JWT Authentication – Interview Questions & Explanations

1. What problem does JWT solve in RESTful authentication?

JWT solves stateless authentication. It allows the server to verify a user without storing session data on the server, making the system scalable and suitable for distributed systems.

2. Why is JWT considered stateless?

Because all required authentication information is contained inside the token itself. The server does not need to store session data; it only verifies the token signature and expiry.

3. Why do we include the user role in the JWT payload?

Including the role allows quick authorization checks without querying the database for every request. It improves performance and preserves statelessness.

4. What fields should be included in a JWT payload?

Only minimal and stable fields: user ID (sub), role, issued-at time (iat), and expiry time (exp).

5. What information should never be included in JWT and why?

Personal data like email, username, phone number, or permissions should not be included because JWT is Base64-encoded, not encrypted. Anyone with the token can decode and read its contents.

6. Why should permissions not be stored inside JWT?

Permissions change frequently, while JWT tokens are issued for a fixed time. Embedding permissions would require token invalidation on every permission change, which is impractical.

7. What is the difference between HS256 and RS256 algorithms?

HS256 uses a shared secret key for signing and verification, while RS256 uses a public-private key pair. RS256 is more secure and suitable for microservices.

8. Why is RS256 preferred over HS256 in production systems?

RS256 separates signing and verification. The private key stays only with the auth service, while other services use the public key to verify tokens, reducing security risks.

9. Can frontend applications verify JWT tokens in RS256?

Yes. Frontend or API gateways can verify JWT signatures using the public key without exposing the private key.

10. Is JWT alone enough for security?

No. JWT provides authentication and authorization hints, but the backend must still enforce role-based access rules, validate permissions, and protect sensitive operations.

11. How are role-based rules enforced if role is already in JWT?

JWT provides the role, but the backend contains logic that defines what each role can or cannot do. This ensures centralized control and prevents privilege escalation.

12. What happens if a user role changes while JWT is still valid?

The old token will still contain the previous role until it expires. This is why JWT tokens should be short-lived and refreshed frequently.