1. What is JWT ?
   - JSON Web Token(JWT) is an open standard for securely transmitting information as a JSON Object. It's digitally signed using HMAC or RSA/ECDSA, ensuring the data's integrity and authenticity.
   - JWT token will be used in the request Authorization Bearer.
2. Why we use JWT ?
   - Authorization
     - Post-login, JWT's allow users to access permitted routes, services, and resources without re-authenticating.
   - Information Exchange:
     - Securely transmit information, ensuring the sender's identity and data integrity.
3. What is JWT structure ?
   - Header:
     - It specifies the token type and signing algorithm. {"alg":"HS256", "typ":"JWT" }
   - Payload:
     - It contains claims (user data and metadata). {"sub":"1234567890", "name":"john", "admin":true}
   - Signature:
     - Verifies the token's integrity.
     - HMACSHA256(
       base64UrlEncode(header) + "." +
       base64UrlEncode(payload), secret)
4. What is HTTP?
   - HTTP stands for Hypertext Transfer Protocol and Http is stateless protocol, which means that each command runs independent of any other command.
   - It is foundation of world wide web(www) and is used to load the web pages using hypertext links.
   - HTTP is application layer protocol designed to transfer information between network devices and runs on the top of other layer of the network stock.
   - A typical HTTP request contains:
     - ➔ Http version type.
     - ➔ URL
     - ➔ Http method.
     - ➔ Http request headers.
     - ➔ Optional Http body.

5. What is HTTP request headers?
   - Http headers contain text information stored in key-value pairs, they are included in every HTTP request (and response, more on the later). These headers communicate core information, such as what browser the client is using and what data is being requested.

6. What is an HTTP Response ?
   - Http response is what web clients (means browsers) receive from an internet server in answer to an Http request. These responses communicate information based on what was asked for in the Http request.
   - A typical Http response contains:
     - ➔ An Http status code.
     - ➔ Http response headers.
     - ➔ Optional Http body.

7. What is an HTTP status code ?
   - Http code 3-digit code which is use to indicate whether the Http request has been successfully completed. Status code is divided into 5 groups, they are
     i) 1xx *Informational*.
     ii) 2xx *Success*.
     iii) 3xx *Redirection*.
     iv) 4xx *Client Error.*
     v) 5xx *Server Error.*

8. What is SMTP ?
   - SMTP (Simple Mail Transfer Protocol) is a common language used to send email.
   - It is a universal set of rules that allow servers and email clients to communicate via internet.
   - Usually, SMTP used port **25.** Today, SMTP instead use port **587** and **465.**

9. What the difference between HTTP and SMTP ?
   - SMTP and HTTP are both network layer protocols that are used to transfer information between hosts.
   - SMTP is used to transfer emails between mail servers, while HTTP is used to transfer data from a web server to a web client.

10. What is API ?
    - API stands for Application Programming Interface
    - APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.
    - In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

11. How do APIs work ?
    - API architecture is usually explained in terms of client and server.
    - The application sending the request is called the client, and the application sending the response is called the server.

12. What are the different ways APIs ?
    - There are four different ways of APIs, they are
      - SOAP APIs
      - RPC APIs
      - WebSocket APIs
      - REST APIs
      - GraphQl APIs

13. What is SOAP API ?
    - These SOAP APIs are known as **Simple Object Access Protocol.** Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.
    - A SOAP API is developed in a more structured and formalized way.

14. How SOAP api is more structural?
    - It uses XML for message formatting and relies on other application layer protocols, such as HTTP or SMTP, for message negotiation and transmission.
    - SOAP is highly standardized which means it has formal specification and defined rules for request and response handling.

15. **What is RPC API ?**
    - These RPC APIs are called **Remote Procedure Calls**. The client completes a function (or procedure) on the server, and the server sends the output back to the client.
    - Remote Procedure Call (RPC) protocol is generally used to communicate between processes on different workstations. However, RPC works just as well for communication between different processes on the same workstation.
    - A Remote Procedure Call (RPC) is a software communication protocol that one program uses to request a service from another program located on a different computer and network, without having to understand the network's details.
    - RPC APIs can be statelessness or Stateful

16. **What is WebSocket API ?**
    - WebSocket API is another modern web API development that uses JSON objects to pass data. A WebSocket API supports two-way communication between client apps and the server. The server can send callback messages to connected clients, making it more efficient than REST API.
    - It has benefits like bi-directional data flow, low-latency communication, smaller data payloads, reduced network overhead, and easier server scalability.
    - WebSocket APIs are stateful Apis.

17. **What is REST APIs ?**
    - REST stands for **Representational State Transfer**. REST defines a set of functions like GET, PUT, DELETE, etc. that clients can use to access server data. Clients and servers exchange data using HTTP.
    - The main feature of REST APIs is statelessness. Statelessness means that servers do not save client data between requests. Client request to the server are similar to URls you type in your browser to visit a website. The response from the server are plain data, without typical graphical rendering of a web page.

18. What is RESTful API ?
    - RESTful API is an interface that two-computer system use to exchange information securely over the internet.
    - RESTful APIs support this information exchange because they follow secure, reliable and efficient software communication standards.
    - RESTful web service requires authentication before responding to requests. Authentication verifies the identity of the client, similar to how you prove your identity with an ID card or driver’s license. RESTful service clients must prove their identity to the server in order to establish trust and gain access to the requested resources.
    - RESTful API has four common authentication methods:

        - HTTP authentication.
            - Basic authentication.
            - Bearer authentication.
        - API keys
        - OAuth

19. What is Basic authentication ?
    - In basic authentication, the client sends the username and password in the request header. It encodes them with base64, which is an encoding technique that converts the pair into set of 64 characters for safe transmission.

20. What is Bearer authentication ?
    - The term Bearer authentication refers to the process of giving access control to the token bearer. The bearer token is typically an encrypted string of characters that the sever generates in response to a login request. The client sends the token in the request headers to access resources.

21. What is API keys ?
    - API keys are another option for REST API authentication. In this approach, the sever assigns a unique generated value to a first-time client.
    - Whenever the client tries to access resources, it uses the unique Api key to verify itself. API keys are less secure because the client has to transmit the key, which makes it vulnerable to network theft.

22. What is OAuth ?
    - OAuth combines passwords and tokens for highly secure login access to any system. The server first requests a password and then asks for an additional token to complete the authentication process. It can check the token at any time and also over time with a specific scope and longevity.

23. **What is GraphQL ?**
    - GraphQL is a query and manipulation language for APIs. GraphQL provides a flexible syntax for developers to specify their data requirements and get back only the needed data in a predictable format. This reduces the unnecessary network calls and bandwidth usage, improving application performance and efficiency.
    - GraphQL also simplifies updating data through “mutations” that describe how the data should change. It supports real-time data updates through “subscriptions”, enabling applications like chat or social media. Overall, GraphQL’s feature make it easier for developers to manage application data.
    - **Syntax:**
      Queries :
```
{
 hero {
  name
  appearsIn
  }
}
```

Response:

```
{
 "data": {
  "hero": {
   "name": "R2-D2",
   "appearsIn": [
    "NEWHOPE",
    "EMPIRE",
    "JEDI"
   ]
  }
 }
}
```

24. What is microservices ?
    - Microservices is an architectural approach where software is built as small, independent services that communicate through well-defined APIs. These services are developed and maintained by small, dedicated teams. A microservices architecture makes applications easier to scale and enables faster development and delivery of new features compared to monolithic architectures.
    - Compare to monolithic application, microservices have less disadvantages in that one disadvantages are communication between services.
    - They communicated to Restful APIs like **http.**
    - The communication issues of microservice can be overcome by **API Gateway, Circuit Breakers.**
    - By using API Gateway to handle communication between external clients and microservices, centralizing routing, security, and rate limiting.
    - By implementing circuit breakers (e.g., Hystrix) to prevent cascading failures when a service is down.
25. What are the different types of API ?
    - APIs can be classified into two criteria: their **architecture** and their **scope** of use. The next mentions that the main type of API architecture have already been covered, and it will now discuss the different scopes of use for APIs.
    - There are commonly four types of API, they are

      i) Private APIs
      - ⇨ These are internal to an enterprise and only used for connecting systems and data within the business.

      ii) Public APIs
      - ⇨ These are open to the public and may be used by anyone. There may or not be some authorization and cost associated with these types of APIs.

      iii) Partner APIs

- ⇨ APIs that are only accessible to authorized external developers to facilities business-to-business partnerships.
- ⇨ They allow companies to securely share data and services with their partners and enable integration between their systems.

iv) Composite APIs

- ⇨ Composite APIs combines two or more different APIs to address complex system requirements or behaviours.
- ⇨ They allow integrating multiple APIs to provide more comprehensive functionality by combining the capabilities of individual APIs.
- ⇨ Composite APIs are useful when a single API cannot fulfil all the requirements, and combing multiple APIs is necessary to meet complex system needs.

---

26. What is message queue ?
    - A message queue is a form of asynchronous service-to-service communication used in serverless and microservices architectures. Messages are stored on the queue until they are processed and deleted. Each message is processed only once, by a single consumer. Message queues can be used to decouple heavyweight processing, to buffer or batch work, and to smooth spiky workloads.

27. What is event broker ?
    - An event broker is message-oriented middleware that enables the transmission of events between different components of a system, acting as a mediator **between publishers and subscribers**. It is the cornerstone of **event-driven architecture**, and all event-driven applications use some form of event broker to send and receive information.
    - There are three basic types of event brokers: **queue-oriented**, **log-oriented**, **subscription-oriented.**

28. What is message broker ?
    - A message broker is software that enables applications, systems and services to communicate with each other and exchange information.
    - The message broker does this by translating messages between formal messaging protocols. This allows interdependent services to "talk" with one another directly, even if they were written in different languages or implemented on different platforms.

29. What is the difference between message broker vs message queue ?
    - Once a message is received, a message broker typically maintains the message in a message queue until a consumer/subscriber can retrieve it. Message queues are data structures that provide reliable temporary storage for messages until they are read.
    - They store messages in the same order in which they were sent, so they can be processed in sequence. Producers and consumers can share the same queue, and messages can be routed to one or many message queues

- Message queues help ensure information doesn't get lost or destroyed due to incompatible systems, network latency, or application failure. Because the message queue is such an important component of a message broker,

30. What is the difference between message broker and event broker ?
    - Event brokers and message brokers both establish loosely coupled relationships between applications using message exchange patterns like publish/subscribe and request/reply, or combine multiple patterns, depending on the specific requirements of the interaction.
    - They both support various qualities of services such as non-persistent and persistent delivery through the **use of queues**, but generally speaking **message brokers are better at point-to-point** and **queue-oriented** data exchange, while **event brokers are better at real-time publish/subscribe** interactions.

31. **What are the problems will we face if we didn't use any of the message broker or event brokers ?**
    - Tight Coupling Between Services
    - Synchronous Communication Bottlenecks
    - Single Point of Failure
    - Lack of Scalability
    - Inability to Handle High Latency or Delays
    - No Easy Retry or Message Durability
    - Difficulty in Implementing Event-Driven Architectures
    - Limited Support for Load Balancing and Traffic Distribution
    - Failure to Manage Burst Traffic Efficiently
    - Harder Debugging and Monitoring
    - State Management Issues
    - Difficulty Implementing Complex Workflows or Orchestrations.

32. What is event driven architecture ?
    - Event-driven architecture (EDA) is a design pattern where components or services in a system communicate with each other by sending and receiving events. An event is any significant change or action in the system that signals something has happened, like a user clicking a button, a system updating data, or an external service sending information.
    - In EDA, there are typically three main parts:
        a. **Event Producers**: These are the components that generate events when something happens (e.g., a user submits a form).
        b. **Event Channels**: These are the pathways (like message queues) that carry events from the producer to the components that need to react.
        c. **Event Consumers**: These are the components that listen for and act upon the events (e.g., a service processing a payment after receiving an "order placed" event).

33.