**Q.1)(development) difference between package.json and package-lock.json**

1)*package.json* - package.jason file contain metadata that is revelent to our project.it contain information about version,name,scripts,dependencies,devDependencies of project.

2)*package-lock.json* – this file contain the exact dependency tree installed in node module .this provide a way for teams working on private apps to ensure that they have the same version of dependencies and sub-dependiences

(*Note 🡺*

In your package.json file, the line "react": "^18.3.1" specifies the version of the React library that your project depends on. Let's break down this syntax:

* **"react"**: This is the name of the package, in this case, the React library.
* **"^18.3.1"**: This is the version specification using the caret (^) symbol.

The caret (^) in semantic versioning allows for updates that do not change the leftmost non-zero digit. For "^18.3.1", this means that your project is compatible with versions >=18.3.1 and <19.0.0. In other words, it will accept any minor or patch updates in the 18.x.x range but will not automatically accept major updates, such as 19.0.0. )

**Q.2)(Javascript)**

let str="jscafe"

str[0]="c"

console.log(str);

output ->jscafe

explain 🡺we can not modify string using is it’s index.insted modification we create new string

let str="jscafe"

str=”cscafe”

console.log(str);

output ->cscafe

concept 🡺*string’s are immutable in javascript because can not modify content of an existing string directly.(refer youtube interview development playlist)*

**Q.3)(javascript)**

let a=[1];

let b=[2];

console.log(a+b);

output->12

**(Note 🡺** when we use addition operator with array then javascript use ‘.toString’ method which convert element into string.hence concenation of these two string gives answer as 12. )

**Q.4)statefull and stateless authentication**

**Ans🡪**Stateful and stateless authentication are two distinct approaches to managing user sessions in web applications

1. **Stateful Authenticatio**

In stateful authentication, the server maintains session data for each user. Upon successful login, the server generates a unique session identifier (session ID) and stores it, along with the user's authentication details, in a database or in-memory store. This session ID is then sent to the client, typically stored in a cookie, and included with each subsequent request. The server uses this ID to retrieve the corresponding session data, ensuring the user's authentication status and related information are preserved throughout their interaction with the application.

**Advantages:**

* **Session Management:** The server has full control over user sessions, allowing for actions like immediate session invalidation or modification.
* **Data Consistency:** Since session data is stored on the server, it can be updated centrally, reducing the risk of clients operating with outdated information.

**Disadvantages:**

* **Scalability Challenges:** Maintaining session state on the server can consume significant resources, especially with a large user base, potentially leading to scalability issues.

2) **Stateless Authentication**

Conversely, stateless authentication does not require the server to store session data. Instead, upon user authentication, the server issues a token—commonly a JSON Web Token (JWT)—that encapsulates the user's identity and authorization details. This token is signed cryptographically to prevent tampering and is sent to the client, which includes it in the header of subsequent requests. The server validates the token's signature and extracts the necessary information to process each request without referencing any stored session data.

**Advantages:**

* **Scalability:** Since the server doesn't maintain session state, it can handle a larger number of users more efficiently, simplifying horizontal scaling.

**Disadvantages:**

* **Session Revocation Limitations:** Once issued, tokens remain valid until they expire, making immediate session revocation challenging without implementing additional mechanisms.
* **Potential for Stale Data:** If user information changes (e.g., role updates) after token issuance, the token may contain outdated data until it is refreshed or expires.

**Q.5) what is token in web development?**

In web development, a **token** is a piece of data generated by a server that represents the user's authentication or authorization status. Tokens are commonly used in token-based authentication systems to verify users' identities without requiring them to re-enter credentials for each request.

**How Token-Based Authentication Works:**

1. **User Login:** A user provides their credentials (e.g., username and password) to the server.
2. **Token Issuance:** Upon successful authentication, the server generates a token—often a JSON Web Token (JWT)—that encodes the user's identity and permissions.
3. **Client Storage:** The client (e.g., a web browser or mobile app) stores this token, typically in local storage or a cookie.
4. **Subsequent Requests:** For future requests to protected resources, the client includes the token, usually in the Authorization header.
5. **Server Validation:** The server verifies the token's validity and, if valid, processes the request accordingly.

**Q.6)what is JWT(json web token)**

When we give user name and password to server then server use user name and password to authenticate user and issue JWT token to user which contain user name ,roll and random token.after this user will store token in the browser local storage to authenticate with API.

**Compact Size:** JWTs are designed to be lightweight, typically consisting of three parts: header, payload, and signature. This compact structure ensures that they can be transmitted quickly and efficiently over networks without significant overhead.

**Stateless Authentication:** JWTs enable stateless authentication, meaning the server doesn't need to maintain session information. This reduces server overhead and simplifies scalability.

**Enhanced Performance:** JWTs contain all necessary user information within the token itself, reducing the need for frequent database lookups and thereby improving application performance.

**Security:** JWTs can be signed and encrypted, ensuring data integrity and confidentiality. The use of digital signatures allows recipients to verify the token's authenticity and detect any tampering.this make secure way to transfer information between client and server.