EXPERIMENT NO. 4

Name of Student	**Prajyot Shinde**
Class Roll No	**55**
D.O.P.	**25-02-25**
D.O.S.	**04-03-35**
Sign and Grade	

AIM:

To design a Flask application that showcases URL building and demonstrates the use of HTTP methods (GET and POST) for handling user input and processing data.

PROBLEM STATEMENT:

Create a Flask application with the following requirements:

1. A homepage (/) with links to a "Profile" page and a "Submit" page using the url_for() function.
2. The "Profile" page (/profile/<username>) dynamically displays a user's name passed in the URL.
3. A "Submit" page (/submit) displays a form to collect the user's name and age. The form uses the POST method to send the data, and the server displays a confirmation message with the input.

THEORY:

1. What is a route in Flask, and how is it defined?

   A **route** in Flask is a URL endpoint that is associated with a specific function in a web application. When a user accesses a particular URL, Flask executes the corresponding function and returns the response.

   Key Points About Routes in Flask:

   - Routes are defined using the @app.route() decorator.
   - A function is assigned to each route to handle the request.
   - The function typically returns HTML, JSON, or plain text as a response.
   - Multiple routes can be defined for different functionalities in the web app.

2. How can you pass parameters in a URL route?

   Flask allows dynamic content in URLs by using placeholders inside angle brackets (<>). These placeholders act as variables that capture values from the URL.

   Types of URL Parameters in Flask:

   - **String (default)** – Captures any text value (<name>).
   - **Integer** – Captures only numbers (<int:number>).
   - **Float** – Captures decimal numbers (<float:value>).
   - **Path** – Captures an entire subpath (<path:subpath>).

3. What happens if two routes in a Flask application have the same URL pattern?

   If two routes are defined with the same URL pattern, Flask overrides the first definition with the second one. The last declared function for that route is the one that gets executed.

   If a user visits /welcome, they will get "Welcome to Version 2", as the second definition replaces the first.

4. What are the commonly used HTTP methods in web applications?

   HTTP methods determine the type of action performed when a request is sent to a web server. Flask allows handling different HTTP methods in route functions.

   Common HTTP Methods:

   - **GET** – Retrieves data from the server (default method).
   - **POST** – Sends data to the server, often used in form submissions.
   - **PUT** – Updates an existing resource on the server.
   - **DELETE** – Removes a resource from the server.
   - **PATCH** – Partially updates a resource.

5. What is a dynamic route in Flask?

   A **dynamic route** allows a web application to generate different responses based on the URL parameters. These parameters are defined inside < > brackets in the route.

   Why Use Dynamic Routes?

   - They make applications flexible and reusable.
   - They allow user-specific content, such as profiles, dashboards, and product pages.
   - They reduce the need to define multiple static routes.

6. Write an example of a dynamic route that accepts a username as a parameter.

```
from flask import Flask

app = Flask(__name__)

@app.route('/user/<username>')
def show_user(username):
    return f"Hello, {username}!"

if __name__ == '__main__':
    app.run(debug=True)
```

7. What is the purpose of enabling debug mode in Flask?

   Debug mode is an important feature for developers that helps in testing and troubleshooting applications.

   Advantages of Debug Mode:

   1. **Automatic Code Reloading:** If you make changes to your code, Flask automatically restarts the server, eliminating the need for manual restarts.
   2. **Detailed Error Messages:** Debug mode provides a full interactive debugger when an error occurs, making it easier to diagnose problems.
   3. **Faster Development Process:** Since code changes take effect immediately, developers can work more efficiently.

8. How do you enable debug mode in a Flask application?

   There are multiple ways to enable debug mode in Flask:

   Method 1: Using app.run(debug=True)

   This is the simplest way to enable debug mode

   Method 2: Using Environment Variables

   You can enable debug mode by setting the FLASK_ENV variable to development.

   Method 3: Using a Configuration File

   You can also enable debug mode in a Flask application by setting the configuration:

   app.config['DEBUG'] = True

   app.run()

CODE: -

App.py

```
from flask import Flask, render_template, request, redirect, url_for

app = Flask(__name__)

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

@app.route('/profile/<username>')
```

```
def profile(username):
    return render_template('profile.html', username=username)

@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
         return render_template('submit.html', message=f'Thank you, {name}. You are {age} years
old!')
    return render_template('submit.html', message=None)

if __name__ == '__main__':
    app.run(debug=True)
```

templates/profile.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <title>Profile</title>
</head>
<body>
    <h1>Profile Page</h1>
    <p>Welcome, {{ username }}!</p>
    <a href="{{ url_for('home') }}">Back to Home</a>
</body>
</html>
```

templates/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <title>Home</title>
</head>
<body>
    <h1>Welcome to the Flask App</h1>

    <!-- Link to Submit Page -->
```

```
    <a href="{{ url_for('submit') }}">Go to Submit Page</a>

    <!-- Profile Form -->
    <h2>Enter Your Name to Visit Profile</h2>
        <form action="{{ url_for('profile', username='USERNAME') }}" method="GET"
onsubmit="return goToProfile();">
        <input type="text" id="username" placeholder="Enter your name" required>
        <button type="submit">Go to Profile</button>
    </form>

    <script>
        function goToProfile() {
            let username = document.getElementById('username').value;
            if (username) {
                                                    let profileUrl = "{{ url_for('profile',
username='USERNAME') }}".replace("USERNAME", username);
                window.location.href = profileUrl;
            }
            return false;
        }
    </script>

</body>
</html>
```

templates/submit.html

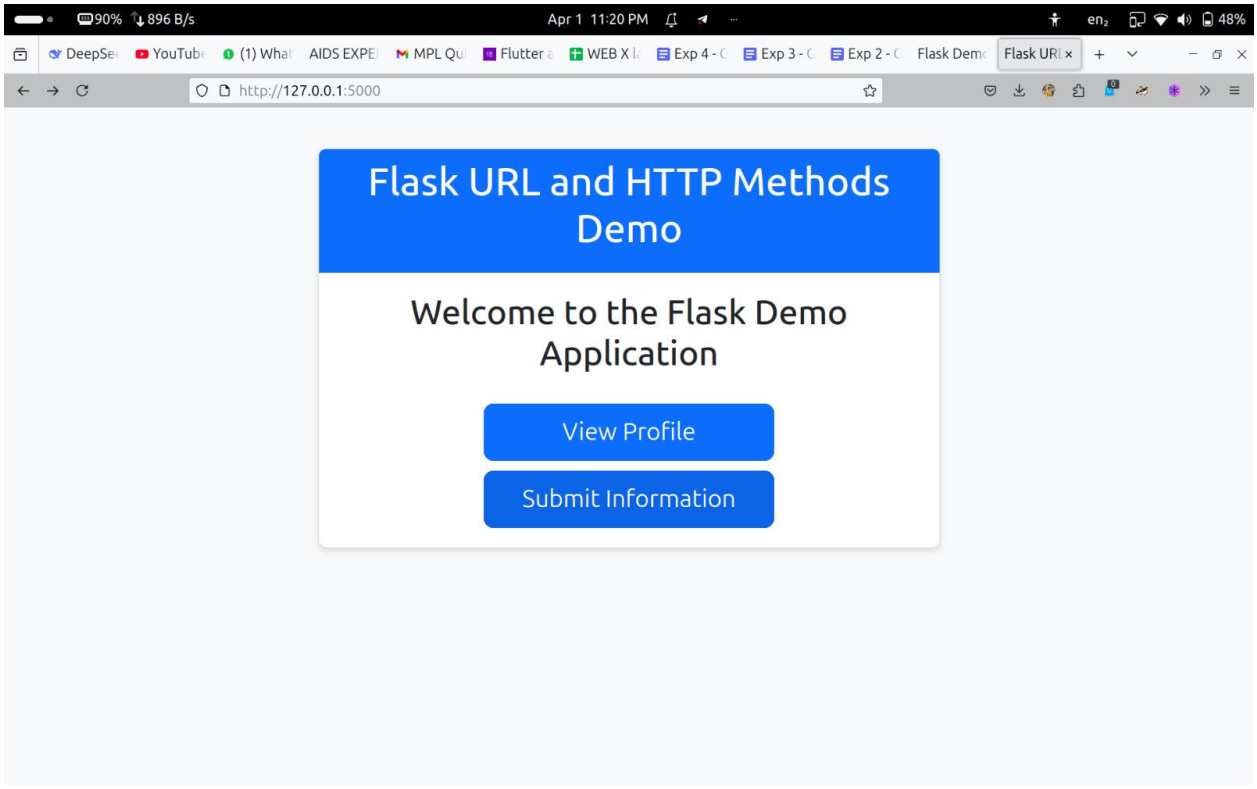
```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <title>Submit</title>
</head>
<body>
    <h1>Submit Your Details</h1>
    <form method="POST">
        <label>Name:</label>
        <input type="text" name="name" required>
        <label>Age:</label>



        <input type="number" name="age" required>
        <button type="submit">Submit</button>
    </form>
    {% if message %}
        <p>{{ message }}</p>
    {% endif %}
</body>
</html>
```

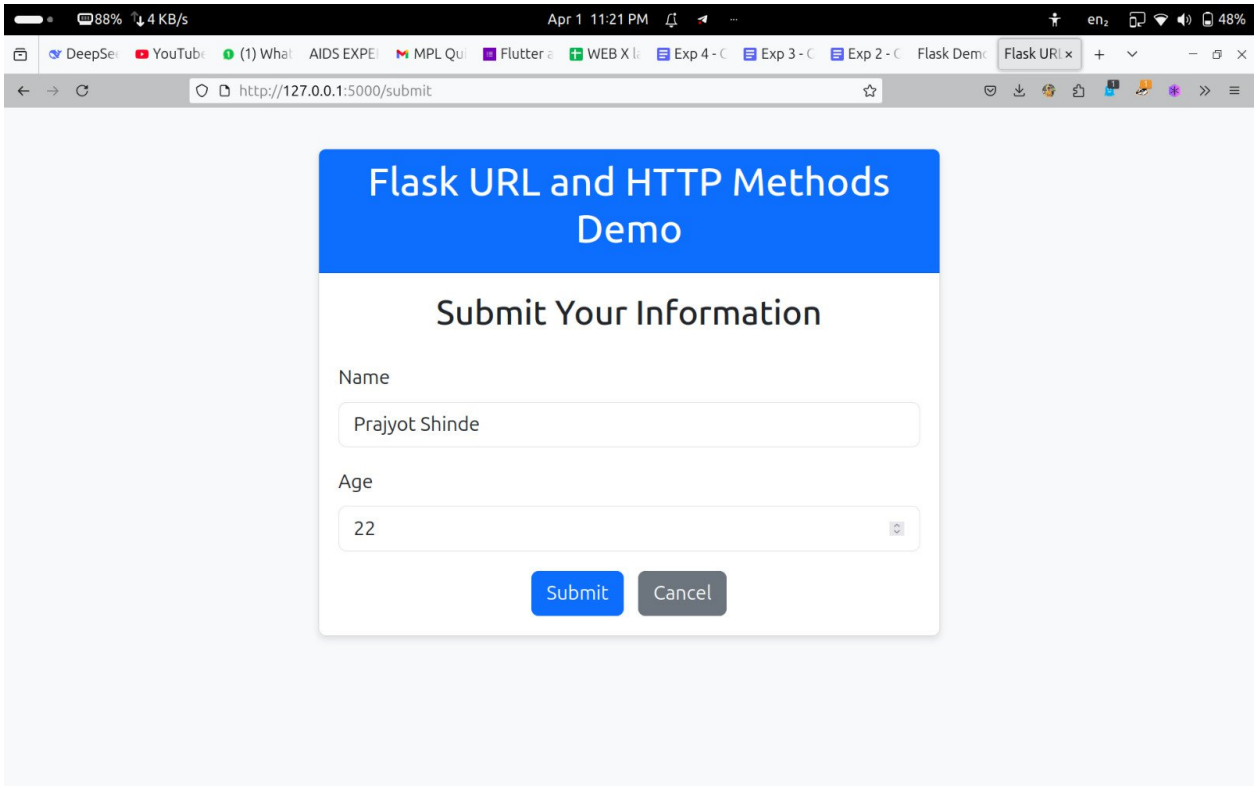
static/style.css

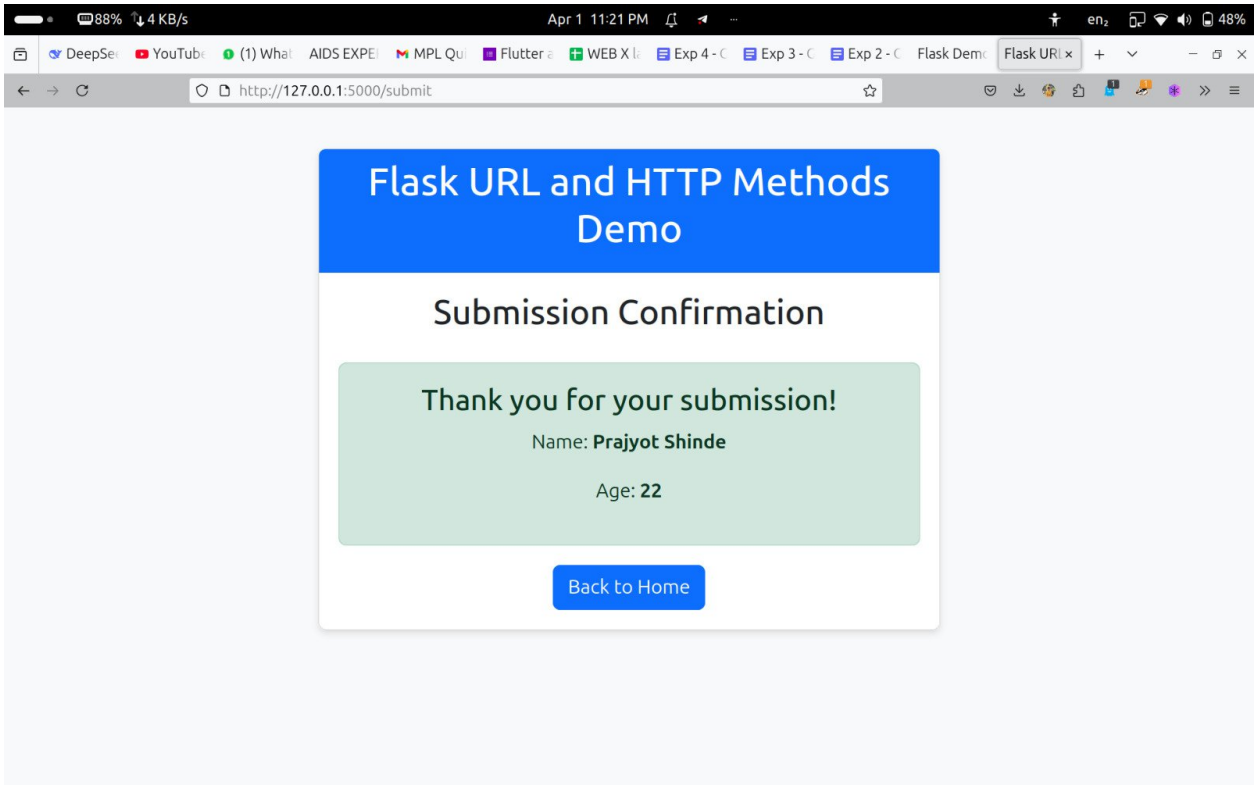
```
body {
```

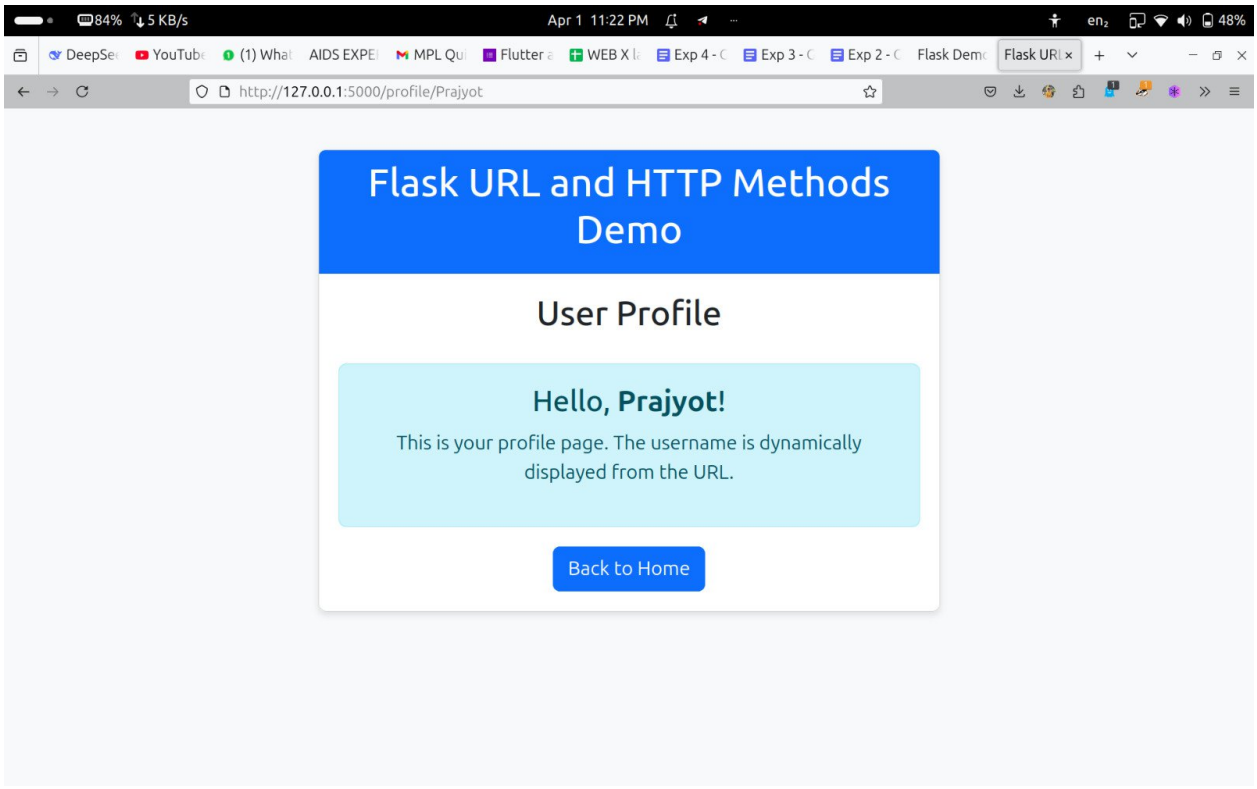
```
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

h1 {
    color: #333;
}

form {
    margin-top: 20px;
}

input, button {
    margin: 5px;
    padding: 10px;
}

button {
    background-color: #5cb85c;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #4cae4c;
}
```

OUTPUT: -


http://127.0.0.1:5000
Flask URL and HTTP Methods Demo
Welcome to the Flask Demo Application
View Profile
Submit Information


http://127.0.0.1:5000/submit
Flask URL and HTTP Methods Demo
Submit Your Information
Name
Prajyot Shinde
Age
22
Submit
Cancel


Apr 1 11:21 PM
http://127.0.0.1:5000/submit
Flask URL and HTTP Methods Demo
Submission Confirmation
Thank you for your submission!
Name: Prajyot Shinde
Age: 22
Back to Home


Apr 1 11:22 PM
http://127.0.0.1:5000/profile/Prajyot
Flask URL and HTTP Methods Demo
User Profile
Hello, Prajyot!
This is your profile page. The username is dynamically displayed from the URL.
Back to Home

Conclusion: -

The Flask application effectively demonstrates URL building using the url_for() function and handles user input through GET and POST methods. The homepage provides navigation links, while the Profile page dynamically displays user information passed in the URL. The Submit page collects user data via a form and processes it using the POST method, displaying a confirmation message. This implementation highlights Flask's capability to manage dynamic routing, handle user input efficiently, and process data seamlessly.