FLASK Assignment

1. What is a Web API?
A Web API allows communication between different software systems over the web using HTTP.

2. How does a Web API differ from a web service?
A Web API is an interface for accessing web-based services, while a web service is a more general term for services on the web.

3. What are the benefits of using Web APIs in software development?
Web APIs enable communication between systems, promote reusability, and allow integration with third-party services.

4. Explain the difference between SOAP and RESTful APIs.
SOAP is a protocol with strict standards, while RESTful APIs are lightweight, using HTTP methods and standard HTTP protocols.

5. What is JSON and how is it commonly used in Web APIs?
JSON (JavaScript Object Notation) is a lightweight data format commonly used for exchanging data in Web APIs.

6. Can you name some popular Web API protocols other than REST?
SOAP, GraphQL, and gRPC are popular Web API protocols.

7. What role do HTTP methods (GET, POST, PUT, DELETE, etc.) play in Web API development?
HTTP methods define the operations to be performed on resources (e.g., GET retrieves, POST creates, PUT updates, DELETE removes).

8. What is the purpose of authentication and authorization in Web APIs?
Authentication verifies user identity, while authorization determines what actions a user can perform.

9. How can you handle versioning in Web API development?
Versioning can be handled through URL path versioning, query parameters, or custom headers.

10. What are the main components of an HTTP request and response in the context of Web APIs?
Request components: method, URL, headers, body. Response components: status code, headers, body.

11. Describe the concept of rate limiting in the context of Web APIs.
Rate limiting restricts the number of API requests a client can make in a given time frame to prevent abuse.

12. How can you handle errors and exceptions in Web API responses?
Errors are handled by returning proper HTTP status codes and meaningful error messages in the response body.

13. Explain the concept of statelessness in RESTful Web APIs.
Statelessness means each API request is independent, with no client context stored between requests.

14. What are the best practices for designing and documenting Web APIs?
Best practices include clear endpoint design, consistent naming conventions, and comprehensive documentation.

15. What role do API keys and tokens play in securing Web APIs?
API keys and tokens authenticate requests and ensure only authorized users access the API.

16. What is REST, and what are its key principles?
REST is an architectural style that uses HTTP and stateless communication with standard HTTP methods.

17. Explain the difference between RESTful APIs and traditional web services.
RESTful APIs use HTTP and are stateless, while traditional web services (e.g., SOAP) have more complex protocols and are stateful.

18. What are the main HTTP methods used in RESTful architecture, and what are their purposes?
GET (retrieve), POST (create), PUT (update), DELETE (remove), PATCH (partially update).

19. Describe the concept of statelessness in RESTful APIs.
Statelessness in REST means that each request is independent, and the server does not store session information.

20. What is the significance of URIs (Uniform Resource Identifiers) in RESTful API design?
URIs uniquely identify resources, allowing clients to interact with specific data or services.

21. Explain the role of hypermedia in RESTful APIs. How does it relate to HATEOAS?
Hypermedia provides information about available actions in REST APIs, with HATEOAS (Hypermedia As The Engine of Application State) guiding client interactions.

22. What are the benefits of using RESTful APIs over other architectural styles?
REST is lightweight, easy to use, and scalable with simple stateless communication.

23. Discuss the concept of resource representations in RESTful APIs.
Resources in REST are represented in formats like JSON or XML and provide data to clients.

24. How does REST handle communication between clients and servers?
REST uses HTTP methods to communicate between clients (requesting resources) and servers (providing resources).

25. What are the common data formats used in RESTful API communication?
Common data formats are JSON, XML, and HTML.

26. Explain the importance of status codes in RESTful API responses.
Status codes indicate the result of an API request, such as success (200) or error (404).

27. Describe the process of versioning in RESTful API development.
API versioning is managed through the URL or headers, allowing clients to use different API versions.

28. How can you ensure security in RESTful API development? What are common authentication methods?
Security is ensured using HTTPS, OAuth, API keys, and JWT (JSON Web Tokens).

29. What are some best practices for documenting RESTful APIs?
Best practices include clear and concise endpoint descriptions, usage examples, and detailed response explanations.

30. What considerations should be made for error handling in RESTful APIs?
Ensure meaningful status codes, detailed error messages, and proper documentation for client-side handling.

31. What is SOAP, and how does it differ from REST?
SOAP is a protocol with strict standards, while REST is an architectural style that uses HTTP.

32. Describe the structure of a SOAP message.
A SOAP message consists of an envelope, header, and body, used to encapsulate the message content.

33. How does SOAP handle communication between clients and servers?
SOAP uses XML messages for communication between clients and servers, often over HTTP or SMTP.

34. What are the advantages and disadvantages of using SOAP-based web services?
Advantages: strict standards, built-in security. Disadvantages: complexity, slower performance.

35. How does SOAP ensure security in web service communication?
SOAP uses WS-Security for message integrity, authentication, and encryption.

36. What is Flask, and what makes it different from other web frameworks?
Flask is a lightweight, flexible Python web framework known for its simplicity and ease of use.

37. Describe the basic structure of a Flask application.
A Flask app consists of routes, views, templates, and a main application instance.

38. How do you install Flask on your local machine?
Flask can be installed using the command pip install Flask.

39. Explain the concept of routing in Flask.
Routing in Flask maps URLs to specific Python functions (views) that handle the requests.

40. What are Flask templates, and how are they used in web development?
Flask templates are HTML files with embedded Jinja2 code, allowing dynamic content rendering in web pages.