Restful API & Flask | ASSIGNMENT

Question 1 : What is a RESTful API?

Answer : REST (Representational State Transfer) is an architectural style that defines a set of constraints to be used for creating web services.

- Web services that adhere to the REST architecture are called RESTful services or REST APIs.
- These APIs allow different software applications to communicate over the internet, often using standard HTTP methods.

Question 2: What is Flask, and why is it popular for building APIs?

Answer: Flask is a lightweight, flexible Python microframework for building web applications.

It's known for its simplicity, ease of use, and ability to be extended with various libraries and tools.

Flask is particularly well-suited for smaller projects, prototypes, and applications where customization and control are valued.

Question 3: What are HTTP methods used in RESTful APIs?

Answer: REST APIs use standard HTTP methods to perform operations on resources.
Each method corresponds to a specific action :

1. **POST**: Creates a new resource
2. **GET**: Retrieves a resource or collection of resources
3. **PUT**: Updates an existing resource
4. **DELETE**: Remove a resource

Question 4: What is the purpose of the @app.route() decorator in Flask?

Answer : The `@app.route()` decorator in Flask serves the purpose of binding a URL path to a specific Python function, effectively creating a route within your Flask web application. When a user navigates to the specified URL in their web browser, the function decorated with `@app.route()` is executed, and its return value is sent back as the response to the user's request.

Question 5: What is the role of Flask-SQLAlchemy?

Answer : Flask doesn't have a built-in way to handle databases, so it relies on SQLAlchemy, a powerful library that makes working with databases easier. SQLAlchemy provides an Object Relational Mapper(OPM), allowing developers to interact with databases using Python code instead of raw SQL.
This brings several advantages:

- Simplifies database management.
- Improves security.
- Supports multiple database systems like SQLite, MySQL and PostgreSQL.
- Easily integrates with Flask using the Flask - SQLAlchemy extension.

Question 6: How do you create a basic Flask application?

Answer: Topics :
Installing Prerequisites (for both Windows and macOS users)
a) Installing Visual Studio Code (VS Code) – Recommended IDE
b) Installing Python
1.

Setting Up the Project Directory
a) Creating and activating a virtual environment
b) Installing the Flask framework using pip
2.

Understanding Flask Project Structure
a) Exploring typical folders and files in a Flask app
3.

Writing Your First Flask Application
a) Creating a simple Python function
b) Exposing it as a route in the Flask application

Question 7:How do you return JSON responses in Flask?

Answer: When building a REST API with Flask, JSON is the standard format used to send data between the server and client. It's simple, readable and works well across platforms. Flask makes it easy to return JSON responses, which is useful for sending data like user info, product details or status messages to web or mobile apps.

For example, if the API wants to return user data, the JSON response might look like this:

```
{
    "name": "Lily",
    "age": 25
}
```

Before building a Flask REST API, make sure Flask library is already installed. Also to install the flask-restful library, use below command:

```
pip install Flask-RESTful
```

Question 8:How do you handle POST requests in Flask?

Answer: The POST method is used to send data to the server for processing. Unlike GET, it does not append data to the URL. Instead, it sends data in the request body, making it a better choice for sensitive or large data.

Example of POST in Flask

We will modify our previous Flask app to use POST instead of GET. The app will still take a number, calculate its square, and display the result.

Our app will have the same three files; the only changes are made in the app.py and squarenum.html file.

Question 9: How do you handle errors in Flask (e.g., 404)?

Answer: A custom error page improves user experience by providing a clean layout, navigation options, or auto-redirecting to the homepage. Flask lets us handle errors and display a custom page easily.

Flask allows defining routes and functions in a Python file so create an **app.py** file for our main flask app. We set up the main page route ('/') and a 404 error handler using Flask's built-in function.

Question 10: How do you structure a Flask app using Blueprints?

Answer: Flask Blueprints provide a mechanism to organize Flask applications into modular, reusable components. They allow for the structuring of an application into distinct logical sections, such as authentication, user profiles, or a blog, each with its own routes, templates, static files, and other resources.

Steps to Use Blueprints:

1. Create a blueprint in a separate module.
2. Define routes inside that blueprint.
3. Register the blueprint in the main application.

A typical Flask project using blueprints looks like this:

```
/flask_app
│── /app
│   │── /routes
│   │   │── __init__.py
│   │   │── user_routes.py
│   │── __init__.py
│── run.py
```

- **user_routes.py** - Contains routes related to users.
- **__init__.py** (inside /routes) - Allows importing different route files.
- **__init__.py** (inside /app) - Initializes the Flask app.
- **app.py** - The entry point of the application.

This app will have:

- A user profile route handled by a separate User Blueprint.
- A template (index.html) to display a dynamic welcome message.
- A modular project structure that keeps the code clean and reusable.