JWT Explaining

JSON Web Token (JWT) is a secure, compact way to represent claims between two parties. It's commonly used for authentication and information exchange in stateless systems.
A JWT has three parts:

1. **Header**: Metadata (e.g., type: JWT, algorithm: HMAC SHA256).
2. **Payload**: User data and claims, such as user ID and roles.
3. **Signature**: Validates integrity using a secret key or public/private key.

How JWT Works

1. User logs in → Server verifies credentials → Issues a JWT.
2. JWT is sent with each request (e.g., in the Authorization header).
3. Server validates JWT and processes the request.

Advantages

- **Stateless**: Eliminates server-side session management.
- **Scalable**: Ideal for distributed systems or microservices.
- **Cross-domain**: Enables Single Sign-On (SSO).
- **Self-contained**: Encodes all necessary information in the token.
- **Secure**: Signature ensures data integrity; supports encryption.

Downsides

- **No Built-in Revocation**: Requires custom blacklist to revoke tokens.
- **Sensitive Data Exposure**: Data is base64-encoded, not encrypted (use encryption for confidential data).
- **Large Tokens**: Increases request size compared to session IDs.
- **Performance Impact**: Validating tokens requires cryptographic operations.
- **Expiration Management**: Expired tokens require re-authentication mechanisms.

When to Use JWT

- **Stateless Systems**: Ideal for REST APIs or microservices.
- **Single Sign-On (SSO)**: Simplifies authentication across domains.
- **Short-lived Tokens**: Use for temporary authentication to minimize risks.

When to Avoid JWT

- **Frequent Revocation Needed**: Revoking tokens without state is challenging.
- **Large Payloads**: Token size impacts performance.
- **High Confidentiality**: Avoid placing sensitive data in JWT payloads unless encrypted.