**----------------General Questions on Backend----------------**

1. ***What is Authentication?***

* Authentication is the process of verifying WHO a user is.

1. ***What is Authorization?***

* Authorization is the process of verifying WHAT someone has access to.

1. ***How do you do role-based authentication?***
2. ***What is hashing?***

* It is a process to convert information to a shorter fixed value known as the key.

1. ***What is encryption?***

* It is the process to encode data securely such that only the authorized user who knows the key or password is able to retrieve the original data.

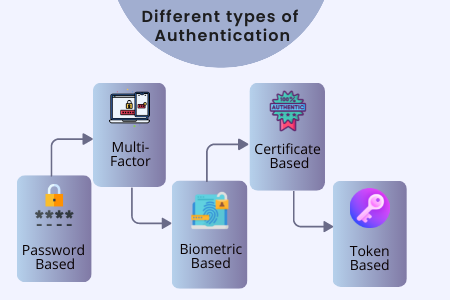
1. ***How is hashing and encryption different?***

|  |  |
| --- | --- |
| **Hashing** | **Encryption** |
| The hash code or key cannot be reversed to the original information by any means. | The original information can be easily retrieved if we know the encryption key and algorithm used for encryption. |
| It is more secure in comparison to encryption. | It is less secure in comparison to hashing. |
| The hashed information is generally of small and fixed length. | The encrypted information is not of fixed length. |

1. ***What is JWT?***

* JSON Web Token. It is an open standard used to share security information between two parties — a client and a server. JWTs are signed using a cryptographic algorithm to ensure that the claims cannot be altered after the token is issued.

1. ***How is JWT different and list the pros and cons of using JWT tokens?***
2. ***What are the different ways to manage authentication?***

******

1. ***What is cookie-based auth?***

* A cookie is a small piece of data created by a server and sent to your browser when you visit a website. Browsers often need to store and send it back to the server to tell that the request is coming from the same browser, to keep the user authenticated. We read the browser cookies as “key-value” pairs.
* Flow--->
  1. The user provides a username and password in the login form and the client/browser sends a login request.
  2. After the request is made, the server validates the user on the backend by querying the database. If the request is valid, it will create a session by using the user information fetched from the database and store them. For each session a unique ID called the session ID is created. By default, the session ID will be given to the client through the Browser.
  3. The Browser will submit this session ID on each subsequent request. The session ID is verified against the database. Based on this session ID, the server will identify the session belonging to which client and then give the request access.
  4. Once a user logs out of the app, the session is destroyed both client-side and server-side.

1. ***What is session management?***

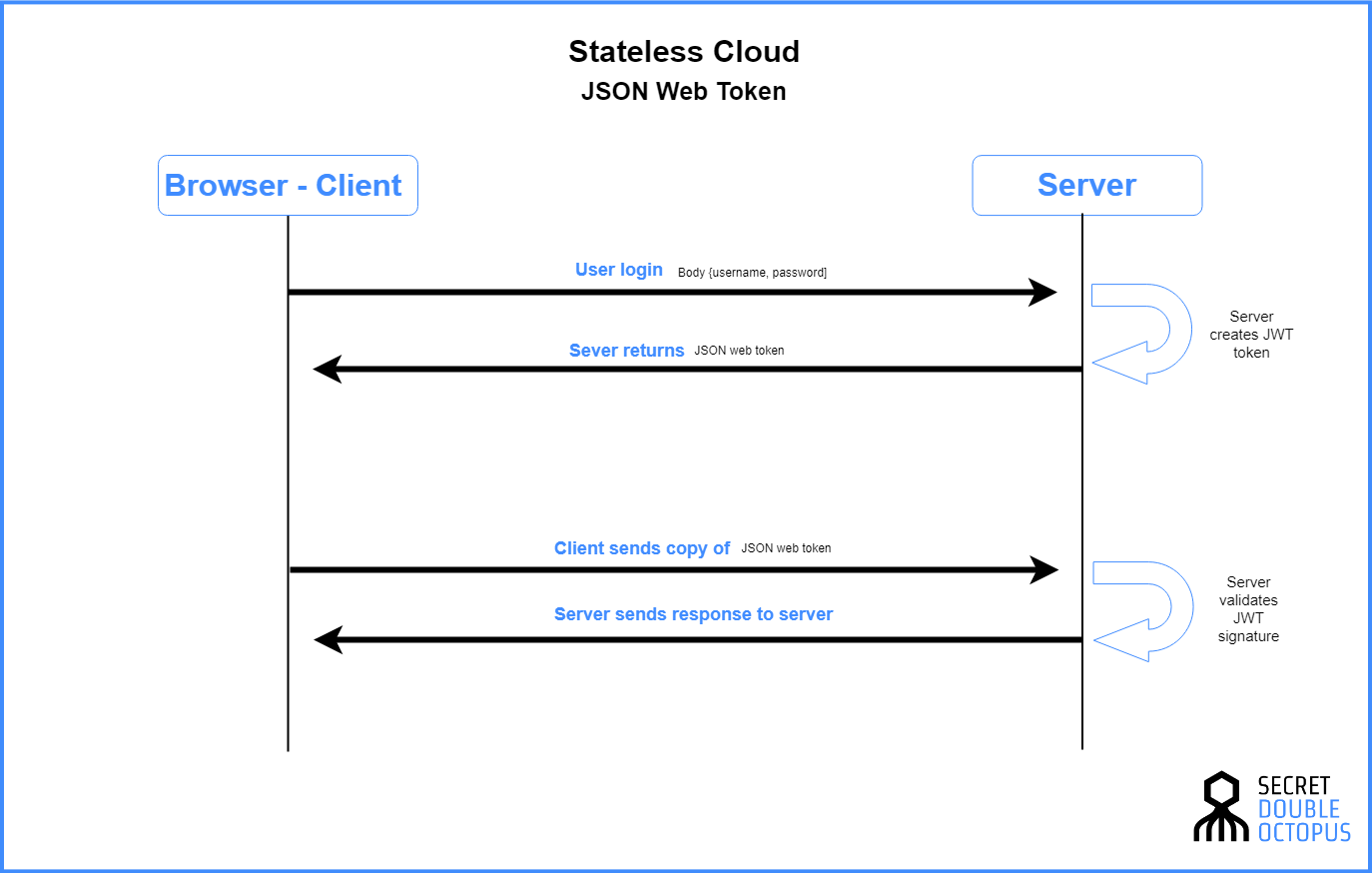
* Session management refers to the process of securely handling multiple requests to a web-based application or service from a single user or entity. Websites and browsers use HTTP to communicate, and a session is a series of HTTP requests and transactions initiated by the same user.

1. ***What is REST api?***

* **R**epresentational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. **REST API** is a way of accessing web services in a simple and flexible way without having any processing.
* REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It’s used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.
* In HTTP there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE.

1. ***What is a stateless backend?***

* The stateless server has no state with regard to the user’s information. It means when the user access any web resource, the server does not keep a track of the user’s identity or actions performed on the page. So every time, the user has to prove the identity to gain access.



1. ***What is GraphQL?***

* GraphQL is a query language and server-side runtime for APIs that prioritizes giving clients exactly the data they request and no more. GraphQL is designed to make APIs fast, flexible, and developer-friendly.

|  |  |
| --- | --- |
| **GraphQL** | **REST** |
| GraphQL is an application layer server-side technology which is developed by Facebook for executing queries with existing data. | REST is a software architectural style that defines a set of constraints for creating Web services. |
| It follows client-driven architecture. | It follows server-driven architecture. |
| GraphQL can be organized in terms of a schema. | REST can be arranged in terms of endpoints. |
| GraphQL is a growing community. | REST is a large community. |
| The development speed in GraphQL is fast. | The development speed in REST is Slow. |
| The learning curve in GraphQL is difficult. | The learning curve in REST is moderate. |
| The identity is separated from how you fetch it. | The endpoint you call in REST is the identity of an object. |
| In GraphQL, the server determines available resources. | The shape and size of the resource are determined by the server in REST. |
| GraphQL provides high consistency across all platforms. | It is hard to get consistency across all platforms. |
| The message format for GraphQL mutations should be a string. | The message format for REST mutations can be anything. |
| It is strongly typed. | It is weakly typed. |
| GraphQL API endpoints are single. | REST API endpoints are multiple. |
| It uses metadata for query validation. | It does not have machine-readable metadata cacheable. |
| Provides consistent and high-quality UX across all operating systems. | It is difficult to get consistency across all operating systems. |
| Partners of GraphQL require API customization. | It offers flexible public API that can easily enable new applications. |

1. ***What is gRPC?***
2. ***What is the client server model?***

* Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients, therefore, initiate communication sessions with servers, which await incoming requests. Examples of computer applications that use the client-server model are email, network printing, and the World Wide Web.

1. ***What is HTTP vs HTTPS?***

|  |  |
| --- | --- |
| **HTTP** | **HTTPS** |
| Hypertext Transfer Protocol. | Hypertext Transfer Protocol Secure. |
| Transmits the data over port number 80. | Ttransmits the data over port number 443. |
| It is unsecured as the plain text is sent, which can be accessible by the hackers. | It is secure as it sends the encrypted data which hackers cannot understand. |
| It is mainly used for those websites that provide information like blog writing. | It is a secure protocol, so it is used for those websites that require to transmit the bank account details or credit card numbers. |
| It is an application layer protocol. | It is a transport layer protocol. |
| It does not use SSL. | It uses SSL that provides the encryption of the data. |
| Google does not give the preference to the HTTP websites. | Google gives preferences to the HTTPS as HTTPS websites are secure websites. |
| The page loading speed is fast. | The page loading speed is slow as compared to HTTP because of the additional feature that it supports, i.e., security. |

1. ***What is throughput?***

* Throughput is the average amount of data that actually passes through over a given period of time.
* It is different from bandwidth, as bandwidth is the maximum amount of data that can pass through the network at any given time.
* Throughput is not necessarily equivalent to bandwidth, because it is affected by latency and other factors.
* *Causes*:-Congestion, Protocol overhead, Latency, etc.
* *How to improve*:- Increasing bandwidth, Improving latency, Protocol choice, etc.

1. ***What is availability?***

* Availability is the amount of time that a system is able to respond, that is the ratio of Uptime / (Uptime + Downtime). Availability is a critical metric of performance for a service, because downtime can both harm users who rely on the systems and cause a business to lose large profits in a short amount of time.
* *Causes*:-Hardware failure, Software bugs, Complex architectures, Dependent service outages, Request overload, Deployment issues, etc.
* *How to improve*:- Failover systems, Clustering, Backups, Geographic redundancy, Automatic testing, deployment, and rollbacks, etc.

1. ***What is latency?***

* Latency is the time it takes for data to pass from one point on a network to another.
* Latency is a measurement of time, not of how much data is downloaded over time.
* *Causes*:-Physical distance, Complex calculations, Congestion, Too many nodes, etc.
* *How to improve*:-Better paths, Caching, Protocol choice, etc.

1. ***What is Caching?***

* Caching is a technique that stores copies of frequently used application data in a layer of smaller, faster memory in order to improve data retrieval times, throughput, and compute costs.

1. ***What are ways to cache on the backend?***

* Web Caching (Browser/Proxy/Gateway)
* Data Caching
* Application/Output Caching
* Distributed Caching

1. ***What is LRU cache?***

* Least Recently Used (LRU) is a cache replacement algorithm that replaces cache when the space is full. It allows us to access the values faster by removing the least recently used values.

1. ***What is rate-limiting?***

* Rate limiting is a feature used to control outgoing and incoming requests in a server. When a user makes more requests than the provided limit in the window duration, then an error message is returned informing the user that the allowed limit is exceeded.
* The rate-limiting feature makes it possible to secure the Backend API from malicious attacks. It allows us to cap the number of requests that a user can make to our APIs.

1. ***What are the different ways to do rate limits?***

* Using a third-party library as middleware
* Using a custom implementation (Redis and Moment)

1. ***What is a load balancer?***

* A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks.
* Load balancers are generally grouped into two categories: Layer 4 and Layer 7. Layer 4 load balancers act upon data found in network and transport layer protocols (IP, TCP, FTP, UDP). Layer 7 load balancers distribute requests based upon data found in application layer protocols such as HTTP.

1. ***What is the difference between SQL and NoSQL databases?***

|  |  |
| --- | --- |
| **SQL** | **NoSQL** |
| RELATIONAL DATABASE MANAGEMENT SYSTEM (RDBMS) | Non-relational or distributed database system. |
| Fixed or static or predefined schema | Dynamic schema |
| Not suited for hierarchical data storage. | Best suited for hierarchical data storage. |
| Best suited for complex queries | Not so good for complex queries |
| Vertically Scalable | Horizontally scalable |
| Follows ACID property | Follows CAP(consistency, availability, partition tolerance) |
| Examples: [MySQL](https://www.geeksforgeeks.org/mysql-common-mysql-queries/), [PostgreSQL](https://www.geeksforgeeks.org/what-is-postgresql-introduction/), Oracle, MS-SQL Server, etc | Examples: [MongoDB](https://www.geeksforgeeks.org/mongodb-tutorial/), [GraphQL](https://www.geeksforgeeks.org/graphql-query/), [HBase](https://www.geeksforgeeks.org/architecture-of-hbase/), [Neo4j](https://www.geeksforgeeks.org/neo4j-introduction/),[Cassandra,](https://www.geeksforgeeks.org/apache-cassandra-nosql-database/) etc |

1. ***What is a web socket?***

* WebSocket is bidirectional, a full-duplex protocol that is used in the same scenario of client-server communication, unlike HTTP it starts from ws:// or wss://. It is a stateful protocol, which means the connection between client and server will keep alive until it is terminated by either party (client or server). After closing the connection by either of the client and server, the connection is terminated from both ends.

|  |  |
| --- | --- |
| **WebSocket Connection** | **HTTP Connection** |
| WebSocket is a bidirectional communication protocol that can send the data from the client to the server or from the server to the client by reusing the established connection channel. The connection is kept alive until terminated by either the client or the server. | The HTTP protocol is a unidirectional protocol that works on top of TCP protocol which is a connection-oriented transport layer protocol, we can create the connection by using HTTP request methods after getting the response HTTP connection get closed. |
| Almost all the real-time applications like (trading, monitoring, notification) services use WebSocket to receive the data on a single communication channel. | Simple RESTful application uses HTTP protocol which is stateless. |
| All the frequently updated applications used WebSocket because it is faster than HTTP Connection. | When we do not want to retain a connection for a particular amount of time or reuse the connection for transmitting data; An HTTP connection is slower than WebSockets. |

1. ***Describe how you design an API?***
2. ***What is Redis? Why do we use it?***
3. ***What is a horizontal and vertical scaling?***

* Horizontal scaling means scaling by adding more machines to your pool of resources (also described as “scaling out”), whereas vertical scaling refers to scaling by adding more power (e.g. CPU, RAM) to an existing machine (also described as “scaling up”).

1. ***How do you build a system which is reliable?***