MERN Stack Interview Questions and Answers

Updated onOctober 16, 2024
Today, the MERN stack is considered one of the most popular technologies in building web applications that consists of four key technologies, i.e. MongoDB, Express.js, React, and Node.js. It uses JavaScript throughout in order to provide a highly powerful and efficient way to develop full-stack applications altogether. The MERN stack offers a great experience infull stack development.
In this blog, you will discuss all the necessary MERN stack interview questions and answers. Moreover, you can see the questions which are primarily categorised into Freshers, Intermediate and Experienced levels, so that you can get all types of exposure regarding these interview questions.

What is MERN Stack Development?

Creating web applications using theMERN stackinvolves utilising the four technologies listed below.
• MongoDB:MongoDB is a type of database that focuses on managing data, unlike traditional table-based databases.
• js:Express.js is a web framework for Node.js designed to simplify the creation of scalable server-side JavaScript applications.
• React:React is a JavaScript library that anyone can access and use to create user interface components, with Facebook overseeing its maintenance.
• js:Node.js is a JavaScript platform that enables the execution of JavaScript code on the server side.
MERN stack development is in demand for a number of reasons:
• Versatility:The flexibility of this framework allows developers to create a range of applications, from easy ones to complex systems of any size.
• Capacity:The technologies within the MERN stack have the capability to handle and manage large amounts of data and traffic effectively, making them ideal for scaling up applications as they grow in size.
Also Read:MERN Stack Developer Salary

MERN Stack Interview Questions With Answers for Freshers

1. What is ReactJS?

ReactJS is a Graphical user interface library invented by Facebook. It is mainly used to develop front-end applications. It facilitates the construction of reusable components, proper state management, and rendering through its virtual DOM efficiently.

2. What is the purpose of MongoDB?

MongoDB is a NoSQL database used for large amounts of unstructured data. It contains a flexible structure with data stored in JSON documents.

3. What are props in React?

Props are inputs received from parent components that are not modifiable by child components. In other words, it is used to pass the data to the child components and it is also quite helpful in the dynamic rendering of the child components whenever the parent component changes.

4. How do you create a simple React component?

You can create a simple React component as a function or a class. Here’s an example using a function:

This component receives a name prop and renders a greeting message.

5. What is the purpose of ExpressJS?

ExpressJS is a node.js web application framework that offers a variety of features. It is recommended for use in the development of mobile and web applications. Server-side coding becomes less tedious with the help of the ExpressJS framework.

6. Explain the MVC architecture.

MVC architectureis a standard architectural way to design apps that distribute one functionality across several components. There are three main components of this:
• Model:It is the part of the application that is responsible for data and business rules.
• View:This is the part that focuses on how the user interacts with the application through its interface.
• Controller:A mediator between the Model and View that processes user input and performs changes in the Model and the View as needed.

7. What are the differences between SQL and NoSQL databases?

Feature	SQL Databases	NoSQL Databases
Data Model	Relational (tables with rows and columns)	Non-relational (document, key-value, etc.)
Schema	Fixed schema	Dynamic schema
Scalability	Vertical (adding more resources to a single server)	Horizontal (adding more servers)
Transactions	Supports ACID transactions	Varies, often supports eventual consistency
Use Cases	Complex queries, structured data	Unstructured or semi-structured data, big data

8. What do you mean by asynchronous API?

An asynchronous API can be described as one, which enables the tasks to take place disregarding the primary program flow. Rather than halting the execution of the program when necessary tasks are ongoing, it allows the program to keep running and manages the result once it becomes available by utilising callbacks, promises or async/await functions.

9. How do you connect a React app to a MongoDB database using Node.js?

When linking a React app to a Node.js backend and MongoDB database the usual method involves configuring an HTTP server using Express and Mongoose. The server supports the requests generated from the React application and also makes some calls to the MongoDB database which lies at the backend.

10. What is reconciliation?

The term reconciliation in the context of ReactJS relates to the updating of the existing DOM by scanning the current DOM, including changes made by the virtual DOM. This is commonly called the reconciliation process in React where changes are detected in the virtual DOM and only the

appropriate sections of the actual DOM where changes occurred are updated in a quick manner.
Code Example:

11. Does MongoDB support foreign key constraints?

MongoDB does not natively support foreign key constraints like traditional relational databases. However, it allows for data modelling with embedded documents or manual reference checks to maintain relationships between collections.

12. How does Node prevent blocking code?

Node.js prevents blocking code by using an event-based approach and non-blocking I/O. Tasks are executed asynchronously with the use of callbacks, events and promises so that the main thread is not stalled while waiting for IO operations such as files or databases to respond.

13. What are the data types in MongoDB?

MongoDB supports various data types, including:
• String:Stores text data.
• Number:Stores integers, doubles, or decimals.
• Boolean:Stores true or false values.
• Array:Stores a list of values.
• Object:Stores documents or embedded documents.
• Date:Stores dates in a specific format.
• Null:Represents a null value.
• ObjectId:A unique identifier for documents.
• Binary Data:Stores binary data like files.

14. What are the key differences between REST and GraphQL?

Feature	REST	GraphQL
Data Fetching	Multiple endpoints for different data	Single endpoint, customizable queries
Over-fetching/Under-fetching	Can over-fetch or under-fetch data	Retrieves only requested data
Versioning	Requires versioning for changes	No versioning needed, schema evolves
Flexibility	Less flexible, rigid structure	Highly flexible, client controls queries
Performance	May require multiple requests	Efficient, reduces network requests

15. What is the virtual DOM in React?

The virtual DOM is a concept upon which the entire application’s efficiency and effectiveness rely. It is used in rendering components within the application. Whenever there are changes made to the state of a React component, there is no need to waste time updating the real DOM. The virtual DOM is updated first and only after the changes have been reconciled does the actual DOM get rendered.

16. What methods are available to obtain transactions and locking in MongoDB?

Code Example:

17. How does Node.js handle child threads?

Node.js is inherently single-threaded although it does manage child threads via the use of the child_process module. In this way, Node.js can create other processes in order to complete a task in parallel with the main thread instead of making an operation wait. Child processes can communicate with the parent process via messaging.
Code Example:

18. What are the differences between JavaScript var, let, and const?

var:
• Function-scoped
• Can be re-declared and updated
• Hoisted to the top of the scope
let:
• Block-scoped
• Can be updated but not re-declared
• Not hoisted, must be declared before use
const:
• Block-scoped
• Cannot be updated or re-declared
• Not hoisted, must be initialised during declaration

19. If Node.js is single-threaded, then how does it handle concurrency?

Concurrency in Node.js can be achieved using its event loop and asynchronous I/o. So while classical JavaScript is single-threaded, Node.js sends all in/ out requests (such as HTTP or file requests) to a kernel thread, which can process all of these queries at the same time. When the requests finish, instead of blocking the main thread to fetch data, Node uses callbacks or promises.

20. What are pure components?

Pure components in React are components that render the same output for the same state and props. They implement shouldComponentUpdate() method to perform a shallow comparison of props and state, preventing unnecessary re-renders and improving performance.

21. What are React Hooks?

React Hooks are functions that let you use state and other React features in functional components. Common hooks include useState for state management and useEffect for handling side effects. Hooks allow you to manage state, lifecycle methods, and other React features without writing class components.

22. Compare the features of MongoDB and MySQL.

Feature	MongoDB	MySQL

Data Model	Document-based (JSON-like)	Relational (tables and rows)
Schema	Dynamic schema, flexible structure	Fixed schema, structured data
Transactions	Supports ACID transactions (since v4.0)	Fully supports ACID transactions
Scalability	Horizontally scalable (sharding)	Vertically scalable
Performance	High for unstructured data	High for structured data and complex queries
Use Cases	Big data, content management	Financial systems, structured data

23. What is aggregation in MongoDB?

Aggregation in MongoDB is the processing of large volumes of documents in a certain way that documents are presented in an ordered manner passing through stages of filtering, grouping, sorting and other computations. The aggregation framework is exactly like a GROUP BY clause in SQL except that there are no limitations.

24. How would you describe the use of JSX in Programming?

JSX (JavaScript XML) is an extension to the JavaScript language used in the React library. JSX brings in the capability to embed HTML-like codes within Javascript and afterwards transforms them into conventional Javascript using tools like Babel. It also helps in easier designing and representation of UI components structures.

25. How do synchronous and asynchronous code execution differ in JavaScript?

• Synchronous:This is a step-by-step code executed whereby each step cannot progress until the previous one is fully completed. This may cause blocking in situations where a particular line of code takes an extended period.
• Asynchronous:In this, no waiting occurs for an incomplete task. Further action is taken to the next task. After that, once the asynchronous action is done a callback, promise, or async/await is employed to take care of the output.

26. What is ReactDOM?

ReactDOM is a library whose purpose is to provide methods for working in the DOM with React components that include rendering the React components in the DOM. It acts as a bridge between the logic of React and the DOM part of the browser, enabling the optimal updation of structural content.

27. Explain the lifecycle methods of components.

Lifecycle methods in React class components allow you to control what happens during different phases of a component's life:
• Mounting:componentDidMount() is called after the component is inserted into the DOM.
• Updating:componentDidUpdate() is called after the component's state or props change.
• Unmounting:componentWillUnmount() is called just before the component is removed from the DOM.

28. What are the differences between GET and POST methods in HTTP?

Feature	GET	POST
Data Transmission	Appends data to the URL (query string)	Sends data in the request body
Security	Less secure (data visible in URL)	More secure (data not visible in URL)
Data Size	Limited by URL length	No size limitations for data
Use Case	Retrieving data	Submitting data (e.g., forms)
Idempotence	Idempotent (multiple requests = same result)	Non-idempotent (multiple requests can have different effects)

29. What is sharding in MongoDB?

A shard in MongoDB is a logical ‘chunk’ of a dataset that has been split for distribution across various servers. Sharding is horizontal scaling by dividing a large dataset into smaller fragments or shards so that they become more manageable. Each shard is situated on a different server therefore performance is enhanced and more data and traffic can be accommodated in the database.

30. What is a stream and what types of streams are available in Node.js?

In Node.js, a stream is an abstract interface that encompasses classes such as, Writable stream, Readable stream, Transform stream, Duplex stream, etc, that deal with the input and output of data through chunks rather than in a single scenario which may take longer to process data. Data streams are useful to prevent excessive memory consumption during the reading and writing of a file.
Types of streams include:
• Readable streams:Used to read data, e.g., fs.createReadStream().
• Writable streams:Used to write data, e.g., fs.createWriteStream().
• Duplex streams:Both readable and writable, e.g., TCP sockets.

31. How do you perform a CRUD operation in MongoDB using Mongoose?

CRUD (Create, Read, Update, Delete) operations in Mongoose involve interacting with MongoDB collections through models. Here’s a brief example:

32. What is prop drilling?

Prop drilling is a process in React where you pass data from a parent component down to deeply nested child components through props. This can make the code difficult to manage as the number of components increases. To avoid prop drilling, you can use context or state management libraries like Redux.

33. How do promises differ from callbacks in JavaScript?

• Callbacks:These are ordinary functions which take other functions as their arguments, and are executed once the required operation is accomplished.
• Promises:These are the objects which are used to complete or fail an asynchronous operation. Promises are preferred over callbacks since they can be easily chained together leading to a reduction in complexity and an increase in code comprehension.

34. How do you use the useState and useEffect hooks in a functional React component?

• useState:Initialises state in a functional component and returns a stateful value along with a function to update it.const [count, setCount] = useState(0);
• useEffect:Runs side effects in a functional component, such as fetching data or updating the DOM.useEffect(() => { document.title = `Count: ${count}`; }, [count]); // Runs only when 'count' changes

MERN Stack Interview Questions With Answers for Intermediate-Level

35. How does Node.js handle child threads?

Node.js is single-threaded but the child_process module allows managing child threads. This module implements the creation of subprocesses by Node.js to carry out work in parallel, sparing the main process by transferring heavy tasks and avoiding blocking. Subprocesses send messages back to the parent process.

36. What is blocking code?

Blocking code refers to code that stops the execution of further instructions until the current task is completed. In a synchronous process, blocking code can halt the entire flow, making the system less responsive. Node.js avoids blocking code by using non-blocking, asynchronous methods.

37. Compare relational and non-relational databases.

Feature	Relational Databases	Non-relational Databases
Data Model	Structured, table-based (rows and columns)	Unstructured or semi-structured (document, key-value, etc.)
Schema	Fixed schema	Dynamic schema
Scalability	Vertically scalable	Horizontally scalable (sharding)
Transactions	Fully supports ACID transactions	Limited or eventual consistency
Flexibility	Less flexible, rigid schema	More flexible, allows for varied data types
Examples	MySQL, PostgreSQL	MongoDB, Cassandra

38. Can you elaborate on the MongoDB aggregation pipeline?

The MongoDB aggregation pipeline is a framework for data aggregation, which processes data in a series of stages. Each stage transforms the documents, and the output of one stage is passed as input to the next. Common stages include:
• $match:Filters documents to pass only those that match the criteria.
• $group:Groups documents by a specified key and performs aggregate functions.
• $project:Reshapes each document by including, excluding, or adding fields.
• $sort:Orders the documents based on specified fields.
• $limit:Limits the number of documents to pass to the next stage.

39. What is middleware in Node.js and how is it used?

Middleware in Node.js refers to functions that run during the request-response cycle in an Express application. They can perform various tasks like logging, authentication, or modifying request and response objects. Middleware functions have access to the req, res, and next objects, allowing them to pass control to the next middleware in the stack.
Example:

40. What is the difference between ShadowDOM and VirtualDOM?

Feature	ShadowDOM	VirtualDOM
Purpose	Encapsulates component styles and structure	Efficiently updates the real DOM
Scope	Component-specific	Entire application
Isolation	Provides style encapsulation	No style encapsulation
Usage	Primarily used in web components	Used in React for rendering optimization
Implementation	Native browser technology	Part of React's reconciliation process

41. How do you set up routing in a React application using React Router?

The library which enables the implementation of routing in React applications is called React Router. When you want to implement routing you need to install React Router, outline the routes then make use of components such as BrowserRouter, Route and Switch.
Example:

42. What's the event loop?

In Node.js, the event loop is the main feature by which asynchrony is achieved. The event loop awaits that the call stack be empty before executing tasks in the callback queue by polling the call stack and callback queue. With this feature installed within Node.js, it is possible to carry out non-blocking I/O operations which in turn contributes to the responsiveness of applications even when under heavy loads.

43. Compare Node.js and traditional web servers like Apache.

Feature	Node.js	Apache
Language	JavaScript	C, Perl, PHP, etc.
Concurrency	Non-blocking, event-driven	Thread-based

Performance	High for I/O-bound tasks	High for CPU-bound tasks
Flexibility	Highly flexible, full-stack development	More rigid, primarily serves static content
Use Case	Real-time applications, APIs	Static content, traditional websites

44. Can you explain CORS?

CORS in the full form Cross-Origin Resource Sharing refers to an unsafe practice of how web resources serve content to other domains only within a web browser.

45. Could you tell us what Mongoose is?

Mongoose is a library dealing with object data modelling (ODM) in node.js based systems and was specifically made for the use of the MongoDB database. It provides a solution/schema to your application data and includes a built-in validator, query builder and middleware. With the use of Mongoose, we can also work with the structure of MongoDB by creating schemas and models for MongoDB collections.

46. How to check if an object is an array or not in JavaScript?

You can check if an object is an array using the Array.isArray() method.
Example:

47. Compare React’s useState and useReducer hooks.

Feature	useState	useReducer
Purpose	Simple state management	Complex state logic, often with multiple actions
API	Returns state and a setter function	Returns state and a dispatch function
Complexity	Suitable for basic state needs	Better for managing state transitions
Example Use Case	Toggling a value, managing a form input	Managing a complex form, implementing a state machine

48. List the two arguments that async.queue takes as input in Node.js.

The async.queue function in Node.js takes two arguments:
• worker:A function for processing each task in the queue.
• concurrency:The maximum number of worker functions that should run concurrently.

49. What is the need for module exports in Node.js?

Module.exports is a feature that allows exporting given functions, objects or primitives within a node.js module to be used in other files. It is a definition of what a module will make available for access by other portions of the application. // math.js function add(a, b) { return a + b; } module.exports = { add }; // main.js const math = require(‘./math’); console.log(math.add(2, 3)); // 5

50. How do you create a MongoDB schema and model using Mongoose?

In Mongoose, the developers create a schema and define what each document in a given MongoDB collection must look like. When the details regarding the schema are complete, the developer will then proceed to make a model out of it. This model will represent the collection in the database which allows the developer to carry out operations on the model.
Example:

51. What is a RESTful API?

RESTful API is the acronym for 'Representational State Transfer' head. It is an architectural style that has no definitive standard for designing networked applications. RESTful APIs are stateless, therefore a client request must be able to stand alone and contain all the necessary information to be fulfilled.

52. Compare the pros and cons of using MongoDB vs. Firebase for a project.

Feature	MongoDB	Firebase
Data Model	Document-based (JSON-like)	Realtime NoSQL database
Scalability	High scalability (sharding)	Scalable with auto-syncing capabilities
Offline Support	Not built-in, requires additional setup	Built-in offline data persistence
Flexibility	Highly flexible, schema-less design	Less flexible, designed for real-time apps
Querying	Powerful querying and aggregation	Limited querying capabilities
Use Case	Complex queries, large datasets	Realtime features, mobile apps
Learning Curve	Requires more setup and knowledge	Easier to start with, especially for beginners

53. What are Node.js buffers?

Buffers in Node.js are used to handle binary data directly, especially when dealing with streams of data like file I/O or network packets. A buffer is a raw memory allocation outside the V8 heap, which makes it efficient for working with binary data.
Example:
const buffer = Buffer.from('Hello, World!'); console.log(buffer.toString()); // Outputs: 'Hello, World!'

54. Explain the concept of a stub in Node.js.

A stub in Node.js is a function or object that mimics the behaviour of real methods in tests. It allows you to control the behaviour of code that interacts with external systems, making it easier to test in isolation. Stubs can simulate different responses and verify how functions are called.
Example:

55. What is a thread pool and which library handles it in Node.js?

A thread pool in Node.js is a number of threads that are pooled together in order to work on tasks that can be done in parallel. Within node js architecture, file operations, dns resolution services, and cryptic activities such as cryptographic key operations are coupled in a single bound using a thread pool whereby multiple operations are accomplished at the same time.

56. What are the key differences between front-end and back-end development?

Aspect	Front-end Development	Back-end Development
Focus	User interface and user experience	Server-side logic, database management
Languages	HTML, CSS, JavaScript, React, Angular	Node.js, Python, Java, Ruby, PHP
Frameworks	React, Angular, Vue.js	Express.js, Django, Spring
Tools	Webpack, Sass, Bootstrap	Databases (SQL, NoSQL), Server management
Interaction	Directly interacts with users	Handles data storage, business logic
Performance	Focused on load times, responsiveness	Focused on server speed, database queries

57. How to make Node modules available externally?

To make Node modules available externally, you use module.exports to define the public interface of your module. Any function, object, or variable assigned to module.exports can be imported into other files using require().
Example:

58. What is the default scope of a Node.js application?

The most common scope of a nodejs application is the module scope. Unlike global scope, which is the scope that obtains throughout the application, node js variables, and functions inside a module is particular to that module and cannot be used outside it unless it is being exported using module.exports.

59. Compare the performance of MongoDB vs. PostgreSQL.

Aspect	MongoDB	PostgreSQL
Data Model	Document-based, flexible schema	Relational, structured schema
Read/Write Speed	High for unstructured data, rapid prototyping	High for complex queries, joins, and ACID transactions
Scalability	Horizontal scalability (sharding)	Vertical scalability, horizontal with extensions
Use Cases	Big data, content management, JSON storage	Financial systems, complex data relationships
Indexing	Indexing on any field, less complex	Advanced indexing, including full-text search

ACID Compliance	Limited, with eventual consistency	Full ACID compliance, strong consistency

MERN Stack Interview Questions With Answers for Experienced

60. What is a document in MongoDB?

A document in MongoDB is a record within a MongoDB collection, comparable to the table row in the relational database. It is a type of structure that is made of fields and their corresponding values; the fields are more like the columns in a table.
Example:

61. What is the Mongo shell?

The Mongo command line interface that allows a user to interact with the Mongo internals and build, manage, and retrieve information fromMongoDB databasesis known as the Mongo shell. It helps you manage your database, run queries, do admin work and check commands for MongoDB. The Mongo shell is a software environment commonly used by developers and database administrators to run database tasks from a command window.
Example:
$ mongo > use myDatabase > db.users.find({ name: “John Doe” })

62. Explain the term “indexing” in MongoDB.

The creation of indexes in MongoDB is a technique used to optimise the search processes over a collection. It is a specialised structure which contains some parts of the data, and it is easy to navigate. MongoDB uses indexes on its fields/texts for finding the necessary documents without scanning through the entire collection, thus enhancing the query performance.
Common Index Types:
• Single field index:Indexes a single field.
• Compound index:Indexes multiple fields within a document.
• Text index:Supports text search queries on string content.
• Geospatial index:Supports location-based queries.

63. How do you delete a document in MongoDB?

If you want to delete one or a group of documents from the collection in MongoDB, you can apply methods deleteOne() or deleteMany() to a particular collection. deleteOne() deletes one document satisfying the requirements, on the contrary deleteMany() gets rid of all satisfying requirements documents.
Example:

64. What are the differences between a web server and an application server?

Feature	Web Server	Application Server
Primary Function	Serves static content, such as HTML, CSS, and images	Executes business logic, serves dynamic content
Processing Capability	Handles HTTP requests and responses	Manages application logic, database interactions
Examples	Apache, Nginx	Tomcat, WebSphere, JBoss

Languages Supported	HTML, CSS, JavaScript	Java, .NET, PHP, Python, etc.
Interaction	Directly interacts with the client's browser	Interacts with databases, APIs, and web servers
Performance	Optimised for serving static files	Optimised for complex transactions and logic processing

65. What is a replica set in MongoDB?

The primary function of a replica set in MongoDB is to ensure redundancy and high uptime through a set of MongoDB servers that hold identical data. There is a primary node and secondary nodes in each replica set. The primary node is responsible for every write operation on the replicas as the secondary nodes only replicate data from the primary server. If the primary node fails, an automatic election process selects a new primary, ensuring minimal downtime.
Key Features:
• Primary Node:Handles write operations and replicate data to secondaries.
• Secondary Nodes:Replicate data from the primary and can serve read operations.
• Arbiter:Participates in elections but does not store data.

66. What is scaffolding in Express.js?

Scaffolding in Express.js refers to the automatic generation of boilerplate code for a new Express application. Tools like express-generator create a basic structure with folders for routes, views, and static files, setting up a minimal working application. This scaffolding allows developers to quickly start building applications by providing a consistent starting point.
Example:
npx express-generator myapp cd myapp npm install npm start

67. What is routing and how does routing work in Express.js?

Routing in Express.js refers to how the application responds to client requests at specific endpoints (URIs). A route defines a specific path and the HTTP methods (GET, POST, etc.) that can be applied to it. Express uses route handlers to manage requests and responses based on the URL pattern and HTTP method.
Example:

68. How can I authenticate users in Express?

User authentication in Express can be implemented using middleware to handle user sessions, tokens, or credentials. Common methods include:
• Session-based authentication:Storing user sessions on the server using libraries like express-session.
• Token-based authentication:Using JSON Web Tokens (JWT) to verify user identity. The token is sent with each request and verified by the server.
• OAuth:Implementing third-party authentication via providers like Google or Facebook using passport.js.

69. Which template engines does Express support?

Express supports various template engines that allow you to generate HTML pages dynamically. Some popular template engines include:
• Pug (formerly Jade):A high-performance template engine with a concise syntax.
• EJS (Embedded JavaScript):A simple template engine that lets you embed JavaScript in your HTML.

• Handlebars:An extension of the Mustache template engine with more powerful features.

70. What function arguments are available to Express.js route handlers?

The route handler of Express.js has three main arguments.
• req (request):This is the request object which contains all the information of the request that was made such as headers, URLs etc.
• res (response):This refers to the response content that will be sent to the client (e.g. status, JSON data, HTML content).
• next (next middleware):This is a function that is called to pass control over to the next middleware or the route handler on the stack. This is used when multiple functions have to be called when handling one request or when multiple functions handle one request, especially in the case of errors.

71. How do I render plain HTML in Express?

To render plain HTML in Express, you can use the res.sendFile() method to send an HTML file from your server to the client, or you can use res.send() to send HTML as a string.
Example:

72. How to connect Node.js with React.js?

In order for Node.JS to communicate with ReactJS, one has to typically establish a Node.js server (with express). The purpose of this server is to serve the React application as well as attend to API requests. To ease the development of the React application, developers need to utilise create-react-app and express to handle the server-side processing.
Steps:
• Create a React app using create-react-app.
• Set up an Express server to handle API routes.
• Use fetch or axios in your React components to make requests to your Node.js server.
Example (Server-side):

73. What is the difference between ShadowDOM and VirtualDOM?

Feature	ShadowDOM	VirtualDOM
Purpose	Encapsulates component styles and structure	Efficiently updates the real DOM
Scope	Component-specific	Entire application
Isolation	Provides style encapsulation	No style encapsulation
Usage	Primarily used in web components	Used in React for rendering optimization
Implementation	Native browser technology	Part of React’s reconciliation process

74. What are higher-order components (HOC) in React?

Higher-Order Components (HOC) are a type of function that takes a component as an argument and returns a new component that has additional capabilities. HOC lets you think of specified component logic such as states, or any behaviour and implement that in as many components as you want but with no code being repeated.
Example:

75. How can you achieve transaction and locking in MongoDB?

MongoDB supports multi-document transactions, allowing you to perform multiple operations atomically. The beginning of the transaction can be done with methods startSession() and startTransaction(), thus either all operations are executed successfully or none at all.
Example:

76. How to avoid callback hell in Node.js?

Callback hell occurs when multiple nested callbacks make code difficult to read and maintain. You can avoid callback hell in Node.js by using:
• Promises:Allows chaining of asynchronous operations.
• Async/Await:Provides a more synchronous-looking flow for asynchronous code.
• Modularization:Break down complex code into smaller, reusable functions or modules.
Example:

77. How do you handle form validation in React?

Form validation in React can be handled by managing form state and validating inputs as the user types or on form submission. You can use controlled components, where form inputs are linked to state, or libraries like Formik for more complex forms.
Example:

78. What are the differences between HTTP and HTTPS?

Feature	HTTP	HTTPS
Security	Unencrypted, vulnerable to attacks	Encrypted using SSL/TLS
Port	Uses port 80 by default	Uses port 443 by default
Data Integrity	Data can be tampered with during transit	Data is secure from tampering
Certificate	No certificates required	Requires an SSL/TLS certificate
Use Case	Non-sensitive data transfer	Sensitive data transfer (e.g., payments)

79. How do you compare for loops and map functions in JavaScript?

Feature	for Loop	map() Function
Purpose	Iterates over elements with custom logic	Creates a new array by applying a function to each element
Mutation	Can mutate the original array or work on other data structures	Returns a new array, does not mutate original
Return Value	No inherent return value	Returns a new array
Readability	Can be less readable with complex logic	More concise and readable

Performance	Generally faster for simple tasks	Slightly slower but more functional

80. Compare the performance of MongoDB vs. PostgreSQL.

Aspect	MongoDB	PostgreSQL
Data Model	Document-based, flexible schema	Relational, structured schema
Read/Write Speed	High for unstructured data, rapid prototyping	High for complex queries, joins, and ACID transactions
Scalability	Horizontal scalability (sharding)	Vertical scalability, horizontal with extensions
Use Cases	Big data, content management, JSON storage	Financial systems, complex data relationships
Indexing	Indexing on any field, less complex	Advanced indexing, including full-text search
ACID Compliance	Limited, with eventual consistency	Full ACID compliance, strong consistency

Conclusion

MERN stack is a sturdy combination of tech stacks for web developers to develop full-stack web applications efficiently using the JavaScript programming language throughout the stack. By understanding key concepts such as MongoDB schema design, Express.js middleware, React components, and Node.js concurrency, you can create scalable and high-performance applications. This guide on MERN stack interview questions has taken into consideration learners at different levels, beginners, and experienced developers as well. Learning the MERN stack will not only help you ace the technical interview but also give you the necessary skills to create modern web applications. In this case, whether it is MongoDB database transactions, setting up authentication in a MERN stack application, or optimising the React components, this information is going to be very helpful in your interview process.
Updated onOctober 16, 2024