■ MERN Stack Full Interview Preparation Guide

MERN Stack Fundamentals

Q: What is the MERN stack and how do its components interact?
A: MERN stands for MongoDB, Express.js, React.js, and Node.js. React handles the frontend, Node and Express manage backend APIs, and MongoDB stores data. React sends requests to Express, which interacts with MongoDB and sends responses back.

Q: How do you structure a MERN stack application?
A: Typically, the project is divided into a frontend folder (React) and a backend folder (Node/Express). The backend contains routes, controllers, and models, while the frontend handles UI components and state management.

Q: Explain how data flows between the frontend (React) and the backend (Express.js/Node.js).
A: The frontend sends HTTP requests (usually via Axios or Fetch) to Express API endpoints. The backend processes the data, communicates with MongoDB, and sends a JSON response back to React for UI updates.

Q: How do you handle authentication and authorization in a MERN app (e.g., JWT)?
A: After login or signup, the backend generates a JWT (JSON Web Token) that is sent to the client. The token is stored in local storage or cookies and attached to future requests for verifying access to protected routes.

Q: What is CORS and how do you address it in a MERN application?
A: CORS (Cross-Origin Resource Sharing) restricts browser requests from different domains. In Express, it can be handled by using the 'cors' middleware package to allow specific frontend domains to access the backend.

Q: How do you deploy a MERN stack application?
A: React frontend can be deployed on Vercel or Netlify, and the backend (Express/Node) on Render or Railway. MongoDB Atlas is used for cloud database hosting, and environment variables are configured for production.

Q: Discuss the advantages and disadvantages of using the MERN stack.
A: Advantages include a unified JavaScript language, scalability, and fast development. Disadvantages include high initial complexity, asynchronous handling issues, and performance limits for extremely large-scale systems.

MongoDB Questions

Q: What is MongoDB and how does it differ from a relational database?
A: MongoDB is a NoSQL database that stores data as JSON-like documents instead of tables. Unlike relational databases, it doesn't require predefined schemas and supports flexible data structures.

Q: Explain concepts like collections, documents, and BSON.
A: A collection in MongoDB is similar to a table in SQL, documents are like rows but stored as JSON objects, and BSON (Binary JSON) is the internal data format MongoDB uses for performance.

Q: What are replica sets and sharding in MongoDB, and why are they used?
A: Replica sets provide data redundancy by keeping multiple copies of data across servers for high availability. Sharding splits data into smaller chunks across multiple servers for better scalability.

Q: How do you perform CRUD operations in MongoDB?
A: Create with insertOne(), Read with find(), Update with updateOne(), and Delete with deleteOne(). In Mongoose, these are abstracted with model methods like Model.create() and Model.find().

Q: Explain the MongoDB aggregation pipeline.
A: The aggregation pipeline processes data through multiple stages (like $match, $group, $sort) to transform and analyze data within the database efficiently.

Express.js & Node.js Questions

Q: What is Node.js and how does it achieve non-blocking I/O?
A: Node.js is a JavaScript runtime built on Chrome's V8 engine. It uses an event-driven, asynchronous model and a single-threaded event loop to handle multiple requests without blocking.

Q: Explain the event loop in Node.js.
A: The event loop continuously checks for queued tasks, executes callbacks, and handles asynchronous operations, enabling Node.js to perform efficiently without multiple threads.

Q: What is Express.js and its role in the MERN stack?
A: Express.js is a lightweight framework that simplifies building RESTful APIs and server logic. It handles routing, middleware, and HTTP requests easily within the Node environment.

Q: Explain middleware in Express.js and provide examples of its use.
A: Middleware functions intercept requests before they reach routes. Examples include authentication (JWT validation), logging (Morgan), and error handling middleware.

Q: How do you create a RESTful API using Express.js?
A: Define routes (GET, POST, PUT, DELETE) using app.get(), app.post(), etc., then connect them to controller functions that perform database operations and return JSON responses.

Q: Discuss error handling in Express.js.
A: Error handling is done via middleware functions with four parameters (err, req, res, next). Developers can also use try-catch blocks and send custom error responses to clients.

React Questions

Q: What is React and its core principles (e.g., Virtual DOM, components, JSX)?
A: React is a UI library based on components and declarative programming. It uses a Virtual DOM to efficiently update the UI and JSX syntax to write HTML-like code in JavaScript.

Q: Explain the component lifecycle methods in class components or the equivalent in functional components with Hooks.
A: Lifecycle methods like componentDidMount and componentWillUnmount are used in class components. In functional components, useEffect serves a similar purpose to handle side effects.

Q: What are React Hooks and how do they improve functional components?
A: Hooks like useState and useEffect allow functional components to use state and lifecycle features without converting them into class components, making code simpler and cleaner.

Q: How do you manage state in a React application (e.g., useState, useContext, Redux)?
A: Local state is managed with useState, global state with useContext or Redux. Redux centralizes the state, making it easier to share data across components.

Q: Explain the difference between state and props.
A: State represents internal data managed by the component itself, while props are inputs passed from parent to child components to make them reusable and dynamic.

Q: How do you handle routing in a React application (e.g., React Router)?
A: React Router manages client-side navigation using routes like /home or /product/:id without reloading the page. It uses BrowserRouter, Routes, and Route components.

Q: Discuss performance optimization techniques in React.
A: Use React.memo to avoid re-renders, lazy load components, use key props efficiently, and prevent unnecessary state updates. Code-splitting also improves performance.

Integration & Advanced Topics

Q: How do you connect Node.js to MongoDB (e.g., using Mongoose)?
A: Mongoose provides an abstraction layer over MongoDB. You connect using mongoose.connect(DB_URI) and define schemas/models for structured data access.

Q: Explain how to implement file uploads in a MERN application.
A: Use multer (Node middleware) for handling multipart/form-data. Uploaded files can be stored locally or on services like Cloudinary or AWS S3 for scalability.

Q: Discuss real-time features using WebSockets in a MERN context.
A: WebSockets (via Socket.io) allow real-time communication like chats or live updates between client and server by maintaining persistent bi-directional connections.

Q: Compare RESTful APIs with GraphQL in the context of MERN.
A: REST APIs use multiple endpoints for each resource, while GraphQL uses a single endpoint where clients specify exactly what data they need, improving efficiency.

Q: How do you ensure the security of a MERN stack application?
A: Use HTTPS, JWT for authentication, bcrypt for password hashing, CORS configuration, input validation, and environment variables to hide sensitive data.