EXPERIMENT NO. 4 - Flask Application using GET and POST

Name of Student	**Kartik Bhat**
Class Roll No	**D15A 3**
D.O.P.	
D.O.S.	
Sign and Grade	

EXPERIMENT 4

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:nam

   e>') 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

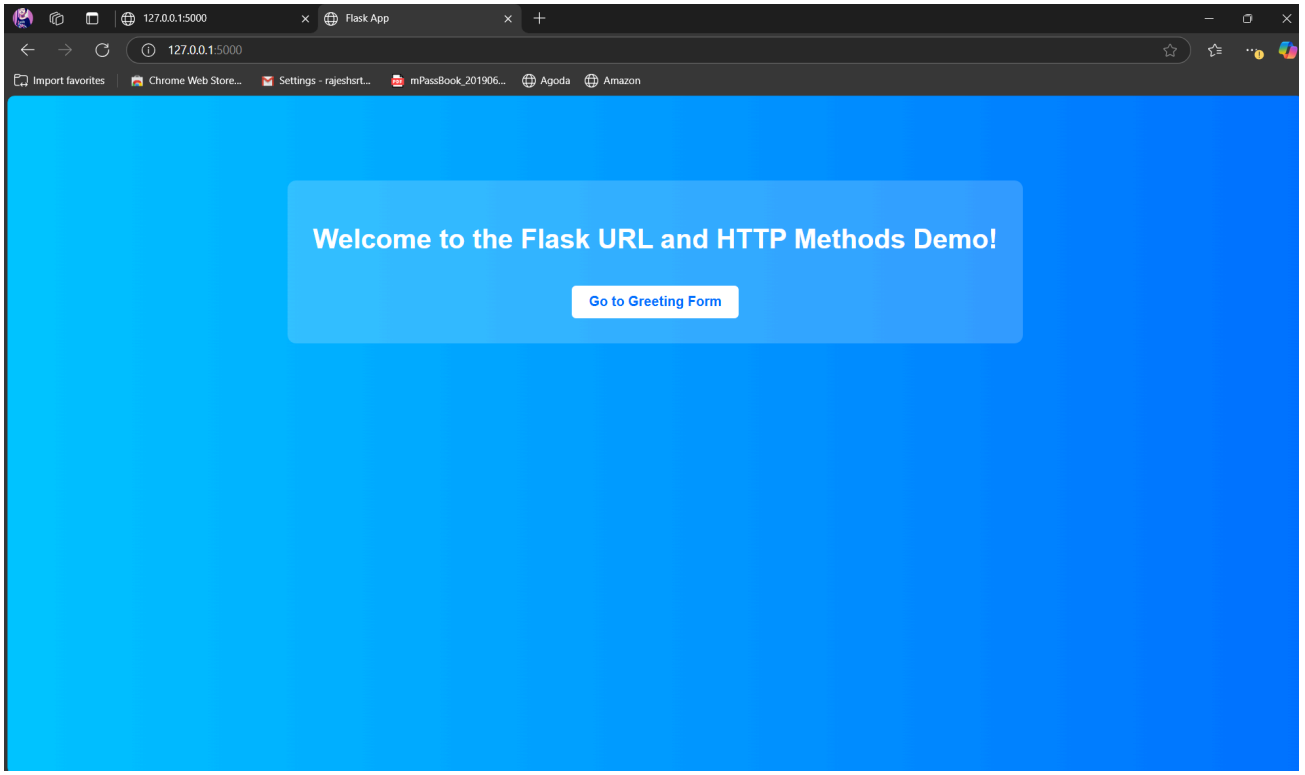
CODE:

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


app = Flask(__name__)


@app.route('/')

def home():

    return render_template('index.html')


@app.route('/greet', methods=['GET', 'POST'])

def greet():

    if request.method == 'POST':

        name = request.form.get('name')

        return redirect(url_for('hello', username=name))

    return render_template('greet.html')


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

def hello(username):
```

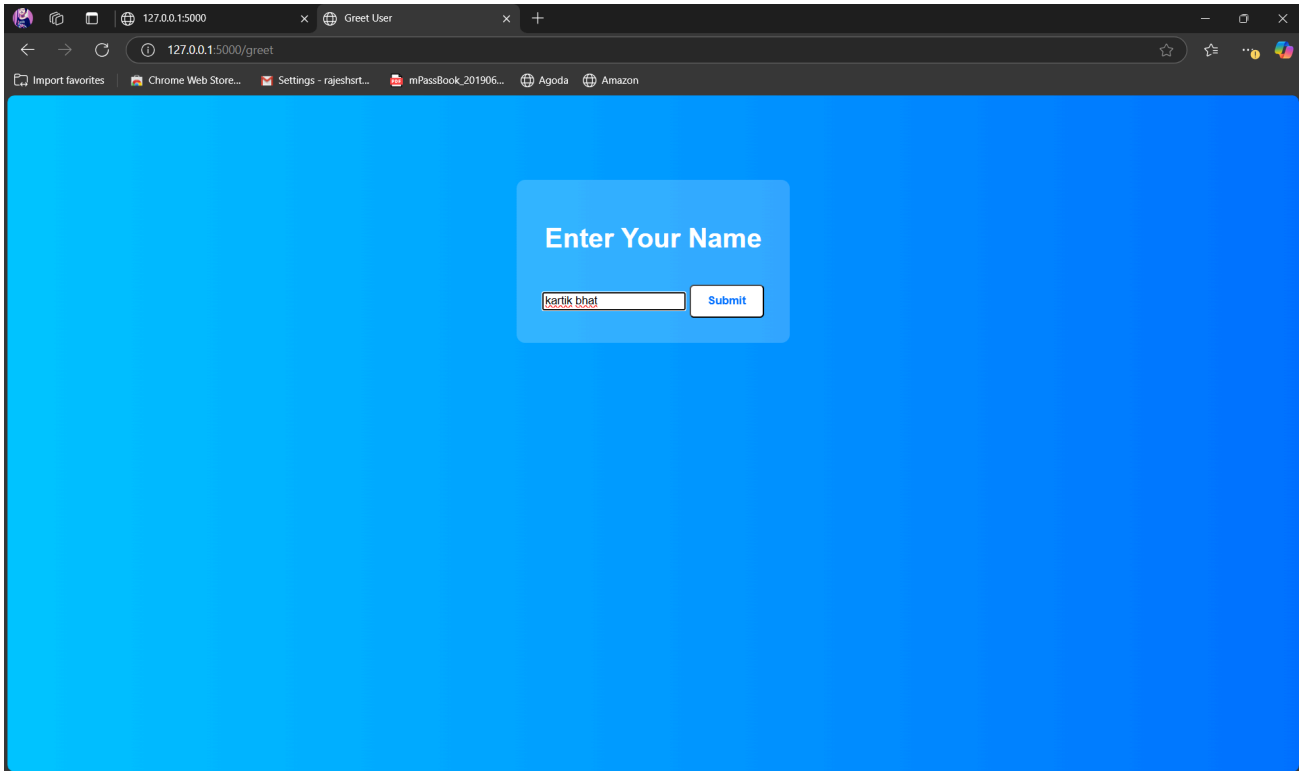
```
    return render_template('hello.html', username=username)


if __name__ == '__main__':

    app.run(debug=True)
```

OUTPUT :

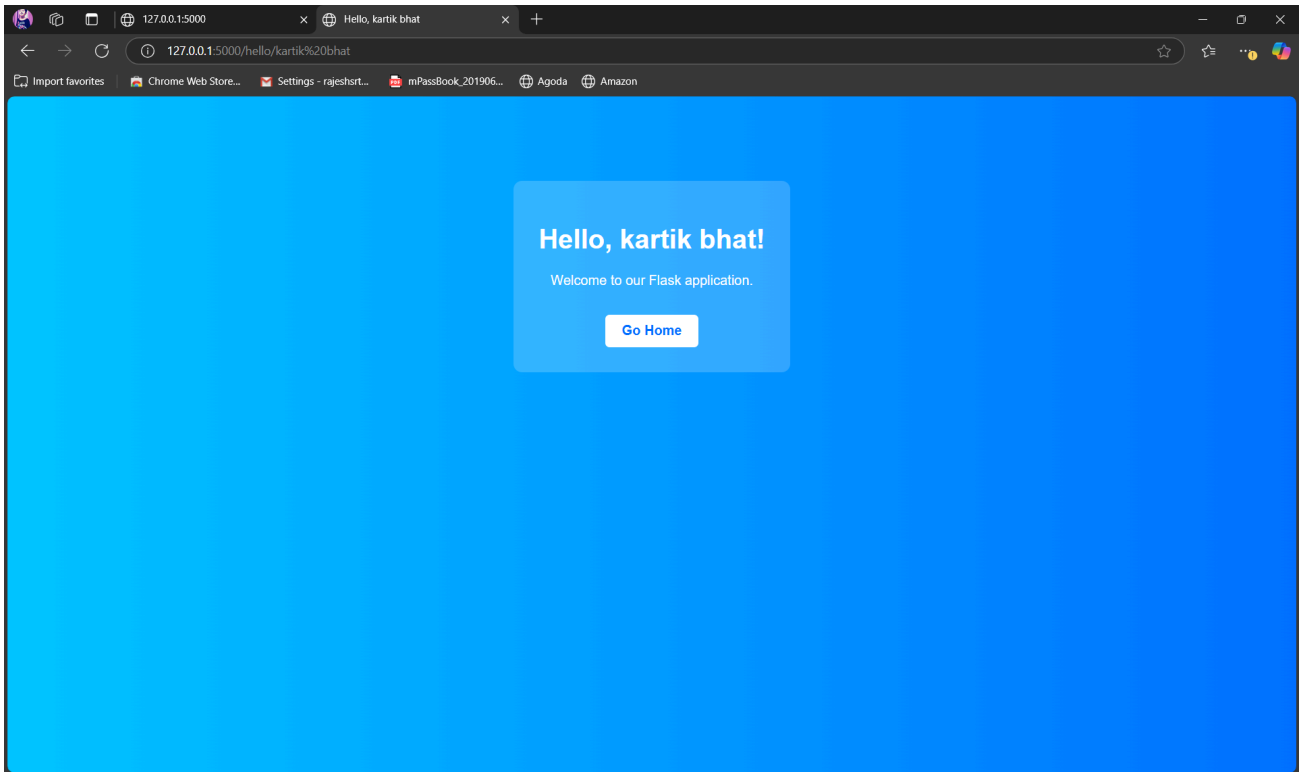
Homepage: Displays a welcome message with navigation links to the "Profile" and "Submit" pages.



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.