Experiment 4 : Flask Application using GET and POST

Name of Student	**Prathamesh Palve**
Class Roll No	**D15A_31**
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?

In Flask, a **route** is a URL pattern that is associated with a specific function in a web application. When a user visits a particular URL, Flask executes the corresponding function and returns the response. Routes define how the application should respond to different URLs.
A route is defined using the `@app.route()` decorator. For example:
python
CopyEdit

```
from flask import

Flask app = Flask(__

name__________)
```

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

In this example, the root URL (`/`) is mapped to the `home` function, which returns a simple text response.

---

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

In Flask, parameters can be passed in a URL route using angle brackets (`< >`). These parameters can be dynamic and help in passing values from the URL to the function.

For example:

python
CopyEdit

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

Here, when a user visits `/user/Sanket`, the function receives `"Sanket"` as the `name` parameter and returns `"Hello, Sanket!"`.
Flask also allows specifying the data type of parameters, such as:

python

CopyEdit

```
@app.route('/square/<int:num>')
def square(num):
    return f"The square of {num} is {num**2}"
```

This ensures that `num` is treated as an integer.

---

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
CopyEdit

```
@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.

In Flask, these methods can be specified in the `methods` parameter of the `@app.route()` decorator:

python
CopyEdit

```
@app.route('/submit', methods=['POST'])
def submit():
    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
CopyEdit

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

2. **Using Environment Variables**
   In the terminal, set the environment variable before running the Flask app:

sh
CopyEdit
```
export FLASK_ENV=development
flask run
```

On Windows (Command Prompt):

sh
CopyEdit
```
set FLASK_ENV=development
flask run
```

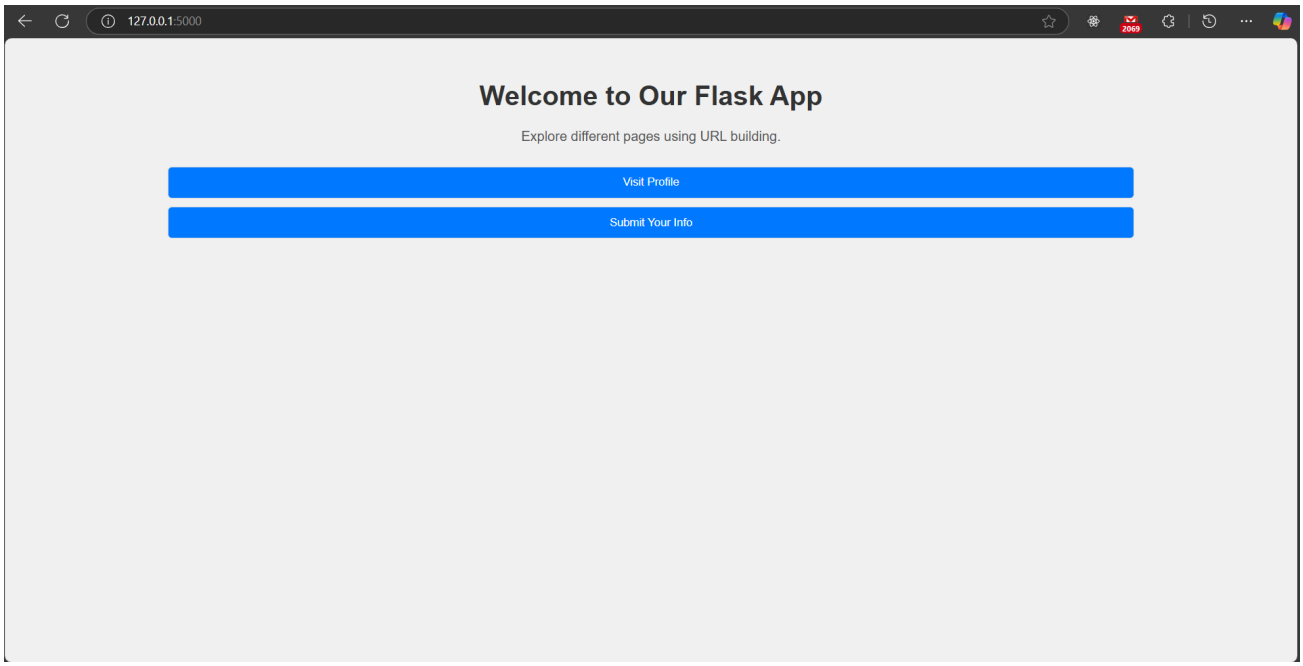
This enables debug mode and allows for easier debugging during development.

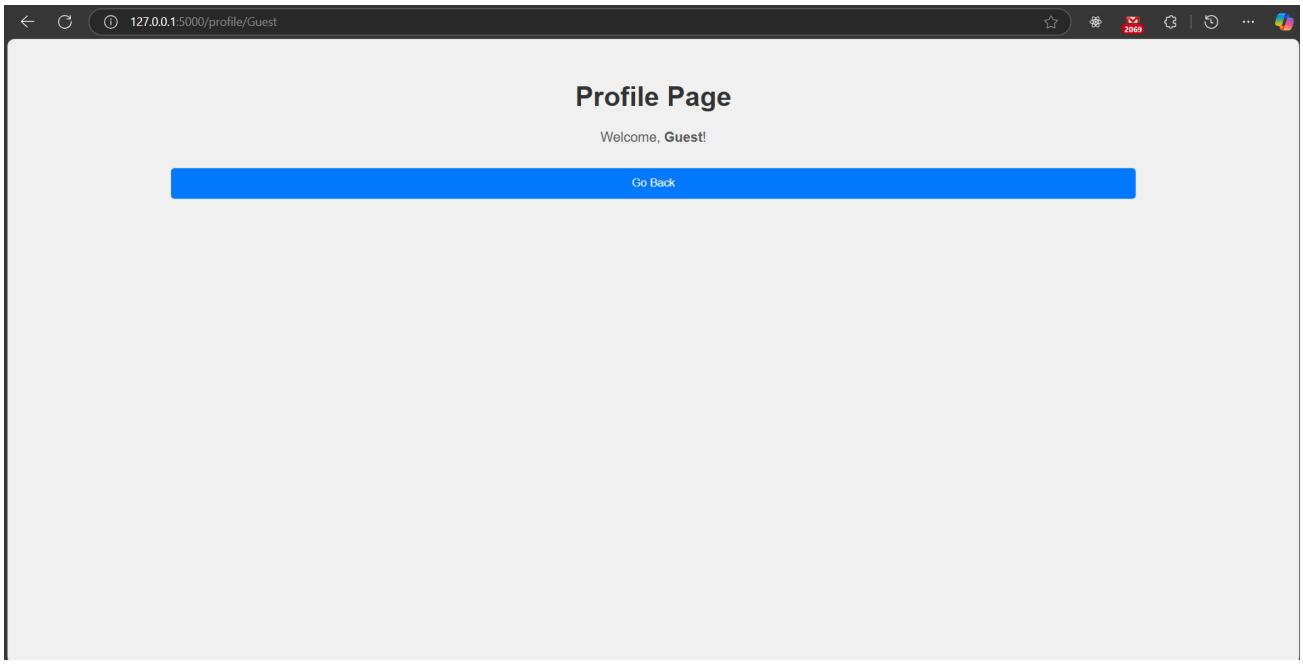
OUTPUT:-

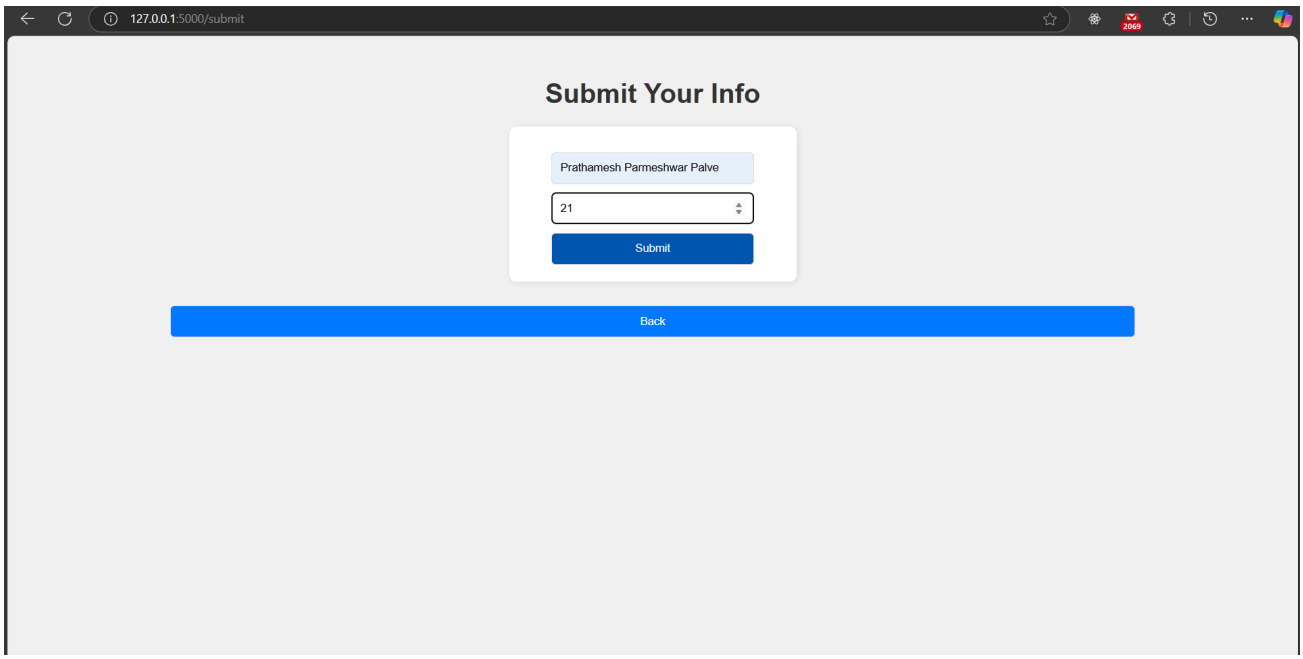
```
from flask import Flask, render_template_string, request, redirect, url_for

app = Flask(__name__)

# CSS Styling (to be used in all pages)
style = '''
    <style>
        body { font-family: Arial, sans-serif; background-color: #f4f4f4;
text-align: center; margin: 50px; }
        h1 { color: #333; }
        p { color: #555; }
        form { background: white; padding: 20px; border-radius: 8px; width: 300px;
margin: auto; box-shadow: 2px 2px 10px rgba(0,0,0,0.1); }
        input, button { margin-top: 10px; padding: 10px; width: 80%; border: 1px
solid #ddd; border-radius: 5px; }
        button { background-color: #007bff; color: white; cursor: pointer; }
        button:hover { background-color: #0056b3; }
        a { text-decoration: none; }
    </style>
'''

# Homepage Route
@app.route('/')
def home():
    return render_template_string(f'''
        {style}
        <h1>Welcome to Our Flask App</h1>
        <p>Explore different pages using URL building.</p>
        <a href="{url_for('profile', username='Guest')}"><button>Visit
Profile</button></a>
        <a href="{url_for('submit')}"><button>Submit Your Info</button></a>
    ''')

# Profile Page Route (Dynamic username in URL)
@app.route('/profile/<username>')
def profile(username):
    return render_template_string(f'''
        {style}
        <h1>Profile Page</h1>
        <p>Welcome, <b>{username}</b>!</p>
        <a href="/"><button>Go Back</button></a>
    ''')

# Submit Page Route (GET & POST)
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        return redirect(url_for('confirmation', username=name, age=age))

    return render_template_string(f'''
        {style}
        <h1>Submit Your Info</h1>
        <form method="post">
            <input type="text" name="name" placeholder="Enter your name"
required><br>
            <input type="number" name="age" placeholder="Enter your age"
required><br>
            <button type="submit">Submit</button>
        </form>
```

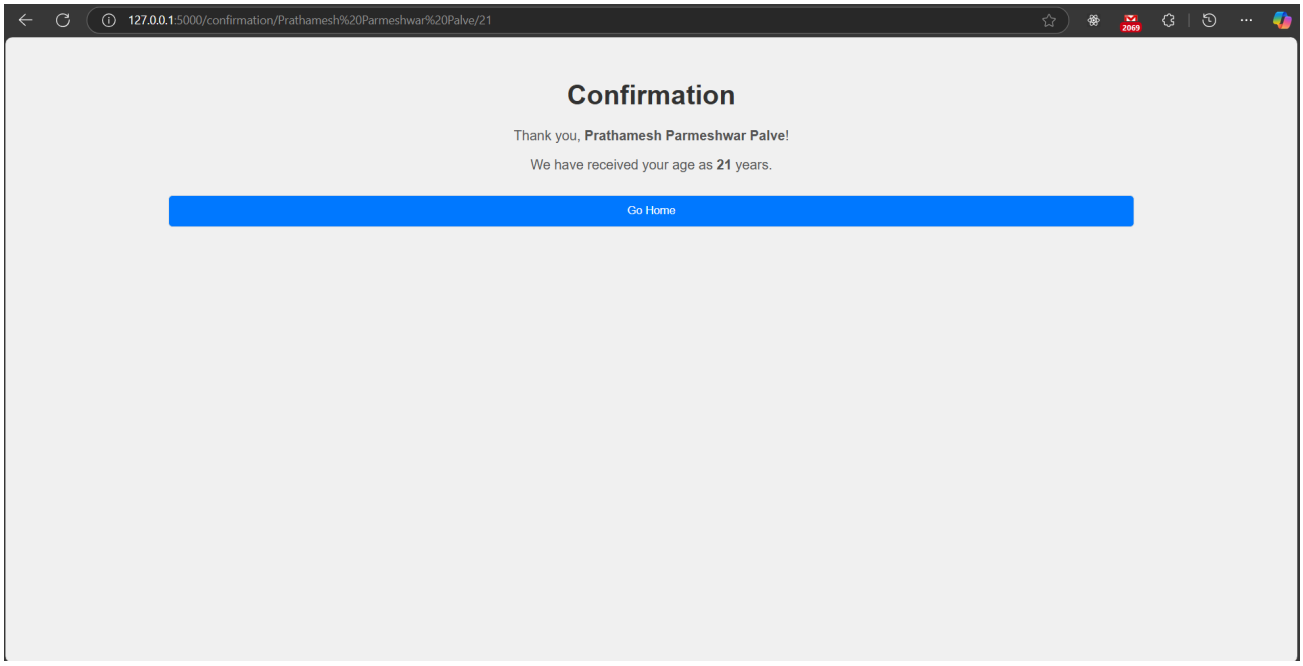
```
        <br>
        <a href="/"><button>Back</button></a>
    ''')

# Confirmation Page (Displays submitted data)
@app.route('/confirmation/<username>/<age>')
def confirmation(username, age):
    return render_template_string(f'''
        {style}
        <h1>Confirmation</h1>
        <p>Thank you, <b>{username}</b>!</p>
        <p>We have received your age as <b>{age}</b> years.</p>
        <a href="/"><button>Go Home</button></a>
    ''')

# Run the Flask app
if __name__ == '__main__':
    app.run(debug=True)
```

Result:-


127.0.0.1:5000
Welcome to Our Flask App
Explore different pages using URL building.
Visit Profile
Submit Your Info


127.0.0.1:5000/profile/Guest
Profile Page
Welcome, Guest!
Go Back






```
prath@prathmzzPC MINGW64 ~/Desktop/Hackathons/temp/temp4
$ python temp.py
 * Serving Flask app 'temp'
 * Debug mode: on
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: 586-033-160
127.0.0.1 - - [30/Mar/2025 23:03:45] "POST /contact HTTP/1.1" 404 -
127.0.0.1 - - [30/Mar/2025 23:03:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2025 23:03:57] "GET /profile/Guest HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2025 23:04:05] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2025 23:04:07] "GET /submit HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2025 23:04:20] "POST /submit HTTP/1.1" 302 -
127.0.0.1 - - [30/Mar/2025 23:04:20] "GET /confirmation/Prathamesh%20Parmeshwar%20Palve/21 HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2025 23:04:27] "GET / HTTP/1.1" 200 -
```