💡 **Q.1** Explain Middlewares in NodeJS?

In Node.js, middleware refers to a function or a set of functions that are executed in a specific order during the processing of an HTTP request. It sits between the incoming request and the final handler, allowing you to perform various operations on the request or response objects. Middleware functions have access to the request and response objects, as well as the next middleware function in the chain.

Middleware functions can be used for a wide range of purposes, such as logging, authentication, authorization, input validation, error handling, and more. They provide a modular and reusable way to add functionality to your Node.js applications.

When a request is made to a Node.js server, the server passes the request through a series of middleware functions before it reaches the final route handler. Each middleware function can modify the request or response objects, and it can also decide whether to pass the request to the next middleware in the chain or respond to the client immediately.

To use middleware in Node.js, you typically use a framework like Express.js, which provides a built-in middleware system. Express.js allows you to easily define and use middleware functions using the `app.use()` method. You can also create custom middleware functions using the `function(req, res, next)` signature and mount them at specific routes or globally.

💡 **Q.2** Why use Express Over NodeJS?

Express is a web application framework for Node.js that simplifies the process of building web applications and APIs. While Node.js provides the core functionality for building server-side applications, Express enhances it by providing a higher-level abstraction and additional features. Here are some reasons why you might choose to use Express over plain Node.js:

1. **Simplicity**: Express provides a simple and intuitive API for handling HTTP requests and building routes. It abstracts away many low-level details and boilerplate code, making it easier to build web applications.

2. **Middleware**: Express has a powerful middleware system that allows you to add functionality to your application in a modular and reusable way. Middleware functions can handle tasks like authentication, logging, error handling, and more.

3. **Routing**: Express provides a flexible routing system that helps in defining and organizing the application's routes. It supports parameters, pattern matching, and route grouping, making it easy to create RESTful APIs.

4. **Ecosystem**: Express has a large and vibrant ecosystem with a wide range of middleware packages and extensions available. These packages can be easily integrated into your Express application, saving development time and effort.

5. **Popularity**: Express is one of the most popular web frameworks for Node.js, which means it has a large and active community. This community provides support, resources, and numerous third-party libraries that can help with various aspects of web development.

However, it's important to note that Express is built on top of Node.js and is not a replacement for it. Express leverages the power of Node.js and provides a higher-level abstraction for web application development.

💡 **Q.3** What are REST APIs?

REST (Representational State Transfer) is an architectural style for designing networked applications. REST APIs (Application Programming Interfaces) are a set of rules and conventions that allow different software applications to communicate with each other over the internet using standard HTTP methods, such as GET, POST, PUT, and DELETE.

REST APIs are based on a few key principles:

1. **Client-Server Architecture**: The client and server are separate entities that communicate over a network. The client sends requests to the server, and the server responds with the requested data or performs the requested actions.

2. **Statelessness**: Each request from the client to the server must contain all the information needed to understand and process the request. The server doesn't store any client-specific information between requests.

3. **Uniform Interface**: REST APIs follow a uniform and standardized set of rules for communication. This includes using standard HTTP methods, using URIs (Uniform Resource Identifiers) to identify resources, and using hypermedia links to navigate between resources.

4. **Resource-Based**: REST APIs treat everything as a resource, which can be accessed and manipulated using a unique URI. Resources can be entities like users, products, articles, etc.

REST APIs are widely used in web development because they provide a scalable, stateless, and interoperable way to build distributed systems. They allow different systems and technologies to communicate and exchange data efficiently.

💡 **Q.4** What is the use of MongoDB?

MongoDB is a popular open-source NoSQL database system that provides high scalability, flexibility, and performance. It is designed to handle large amounts of unstructured or semi-structured data, making it well-suited for modern web applications.

The key uses of MongoDB include:

1. **Flexible Data Model**: MongoDB uses a flexible document model, known as BSON (Binary JSON), to store data. BSON documents can have varying structures, allowing you to store data with different fields and types in a single collection. This flexibility makes it easier to evolve the data model as your application requirements change.

2. **Scalability**: MongoDB is horizontally scalable, which means it can distribute data across multiple servers or clusters to handle high traffic loads and large datasets. It supports automatic sharding, allowing you to partition data across multiple machines for improved performance and scalability.

3. **High Performance**: MongoDB offers high write and read throughput by using memory-mapped storage and a flexible indexing system. It can efficiently handle large volumes of concurrent read and write operations.

4. **Aggregation and Querying**: MongoDB provides a powerful aggregation framework and a rich query language that allows you to perform complex queries and aggregations on your data. It supports various query types, including range queries, geospatial queries, and text search.

5. **Schema Flexibility**: Unlike traditional relational databases, MongoDB does not enforce a predefined schema. You can store documents with different structures in the same collection, which allows for more dynamic and iterative development.

MongoDB is commonly used in a wide range of applications, including content management systems, real-time analytics, e-commerce platforms, social networks, and more.

💡 **Q.5** What is Mongoose and how does it relate to MongoDB?

Mongoose is an Object-Data Modeling (ODM) library for MongoDB and Node.js. It provides a higher-level abstraction over the MongoDB driver and simplifies the interaction with MongoDB databases. Mongoose allows you to define schemas, models, and relationships between data, making it easier to work with MongoDB in a structured and organized manner.

Here's how Mongoose relates to MongoDB:

1. **Schema Definition**: Mongoose allows you to define schemas that define the structure and validation rules for documents in a MongoDB collection. Schemas provide a way to enforce consistency and structure in your data, similar to how tables and columns do in traditional relational databases.

2. **Model Creation**: With Mongoose, you can create models based on the defined schemas. Models represent collections in MongoDB and provide an interface to interact with the data. They encapsulate various operations like creating, querying, updating, and deleting documents.

3. **Data Validation**: Mongoose provides built-in validation capabilities for data stored in MongoDB. You can define validation rules for fields in a schema, such as required fields, data type checks, custom validation functions, and more. Mongoose ensures that the data adheres to the specified rules before saving it to the database.

4. **Middleware Support**: Mongoose supports middleware functions that can intercept and modify document operations, such as pre-save or post-remove hooks

. This allows you to add custom logic, perform transformations, or trigger additional actions before or after certain database operations.

Overall, Mongoose enhances the MongoDB experience by providing a higher level of abstraction, data modeling capabilities, validation mechanisms, and convenient APIs for working with MongoDB databases in Node.js applications.