**Flask API Mini-Project**

**Task 1:**

1. **What is Flask?**

* Flask is a lightweight and flexible Python web framework that provides essential tools for building web applications. It's considered a microframework because it doesn't require particular tools or libraries. This gives developers the freedom to choose the components they want to use.
* **Key Features:**
  + **Lightweight:** Minimal core with extension possibilities.
  + **Flexible:** Unopinionated, allowing developers to choose their tools.
  + **Routing:** Easy and clean URL routing.
  + **Templating:** Support for Jinja2 templating engine.
  + **WSGI Compliance:** Based on Werkzeug WSGI toolkit.
  + **Extensible:** Many extensions available for added functionality.

1. **How do you install and run Flask?**

* To install Flask, Open your terminal or command prompt and use pip (Python package installer):
* *pip install flask*
* To run Flask, you need to create a simple Flask app/file. For example you can name your file as “app.py”
* from flask import Flask

app = Flask(\_\_name\_\_)

@app.route("/")

def home():

return "Hello, Flask!"

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

* To run the Flask after you have created your file;
* *python app.py*
* By default, Flask runs on **http://127.0.0.1:5000** (localhost port 5000). The debug=True parameter enables **auto-reload** and error debugging.

1. How do you define API routes and return JSON responses?

* Flask makes it easy to create API endpoints that return JSON responses using the Flask.jsonify() method.

Example: Creating API Routes

from flask import Flask, jsonify

app = Flask(\_\_name\_\_)

@app.route("/api", methods=["GET"])

def api():

data = {"message": "Welcome to the API", "status": "success"}

return jsonify(data) # Returns JSON response

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**Key points:**

* + Use @app.route("/endpoint") to define a route.
  + Use jsonify() to return JSON responses.
  + You can specify HTTP methods (GET, POST, etc.) inside methods=[...].

1. What is Jinja2, and how is it used in Flask for rendering HTML Templates?

* Jinja2 is a powerful templating engine used in Flask to generate dynamic HTML content. It allows developers to embed Python-like expressions inside HTML files, making web pages more interactive
* Steps to Use Jinja2 in Flask

1. Create a templates folder (Flask automatically looks for templates inside a folder named ***templates.***)
2. Create an HTML file (index.html) inside the *templates/* folder.
3. Use render\_template() in Flask to serve the HTML file.

* Examples on how to Render an HTML Template

The project structure should be like this;

/project

├── app.py

├── templates/

├── index.html

app.py (Flask app):

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route("/")

def home():

return render\_template("index.html", title="Welcome Page")

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

templates/index.html (Jinja2 template):

<!DOCTYPE html>

<html lang="en">

<head>

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

</head>

<body>

<h1>{{ title }}</h1>

<p>Welcome to my Flask app!</p>

</body>

</html>

**Jinja2 Syntax in HTML:**

* {{ variable }} → Outputs a variable inside the template.
* {% ... %} → Used for control statements (loops, conditionals).

Example for Looping through a list in Jinja2

<ul>

{% for item in ["Apple", "Banana", "Cherry"] %}

<li>{{ item }}</li>

{% endfor %}

</ul>

**Summary:**

* Flask is a lightweight Python web framework for building web applications and APIs.
* Install Flask using pip install flask, and run it using python app.py.
* Define API routes using @app.route() and return JSON responses with jsonify().
* Jinja2 is Flask’s templating engine, allowing dynamic content rendering in HTML templates using *render\_template().*