EXPERIMENT NO.4 : Flask

Name of Student	**Komal Milind Deolekar**
Class Roll No	**14**
D.O.P.	**25-02-2025**
D.O.S.	**04-03-2025**
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.

Github Link :

https://github.com/KomalDeolekar0607/Webx_Lab/tree/main/Webx_Lab_Exp_4

Theory :

- **What is a route in Flask, and how is it defined?**

  A **route** in Flask is a URL pattern that maps to a specific function. It tells Flask what action to take when a user visits a particular URL. Routes help define how an application responds to client requests.

  How is it Defined?

  In Flask, routes are defined using the @app.route() decorator. The function beneath it executes when the route is accessed.

  Example:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')  # Route for the home page
def home():
```

```
    return "Welcome to the Flask App!"

if __name__ == "__main__":
    app.run()
```

Explanation:

- The @app.route('/') maps the root URL (/) to the home() function.
- When a user visits http://127.0.0.1:5000/, they will see "Welcome to the Flask App!".

- **How can you pass parameters in a URL route?**

  Flask allows **dynamic routing**, where variables can be passed in the URL using angle brackets < >.

  Example:

```
@app.route('/user/<name>')  # 'name' is a dynamic variable in the URL
def greet_user(name):
    return f"Hello, {name}!"
```

  How It Works?

  - If a user visits /user/Alex, the output will be:
    "Hello, Alex!"

  - If they visit /user/Komal, the output will be:
    "Hello, Komal!"

  You can also define multiple parameters:

```
@app.route('/user/<name>/<int:age>')  # Age must be an integer
def user_info(name, age):
    return f"User {name} is {age} years old."
```

- **What happens if two routes in a Flask application have the same URL pattern?**

  If two routes have **the same URL pattern**, Flask will **override the first one**, keeping only the last-defined function.

  Example (Incorrect Usage):

```
@app.route('/about')
def about_v1():
    return "This is About Page - Version 1"
@app.route('/about')
def about_v2():
    return "This is About Page - Version 2"
```

  Here, /about will return "This is About Page - Version 2", and "Version 1" will be ignored.

To avoid this issue, use **different route names** or **methods**:

```
@app.route('/about/v1')
def about_v1():
   return "This is About Page - Version 1"
@app.route('/about/v2')
def about_v2():
   return "This is About Page - Version 2"
```

- **What are the commonly used HTTP methods in web applications?**

  HTTP methods define how data is sent between the client (browser) and the server. The most commonly used methods are:

Method	Description
GET	Requests data from the server (e.g., retrieving a webpage).
POST	Sends data to the server (e.g., submitting a form).
PUT	Updates an existing resource on the server.
DELETE	Removes a resource from the server.
PATCH	Partially updates a resource on the server.

Example: Handling GET and POST Requests

```
@app.route('/submit', methods=['GET', 'POST'])
def submit_form():
   if request.method == 'POST':
      return "Data Submitted"
   else:
      return "Fill the form first"
```

A **GET request** loads the form.

A **POST request** submits the data.

- **What is a dynamic route in Flask?**

  A **dynamic route** allows variables to be passed through the URL, making routes flexible.

  Example: Dynamic Route

```
@app.route('/profile/<username>')
def profile(username):
    return f"Welcome to {username}'s profile!"
```

  - If a user visits /profile/Alex, the output will be:
    "Welcome to Alex's profile!

  Dynamic routes allow personalized responses based on user input.

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

```
from flask import Flask
app = Flask(__name__)

@app.route('/hello/<username>')  # Dynamic route with username
def say_hello(username):
    return f"Hello, {username}! Welcome to Flask."

if __name__ == "__main__":
    app.run()
```

  How It Works?

  Visiting /hello/Komal outputs:

  "Hello, Komal! Welcome to Flask."

  Visiting /hello/John outputs:

  "Hello, John! Welcome to Flask."

- **What is the purpose of enabling debug mode in Flask?**

  Enabling **debug mode** in Flask helps developers by:

  - **Automatically reloading the server** when code changes.
  - **Displaying detailed error messages** instead of generic ones.
  - **Making it easier to debug issues** quickly.

- **How do you enable debug mode in a Flask application?**

  There are **two ways** to enable debug mode:

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

  app.run(debug=True)

  This enables auto-reloading and debugging in the console.

  Method 2: Using Environment Variables

  Set the FLASK_ENV variable to development:

  For **Linux/macOS** (Terminal):

  export FLASK_ENV=development
  flask run

  For **Windows (Command Prompt):**

  set FLASK_ENV=development
  flask run

  This ensures Flask runs in **debug mode** automatically.

Code :

app.py

```
from flask import Flask, request, url_for

app = Flask(__name__)

@app.route('/')
def home():
    return f"
    <html lang="en">
    <head>
        <title>Flask App</title>
        <link rel="stylesheet" href="{url_for('static', filename='styles.css')}">
    </head>
    <body>
        <div class="container">
            <h1>Welcome to the App!!!</h1>
            <p><a href="{url_for('profile', name='Komal')}">Go to Profile Page</a></p>
            <p><a href="{url_for('submit')}">Go to Submit Page</a></p>
        </div>
    </body>
    </html>
```

```
    '''

@app.route('/profile/<name>')
def profile(name):
    return f'''
    <html lang="en">
    <head>
        <title>Profile</title>
        <link rel="stylesheet" href="{url_for('static', filename='styles.css')}">
    </head>
    <body>
        <div class="container">
            <h1>Welcome, {name}!</h1>
            <a href="{url_for('home')}">Back to Home</a>
        </div>
    </body>
    </html>
    '''

@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        name = request.form.get('name')
        age = request.form.get('age')
        return f'''
        <html lang="en">
        <head>
            <title>Submission Successful</title>
            <link rel="stylesheet" href="{url_for('static', filename='styles.css')}">
        </head>
        <body>
            <div class="container">
                <h1>Thank You, {name}!</h1>
                <p>Your age ({age}) has been submitted successfully.</p>
                <a href="{url_for('home')}">Back to Home</a>
            </div>
        </body>
        </html>
        '''
    return f'''
    <html lang="en">
    <head>
        <title>Submit Form</title>
        <link rel="stylesheet" href="{url_for('static', filename='styles.css')}">
    </head>
    <body>
        <div class="container">
            <h1>Submit Your Details</h1>
            <form method="post">
                <input type="text" name="name" placeholder="Enter your name" required>
                <input type="number" name="age" placeholder="Enter your age" required>
```

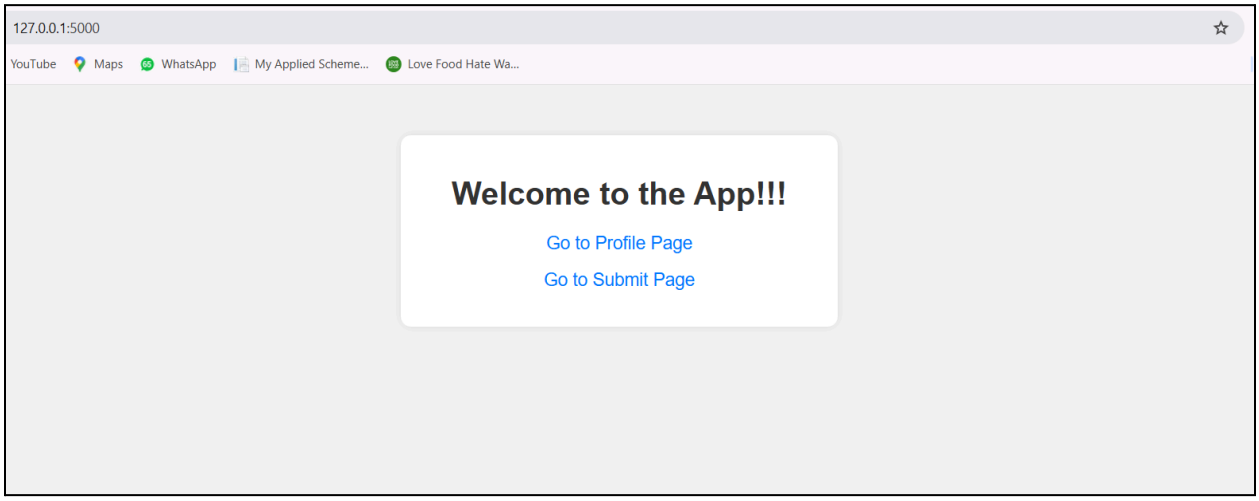
```
            <button type="submit">Submit</button>
        </form>
    </div>
</body>
</html>
'''

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

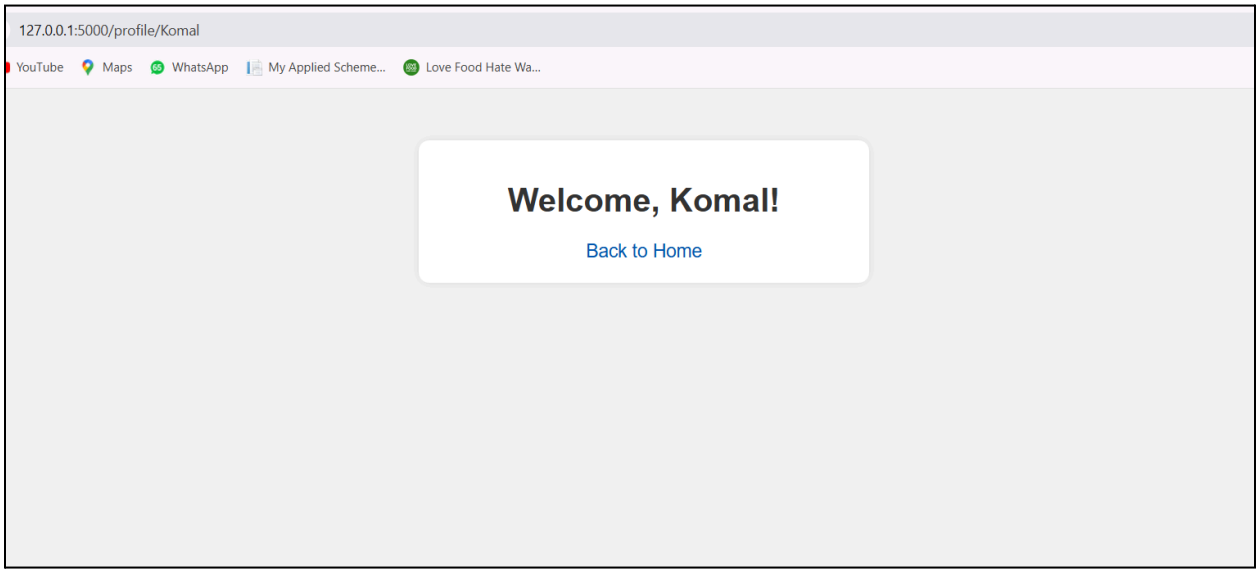
static/styles.css

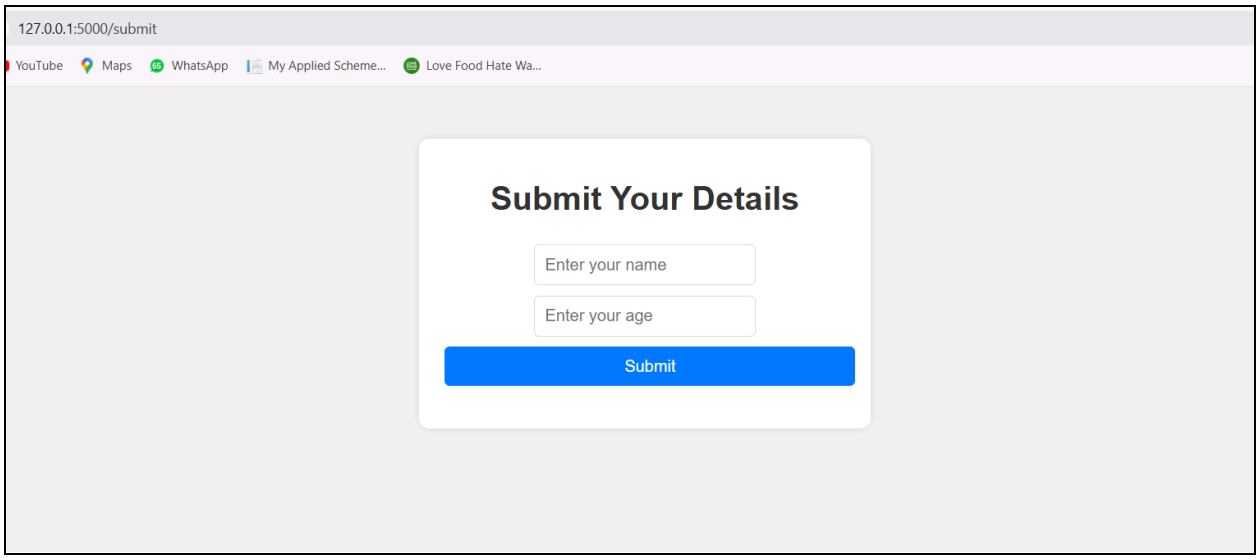
```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin: 50px;
    background-color: #f4f4f4;
}

h1 {
    color: #333;
}

.container {
    background: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);
    width: 400px;
    margin: auto;
}

a {
    text-decoration: none;
    color: #007BFF;
    font-size: 18px;
}

a:hover {
    color: #0056b3;
}

input, button {
    padding: 10px;
    font-size: 16px;
    margin: 5px;
    border-radius: 5px;
    border: 1px solid #ddd;
}

button {
```

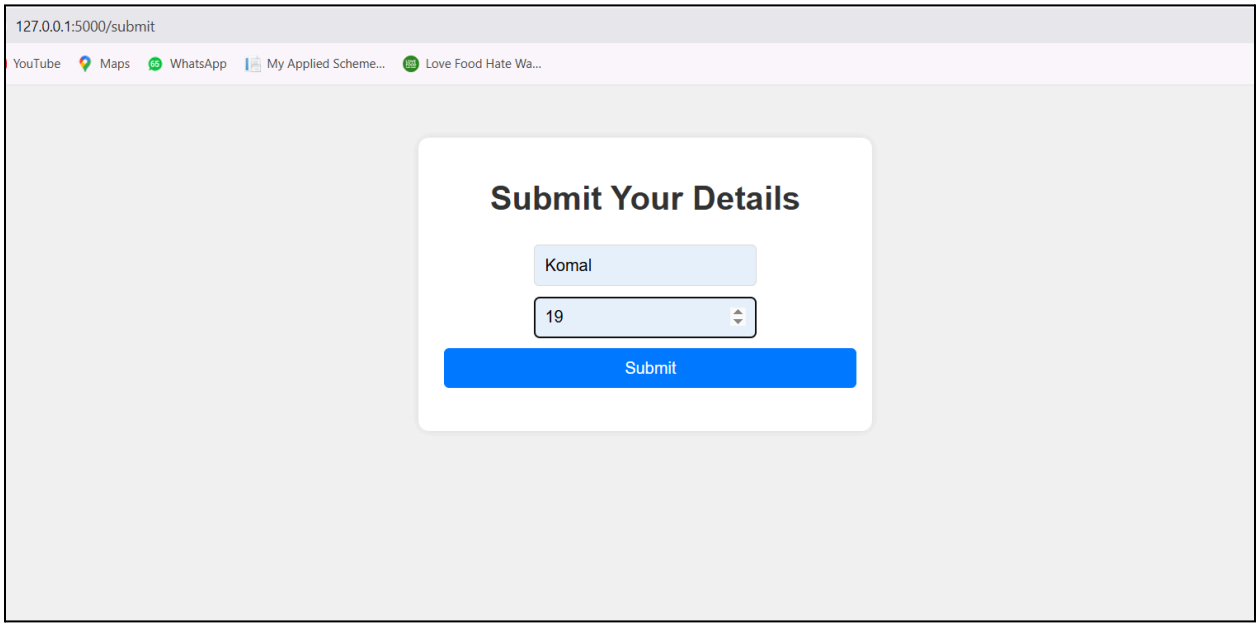
```
    background-color: #007BFF;
    color: white;
    border: none;
    cursor: pointer;
    width: 100%;
}

button:hover {
    background-color: #0056b3;
}
```

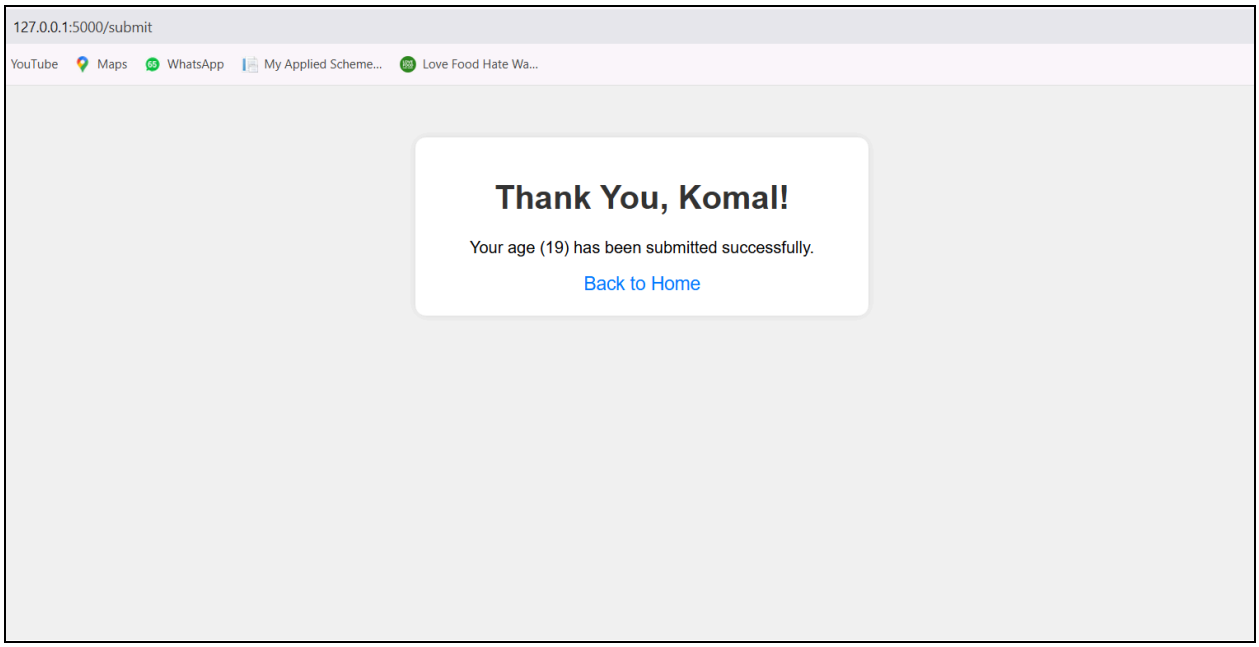
Output :






127.0.0.1:5000/submit
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Submit Your Details
Enter your name
Enter your age
Submit


127.0.0.1:5000/submit
YouTube
Maps
WhatsApp
My Applied Scheme...
Love Food Hate Wa...
Submit Your Details
Komal
19
Submit



Conclusion :

In this experiment, we successfully designed a Flask web application that demonstrates URL building using url_for() and handles user input through GET and POST methods. The application includes a dynamic profile page that receives a username from the URL and a form submission page that collects and processes user data. This practice enhances understanding of routing, dynamic URLs, form handling, and HTTP methods in Flask, which are essential components of modern web development.