**Experiment 06- Creating a Web Service**

**Learning Objective:** Create a Web Service

**Tools:**  Python, Flask, SQLite3, HTML/CSS

**Theory:** A web service is a software system designed to support interoperable machine-to-machine interaction over a network. In this experiment, we used Python's Flask framework to create a simple web service that performs CRUD (Create, Read, Update, Delete) operations on user data stored in an SQLite3 database. The service provides an API to interact with the data and includes a frontend built using HTML and CSS for user interaction.

**Code Description:**The web service is built using Flask, a lightweight web framework for Python. The application provides a simple user management system:

1. **Database Setup:**
   * SQLite3 is used as the database to store user information. A setup\_db.py script initializes the database with a users table.
2. **Flask Application (app.py):**
   * **Home Route (/):** Displays a web page with a form to add new users and a list of existing users.
   * **API Route (/api/data):** Handles both GET requests to retrieve user data and POST requests to add new users to the database.
3. **Frontend:**
   * The frontend is built using HTML and styled with CSS. It includes a form for adding users and displays the list of users retrieved from the database.
4. **Static Files:**
   * CSS files are stored in the static directory and are used to style the HTML templates located in the templates directory.

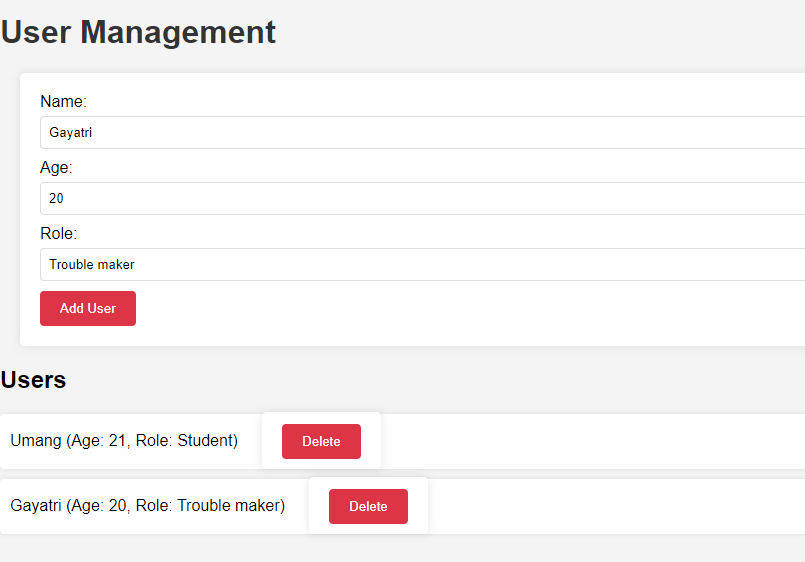
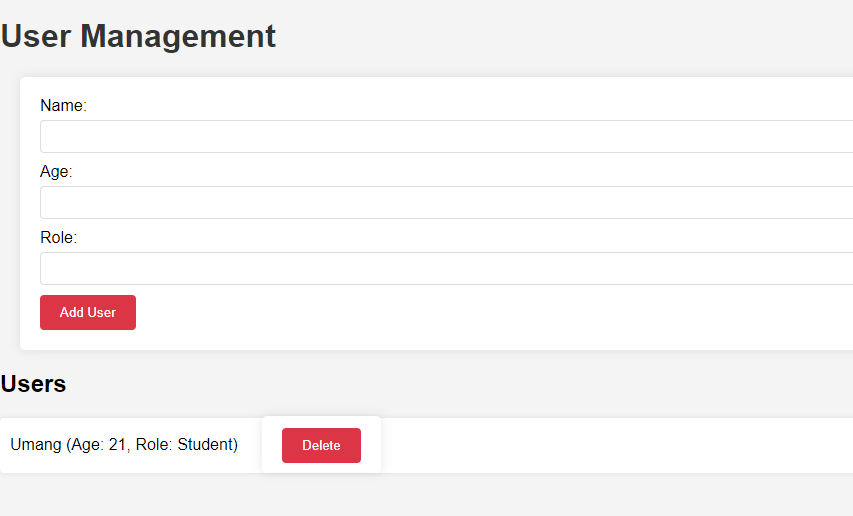
#### 

#### **Code:**

| from flask import Flask, jsonify, request, render\_template, redirect, url\_for  import sqlite3  app = Flask(\_\_name\_\_)  def connect\_db():  return sqlite3.connect('database.db')  @app.route('/')  def home():  conn = connect\_db()  cursor = conn.execute('SELECT \* FROM users')  users = cursor.fetchall()  conn.close()  return render\_template('index.html', users=users)  @app.route('/api/users', methods=['GET'])  def get\_users():  conn = connect\_db()  cursor = conn.execute('SELECT \* FROM users')  users = [  {"id": row[0], "name": row[1], "age": row[2], "role": row[3]}  for row in cursor.fetchall()  ]  conn.close()  return jsonify(users)  # API route to add a new user  @app.route('/api/users', methods=['POST'])  def add\_user():  data = request.get\_json()  name = data['name']  age = data['age']  role = data['role']  conn = connect\_db()  conn.execute('INSERT INTO users (name, age, role) VALUES (?, ?, ?)', (name, age, role))  conn.commit()  conn.close()    return jsonify({"message": "User added successfully!"}) | @app.route('/api/users/<int:user\_id>', methods=['PUT'])  def update\_user(user\_id):  data = request.get\_json()  name = data['name']  age = data['age']  role = data['role']  conn = connect\_db()  conn.execute('UPDATE users SET name = ?, age = ?, role = ? WHERE id = ?', (name, age, role, user\_id))  conn.commit()  conn.close()  return jsonify({"message": "User updated successfully!"})  # API route to delete a user  @app.route('/api/users/<int:user\_id>', methods=['DELETE'])  def delete\_user(user\_id):  conn = connect\_db()  conn.execute('DELETE FROM users WHERE id = ?', (user\_id,))  conn.commit()  conn.close()    return jsonify({"message": "User deleted successfully!"})  # Frontend route to add a new user via form  @app.route('/add\_user', methods=['POST'])  def add\_user\_form():  name = request.form['name']  age = request.form['age']  role = request.form['role']  conn = connect\_db()  conn.execute('INSERT INTO users (name, age, role) VALUES (?, ?, ?)', (name, age, role))  conn.commit()  conn.close()    return redirect(url\_for('home'))  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |
| --- | --- |

#### 

**Output:**

****

**Learning Outcomes:** The student should have the ability to:

LO 1: Design and implement a simple web service using Flask.

LO 2: Integrate a SQLite3 database with a Flask application for data persistence.

LO 3: Create a basic web frontend using HTML and CSS to interact with a Flask backend

**Course Outcomes:** Upon completion of the course students will be able to understand and explain the fundamentals of web services and their implementation using Flask, how to interact with a database through a web service, the process of building and styling a simple web frontend for user interaction.

**Conclusion:**

**For Faculty Use**

| **Correction Parameters** | **Formative Assessment [40%]** | **Timely completion of Practical [ 40%]** | **Attendance / Learning Attitude [20%]** | **Total** |
| --- | --- | --- | --- | --- |
| **Marks Obtained** |  |  |  |  |