***Node.js Toolkit:***

**1. What is Node.js and how does it work?**

Node.js is an open-source, cross-platform,javascript runtime environment that executes javascript code outside of a browser. Node JS was created by Ryan Dhal.

Node.js uses V8 javascript engine from google chrome to execute javascript code on the server.Node.js provides an event-driven, non-blocking I/O model,making it highlight and efficient for building scalable network applications.

**2.What is the difference between Node.js and JavaScript?**

Node.js is runtime environment for executing javascript code outside of a browser,while javascript is a programming language that can run in various environments, including web browsers.Node.js provides additional APIs and modules that are not available in the browsers javascript runtime, such as file system access and networking capabilities.

**3.How do you handle asynchronous operations in Node.js?**

Node.js uses callbacks, promises, and async/await for handling asynchronous operation.Callbacks are the traditional approach where a function accepts a callback function as a parameter, which is called when the operation is complete. Promises provides a more structured way if handling asynchronous operations, allowing and error handling.Async/await is a syntactical sugar built on top of promises,making asynchronous code look more like synchronous code,improving readability.

**4.What is the Event Loop in Node.js?**

An event loop is an endless loop, which waits for tasks, executes them, and then sleeps until it receives more tasks. The event loop executes tasks from the event queue only when the call stack is empty i.e. there is no ongoing task. The event loop allows us to use callbacks and promises.

**5.Explain the concept of middleware in Node.js.**

Middleware in Node.js refers to functions that have access to the request and response objects in an applications request -response cycle. It sits between the initial processing of a request and the final sending of the response Middleware functions can perform various tasks such as modifying the request/response objects,logging,authentication,error,handeling etc.They can be chained together to form a pipeline,where each middleware function can modify the request/response or pass it to the next middleware in the chain.

**6.How do you handle errors in Node.js?**

In Node.js errors can be handled using try-catch block for synchronous code or using error-first callbacks, promise, or try-catch with async/await for asynchronous code.Error-first callbacks follow a convention where the first parameter of a callback function is reserved for an error object.Promises allow error handling through the .catch() method, and async/await relies on try-catch blocks to catch and handle errors.

**7.What is npm and what is its purpose in Node.js?**

Npm (Node Package Manager) is the default package manager for Node.js. It is a command-line tool used to install,publish, and manage third party Node.js modules and dependencies.npm provides access to a vast ecosystem of reusable packages, making it easier to develop Node.js applications by leveraging existing code and libraries

**8.How can you read data from the request body in Node.js?**

To read data from the request body in Node.js, you need to use the req object, which represents the incoming request. Depending on the content type of the request, you can use different modules such as body-parser or the built-in querystring or fs modules to parse and extract data from the request body.

**9.Explain the concept of modules in Node.js.**

Modules in Node.js are reusable blocks of code that encapsulate related functionality. They allow you to organize and modularize your codebase. Node.js uses the CommonJS module system, where each file is treated as a separate module, and you can export functions, objects, or variables using the module.exports or exports object. Modules can be imported using the require() function and can be either built-in core modules or external modules installed via npm.

**10.What is package json?**

The package.json file in Node.js the heart of the entire application.It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

**11.Explain the purpose of ExpressJS package?**

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware module of Node.js called connect. The connect module further makes use of http module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

**12.Why is NodeJS Single threaded?**

Node.js is single-threaded for async processing.By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved as opposed to the typical thread-based implementation

***MongoDB Toolkit:***

**1.What is MongoDB and how does it differ from a traditional relational database?**

MongoDB is a popular open-source NoSQL database that stores data in flexible, JSON-like documents called BSON. Unlike traditional relational databases, MongoDB does not use tables and rows. Instead, it uses collections of documents with dynamic schemas, allowing for more flexible data models.

**2.What are the advantages of using MongoDB?**

→Scalability

→Flexibility

→Performance

→Replication

→JSON-like Documents

**3. What are the different types of NoSQL databases?**

NoSQL is a non-relational DBMS, that does not require a fixed schema, avoids joins, and is easy to scale. The purpose of using a NoSQL database is for distributed data stores with humongous data storage needs. NoSQL is used for Big data and real-time web apps.

Types of NoSQL Databases:

i.Document databases

ii.Key-value stores

iii.Column-oriented databases

iv.Graph databases

**4.What is BSON in MongoDB?**

BSON (Binary JSON) is a binary representation of JSON-like documents used in MongoDB. It adds additional data types like ObjectId, ISODate, BinData, and more, which are not available in JSON. BSON provides efficient serialization and deserialization of data, making it suitable for storing and retrieving data in MongoDB.

**5.What is a collection in MongoDB?**

A collection in MongoDB is a group of related documents. It is similar to a table in a relational database, but without a fixed schema. Collections are schema-less, meaning they can contain documents with different structures and fields. Collections provide a way to organize and store related data in MongoDB.

**6.What is the difference between a SQL query and a MongoDB query?**

SQL queries are structured using the SQL language and operate on tables and rows. MongoDB queries, on the other hand, use a JSON-like syntax to query documents in collections. MongoDB queries are more flexible and powerful, supporting complex conditions, nested queries, and various operators

**7.What is an index in MongoDB and why is it important?**

In MongoDB, an index is a data structure that improves the speed of data retrieval operations on specific fields in a collection. It allows queries to quickly locate and access the desired data, similar to an index in a book. Indexes are essential for optimizing query performance, especially for frequently queried fields or fields used in sorting or filtering operations.

**8.How do you perform data modeling in MongoDB?**

Data modeling in MongoDB involves designing the structure and relationships of the documents within collections. It requires analyzing the application's requirements, understanding the data access patterns, and denormalizing data when necessary. Data modeling in MongoDB often follows a "store what you query" approach, optimizing for read operations.

**9.What are the different types of MongoDB indexes?**

MongoDB supports various types of indexes,

→Single Field Index: Indexes a single field in a document.

→Compound Index: Indexes multiple fields together.

→Multi-Key Index: Indexes arrays and sub-documents.

→Text Index: Enables full-text search on text fields.

→Geospatial Index: Indexes geospatial data for spatial queries.

→Hashed Index: Used for sharding and random access to data.

**10.What is the significance of the "\_id" field in MongoDB?**

The "\_id" field is a unique identifier automatically assigned to every document in a collection. It uniquely identifies each document and is indexed by default. MongoDB uses this field as the primary key and ensures its uniqueness across the collection.

**11.How do you perform aggregation in MongoDB?**

Aggregation in MongoDB involves processing and transforming data, typically across multiple documents, using the MongoDB Aggregation Pipeline. The pipeline consists of stages that perform various operations like filtering, grouping, sorting, projecting, and aggregating data. It provides powerful data manipulation capabilities and allows complex data transformations.

**12.What is sharding in MongoDB and why is it used?**

Sharding is a technique used in MongoDB for horizontal scaling. It involves partitioning data across multiple machines or servers called shards. Each shard holds a subset of the data, allowing for distributed storage and parallel processing. Sharding improves performance, scalability, and availability of MongoDB deployments.

**13.What are the key features of MongoDB?**

→Document Oriented and NoSQL database.

→Supports Aggregation

→Uses BSON format

→Sharding (Helps in Horizontal Scalability)

→Supports Ad Hoc Queries

→Schema Less

→Capped Collection

→Indexing (Any field in MongoDB can be indexed)

→MongoDB Replica Set (Provides high availability)

→Supports Multiple Storage Engines.

**14.What are the difference between SQL and NoSQL?**

**SQL:**

→relational

→use structured query language and have a predefined schema.

→are vertically scalable

→are table based

→are better for multi-row transactions

**NoSQL:**

→Non-relational

→NoSQL databases have dynamic schemas for unstructured data.

→are horizontally scalable.

→are document, key-value, graph or wide-column stores.

→are better for unstructured data like documents or JSON.