'student' not in session:
        return redirect(url_for('login'))

    connection = get_db_connection()
    cursor = connection.cursor()
    sender_username = session['student']

    # Chỉ người gửi được phép xóa tin nhắn
    cursor.execute("DELETE FROM messages WHERE id = %s AND sender_username = %s", (message_id, sender_username))
    connection.commit()
    flash("Message deleted successfully!", 'success')
    return redirect(request.referrer)
```

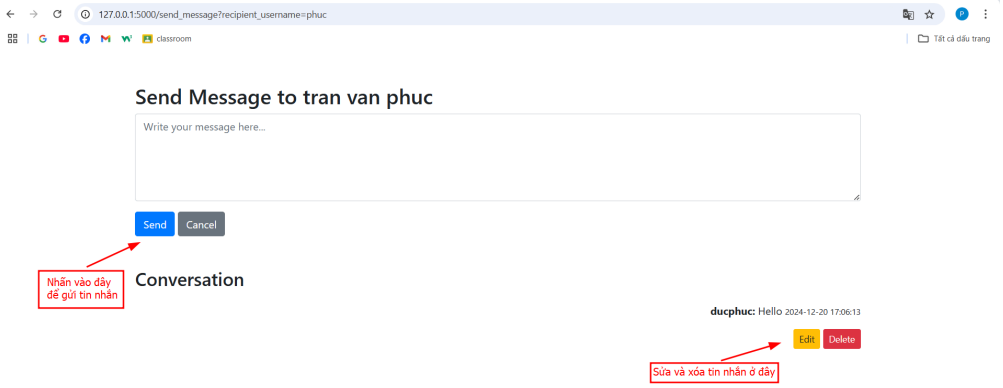
HTML của phần nhắn tin :

```
<body>
    <div class="container mt-5">
        <h2>Send Message to {{ recipient[3] }}</h2>

        <!-- Form gửi tin nhắn -->
        <form method="POST">
            <input type="hidden" name="message_id" id="message_id">
            <div class="form-group">
                <textarea name="message_content" id="message_content" class="form-control" rows="5" placeholder="Write your message here..." required>
</textarea>
```

```
        </div>            <button type="submit" class="btn btn-primary">Send</button>
        <a href="{{ url_for('student_dashboard') }}" class="btn btn-secondary">Cancel</a>
    </form>
    <!-- Tin nhắn giữa người gửi và người nhận -->
    <h3 class="mt-5">Conversation</h3>
    <div class="chat-box">
        {% for msg in conversation %}
            <div class="message {% if msg[1] == session['student'] %}sent{% else %}received{% endif %}">
                <p><strong>{{ msg[1] }}:</strong> {{ msg[3] }} <small>{{ msg[4] }}</small></p>
                {% if msg[1] == session['student'] %}  <!-- Chỉ hiển thị nút Sửa và Xóa nếu là người gửi -->
                    <button class="btn btn-warning btn-sm edit-btn" data-message-id="{{ msg[0] }}" data-message="{{ msg[3] }}">Edit</button>
                    <form action="{{ url_for('delete_message', message_id=msg[0]) }}" method="POST" style="display:inline;">
                        <button type="submit" class="btn btn-danger btn-sm">Delete</button>
                    </form>                {% endif %}
            </div>
        {% endfor %}
    </div>
</div>
<script>        // Script để điền nội dung tin nhắn vào form khi nhấn "Edit"
    document.querySelectorAll('.edit-btn').forEach(button => {
        button.addEventListener('click', () => {
            const messageId = button.getAttribute('data-message-id');
            const messageContent = button.getAttribute('data-message');
            document.getElementById('message_id').value = messageId;
            document.getElementById('message_content').value = messageContent;
        });
    });
</script>

<style>        .message {
        margin-bottom: 15px;
    }

    .sent {
```

```
            text-align: right;
        }

        .received {
            text-align: left;
        }

        .chat-box {
            margin-top: 20px;
        }
    </style>
</body>
```

Hãy thực hiện nhắn tin với người bạn muốn nhắn ở đây



4. Phần bài tập

4.1 Giảng Viên

Chức năng giao bài dành cho giảng viên gồm có:

- Gửi bài tập cho sinh có phần title, decription, upload file

- Phần xem lại danh sách bài tập đã giao và có thể sửa lại một số thông tin nếu muốn
- Phần bài tập của sinh viên gửi lên
  MySQL để lưu trữ bài tập đã gửi:

```
CREATE TABLE BAITAP (
    id INT AUTO_INCREMENT PRIMARY KEY,
    tieude VARCHAR(255) NOT NULL,
    mota TEXT,
    ten_file VARCHAR(255),
    ngay_tao DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Code Python để có thể thực hiện đăng bài tập

```
# Thêm bài tập mới
elif 'tieude' in request.form:
    tieude = request.form['tieude']
    mota = request.form['mota']
    ten_file = request.files['ten_file']

    # Kiểm tra xem thư mục 'uploads' đã tồn tại chưa
    if not os.path.exists('uploads'):
        os.makedirs('uploads')
    # Lưu file vào thư mục 'uploads'
    file_path = f'uploads/{ten_file.filename}'
    ten_file.save(file_path)

    cursor.execute("INSERT INTO BAITAP (tieude, mota, ten_file) VALUES (%s, %s, %s)",
                   (tieude, mota, file_path))
    conn.commit()
    flash('Assignment added successfully!', 'success')
```

```
# giáo viên xóa bài tập
@app.route('/delete_assignment/<int:assignment_id>', methods=['POST'])
def delete_assignment(assignment_id):
    if 'teacher_id' not in session:
        flash('You need to log in first!', 'danger')
        return redirect(url_for('login_teacher'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Xóa bài tập khỏi cơ sở dữ liệu
    cursor.execute("DELETE FROM BAITAP WHERE id = %s", (assignment_id,))
    conn.commit()
    conn.close()

    flash('Assignment deleted successfully!', 'success')
    return redirect(url_for('teacher_dashboard'))  # Quay lại giao diện giảng viên

# giao viên sửa đề bài

# Hàm lấy thông tin bài tập theo assignment_id
def get_assignment_by_id(assignment_id):
    conn = get_db_connection()  # Kết nối đến cơ sở dữ liệu
    cursor = conn.cursor(dictionary=True)  # Đảm bảo kết quả trả về là dictionary
    cursor.execute('SELECT * FROM BAITAP WHERE id = %s', (assignment_id,))
    assignment = cursor.fetchone()  # Lấy một bài tập duy nhất
    conn.close()  # Đóng kết nối
    return assignment

# Hàm cập nhật thông tin bài tập
def update_assignment(assignment_id, title, description, file_name):
    conn = get_db_connection()  # Kết nối đến cơ sở dữ liệu
    cursor = conn.cursor()  # Tạo đối tượng cursor để thực thi câu lệnh SQL
```

```
    cursor.execute('''
        UPDATE BAITAP       SET tieude = %s, mota = %s, ten_file = %s       WHERE id = %s    ''', (title, description, file_name, assignment_id))
    conn.commit()  # Lưu thay đổi vào cơ sở dữ liệu
    conn.close()  # Đóng kết nối

# Sửa lại bài tập
@app.route('/edit_assignment/<int:assignment_id>', methods=['GET', 'POST'])
def edit_assignment(assignment_id):
    # Lấy thông tin bài tập từ cơ sở dữ liệu theo assignment_id
    assignment = get_assignment_by_id(assignment_id)

    if request.method == 'POST':
        # Xử lý cập nhật bài tập
        title = request.form['title']
        description = request.form['description']
        file_name = request.form['file_name']
        # Lưu các thay đổi vào cơ sở dữ liệu
        update_assignment(assignment_id, title, description, file_name)
        flash('Assignment updated successfully!', 'success')
        return redirect(url_for('teacher_dashboard'))  # Quay lại giao diện giảng viên

    # Nếu là GET, hiển thị form với dữ liệu bài tập hiện tại     return render_template('edit_assignment.html', assignment=assignment,
assignment_id=assignment_id)
```

HTML để hiển thị ra giao diện đăng bài tập

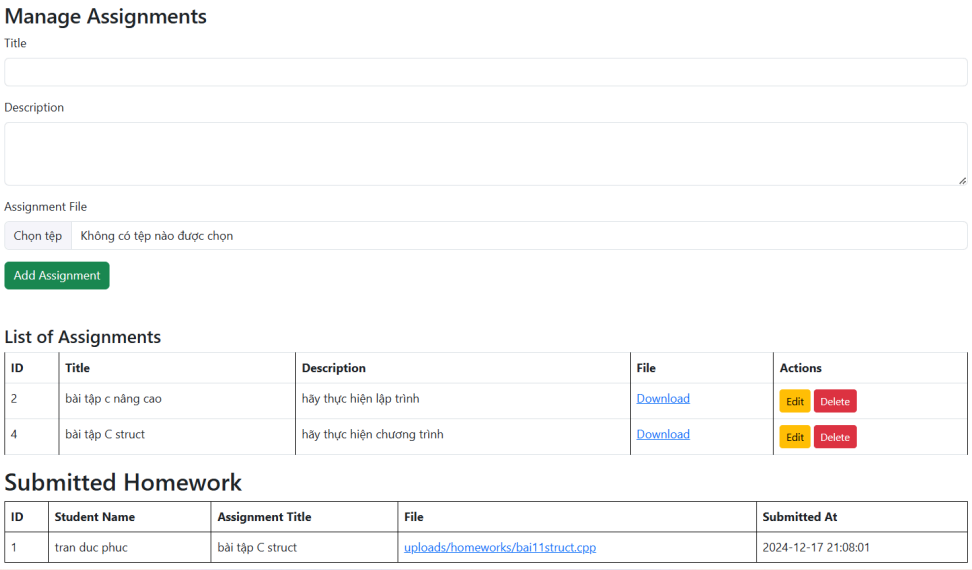
```
<!-- Assignment Section -->
<h3>Manage Assignments</h3>

<!-- Add Assignment Form -->
<form action="/teacher_dashboard" method="POST" enctype="multipart/form-data">
    <div class="mb-3">
        <label for="tieude" class="form-label">Title</label>
```

```
        <input type="text" class="form-control" id="tieude" name="tieude" required>
    </div>    <div class="mb-3">
        <label for="mota" class="form-label">Description</label>
        <textarea class="form-control" id="mota" name="mota" rows="3" required></textarea>
    </div>    <div class="mb-3">
        <label for="ten_file" class="form-label">Assignment File</label>
        <input type="file" class="form-control" id="ten_file" name="ten_file" required>
    </div>    <button type="submit" class="btn btn-success">Add Assignment</button>
</form>

<!-- Assignment Table -->
<h4 class="mt-5">List of Assignments</h4>
<table class="table table-bordered">
    <thead>
            <tr>
                    <th>ID</th>
            <th>Title</th>
            <th>Description</th>
            <th>File</th>
            <th>Actions</th>
        </tr>
     </thead>
     <tbody>
             {% for assignment in assignments %}
        <tr>
            <td>{{ assignment.id }}</td>
            <td>{{ assignment.tieude }}</td>
            <td>{{ assignment.mota }}</td>
            <td><a href="{{ assignment.ten_file }}" target="_blank">Download</a></td>
            <td>
                    <!-- Edit Button -->
                <a href="/edit_assignment/{{ assignment.id }}" class="btn btn-warning btn-sm">Edit</a>
                <!-- Delete Button -->
                <form action="/delete_assignment/{{ assignment.id }}" method="POST" style="display:inline;">
```

```
                    <button type="submit" class="btn btn-danger btn-sm">Delete</button>
                </form>
            </td>
        </tr>
         {% endfor %}
    </tbody>
</table>

<h2>Submitted Homework</h2>

<style>
    table {
        width: 100%;
        border-collapse: collapse;
    }
    table, th, td {
        border: 1px solid black;
    }
    th, td {
        padding: 8px;
        text-align: left;
    }
</style>
<table>
    <thead>
            <tr>
                    <th>ID</th>
            <th>Student Name</th>
            <th>Assignment Title</th>
            <th>File</th>
            <th>Submitted At</th>
        </tr>
    </thead>
     <tbody>
```

```
            {% for homework in homeworks %}
        <tr>
            <td>{{ homework.bailam_id }}</td>
            <td>{{ homework.fullname }}</td>
            <td>{{ homework.tieude }}</td>
            <td>
            <a href="{{ url_for('static', filename=homework.ten_file) }}" download>{{ homework.ten_file }}</a>
            </td>
         <td>{{ homework.ngay_tao }}</td>
        </tr>
        {% endfor %}
    </tbody>
</table>
```

Kết quả của phần giao bài tập



4.2 Sinh viên

Đối với phần bài tập của sinh viên chúng ta sẽ có một bảng hiển thị bài tập mà Giảng viên đã giao.

- Chúng ta tiến hành Download file đề bài mà Giảng viên đã gửi.
- Sau khi làm bài tập xong thì chúng ta có thể upload bài tập lên gửi cho giảng viên.
- Bài tập gửi cho giảng viên sẽ được lưu trong phần bài làm

MySQL:

```
CREATE TABLE BAILAM (
    id INT AUTO_INCREMENT PRIMARY KEY,
    id_baitap INT NOT NULL,
    id_sinhvien INT NOT NULL,
    ten_file VARCHAR(255),
    ngay_tao DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_baitap) REFERENCES BAITAP(id),
    FOREIGN KEY (id_sinhvien) REFERENCES students(id)
);
```

Code Python để có thể hiển thị được bảng bài tập, download bài tập, upload bài tập

```
# Danh sách bài tập
@app.route('/list_baitap')
def list_baitap():
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM BAITAP")
    baitaps = cursor.fetchall()
    cursor.close()
    connection.close()

    return render_template('list_baitap.html', baitaps=baitaps)
```

```python
 # Lưu file về
@app.route('/download_assignment/<filename>')
def download_assignment(filename):
    # Đảm bảo file được lưu trong thư mục đúng
    upload_folder = os.path.join(app.root_path, 'static/uploads')

    # Kiểm tra xem file có tồn tại không
    if os.path.exists(os.path.join(upload_folder, filename)):
        return send_from_directory(upload_folder, filename, as_attachment=True)
    else:
        flash('File not found', 'danger')
        return redirect(url_for('student_dashboard'))

# Sinh viên upload bài làm
@app.route('/upload_homework/<int:assignment_id>', methods=['POST'])
def upload_homework(assignment_id):
    # Xử lý file tải lên
    homework_file = request.files['homework_file']
    if homework_file.filename == '':
        flash('No file selected', 'danger')
        return redirect(request.url)

    # Tạo đường dẫn lưu file
    file_path = f'uploads/homeworks/{homework_file.filename}'
    homework_file.save(file_path)

    # Lưu bài làm vào cơ sở dữ liệu
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute(
        "INSERT INTO BAILAM (id_baitap, id_sinhvien, ten_file) VALUES (%s, %s, %s)",
        (assignment_id, session['student_id'], file_path)
    )
    conn.commit()
```

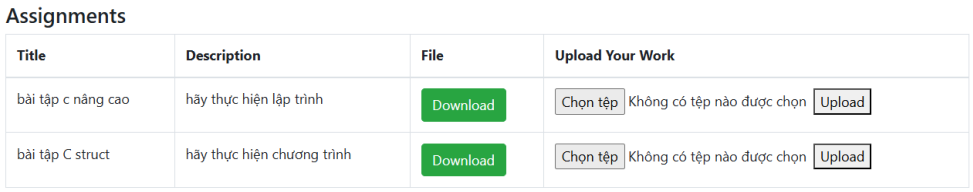
```
    flash('Homework uploaded successfully!', 'success')
    conn.close()
    return redirect(url_for('student_dashboard'))
```

HTML Để hiển thị bảng bài tập:

```
<!-- Bài tập -->
<h4>Assignments</h4>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Title</th>
        <th>Description</th>
        <th>File</th>
        <th>Upload Your Work</th>
      </tr>
    </thead>
    <tbody>
        {% for assignment in assignments %}
        <tr>
            <td>{{ assignment['tieude'] }}</td>  <!-- Hiển thị tiêu đề bài tập -->
            <td>{{ assignment['mota'] }}</td>  <!-- Hiển thị mô tả bài tập -->
            <td>
                {% if assignment['ten_file'] %}
                    <!-- Hiển thị nút download nếu có file -->
                    <a href="{{ url_for('download_assignment', filename=assignment['ten_file']) }}" class="btn btn-success" download>Download</a>
                {% else %}
                    <span>No file available</span>
                {% endif %}
            </td>
            <td>
                <form action="/upload_homework/{{ assignment.id }}" method="POST" enctype="multipart/form-data">
```

```
                <input type="file" name="homework_file" required>
                <button type="submit">Upload</button>
            </form>
        </td>
    </tr>
    {% endfor %}
</tbody>
</table>
```

Kết quả



5. Phần thử thách

5.1 Giảng viên

Giáo viên tạo challenge, trong đó cần thực hiện: upload lên 1 file txt có nội dung là 1 bài thơ, văn,…, tên file được viết dưới định dạng không dấu và các từ cách nhau bởi 1 khoảng trắng. Sau đó nhập gợi ý về quyển sách và submit. (Đáp án chính là tên file mà giáo viên upload lên. Không lưu đáp án ra file, DB,…)

MySQL để lưu trữ thử thách

```
CREATE TABLE challenges (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    file_path VARCHAR(255) NOT NULL,
```

```
    hint TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

Code Python để thực hiện chương trình

```
# Thêm thử thách mới
elif 'challenge_file' in request.files:
    # Lấy dữ liệu từ form
    title = request.form['challenge_title']
    description = request.form['challenge_description']
    hint = request.form['challenge_hint']
    file = request.files['challenge_file']

    # Xử lý file
    if file and file.filename.endswith('.txt'):
        original_filename = secure_filename(file.filename)
        no_diacritics_filename = unidecode(original_filename).replace(" ", "_")
        challenge_folder = 'uploads\challenges'

        # Tạo thư mục nếu chưa có
        os.makedirs(challenge_folder, exist_ok=True)
        file_path = os.path.join(challenge_folder, no_diacritics_filename)
        file.save(file_path)

        # Lưu thử thách vào database
        cursor.execute(
            "INSERT INTO challenges (title, description, file_path, hint) VALUES (%s, %s, %s, %s)",
            (title, description, file_path, hint)
        )
        conn.commit()
```

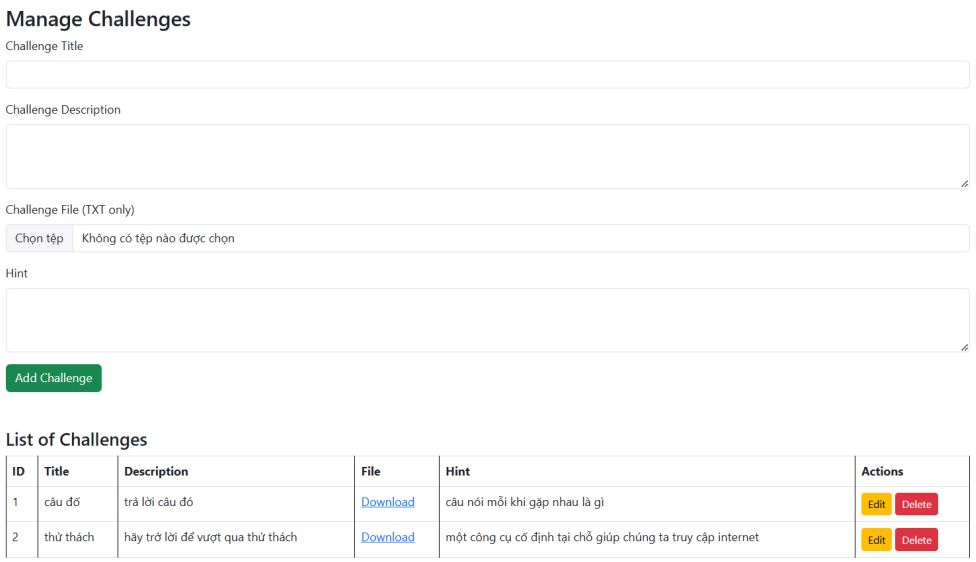
```
            flash('Challenge added successfully!', 'success')
        else:
            flash('Invalid file format. Only .txt files are allowed.', 'danger')

        # Lấy danh sách thử thách
        cursor.execute("SELECT * FROM challenges")
        challenges = cursor.fetchall()
```

HTML để hiển thị phần thêm thử thách và bảng lưu thử thách

```
<!-- Challenge Section -->
<h3 class="mt-5">Manage Challenges</h3>

<!-- Add Challenge Form -->
<form action="/teacher_dashboard" method="POST" enctype="multipart/form-data">
    <div class="mb-3">
        <label for="challenge_title" class="form-label">Challenge Title</label>
        <input type="text" class="form-control" id="challenge_title" name="challenge_title" required>
    </div>
     <div class="mb-3">
        <label for="challenge_description" class="form-label">Challenge Description</label>
        <textarea class="form-control" id="challenge_description" name="challenge_description" rows="3" required></textarea>
    </div>
    <div class="mb-3">
        <label for="challenge_file" class="form-label">Challenge File (TXT only)</label>
        <input type="file" class="form-control" id="challenge_file" name="challenge_file" accept=".txt" required>
    </div>
    <div class="mb-3">
        <label for="challenge_hint" class="form-label">Hint</label>
        <textarea class="form-control" id="challenge_hint" name="challenge_hint" rows="3" required></textarea>
    </div>
    <button type="submit" class="btn btn-success">Add Challenge</button>
</form>
```

```
<!-- Challenge Table -->
<h4 class="mt-5">List of Challenges</h4>
<table class="table table-bordered">
    <thead>
            <tr>
                    <th>ID</th>
            <th>Title</th>
            <th>Description</th>
            <th>File</th>
            <th>Hint</th>
            <th>Actions</th>
        </tr>
    </thead>
        <tbody>
                {% for challenge in challenges %}
        <tr>
            <td>{{ challenge.id }}</td>
            <td>{{ challenge.title }}</td>
            <td>{{ challenge.description }}</td>
            <td><a href="{{ challenge.file }}" target="_blank">Download</a></td>
            <td>{{ challenge.hint }}</td>
            <td>
                    <!-- Edit Button -->
                <a href="/edit_challenge/{{ challenge.id }}" class="btn btn-warning btn-sm">Edit</a>
                <!-- Delete Button -->
                <form action="/delete_challenge/{{ challenge.id }}" method="POST" style="display:inline;">
                    <button type="submit" class="btn btn-danger btn-sm">Delete</button>
                </form>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
```

Kết quả



5.2 Sinh Viên

Sinh viên xem gợi ý và nhập đáp án. Khi sinh viên nhập đúng thì trả về nội dung bài thơ, văn,… lưu trong file đáp án.

Code python để thực hiện chương trình

```
if student:
    # Lấy danh sách các thử thách
    cursor.execute("SELECT id, title, hint FROM challenges")
    challenges = cursor.fetchall()

    if request.method == 'POST':
        # Xử lý khi sinh viên gửi câu trả lời thử thách
        challenge_id = request.form.get('challenge_id')  # Sử dụng .get() để tránh KeyError
        student_answer = request.form.get('answer')  # Sử dụng .get() để tránh KeyError
```

```
    if challenge_id and student_answer:  # Kiểm tra cả hai trường hợp
        # Lấy file_path của thử thách dựa trên challenge_id
        cursor.execute("SELECT file_path FROM challenges WHERE id = %s", (challenge_id,))
        challenge = cursor.fetchone()

        if challenge and challenge['file_path']:
            # Chuẩn hóa đường dẫn file để tránh vấn đề trên các hệ điều hành
            file_path = os.path.normpath(challenge['file_path'])

            # Kiểm tra file có tồn tại hay không
            if os.path.exists(file_path):
                # Lấy đáp án đúng từ tên file (bỏ phần mở rộng)
                correct_answer = os.path.splitext(os.path.basename(file_path))[0]

                # So sánh đáp án của sinh viên với đáp án đúng
                if student_answer.strip() == correct_answer:
                    # Đọc nội dung file nếu đáp án đúng
                    try:
                        with open(file_path, 'r', encoding='utf-8') as file:
                            content = file.read()
                        flash('Correct answer! File content displayed.', 'success')
                    except Exception as e:
                        flash(f"Error reading file: {str(e)}", 'danger')
                else:
                    flash('Wrong answer. Please try again!', 'danger')
            else:
                flash(f"File not found at path: {file_path}", 'danger')
        else:
            flash('Challenge not found or file path missing.', 'danger')

    # Render giao diện và truyền dữ liệu
    return render_template(
        'student_dashboard.html',
```

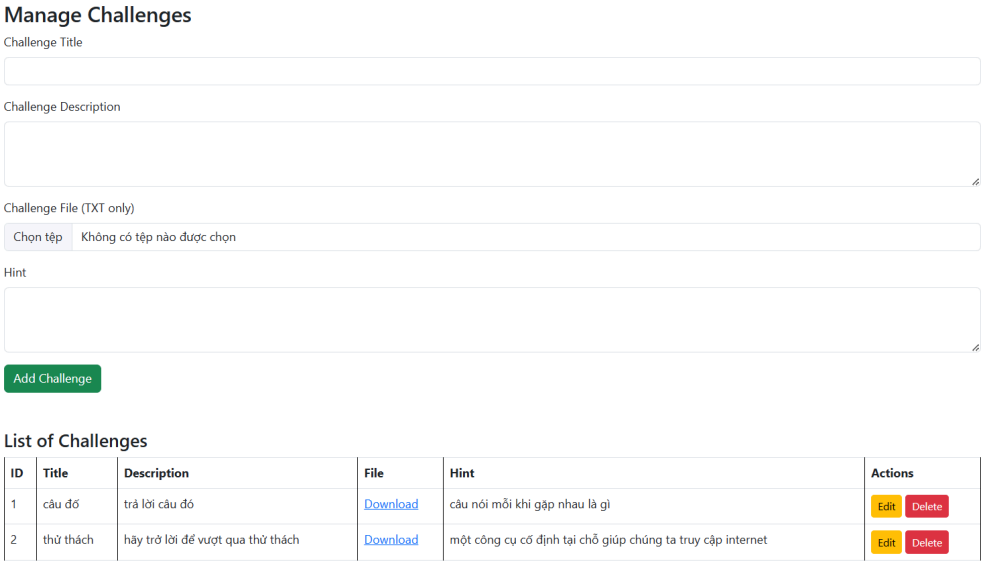
```
            student=student,
            challenges=challenges,
            content=content
        )
```

HTML để hiển thị giao diện thử thách của sinh viên

```
<!-- Thử thách -->
<h4>Challenges</h4>
<ul class="list-group mb-4">
    {% for challenge in challenges %}
        <li class="list-group-item">
            <strong>{{ challenge['title'] }}</strong>  <!-- Thay challenge.title thành challenge['title'] -->
            <p>Hint: {{ challenge['hint'] }}</p>  <!-- Thay challenge.hint thành challenge['hint'] -->
        </li>
    {% endfor %}
</ul>

<h4>Submit Your Answer</h4>
<form method="POST" action="/student_dashboard">
    <div class="mb-3">
        <label for="challenge_id" class="form-label">Select Challenge</label>
        <select name="challenge_id" id="challenge_id" class="form-select" required>
            {% for challenge in challenges %}
                <option value="{{ challenge['id'] }}">{{ challenge['title'] }}</option>  <!-- Thay challenge.id thành challenge['id'] -->
            {% endfor %}
        </select>
    </div>
    <div class="mb-3">
        <label for="answer" class="form-label">Your Answer</label>
        <input type="text" name="answer" id="answer" class="form-control" placeholder="Enter your answer" required>
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
```

```
</form>

{% if content %}
    <h4 class="mt-5">Challenge Content</h4>
    <pre class="border p-3">{{ content }}</pre>
{% endif %}
```

Kết quả



6.Tạo file XML

File XML (Extensible Markup Language) là một loại tệp văn bản được sử dụng để lưu trữ và vận chuyển dữ liệu. XML có cấu trúc dạng cây, cho phép dữ liệu được tổ chức theo các thẻ (tags) có thể được tùy chỉnh, giúp dễ dàng trao đổi dữ liệu giữa các hệ thống khác nhau, đặc biệt là trong các ứng dụng web và cơ sở dữ liệu.

Cấu trúc của một file XML bao gồm các phần chính:

- **Thẻ mở đầu (** `<?xml version="1.0" encoding="UTF-8"?>` **)**: Thông báo rằng đây là một tệp XML và chỉ định phiên bản XML cùng với mã hóa ký tự.
- **Thẻ gốc (** `<root>` **)**: Một tệp XML phải có một thẻ gốc chứa tất cả các dữ liệu khác.
- **Thẻ con**: Các phần tử chứa dữ liệu hoặc các thẻ con khác, mỗi thẻ có tên và giá trị.
- **Dữ liệu**: Nội dung của các thẻ, có thể là văn bản, số liệu hoặc thậm chí các thẻ con khác.

Yêu cầu đề bài

- Tạo chức năng cho phép upload một tệp xml để thêm một lúc nhiều user. Mẫu như sau:

```
<?xml version="1.0" encoding="UTF-8"?>

<user>

<username>johndoe</username>

<email>johndoe@example.com</email>

<phone>+1 (555) 1234567</phone>

<fullname>John Doe</fullname>

<role>admin</role>

</user>
```

- Chức năng được phân quyền chỉ có giáo viên được sử dụng.
- Sau khi upload file xml lên, cần hiển thị ra danh sách user gồm các thông tin thành một bảng: username, email, phone, fullname, role, password được đặt mặc định là `123qweaA@`

- Giáo viên có thể ấn chọn lưu thông tin để cập nhật danh sách user vào database.
  MySQL để lưu thông tin những người trong XML:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    fullname VARCHAR(100),
    role VARCHAR(20),
    password VARCHAR(100)
);
```

Code Python để thực hiện thêm nhiều người dùng băng file XML:

```
# Xử lý upload file XML
elif 'xml_file' in request.files:
    xml_file = request.files['xml_file']
    if xml_file and xml_file.filename.endswith('.xml'):
        try:
            # Parse file XML
            tree = ET.parse(xml_file)
            root = tree.getroot()

            # Kiểm tra dữ liệu trong XML
            print(f"Root: {root.tag}")

            uploaded_users = []
            for user_elem in root.findall('user'):
                username = user_elem.find('username').text
                email = user_elem.find('email').text
                phone = user_elem.find('phone').text
```

```
                fullname = user_elem.find('fullname').text
                role = user_elem.find('role').text

                # In ra các giá trị đã lấy được
                print(f"User found: {username}, {email}, {phone}, {fullname}, {role}")

                # Thêm vào danh sách
                user = {
                    'username': username,
                    'email': email,
                    'phone': phone,
                    'fullname': fullname,
                    'role': role,
                    'password': '123qweaA@'  # Đặt mật khẩu mặc định
                }
                uploaded_users.append(user)

            if uploaded_users:
                # Lưu danh sách người dùng vào session
                session['uploaded_users'] = uploaded_users
                flash('XML file processed successfully!', 'success')
            else:
                flash('No users found in the XML file.', 'danger')

        except Exception as e:
            flash(f'Error processing XML file: {str(e)}', 'danger')

# Lưu danh sách người dùng vào cơ sở dữ liệu khi người dùng nhấn nút "Save Users"
elif 'save_users' in request.form:
    uploaded_users = session.get('uploaded_users', [])
    if uploaded_users:
        for user in uploaded_users:
            try:
                cursor.execute("""
```

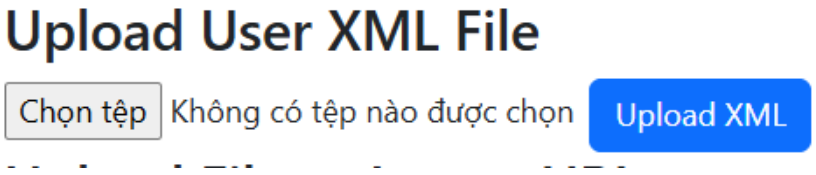
```
                INSERT INTO users (username, email, phone, fullname, role, password)                VALUES (%s, %s, %s, %s, %s, %s)                """,
(user['username'], user['email'], user['phone'], user['fullname'], user['role'],
                 user['password']))
            conn.commit()  # Đảm bảo commit vào cơ sở dữ liệu
            print(f"User {user['username']} saved successfully.")  # Kiểm tra từng user
        except Exception as e:
            print(f"Error saving user {user['username']}: {str(e)}")  # In lỗi nếu có
            flash(f'Error saving user {user["username"]}: {str(e)}', 'danger')

    flash('All users have been saved to the database successfully!', 'success')

    # Xóa khỏi session sau khi lưu thành công
    session.pop('uploaded_users', None)

else:
    flash('No users to save!', 'danger')  # Đảm bảo rằng danh sách người dùng không trống
url = request.form.get('url')  # Sử dụng .get() để tránh KeyError
```

HTML :

```
<!-- Phần upload file XML -->
<h3>Upload User XML File</h3>
<form method="POST" enctype="multipart/form-data">
    <input type="file" name="xml_file" accept=".xml" required>
    <button type="submit" class="btn btn-primary">Upload XML</button>
</form>

{% if uploaded_users %}
    <h4>Uploaded User List</h4>
    <form method="POST">
        <table class="table table-bordered">
            <thead>                <tr>                    <th>Username</th>
                    <th>Email</th>
```

```
                <th>Phone</th>
                <th>Full Name</th>
                <th>Role</th>
            </tr>          </thead>          <tbody>              {% for user in uploaded_users %}
            <tr>
                <td>{{ user.username }}</td>
                <td>{{ user.email }}</td>
                <td>{{ user.phone }}</td>
                <td>{{ user.fullname }}</td>
                <td>{{ user.role }}</td>
            </tr>              {% endfor %}
        </tbody>
    </table>       <button type="submit" name="save_users">Save Users</button>
</form>{% endif %}

{% if success_message %}
<div class="alert alert-success">{{ success_message }}</div>
{% endif %}
```

Kết quả: Khi upload XML thì nhưng thông tin trong đó sẽ được lưu vào database



7.Thực hiện phần thêm một URL vào và hiển thị preview cho tài liệu

Học sinh hoặc giáo viên thực hiện điền một url của ảnh bài làm, tài liệu. Website sẽ tự động truy cập và hiển thị preview cho tài liệu.

Code Python

```
if url:
    try:
        # Xử lý URL để lấy thông tin preview
        preview = get_url_preview(url)
        flash('URL preview generated successfully!', 'success')
    except Exception as e:
        flash(f"Error processing URL: {str(e)}", 'danger')
# Thêm phần xử lý URL cho giáo viên
url = request.form.get('url')  # Sử dụng .get() để tránh KeyError
if url:
    try:
        # Xử lý URL để lấy thông tin preview
        preview = get_url_preview(url)
        flash('URL preview generated successfully!', 'success')
    except Exception as e:
        flash(f"Error processing URL: {str(e)}", 'danger')
```

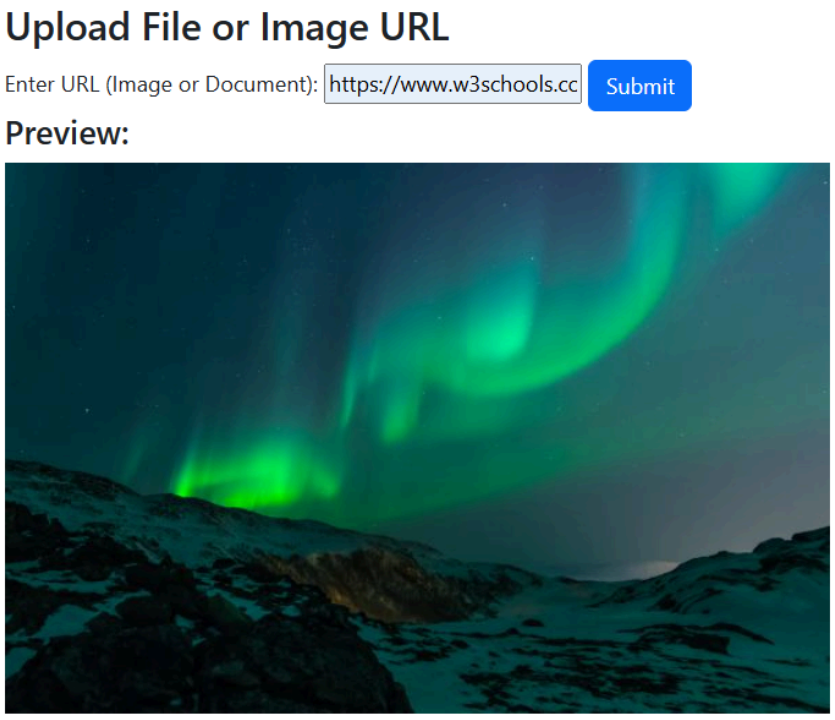
HTML:

```
<!-- Phần upload URL -->
<h3>Upload File or Image URL</h3>
<form method="POST">
    <label for="url">Enter URL (Image or Document):</label>
    <input type="url" id="url" name="url" placeholder="Enter URL of image or document" required>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>

<!-- Phần xem trước -->
{% if preview %}
    <h4>Preview:</h4>
    <div>        {% if preview['type'] == 'image' %}
            <img src="{{ preview['url'] }}" alt="Preview" style="max-width: 100%; max-height: 400px;">
```

```
        {% elif preview['type'] == 'pdf' %}
            <embed src="{{ preview['url'] }}" type="application/pdf" width="100%" height="600px">
        {% elif preview['type'] == 'document' %}
            <a href="{{ preview['url'] }}" target="_blank">Download Document</a>
        {% else %}
            <p>Unable to preview this URL.</p>
        {% endif %}
    </div>
{% endif %}
```

Ở đây tôi đã nhập vào một URL là `https://www.w3schools.com/w3css/img_lights.jpg`

Kết quả



8. Backup và Import người dùng

Yêu cầu

Tạo chức năng backup người dùng:

- Chức năng được phân quyền chỉ có giao viên được sử dụng.
- Giáo viên chọn backup người dùng, website sẽ thực hiện lưu các thông tin người dùng trong databse dưới dạng Object và cho phép tải về.
- Giáo viên thực hiện import file backup, website sẽ thực hiện đọc nội dung trong file backup để lưu lại vào database.

Chức năng backup người dùng giúp lưu trữ lại các thông tin người dùng trong cơ sở dữ liệu dưới dạng Object, và cho phép giáo viên tải về. Điều này giúp tạo bản sao của dữ liệu người dùng để phục hồi hoặc lưu trữ.

Chức năng import:

- Giáo viên có thể tải lên một file backup đã được tạo ra trước đó.
- Hệ thống cần phải thực hiện việc đọc nội dung của file backup, xác định cấu trúc của dữ liệu trong file, và lưu lại thông tin vào cơ sở dữ liệu.

MySQL

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) NOT NULL,
    phone VARCHAR(20),
    fullname VARCHAR(100),
    role VARCHAR(20),
    password VARCHAR(100)
);
```

Code Python

```python
# BACKUP
@app.route('/backup_users', methods=['GET'])
def backup_users():
    if 'teacher_id' not in session:
        flash('You need to log in first!', 'danger')
        return redirect(url_for('login_teacher'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Lấy thông tin người dùng từ cơ sở dữ liệu
    cursor.execute("SELECT username, email, phone, fullname, role FROM users")
    users = cursor.fetchall()

    # Lưu dữ liệu người dùng vào file JSON
    backup_data = json.dumps(users, indent=4)

    # Lưu vào file tạm
    backup_filename = "user_backup.json"
    with open(backup_filename, "w") as f:
        f.write(backup_data)

    # Trả về file cho giáo viên tải về
    return send_file(backup_filename, as_attachment=True)


# IMPORT
@app.route('/import_users', methods=['POST'])
def import_users():
    # Kiểm tra xem người dùng có đăng nhập hay không
    if 'teacher_id' not in session:
        flash('You need to log in first!', 'danger')
        return redirect(url_for('login_teacher'))
```

```
    # Kiểm tra xem có file được chọn không
    if 'backup_file' not in request.files:
        flash('No file selected!', 'danger')
        return redirect(url_for('teacher_dashboard'))  # Trở lại giao diện giảng viên

    backup_file = request.files['backup_file']

    # Kiểm tra định dạng file
    if backup_file.filename.endswith('.json'):
        try:
            # Đọc file JSON
            data = json.load(backup_file)

            # Kiểm tra nếu dữ liệu JSON không rỗng
            if not data:
                flash('The file is empty.', 'danger')
                return redirect(url_for('teacher_dashboard'))

            # Kết nối đến cơ sở dữ liệu
            conn = get_db_connection()  # Đảm bảo bạn đã tạo hàm get_db_connection
            cursor = conn.cursor()

            # Duyệt qua danh sách người dùng và thêm vào cơ sở dữ liệu
            for user in data:
                # Kiểm tra dữ liệu đầu vào
                if not all(key in user for key in ('username', 'email', 'phone', 'fullname', 'role')):
                    flash('Invalid data format in JSON file!', 'danger')
                    return redirect(url_for('teacher_dashboard'))

                cursor.execute("""
                    INSERT INTO users (username, email, phone, fullname, role, password)                    VALUES (%s, %s, %s, %s, %s, %s)                """,
(user['username'], user['email'], user['phone'], user['fullname'], user['role'],
                     '123qweaA@'))  # Đặt mật khẩu mặc định
```

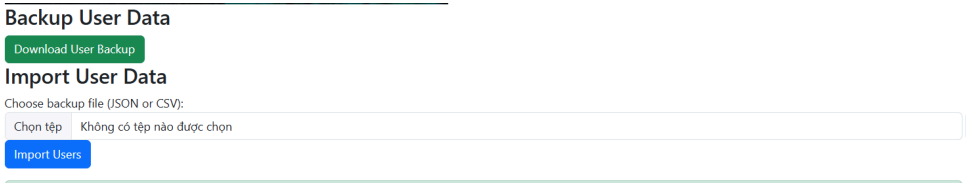
```
            conn.commit()  # Commit để lưu vào DB
            flash('Users imported successfully!', 'success')
        except json.JSONDecodeError:
            flash('Error decoding JSON file.', 'danger')
        except psycopg2.DatabaseError as e:
            flash(f'Database error: {str(e)}', 'danger')
        except Exception as e:
            flash(f'Error importing users: {str(e)}', 'danger')
    else:
        flash('Invalid file format. Please upload a JSON file.', 'danger')

    return redirect(url_for('teacher_dashboard'))  # Quay lại giao diện giảng viên
```

HTML

```
<!-- Phần backup dữ liệu người dùng -->
<h3>Backup User Data</h3>
<a href="{{ url_for('backup_users') }}" class="btn btn-success">Download User Backup</a>

<!-- Phần import dữ liệu người dùng -->
<h3>Import User Data</h3>
<form action="{{ url_for('import_users') }}" method="POST" enctype="multipart/form-data">
    <div class="form-group">
        <label for="file">Choose backup file (JSON or CSV):</label>
        <input type="file" name="backup_file" class="form-control" required>
    </div>    <button type="submit" class="btn btn-primary">Import Users</button>
</form>

<!-- Flash Messages -->
{% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
        <div class="mt-3">
```

```
            {% for category, message in messages %}
                <div class="alert alert-{{ category }}">{{ message }}</div>
            {% endfor %}
        </div>
    {% endif %}
{% endwith %}
```

Kết quả khi chạy chương trình



Đây là file backup khi tải về, các thông tin người dùng trong cơ sở dữ liệu sẽ được chuyển đổi thành một Object (ví dụ, dưới dạng JSON hoặc XML).

```
[
    {
        "username": "user1",
        "email": "user1@example.com",
        "phone": "123456789",
        "fullname": "John Doe",
        "role": "student"
    },
    {
        "username": "user2",
        "email": "user2@example.com",
        "phone": "987654321",
        "fullname": "Jane Doe",
        "role": "teacher"
    },
    {
        "username": "phuc666",
        "email": "phucductran@example.com",
        "phone": "023456789",
        "fullname": "Tran Duc Phuc",
        "role": "student"
    },
    {
        "username": "huuich",
        "email": "chi44@example.com",
        "phone": "0987654321",
        "fullname": "Tran Huu Chi",
        "role": "teacher"
    }
]
```