**JWT (JSON Web Token) – Interview Notes**

**1. What is JWT?**

* JWT (JSON Web Token) is a **compact, self-contained** way to securely transmit information between parties.
* It is commonly used for **authentication and authorization** in web applications.
* It is a **token-based** authentication method that does not require storing user sessions on the server.

2. **Authentication vs Authorization**

|  |  |  |
| --- | --- | --- |
| Feature | Authentication | Authorization |
| Definition | Process of verifying identity | Process of granting permissions |
| Purpose | Confirms **who the user is** | Confirms **what user can do** |
| Example | Logging into an application | Accessing admin panel after login |

**3. Why Use JWT? (Problems It Solves)**

**❌ Problems in Traditional Authentication:**

* **Session-based authentication** stores user sessions on the server → **Consumes memory** and **can be hijacked**.
* **Cookies-based authentication** is vulnerable to **CSRF (Cross-Site Request Forgery) attacks**.

**✅ How JWT Solves These Problems:**

* JWT **removes the need for sessions** – It is a **stateless** authentication method.
* JWT is **digitally signed**, so it cannot be **tampered with**.
* JWT **works across different platforms and devices** (Mobile, Web, APIs).

**4. How Does JWT Work?**

1️⃣ User logs in with **username & password**.  
2️⃣ Server **verifies credentials** and generates a **JWT token**.  
3️⃣ Token is sent to the client (browser, mobile app).  
4️⃣ Client **stores** the token (in local storage, cookies).  
5️⃣ On **each request**, the token is sent to the server in the **Authorization header**.  
6️⃣ Server **validates the token** → If valid, grants access; otherwise, denies it.

**5. Advantages of Using JWT**

✅ **Stateless** – No need to store sessions on the server.  
✅ **More Secure** – Prevents session hijacking and CSRF attacks.  
✅ **Fast & Scalable** – Reduces database and memory usage.  
✅ **Cross-Platform Support** – Works with Web, Mobile, and APIs.

**6. Parts of a JWT Token**

A JWT token consists of **three parts**, separated by dots (.):

**1. Header**

* Contains **token type (JWT)** and **signing algorithm (HS256, RS256, etc.)**.
* Example:

{

"alg": "HS256",

"typ": "JWT"

}

**2. Payload (Claims)**

* Contains user **data (claims)** such as **UserId, Role, Expiry time**.
* Example:

{

"UserId": 123,

"Role": "Admin",

"exp": 1712345678

}

**3. Signature**

* The **Signature** is the third part of a JWT (Header.Payload.Signature).
* It is used to **verify the authenticity and integrity** of the token.
* The signature ensures that **the token was not modified** after it was issued.

**How is the Signature Created?**

**🔢 Formula for JWT Signature:**

1️⃣ **Header + Payload** are combined and encoded using **Base64Url**.  
2️⃣ A **secret key** (known only to the server) is used for signing.  
3️⃣ The result is a **cryptographic signature** that can be verified later.

**Why is the Signature Important?**

✅ **Prevents Token Tampering:** If anyone modifies the token, the signature will become invalid.  
✅ **Ensures Authenticity:** Only the server (with the secret key) can generate and verify the token.  
✅ **Prevents Fake Tokens:** Hackers cannot create valid tokens without the secret key.

**How Does the Server Verify the Signature?**

1️⃣ When a user sends a JWT, the server **separates the Header, Payload, and Signature**.  
2️⃣ The server **recalculates the signature** using the **same secret key**.  
3️⃣ If the **calculated signature matches** the received signature → The token is valid.  
4️⃣ If they **do not match** → The token is **rejected** (possibly tampered with).

**What Happens If the Signature Is Missing?**

🚨 **If the signature is missing or invalid:**  
❌ The server **rejects the token** → User access is denied.  
❌ Attackers can **modify payload data** (e.g., changing user role to "Admin").  
✅ **Solution:** Always verify the signature before trusting the token!

**7. Token Expiry – Why It’s Important?**

❌ **Problems with Non-Expiring Tokens:**

* If stolen, they can be misused **forever**.
* Difficult to revoke access.

✅ **Benefits of Expiring Tokens:**

* **Improves security** → Tokens become invalid after a certain time.
* **Forces re-authentication** → Users must log in again after expiry.
* **Reduces token misuse**.

.