Restful API & Flask

Q1. What is a RESTful API ?

ANS.

- REST API stands for REpresentational State Transfer API.
- It is a type of API (Application Programming Interface) that allows communication between different systems over the internet.
- REST APIs work by sending requests and receiving responses, typically in JSON format, between the client and server.

Q2.Explain the concept of API specification ?

ANS.

- An API specification is a formal, structured document that defines how software components should interact through an Application Programming Interface (API).
- It outlines the expected behavior, structure, and rules for using an API, enabling consistent communication between different software systems.

Key Components of an API Specification:

1.Endpoints: The URL paths where the API can be accessed (e.g., /users, /products/{id})

2.Methods: The HTTP methods supported (e.g., GET, POST, PUT, DELETE).

3.Response Format:

Status codes (e.g., 200 OK, 404 Not Found)

- Response body schema (data structure returned by the API)
- Authentication and Authorization: How clients must authenticate (e.g., API keys, OAuth tokens).
- Error Handling: Description of error formats and codes.

- Data Models/Schemas: Definitions of objects used in the API, typically using formats like JSON Schema.

Why API Specifications Matter:

- Consistency: Ensures all developers use the API the same way.
- Automation: Tools can generate code, documentation, or tests from specs (e.g., Swagger/OpenAPI).
- Interoperability: Facilitates integration between systems, especially in microservices.
- Documentation: Serves as clear reference material for developers and stakeholders.

Q3.What is Flask, and why is it popular for building API ?

ANS.

- Flask is a lightweight, open-source web framework written in Python.
- It is designed to make getting started quick and easy, with the ability to scale up to complex applications.
- Flask is especially popular for building APIs (Application Programming Interfaces) due to its simplicity, flexibility, and extensive community support
-

Why Flask Is Popular for Building APIs:

1. Minimal and Lightweight

- Flask follows the "micro-framework" philosophy.
- It provides the core features of a web framework but lets developers choose and integrate additional tools as needed.
- This makes it ideal for building small to medium-sized APIs quickly.

2.Easy to Learn and Use

- With a simple and intuitive syntax, Flask is beginner-friendly.
- Developers can create an API with just a few lines of code.
- tuilt-in Development Server and Debugger
- Includes a development server and interactive debugger, which makes local development and testing easier.

Q4.What is routing in Flask?

ANS.

- App Routing means mapping the URLs to a specific function that will handle the logic for that URL.
- Modern web frameworks use more meaningful URLs to help users remember the URLs and make navigation simpler.
-

Q5.How do you create a simple Flask application.

ANS.

Creating a simple Flask application is straightforward. Below is a step-by-step guide to help you set up a basic Flask app.

CREATE FLASK.

```
from flask import Flask

app = Flask(__name__)  # Create an instance of the Flask application

@app.route('/')  # Define a route for the root URL
def home():
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run(debug=True)  # Run the app in debug mode
```

Q6. What are HTTP methods used in RESTful API.

ANS

- **HTTP methods such as GET, POST, PUT, PATCH, and DELETE are used in RESTful API development to specify the type of action being performed on a resource.**
- **RESTful HTTP methods are an essential component of developing web APIs in the REST architectural.**

Q7.What is the purpose of the @app.route() decorator in Flask

- **ANS.**
- **In Flask, the @app.route() decorator is used to define routes—that is,**
- **to map a URL path to a function that should run when that URL is accessed. It's a key part of how Flask handles web requests.**
- **EXAMPLE:**

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return 'Welcome to the homepage!'
```

Q8.What is the difference between GET and POST HTTP methods

- **ANS.**
- **The GET and POST HTTP methods are two of the most commonly used methods in web development,**
- **especially in form handling and API requests. Here's a clear breakdown of their key differences:**

Purpose:

- GET: Used to retrieve data from the server.
- POST: Used to send data to the server, typically for creating or updating resources.

Q9.How do you handle errors in Flask APIs

ANS.

In Flask, you can handle errors in your APIs using several built-in and customizable methods. Here's how it's commonly done.

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.errorhandler(404)
def not_found(error):
    return jsonify({'error': 'Not Found'}), 404

@app.errorhandler(500)
def internal_error(error):
    return jsonify({'error': 'Internal Server Error'}), 500
```

Q10.How do you connect Flask to a SQL database

- ANS.
- To connect Flask to a SQL database, you typically use an ORM (Object-Relational Mapper) like SQLAlchemy or work directly with SQLite,
- PostgreSQL, MySQL, etc., using their respective drivers. Below is a step-by-step guide using Flask-SQLAlchemy, which is the most common method.

Q11. What Is Flask-SQLAlchemy.

- ANS.
  - Flask-SQLAlchemy is an extension for Flask that adds support for SQLAlchemy, a powerful Object-Relational Mapper (ORM) for Python.
  - It simplifies the process of integrating databases into Flask applications by providing a high-level, Pythonic interface to interact with SQL databases.
  -

Q12. What is a Blueprint?

ANS

- A Blueprint is like a "mini app" that defines part of your application. You register these blueprints in your main app.
- Think of it as a way to group related routes and logic into reusable components.

WHY BLUEPRINT ARE USEFULL

- Modularization: Split your app into separate modules (e.g., auth, blog, admin)
- Reusability: Reuse components across different apps.
- Clean Routing: Keep route definitions organized and easy to manage.

Q13. What is the purpose of Flask's request object

ANS.

- The purpose of Flask's request object is to give you access to incoming request data
- When a client (like a browser or API consumer) makes an HTTP request to your Flask application.

- It’s part of Flask’s flask module and is essential for building dynamic, data-driven applications

Q14. How do you create a RESTful API endpoint using Flask.

ANS.

- Creating a RESTful API endpoint with Flask is quite simple and straightforward.
- Flask allows you to define routes that handle different HTTP methods like GET,
- POST, PUT, and DELETE, which are the core of RESTful

Q15.What is the purpose of Flask's jsonify() function,

ANS.

- The jsonify() function in Flask is used to convert Python data structures (like dictionaries, lists, etc.) into JSON format and return it as a HTTP response.
- It's a quick and easy way to send JSON data from your Flask API to the client,
- ensuring that the response is correctly formatted as JSON and has the right Content-Type header (application/json).

Q16. Explain Flask’s url_for() function

ANS.

- The url_for() function in Flask is used to generate the URL for a specific endpoint in your application dynamically.
- It allows you to reference routes in your Flask app by their endpoint name (which is usually the name of the view function),

- Rather than hardcoding the URL paths. This helps to make your application more flexible and maintainable, as the URLs won't break if you change the route definitions.

Q17.How does Flask handle static files (CSS, JavaScript, etc.)

ANS.

- Flask handles static files like CSS, JavaScript, images, and other assets through a static folder. When you create a Flask app,
- It automatically provides a route for serving static files without any extra configuration,
- making it easy to link to your CSS, JavaScript, or image files.

Q18.What is an API specification, and how does it help in building a Flask API

ANS.

- An API specification is a document or contract that defines how an API should behave.
- It specifies the structure, endpoints, methods, parameters, responses, and error handling of the API.
- An API specification serves as the blueprint for both developers and consumers, ensuring that everyone understands how the API should work.

Q19.What are HTTP status codes, and why are they important in a Flask API

- ANS.
- What are HTTP status codes, and why are they important in a Flask API
- ANS.

- HTTP status codes are standard responses provided by the server to indicate the outcome of an HTTP request.

They are part of the HTTP protocol and help to communicate the result of a request to the client (browser, API consumer, etc.).
These codes are sent in the HTTP response header along with other data, such as the response body or metadata.

HTTP status codes are grouped into five categories based on their meaning:

- 1xx - Informational: The request was received and understood, and the process is continuing.
- 2xx - Success: The request was successfully processed.
- 3xx - Redirection: Further action is needed to fulfill the request (e.g., redirecting the client).
- 4xx - Client Error: The request contains incorrect syntax or cannot be fulfilled due to client-side issues.
- 5xx - Server Error: The server failed to fulfill a valid request.

Q20. How do you handle POST requests in Flask

ANS.

- In Flask, you handle POST requests by defining a route that explicitly allows the POST HTTP method and then accessing the incoming data using.
- Flask's request object.

Q21. How would you secure a Flask API

ANS.

- Securing a Flask API is crucial to protect it from unauthorized access, data breaches, and other attacks.
- Here's a breakdown of how to secure a Flask API effectively, covering both basic and advanced techniques.

Q22.What is the Significance of the Flask-RESTful Extension,

ANS.

- Flask-RESTful is an extension for Flask that helps you build RESTful APIs quickly and efficiently.
- It provides tools and abstractions that streamline routing, request handling, input validation, and response formatting — all essential parts of an API.

Q23.What is the Role of Flask's session Object?

ANS

- In Flask, the session object is used to store data across multiple requests from the same user — essentially enabling user session management.
- It acts like a temporary, secure dictionary that allows you to save information between requests, such as:

Authentication status
User preferences
Temporary data (e.g. shopping cart contents)