

### Experiment 4 : Flask Application using GET and POST

Name of Student	Bhagyesh Patil
Class Roll No	D15A_35
D.O.P.	<u>27/02/2025</u>
D.O.S.	<u>06/03/2025</u>
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?**

In Flask, a **route** is a way to map a URL to a specific function that handles the request. Whenever a user accesses a particular URL in the browser, Flask uses the route to identify which function should handle that request.

##### **How to Define a Route:**

- Use the `@app.route()` decorator above a function to define a route.
- ```
from flask import Flask
app = Flask(__name__)
```

```
@app.route('/') # This is the route for the homepage
def home():
    return "Welcome to the Home Page!"

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

### **Explanation:**

- The / route maps to the home() function.
- Accessing **http://127.0.0.1:5000/** will show "Welcome to the Home Page!"

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

Flask allows you to define **dynamic routes** that accept parameters. These parameters are passed through the URL and can be captured in the function.

### **Example:**

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

### **Explanation:**

- <username> is a dynamic parameter.
- If you access **http://127.0.0.1:5000/user/John**, the output will be "Hello, John!"

### **Parameter Types:**

You can specify the data type like this:

- <int:id> — Integer parameter
- <float:price> — Floating-point parameter
- <string:name> — String parameter (default)

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

If two routes have the same URL pattern in a Flask application, only the last defined route will take effect, and the previous one will be overridden. Flask does not allow duplicate routes with the same URL pattern, as it would cause ambiguity.

Example of conflicting routes:

```
python
CopyEd
it
@app.route('/hello') def
hello1():
    return "Hello from function 1"

@app.route('/hello') def
hello2():
    return "Hello from function 2"
```

In this case, when a user visits `/hello`, Flask will only execute `hello2()`, and `hello1()` will be ignored.

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

The most commonly used HTTP methods in web applications are:

1. **GET** – Requests data from the server (e.g., retrieving a webpage).
  2. **POST** – Sends data to the server (e.g., submitting a form).
  3. **PUT** – Updates existing data on the server.
  4. **DELETE** – Removes a resource from the server.
  5. **PATCH** – Partially updates a resource.
- ```
return "Form submitted!"
```

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

A **dynamic route** in Flask is a route that contains variables, allowing it to handle multiple different URLs with a single function. Dynamic routes make the web application more flexible by enabling the use of parameters within the URL.

Example:

```
python CopyEdit
@app.route('/user/<username>') def profile(username):
    return f"User Profile: {username}"
```

If a user visits `/user/Sanket`, the function receives `"Sanket"` as a parameter and responds accordingly.

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

```
python
CopyEd
it
from flask import Flask

app = Flask(__name__)

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

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

In this example, the route `/user/<username>` accepts a `username` parameter from the URL and returns a personalized welcome message.

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

Enabling **debug mode** in Flask is useful for development because it provides:

1. **Automatic Code Reloading** – The server automatically restarts when changes are made to the code.
2. **Detailed Error Messages** – Flask displays an interactive debugger when an error occurs, making it easier to identify and fix issues.

However, debug mode should **not** be enabled in a production environment due to security risks.

---

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

Debug mode can be enabled in two ways:

### 1. Setting **debug=True** in **app.run()**

python

CopyEd

it

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

.

## Code :-

### App.py

```
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/process', methods=['POST'])
def process():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        return redirect(url_for('result', name=name, age=age))

@app.route('/result')
def result():
    name = request.args.get('name')
    age = request.args.get('age')
    return render_template('result.html', name=name, age=age)

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

### Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flask App - URL Building</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Enter Your Details</h1>
        <p>Please fill in the form below:</p>
        <form action="{{ url_for('process') }}" 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>
```

### **Result.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result Page</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="result-container">
        <h2>Submitted Details</h2>
        <div class="card">
            <p><strong>Name:</strong> {{ name }}</p>
            <p><strong>Age:</strong> {{ age }}</p>
        </div>
        <a href="{{ url_for('index') }}">Go Back</a>
    </div>
</body>
</html>
```

```

body {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
  background-color: #e8f0fe;
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
  color: #333;
  background-image: linear-gradient(135deg, #6dd5ed, #2193b0);
}

.container, .result-container {
  background-color: #ffffff;
  padding: 30px;
  border-radius: 15px;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);
  text-align: center;
  width: 350px;
  margin: 20px;
  opacity: 0.95;
}

h1, h2 {
  margin-bottom: 20px;
  color: #34495e;
  font-size: 1.8em;
}

p {
  margin-bottom: 20px;
  color: #555;
  font-size: 1em;
}

form {

```

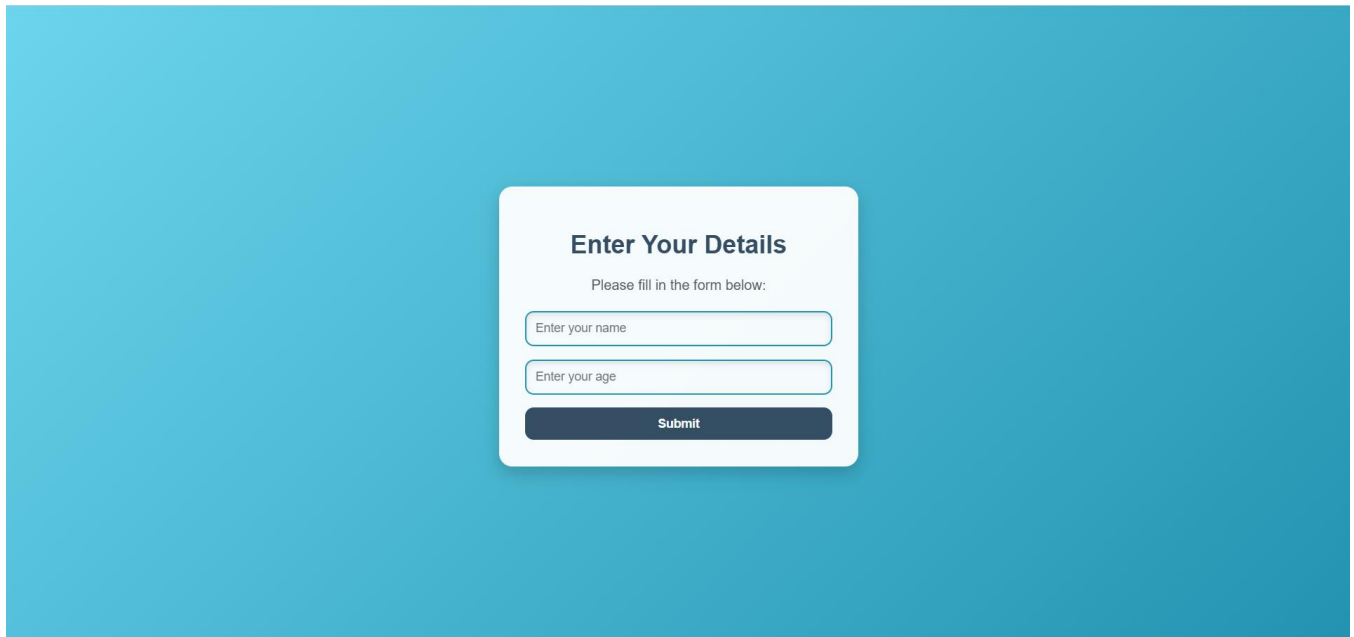
```

35 }
36
37 form {
38   display: flex;
39   flex-direction: column;
40 }
41
42 input {
43   margin-bottom: 15px;
44   padding: 10px;
45   border-radius: 10px;
46   border: 2px solid #2193b0;
47   font-size: 14px;
48   box-shadow: inset 0 3px 6px rgba(0, 0, 0, 0.1);
49   outline: none;
50   transition: border 0.3s;
51 }
52
53 input:focus {
54   border-color: #6dd5ed;
55   box-shadow: 0 0 10px rgba(109, 213, 237, 0.4);
56 }
57
58 button {
59   padding: 10px 20px;
60   background-color: #34495e;
61   color: white;
62   border: none;
63   border-radius: 10px;
64   cursor: pointer;
65   font-weight: bold;
66   font-size: 14px;
67   transition: background-color 0.3s, transform 0.2s;
68 }
69
70 button:hover {
71   background-color: #2c3e50;

```



## Result:-

A screenshot of a web form titled "Enter Your Details" on a blue gradient background. The form is a white rounded rectangle with a shadow. It contains a title, a subtitle, two empty input fields, and a submit button.

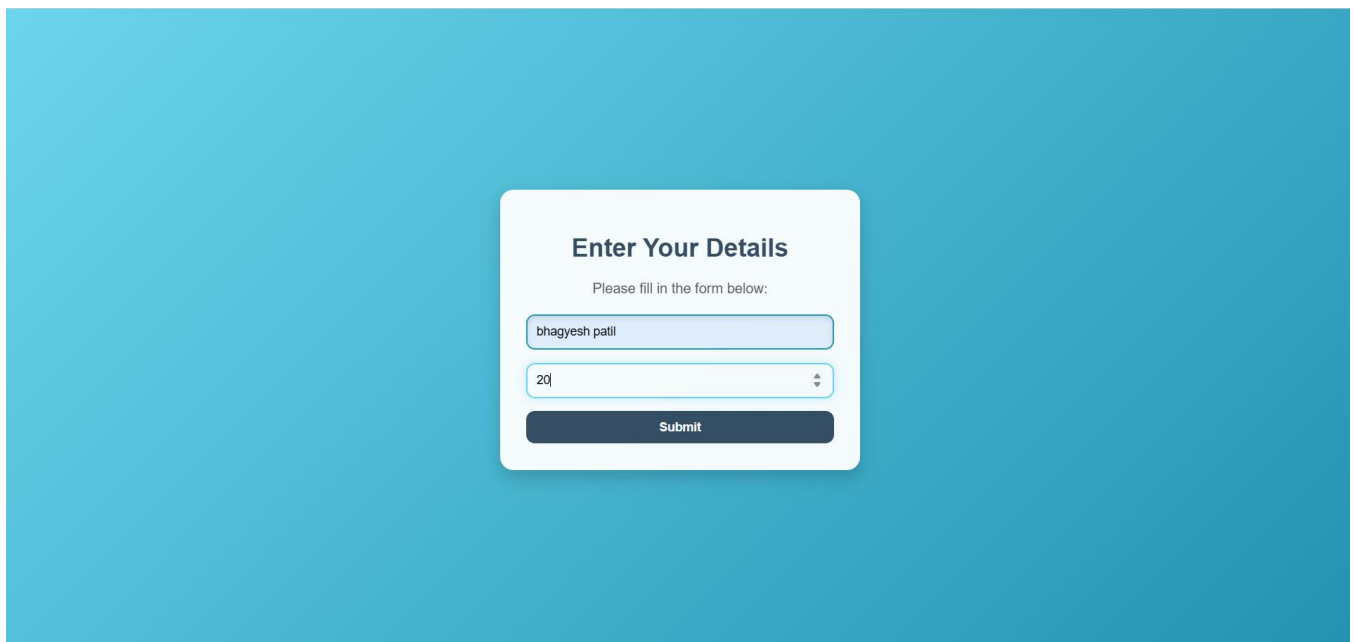
**Enter Your Details**

Please fill in the form below:

Enter your name

Enter your age

Submit

A screenshot of the same web form, but now filled with data. The input fields contain "bhagyesh patil" and "20".

**Enter Your Details**

Please fill in the form below:

bhagyesh patil

20

Submit

## Submitted Details

Name: bhagyesh patil

Age: 20

[Go Back](#)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

**WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.**

\* Running on http://127.0.0.1:5000

Press CTRL+C to quit

\* Restarting with stat

\* Debugger is active!

\* Debugger PIN: 109-556-029

```
127.0.0.1 - - [26/Mar/2025 21:24:23] "GET /result?name=bhagyesh+patil&age=19 HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:24:23] "GET /static/styles.css HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:24:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:24:25] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [26/Mar/2025 21:24:29] "POST /process HTTP/1.1" 302 -
127.0.0.1 - - [26/Mar/2025 21:24:29] "GET /result?name=bhagyesh+patil&age=20 HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:24:29] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [26/Mar/2025 21:24:31] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:24:31] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [26/Mar/2025 21:34:57] "POST /process HTTP/1.1" 302 -
127.0.0.1 - - [26/Mar/2025 21:34:57] "GET /result?name=bhagyesh+patil&age=20 HTTP/1.1" 200 -
127.0.0.1 - - [26/Mar/2025 21:34:57] "GET /static/styles.css HTTP/1.1" 304 -
█
```