EXPERIMENT NO. 5

Name of Student	kartik bhat
Class Roll No	D15A 3
DOP	
DOS	
Sign and Grade	

EXPERIMENT 5

AIM

To create a Flask application that demonstrates template rendering by dynamically generating HTML content using the `render_template()` function.

PROBLEM STATEMENT

Develop a Flask application that includes:

1. A homepage route (`/`) displaying a welcome message with links to additional pages.

2. A dynamic route (`/user/<username>`) that renders an HTML template with a personalized greeting.

3. Use Jinja2 templating features, such as variables and control structures, to enhance the templates.

Theory

1. What does the render_template() function do in a Flask application?

The `render_template()` function in Flask is used to render HTML templates and return them as responses to client requests. Instead of returning plain text or manually writing HTML inside the Python code, Flask allows the use of separate HTML files stored in the `templates` folder.

Usage Example:
python CopyEdit `from flask import Flask,`
```
render_template   app = Flask(__name__)

@app.route('/')  def home():
return render_template('index.html')
```

Here, `render_template('index.html')` loads the `index.html` file from the `templates` folder and sends it as a response. This helps in separating logic from presentation, making web applications more organized and maintainable.

Additionally, `render_template()` supports passing dynamic data to templates:

python CopyEdit
```
@app.route('/user/<name>')  def user(name):
return render_template('user.html', username=name)
```

In `user.html`, we can access `username` using Jinja2 templating:

html CopyEdit
```
<p>Hello, {{ username }}!</p>
```

---

2. What is the significance of the templates folder in a Flask project?

The `templates` folder holds all the HTML files used for rendering web pages in a Flask application. Flask automatically looks for template files inside this directory, making it a convention that helps in maintaining a well-structured project.

Key Significance:

1. **Separation of Concerns** – Keeps the HTML structure separate from Python logic, improving code readability.
2. **Easy Management** – All templates are stored in one location, simplifying maintenance.
3. **Supports Jinja2** – Enables the use of dynamic content within HTML files through Jinja2 templating.
4. **Enables Code Reusability** – Common UI components, such as headers and footers, can be stored in separate template files and reused across multiple pages using template inheritance.

Project Structure Example:

bash CopyEdit
```
/my_flask_app
|── app.py
|── /templates
```

```
|    |__ index.html
|    |__ user.html
|    |__ base.html
|── /static
|    |__ styles.css
|    |__ script.js
```

Here, `index.html` and `user.html` are stored inside the `templates` folder and can be rendered using `render_template()`.

---

3. What is Jinja2, and how does it integrate with Flask?

Jinja2 is a powerful templating engine used in Flask to generate dynamic HTML content. It allows embedding Python-like expressions inside HTML, making web pages more interactive and adaptable based on user input or backend data.

Integration with Flask:

Flask uses Jinja2 by default when rendering templates through `render_template()`. The syntax includes:

- **Variables** – `{{ variable_name }}`
- **Control Structures** – `{% if condition %} ... {% endif %}`
- **Loops** – `{% for item in list %} ... {% endfor %}`

Example Usage:

Python Code (Flask App)

python CopyEdit
```
@app.route('/greet/<name>')  def greet(name):
return render_template('greet.html', username=name)
```

Jinja2 Template (`greet.html`)

html CopyEdit

```
<!DOCTYPE html>
<html>
<head>
    <title>Greeting</title>
</head>
<body>
    <h1>Hello, {{ username }}!</h1>
</body>
</html>
```

Features of Jinja2 in Flask:

1. **Template Inheritance** – Allows reusing base layouts using `{% extends "base.html" %}` and `{% block content %} ... {% endblock %}`.

2. **Filters** – Modify data output (e.g., `{{ name.upper() }}` converts text to uppercase).

3. **Control Structures** – Supports conditionals and loops for dynamic content.

Jinja2 enhances the flexibility of Flask applications by enabling dynamic content generation within HTML templates.

Implementation of the Flask Application

Step 1: Install Flask (if not already installed)

sh CopyEdit
```
pip
install flask
```

Step 2: Create the Flask Application `app.py`

python CopyEdit
```
from flask import Flask,
render_template

app = Flask(__name__)
```

```
@app.route('/')
def home():
    return render_template('index.html')

@app.route('/user/<username>')
def user(username):
    return render_template('user.html', username=username)
 if __name__ ==
'__main__':
app.run(debug=True)
```

Step 3: Create the `templates` folder and add the following files

1. `templates/index.html`

html CopyEdit

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{{ title }}</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>{{ message }}</h1>
        <ul>
            <li><a href="{{ url_for('kartik') }}">Kartik's Page</a></li>
            <li><a href="{{ url_for('joe') }}">Joe's Page</a></li>
        </ul>
    </div>
</body>
</html>
```

2. **`templates/user.html`**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{{ title }}</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body class="user-page">
    <div class="container">
        <h1>Welcome, {{ username }}!</h1>
        <p>This is your personalized page.</p>
        <div class="nav-links">
            <a href="{{ url_for('home') }}">Home</a>
            <a href="{{ url_for('kartik') }}">Kartik's Page</a>
            <a href="{{ url_for('joe') }}">Joe's Page</a>
            <a href="{{ url_for('about') }}">About</a>
            <a href="{{ url_for('contact') }}">Contact</a>
        </div>
    </div>
</body>
</html>
```

Step 4: Run the Flask Application

Open a terminal in the project directory and run:

sh CopyEdit
```
python
app.py
```

If Flask is correctly set up, you should see:

csharp CopyEdit
```
* Running on http://127.0.0.1:5000/
```

Visit http://127.0.0.1:5000/ to see the homepage and http://127.0.0.1:5000/user/YourName for a personalized greeting.

Code:

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

app = Flask(__name__)

# Homepage displaying welcome message and navigation links
@app.route('/')
def home():
    return render_template('index.html', title='Home', message='Welcome to Our Flask App! Navigate using the links below.', links=["Kartik's Page", "Joe's Page"])

# Kartik's Page
@app.route('/kartik')
def kartik():
    return render_template('user.html', title="Kartik's Page", username='Kartik', links=['Home', "Joe's Page", 'About', 'Contact'], page_color='blue')

# Joe's Page
@app.route('/joe')
def joe():
    return render_template('user.html', title="Joe's Page", username='Joe', links=['Home', "Kartik's Page", 'About', 'Contact'], page_color='green')
```

```python
# Greeting form route
@app.route('/greet', methods=['GET', 'POST'])
def greet():
if request.method == 'POST':
name = request.form.get('name')
return redirect(url_for('user', username=name))
return render_template('greet.html', title='Greet User',
links=['Home', 'About', 'Contact'])

# About page route
@app.route('/about')
def about():
return render_template('about.html', title='About Us', message='This
Flask app demonstrates dynamic template rendering using Jinja2.',
links=['Home', "Kartik's Page", "Joe's Page", 'Contact'])

# Contact form route
@app.route('/contact', methods=['GET', 'POST'])
def contact():
if request.method == 'POST':
message = request.form.get('message')
return render_template('contact.html', title='Contact Us',
message=message, links=['Home', "Kartik's Page", "Joe's Page", 'About'])
return render_template('contact.html', title='Contact Us',
message=None, links=['Home', "Kartik's Page", "Joe's Page", 'About'])

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

templates/index.html

```html
<!DOCTYPE html>

<html lang="en">

<head>

    <title>{{ title }}</title>

    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">

</head>

<body>

    <div class="container">

        <h1>{{ message }}</h1>

        <ul>

            <li><a href="{{ url_for('kartik') }}">Kartik's Page</a></li>

            <li><a href="{{ url_for('joe') }}">Joe's Page</a></li>

        </ul>

    </div>

</body>

</html>
```

templates/user.html

```html
<!DOCTYPE html>

<html lang="en">

<head>

    <title>{{ title }}</title>

    <link rel="stylesheet" href="{{ url_for('static', 
filename='styles.css') }}">

</head>

<body class="user-page">

    <div class="container">

        <h1>Welcome, {{ username }}!</h1>

        <p>This is your personalized page.</p>

        <div class="nav-links">

            <a href="{{ url_for('home') }}">Home</a>

            <a href="{{ url_for('kartik') }}">Kartik's Page</a>

            <a href="{{ url_for('joe') }}">Joe's Page</a>

            <a href="{{ url_for('about') }}">About</a>

            <a href="{{ url_for('contact') }}">Contact</a>

        </div>

    </div>

</body>
```

```
</html>
```

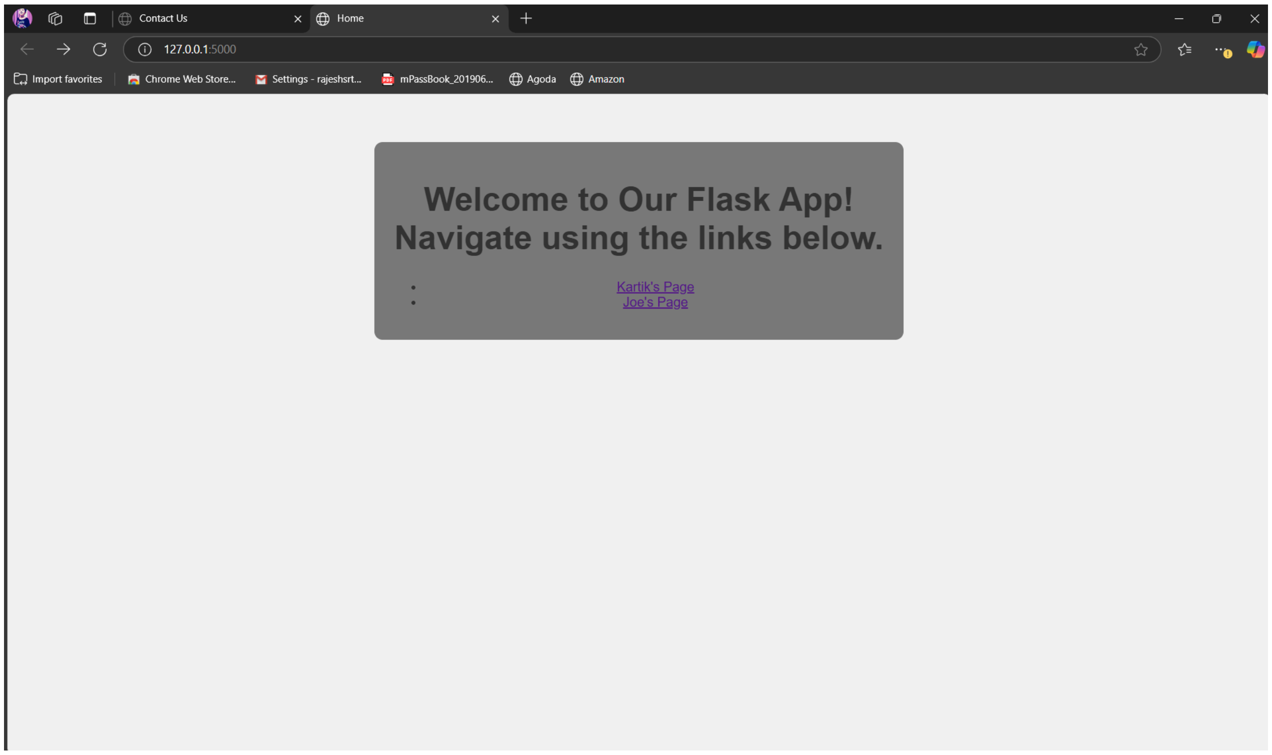
---

static/style.css

```
/* General Styling */

body {

    font-family: Arial, sans-serif;

    text-align: center;

    background-color: #f4f4f4;

    color: #333;

    padding: 50px;

}



/* User Page Styling */

.user-page {

    background-color: #2d89ef; /* Blue Theme */

    color: white;

}



.container {
```

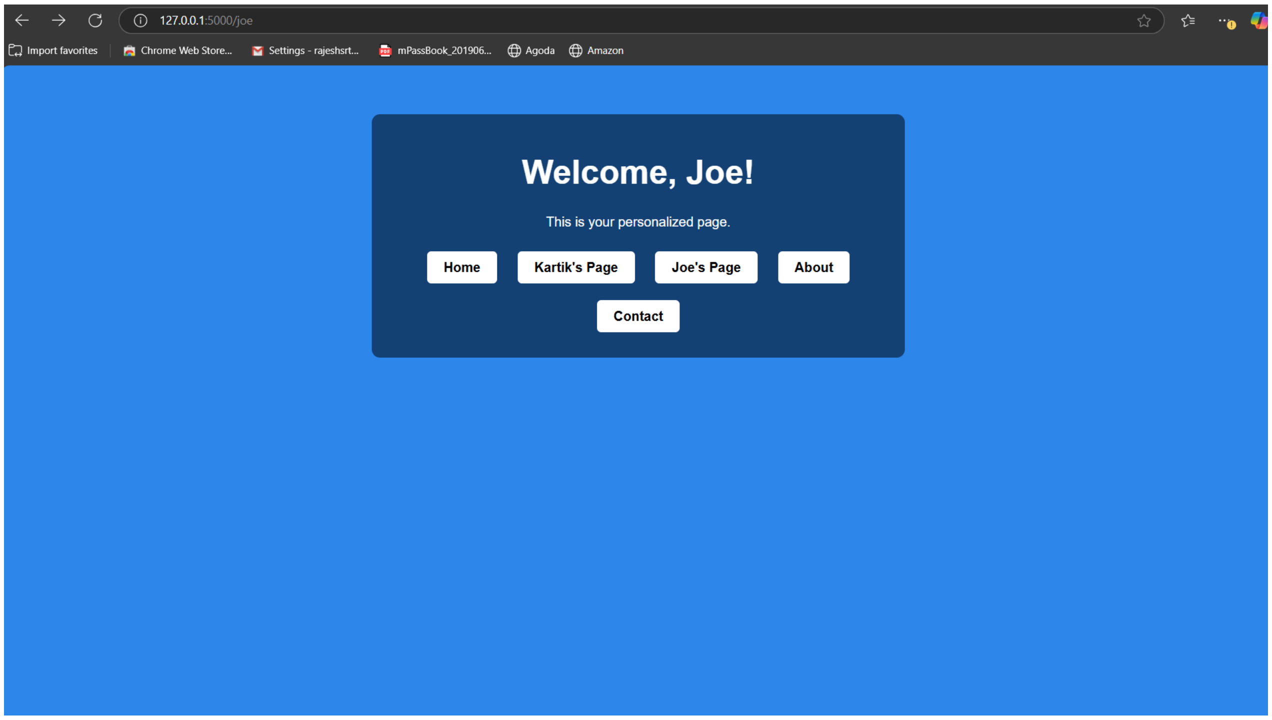
```css
    max-width: 600px;

    margin: auto;

    padding: 20px;

    background: rgba(0, 0, 0, 0.5);

    border-radius: 10px;

}



h1 {

    font-size: 2.5em;

}



/* Navigation Links */

.nav-links a {

    display: inline-block;

    margin: 10px;

    padding: 10px 20px;

    background: white;

    color: black;

    text-decoration: none;

    font-weight: bold;

    border-radius: 5px;
    transition: 0.3s;

}



.nav-links a:hover {

    background: lightgray;

}
```

Result:



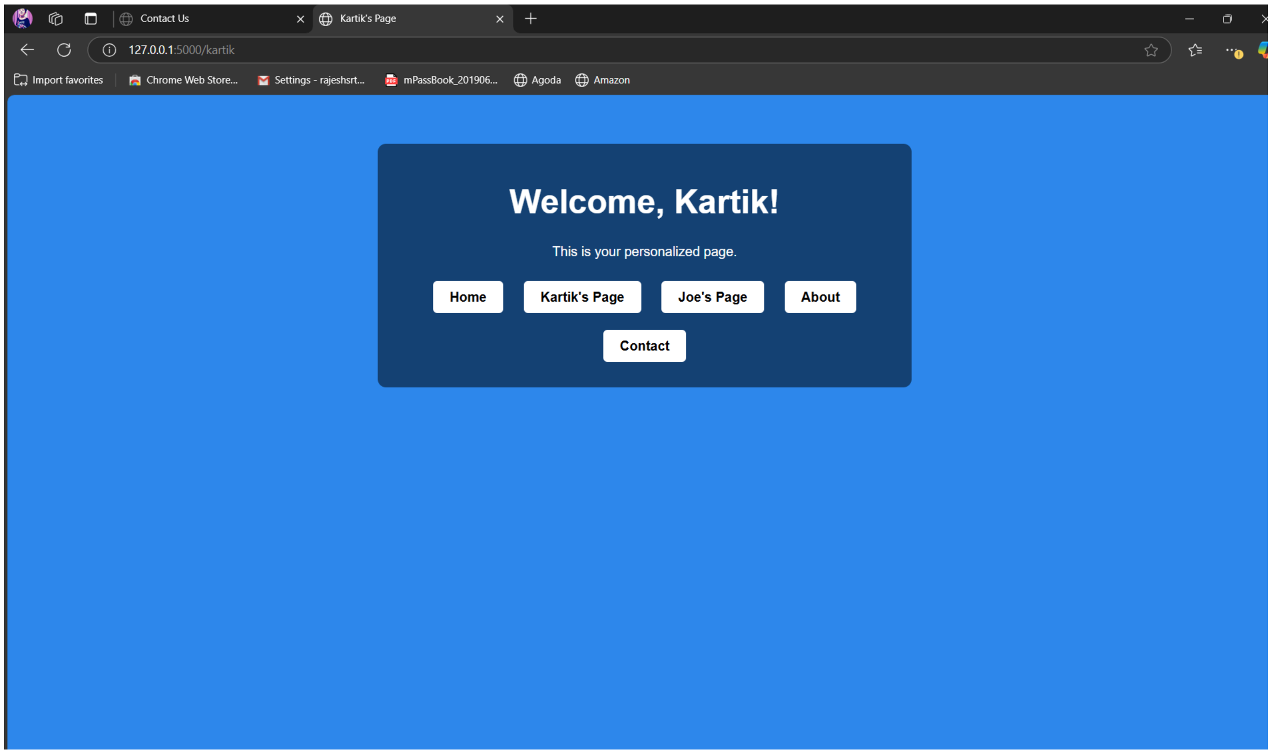
Welcome to Our Flask App! Navigate using the links below.

- Kartik's Page
- Joe's Page



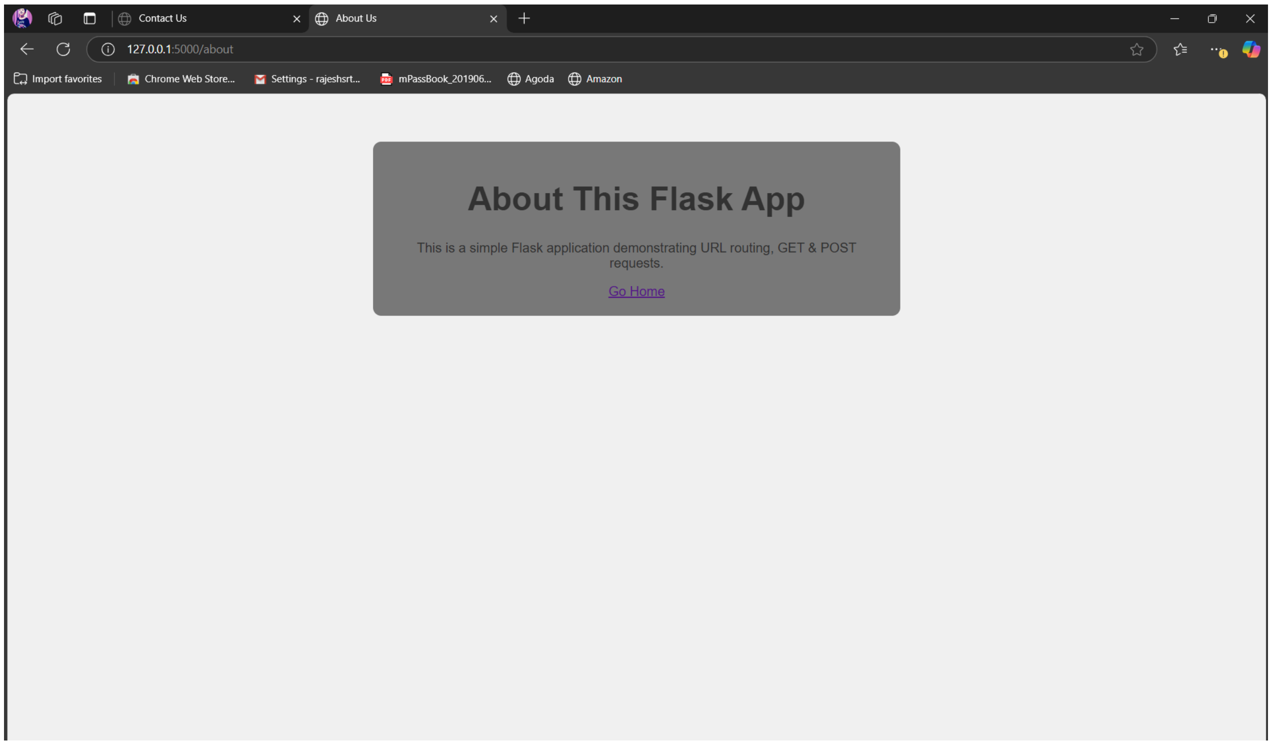
Welcome, Joe!

This is your personalized page.



Welcome, Kartik!

This is your personalized page.


Contact Us | About Us

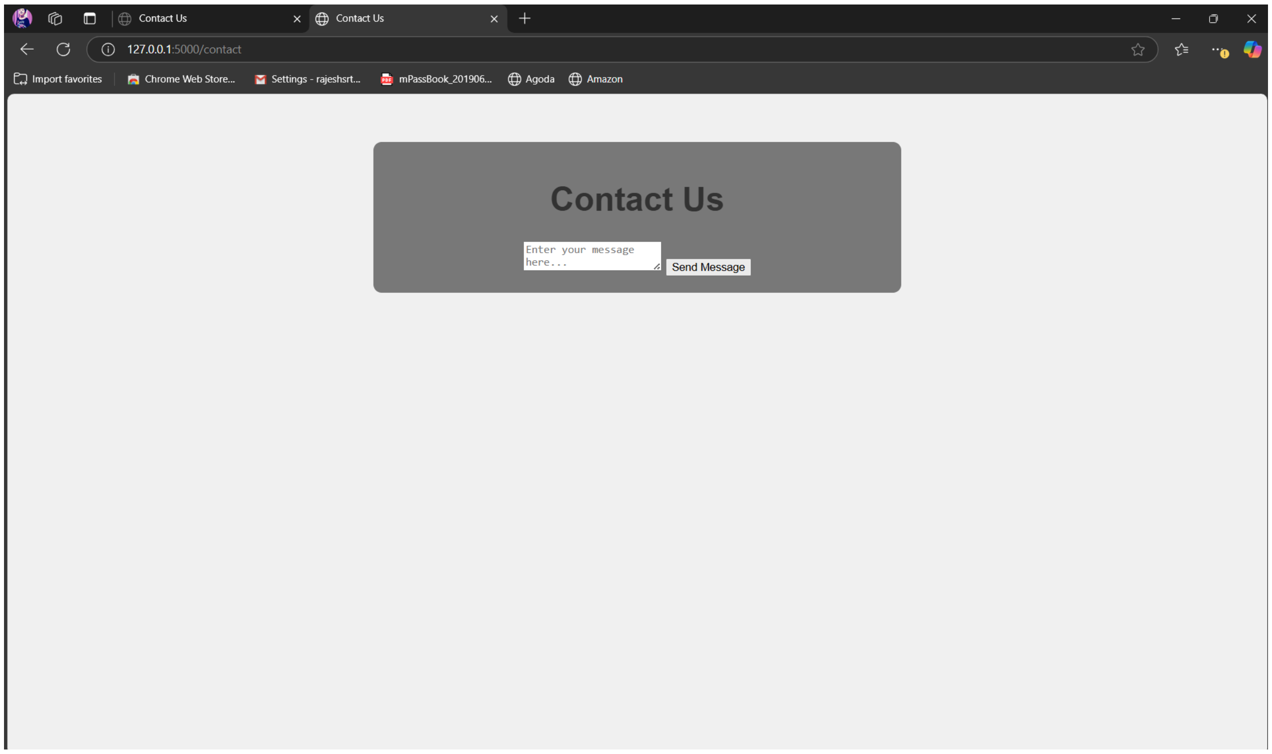
127.0.0.1:5000/about

Import favorites | Chrome Web Store... | Settings - rajeshsrt... | mPassBook_201906... | Agoda | Amazon

About This Flask App

This is a simple Flask application demonstrating URL routing, GET & POST requests.

Go Home


Contact Us | Contact Us

127.0.0.1:5000/contact

Import favorites | Chrome Web Store... | Settings - rajeshsrt... | mPassBook_201906... | Agoda | Amazon

Contact Us

Enter your message here...

Send Message