• Introduction to embedding HTML within Python using web frameworks like Django or Flask.

Ans: Flask is a lightweight web framework that is easy to set up and use. It uses Jinja2 as its templating engine, which allows you to embed Python code within HTML.

1 .Install :

Pip install flask

**2.Create a Basic Flask Application**: Here’s a simple example of a Flask application that renders an HTML template.

From flask import flask,render\_template

3. **Create an HTML Template**: Create a folder named **templates** in the same directory as your Python file, and inside it, create a file named **index.html**.

<html>

<head>

<title>

</title>

</head>

<body>

</body>

<html>

4. **Run the Application**: Execute your Python script, and navigate to **http://127.0.0.1:5000/** in your web browser to see the rendered HTML

**Django**

Django is a more feature-rich web framework that follows the Model-View-Template (MVT) architecture. It also uses a templating engine to embed Python code within HTML.

**1.install**

**Pip install django**

**2. Create a Django Project**: Create a new Django project and an application:

Django-admin stratproject myproject

Cd myproject

Django-admin stratapp myapp

3. **Configure the Application**: Add your app to the **INSTALLED\_APPS** list in **settings.py**:

INSTALLED\_APPS = [

...

'myapp',

]

4. **Create a View**: In **myapp/views.py**, create a view that renders an HTML template.

from django.shortcuts import render

def home(request):

return render(request, 'index.html')

5. **Set Up URLs**: In **myapp/urls.py**, define a URL pattern for your view.

from django.urls import path

from .views import home

urlpatterns = [

path('', home, name='home'),

]

6. **Create an HTML Template**: Create a folder named **templates** inside your app directory (**myapp/templates/**) and create an **index.html** file.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Django App</title>

</head>

<body>

<h1>Welcome to Django!</h1>

<p>This is a simple Django application.</p>

</body>

</html>

7. **Run the Application**: Start the Django development server:

python manage.py runserver

• Generating dynamic HTML content using Django templates.

The user wants an explanation and example about generating dynamic HTML content using Django templates. This involves demonstrating how Django templates can take dynamic data passed from views and render it in HTML. The content should explain template variables, template tags (like loops and conditionals), and how dynamic data is accessed in templates.

**1. Passing Dynamic Data from Views**

In your Django view, you prepare data (typically in a dictionary called **context**) and pass it to the template using the **render()** function.

from django.shortcuts import render

def profile\_view(request):

user\_data = {

'name': 'Alice',

'age': 30,

'hobbies': ['Reading', 'Hiking', 'Coding'],

'is\_premium\_user': True,

}

return render(request, 'profile.html', context=user\_data)

**2. Accessing Data in the Template**

In your HTML template, you use Django’s template language to access variables and control flow elements.

<h1>Welcome, Alice!</h1>

<p>Age: 30</p>

<h3>Hobbies:</h3>

<ul>

<li>Reading</li>

<li>Hiking</li>

<li>Coding</li>

</ul>

<p>Thank you for being a premium user! ★</p>

#### 3. Result

When accessing the **/profile\_view/** URL (configured accordingly), Django renders the template replacing variables and executing control structures, for example:

<h1>Welcome, Alice!</h1>

<p>Age: 30</p>

<h3>Hobbies:</h3>

<ul>

<li>Reading</li>

<li>Hiking</li>

<li>Coding</li>

</ul>

<p>Thank you for being a premium user! ★</p>