

EXPERIMENT NO. 4 - Flask Application using GET and POST

| | |
|-----------------|-------------|
| Name of Student | Mohit Patil |
| Class Roll No | 36 |
| 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 in Flask is a URL pattern linked to a function 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 in a route using angle brackets (< >) in the URL. Flask

captures these values and passes 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 pattern, Flask will use the last-defined route, overriding the previous one. This causes unexpected behavior.

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.

Example:

```
@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(debug=True)`
- `export FLASK_ENV=development`
- `flask run`

CODE

```
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 {{
            font-family: 'Arial', sans-serif;
            background-color: #4f4f9;
            color: #333;
            text-align: center;
            padding: 20px;
        }}
        a {{
            display: inline-block;
            margin: 10px;
            padding: 12px 24px;
            background-color: #4CAF50;
            color: white;
            text-decoration: none;
            border-radius: 5px;
            font-weight: bold;
        }}
        a:hover {{
            background-color: #45a049;
        }}
        h1 {{
            color: #333;
            font-size: 32px;
        }}
    </style>
    </head>
    <body>
        <h1>Hello, World!</h1>
        <a href="/about">About</a>
        <a href="/contact">Contact</a>
    </body>
    </html>
    """
```

```

    }}
    </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 {{
        font-family: 'Arial', sans-serif;
        background-color: #f0f8ff;
        text-align: center;
        color: #333;
        padding: 20px;
    }}
    h1 {{
        color: #5a5a5a;
        font-size: 32px;
    }}
    p {{
        font-size: 18px;
        margin-top: 10px;
    }}
    a {{
        margin-top: 20px;
        display: inline-block;
        padding: 10px 20px;
        background-color: #ff6347;
        color: white;
        text-decoration: none;
        border-radius: 5px;
    }}
    a:hover {{
        background-color: #ff4500;
    }}
</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 {
  font-family: 'Arial', sans-serif;
  background-color: #afad2;
  text-align: center;
  color: #333;
  padding: 20px;
  margin: 0;
}
.form-container {
  background-color: #ffffff;
  padding: 30px;
  border-radius: 8px;
  box-shadow: 0 6px 12px rgba(0, 0, 0, 0.1);
  display: inline-block;
  width: 60%;
  text-align: left;
}
h1 {
  color: #333;
  font-size: 36px;
  margin-bottom: 20px;
}
label {
  font-size: 18px;
  margin-bottom: 10px;
  display: block;
  color: #555;
}
input {
  margin: 10px 0;
  padding: 12px 20px;
  border-radius: 5px;
  border: 1px solid #ccc;
  font-size: 16px;
  width: 100%;
}
input[type="submit"] {
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  width: 100%;
  padding: 15px;
  font-size: 18px;
  border-radius: 5px;
}
input[type="submit"]:hover {
  background-color: #45a049;
}
.back-link {
  margin-top: 20px;
  display: inline-block;
  padding: 12px 24px;
  background-color: #ff6347;
  color: white;
  text-decoration: none;
  border-radius: 5px;
}
.back-link:hover {
  background-color: #ff4500;
}
</style>
</head>
<body>
  <div class="form-container">
    <h1>Submit Your Information</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')}}" class="back-link">Back to Homepage</a>
</div>
</body>
</html>
'''

```

```

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 {{
        font-family: 'Arial', sans-serif;
        background-color: #fafad2;
        text-align: center;
        color: #333;
        padding: 20px;
    }}
    form {{
        background-color: #ffffff;
        padding: 20px;
        border-radius: 8px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        display: inline-block;
    }}
    input {{
        margin: 10px;
        padding: 12px 20px;
        border-radius: 5px;
        border: 1px solid #ccc;
        font-size: 16px;
    }}
    input[type="submit"] {{
        background-color: #4CAF50;
        color: white;
        cursor: pointer;
    }}
    input[type="submit"]:hover {{
        background-color: #45a049;
    }}
    h1 {{
        color: #333;
        font-size: 32px;
    }}
    a {{
        display: inline-block;
        margin-top: 20px;
        padding: 12px 24px;
        background-color: #ff6347;
        color: white;
        text-decoration: none;
        border-radius: 5px;
    }}
    a:hover {{
        background-color: #ff4500;
    }}
</style>
</head>
<body>

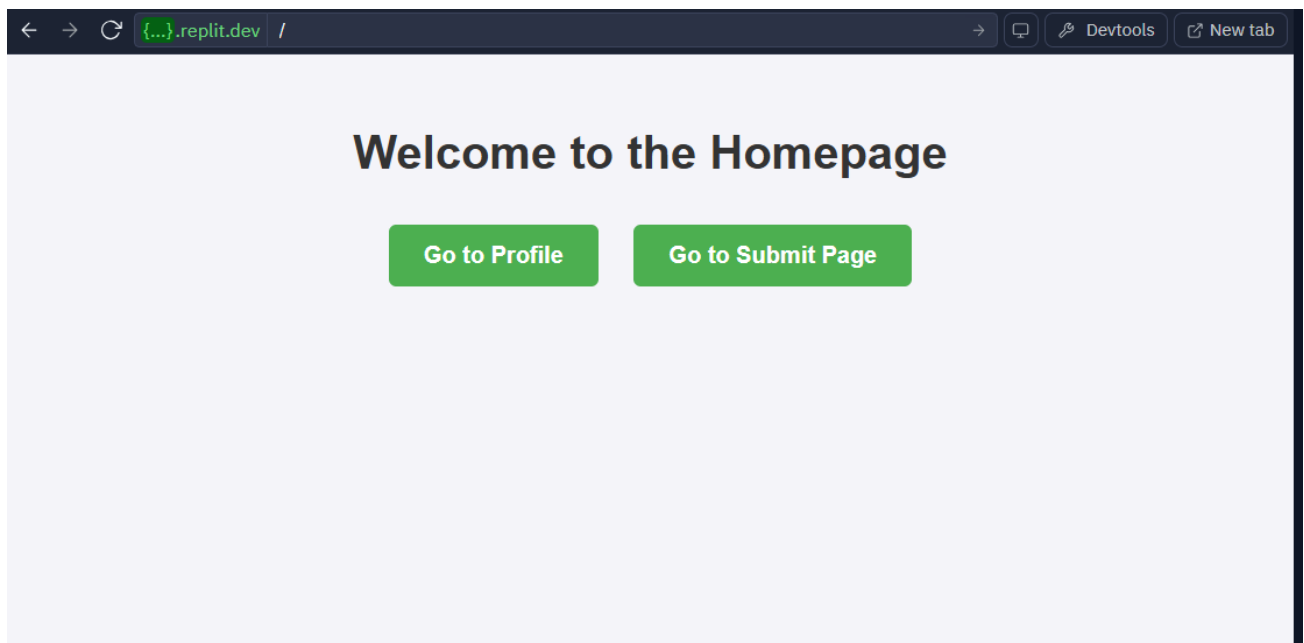
```

```
<h1>Submit Your Information</h1>
<form method="post">
  <label for="name">Name:</label><br>
  <input type="text" id="name" name="name" required><br>
  <label for="age">Age:</label><br>
  <input type="number" id="age" name="age" required><br><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:** The homepage will display two links: one to the "Profile" page and another to the "Submit" page.



← → ↻ {...}.replit.dev /submit → 🖨️ 🔧 Devtools 🔄 New tab

Submit Your Information

Name:

Age:

Submit

Back to Homepage

← → ↻ {...}.replit.dev /profile → 🖨️ 🔧 Devtools 🔄 New tab

Profile Page

Name: Mohit patil

Age: 20

Back to Homepage

- **Profile Page:** After submitting the form with your name and age, the profile page will display your details dynamically.

- **Submit Page:** The form will allow users to enter their name and age, and upon submission, they will be redirected to the profile page with the entered details

CONCLUSION

The experiment successfully demonstrated the implementation of **GET and POST** methods in a Flask application. It involved creating a **homepage (/)** with links to other pages, a **dynamic profile page (/profile/<username>)** that displayed user-specific data from the URL, and a **form-based submit page (/submit)** that handled user input via the **POST** method.

Through this experiment, key Flask concepts such as **URL building (url_for)**, **dynamic routing**, **form handling**, **HTTP methods**, and **enabling debug mode** were explored. This practical implementation highlighted Flask's capability to create interactive web applications with minimal code.

