EXPERIMENT NO. 4 - Flask Application using GET and POST

Name of Student	**Sneha Patra**
Class Roll No	**D15A_40**
D.O.P.	**27/02/2025**
D.O.S.	**06/03/2025**
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.

Theory:

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

   A route is a URL pattern linked to a function in Flask using the @app.route() decorator.

   Example:

   ```
   @app.route('/')
   def home():
       return "Welcome!"
   ```

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

   Parameters can be passed using angle brackets (`< >`) in the route. Flask will capture these values and pass them to the function as arguments. You can also specify data types like `<int:id>` or `<string:name>`.

   Example:

```
@app.route('/user/<string:name>')
def greet_user(name):
    return f"Hello, {name}!"
```

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

   If two routes share the same URL, Flask will use the last-defined route and override the previous one. This causes unexpected behavior and conflicts.

   Example:

```
@app.route('/hello')
def hello1():
    return "Hello from function 1"
@app.route('/hello')
def hello2():
    return "Hello from function 2"
# Only "Hello from function 2" will be shown.
```

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

- HTTP methods define the type of request a client sends to a server.
- GET: Retrieve data (e.g., accessing a web page).
- POST: Send data to the server (e.g., submitting a form).
- PUT: Update existing data.
- DELETE: Remove data.
- PATCH: Partially update data.

5. **What is a dynamic route in Flask?**

   A dynamic route allows variables to be embedded within the URL, making it more flexible. The data in the URL is passed to the function for further processing.

   Example:

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

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

   @app.route('/user/<username>')

   def welcome_user(username):

      return f"Hello, {username}! Glad to see you here."

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

   Debug Mode is used during development for easy troubleshooting. It enables:

   Automatic Code Reloading: The app restarts when code changes.

   Detailed Error Messages: Displays an interactive debugger in case of an error. It should be disabled in production for security reasons.

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

   You can enable debug mode using one of these methods:

   Using app.run()

   export FLASK_ENV=development

   flask run

GitHub Link: https://github.com/Sneha0321/WebX_Exp_4

CODE:

app.py

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


app = Flask(__name__)

app.secret_key = "supersecretkey"


@app.route('/')

def home():

    return f"'

    <html>

    <head>
```

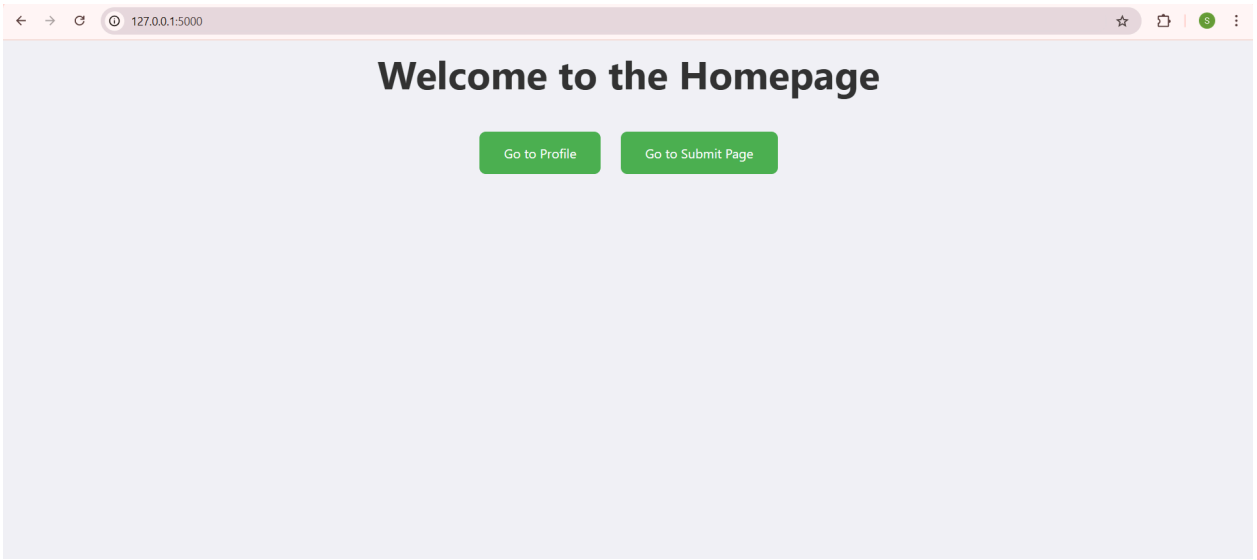
```
        <style>
            body {{ background: linear-gradient(to right, #00c9ff, #92fe9d); text-align: center; color: white; }}
            a {{ display: inline-block; margin: 10px; padding: 10px; background: rgba(0, 0, 0, 0.2); text-decoration: none; color: white; border-radius: 5px; }}
            a:hover {{ background: rgba(0, 0, 0, 0.5); }}
        </style>
    </head>
    <body>
        <h1>Welcome to the Homepage</h1>
        <a href="{url_for('profile')}">Go to Profile</a>
        <a href="{url_for('submit')}">Go to Submit Page</a>
    </body>
    </html>
    '''

@app.route('/profile')
def profile():
    name = session.get('name', 'Guest')
    age = session.get('age', 'Unknown')
    return f'''
    <html>
    <head>
        <style>
            body {{ background: linear-gradient(to right, #007adf, #00ecbc); text-align: center; color: white; }}
        </style>
    </head>
```

```
    <body>
        <h1>Profile Page</h1>
        <p><strong>Name:</strong> {name}</p>
        <p><strong>Age:</strong> {age}</p>
        <a href="{url_for('home')}">Back to Homepage</a>
    </body>
    </html>
    '''

@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        session['name'] = request.form.get('name', 'Unknown')
        session['age'] = request.form.get('age', 'Unknown')
        return redirect(url_for('profile'))

    return '''
    <html>
    <head>
        <style>
                body { background: linear-gradient(to right, #ff512f, #dd2476); text-align: center; color: ; }
                    form { display: inline-block; background: rgba(0, 0, 0, 0.2); padding: 20px; border-radius: 10px; }
            input { margin: 5px; padding: 10px; border-radius: 5px; border: none; }
            input[type="submit"] { background: rgba(0, 0, 0, 0.5); color: white; cursor: pointer; }
        </style>
    </head>
```

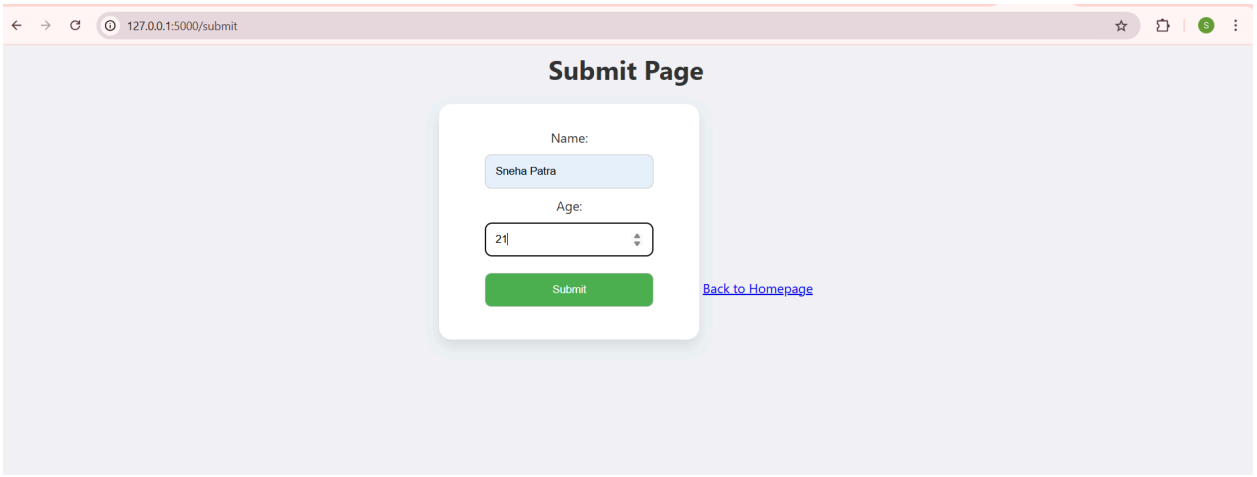
```
    <body>
        <h1>Submit Page</h1>
        <form method="post">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name" required><br>
            <label for="age">Age:</label>
            <input type="number" id="age" name="age" required><br>
            <input type="submit" value="Submit">
        </form>
        <a href="{url_for('home')}">Back to Homepage</a>
    </body>
    </html>
    '''

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

OUTPUT :

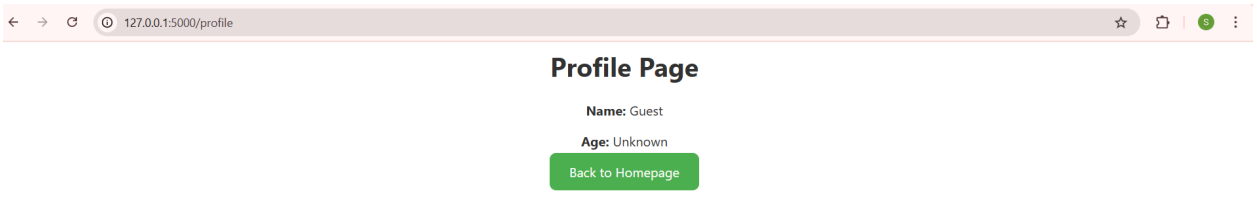
Homepage: Displays a welcome message with navigation links to the "Profile" and "Submit" pages.

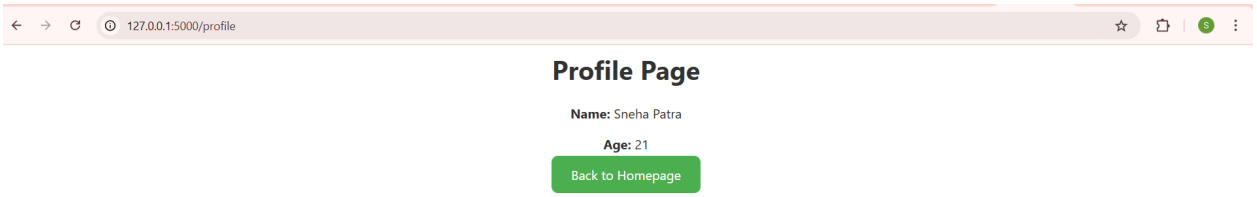


Submit Page: Displays a form to collect the user's name and age. Upon submitting, the data is stored using a session, and the user is redirected to the Profile page.



Profile Page: Displays the submitted name and age dynamically. If no data is submitted, it shows "Guest" and "Unknown" as default values**.**





Conclusion:

The experiment demonstrated the creation of a simple Flask application using GET and POST methods. The application effectively handled dynamic routing using URL parameters and processed user input via a form using the POST method. Additionally, it showcased URL building using the `url_for()` function and maintained user data using sessions.