Practical 3

Web Application Development

October 7, 2024

Instructions

This assignment focuses on using Flask (Python) to build a backend web application. You will implement different routes to handle web requests, interact with a MySQL database using Flask-SQLAlchemy, and create APIs. Complete the following exercises.

Tools Required:

- Python 3.x
- Flask and Flask-SQLAlchemy packages
- MySQL Database (or SQLite)
- Postman or cURL for API testing

Exercise 1: Install Flask Development Environment

Download and install one of the following Python development environments:

- Anaconda (with Flask and SQLAlchemy)
- Any text editor (VSCode, PyCharm) with Flask installation using `pip install flask`

Make sure to install **Flask** and **SQLAlchemy** for database handling.

Exercise 2: Flask Route to Display Strings

- Create a Flask application and write a route that displays the following strings:
  - ”Welcome to Flask Development!”
  - ”This is Labwork 3: Flask/MySQL/API”
- Display these strings at the `/welcome` route.

Exercise 3: Display a Table Using Flask and Jinja2

- Write a Flask route that renders a simple HTML table using Jinja2 templating. The table should display the following data:
  - Column 1: Names
  - Column 2: Ages
- Use a Python list or dictionary to pass the data to the template and display it at the `/table` route.

Sample Table:

Name	**Age**
Alice	22
Bob	19
Charlie	25
David	24
Eve	21

Table 1: Table of Names and Ages

The Flask route should render a table similar to this in HTML. Use the following Python list or dictionary to pass the data to the Jinja2 template:

- `data = ['name': 'Alice', 'age': 22, 'name': 'Bob', 'age': 19, 'name': 'Charlie', 'age': 25, 'name': 'David', 'age': 24, 'name': 'Eve', 'age': 21]`

Flask Route Example:

```
@app.route('/table')
def display_table():
    data = [{'name': 'Alice', 'age': 22}, {'name': 'Bob', 'age': 19},
            {'name': 'Charlie', 'age': 25}, {'name': 'David', 'age': 24},
            {'name': 'Eve', 'age': 21}]
    return render_template('table.html', students=data)
```

In your Jinja2 template (`table.html`), use a loop to generate the rows dynamically:

```
<table border="1">
    <tr>
        <th>Name</th>
        <th>Age</th>
    </tr>
    {% for student in students %}
    <tr>
        <td>{{ student.name }}</td>
        <td>{{ student.age }}</td>
    </tr>
    {% endfor %}
</table>
```

Exercise 4: Flask Function to Calculate Factorial

- Write a Flask route that accepts a number as a URL parameter and calculates the **factorial** of the number.
- Example URL: `/factorial/5`
- The function should return the factorial of the number using Python's math module.

Exercise 5: Flask Function to Check Prime Number

- Write a Flask route that accepts a number as a URL parameter and checks if the number is **prime**.
- Example URL: `/is_prime/7`
- Return `"Prime"` if the number is prime, otherwise return `"Not Prime"`.

Exercise 6: Flask Function to Sort a Numerical Array

- Write a Flask route that accepts a list of numbers (via URL query parameters) and returns the sorted array.
- Example URL: `/sort?numbers=4,2,9,1`
- Return the sorted array in ascending order.

Exercise 7: Flask Function to Reverse a String

- Write a Flask route that accepts a string as a URL parameter and returns the **reversed string**.
- Example URL: `/reverse_string/hello`
- Output should be `"olleh"`.

Exercise 8: Use Flask with MySQL for Database Operations

1. **Create a MySQL Database and Table Using Flask:**
    - Use **Flask-SQLAlchemy** to create the following table in MySQL:
        - `students`: id (Primary Key), name, class, and mark

    - Write a Flask route that creates this table.

2. **Insert Data into the Table:**
    - Write a route that inserts students' data into the MySQL database.
    - Example student data: Name: John, Class: One, Mark: 80

3. **Update Data Based on Condition:**
    - Write a route to update the class of students where the mark is less than 60.
    - Students with marks less than 60 should have their class updated to "Two."

4. **Select and Display Students by Groups:**
    - Write Flask routes to query and display students from the database grouped by marks:
        - **Excellent students**: mark > 75
        - **Good students**: $60 \leq$ mark ≤ 75
        - **Average students**: mark < 60
    - Display the result in three separate HTML tables using Jinja2 templating.

Exercise 9: Build a Frontend for an API with Flask

- Study the Dummy static API at https://dummyapi.io/.
- Create a **Flask application** that serves as a backend API. Your API should perform the following:
    1. **Users:**
        - List available users at the `/users` route.
        - Show detailed info of a user profile at `/user/{user_id}`.
        - List all posts for a specific user at `/user/{user_id}/posts`.
    2. **Posts:**

- List available posts at the `/posts` route.
- List all comments for a specific post at `/post/{post_id}/comments`.