Q1. What is Middleware in Express.js?

→Middleware in Express.js is a function that has access to the request (req), response (res), and the next middleware function in the application's request-response cycle. Middleware can execute code, modify the request or response objects, and terminate the request-response cycle or pass control to the next middleware function using the next() function.

Types of Middleware in Express.js:

1. Built-in Middleware:
    - Example: express.json(), express.static().
2. Third-party Middleware:
    - Example: body-parser, cors, etc.
3. Custom Middleware:
    - Middleware functions defined by the developer for specific needs.

Q2. What is json web tokens?

→JSON Web Tokens (JWTs) are a compact, URL-safe means of representing claims to be transferred between two parties (e.g., client and server). JWTs are used for secure data transmission, commonly in authentication and authorization mechanisms.

Structure of a JWT: A JWT consists of three parts, separated by dots (.):

1. Header: Contains metadata about the token, such as the algorithm used for signing.
2. Payload: Contains the claims (data) like user ID, roles, etc.
3. Signature: Verifies the authenticity of the token.

Use Cases:

- **Authentication**: After logging in, a JWT is issued to the user and sent with subsequent requests.
- **Authorization**: JWTs can store roles/permissions for access control.

Q3. What is different between encryption and hashing

1. Definition:
    - Encryption converts data into a secure format that can be decrypted using a specific key.
    - Hashing transforms data into a fixed-length value using a hash function, which is irreversible.
2. Purpose:
    - Encryption protects sensitive data during transmission or storage by making it unreadable to unauthorized users.
    - Hashing ensures data integrity and secures password storage by comparing hash values.
3. Reversibility:
    - Encryption is reversible; encrypted data can be decrypted with the correct key.
    - Hashing is irreversible; hashed data cannot be converted back to its original form.
4. Use Cases:
    - Encryption is used for secure messaging (e.g., WhatsApp) and data storage (e.g., encrypting files).
    - Hashing is used for password security (e.g., login systems) and file verification (e.g., checksums).
5. Algorithms:
    - Common encryption algorithms include AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman).
    - Common hashing algorithms include SHA (Secure Hash Algorithm), MD5, and bcrypt.
6. Output:
    - Encryption produces a random sequence of characters, varying with each application.

- Hashing generates a fixed-length output for the same input, with added "salting" for uniqueness.

7. Example Scenario:
    - Encryption secures credit card information for online transactions.
    - Hashing stores passwords securely in login systems.