50 machine coding round interview questions MERN

Create By - Ankit Sharma (https://www.linkedin.com/in/ankit7rma/)

Beginner-Level Questions (1-15)

1. **Build a Simple To-Do List Application**

   Create a React frontend with a Node.js/Express backend to add, view, and delete tasks, storing them in MongoDB.

2. **User Registration Form**

   Design a React form to collect user details (name, email, password) and save them to MongoDB via an Express API.

3. **Counter Application**

   Implement a counter in React with increment, decrement, and reset buttons, using useState hook.

4. **API to Fetch and Display Users**

   Write an Express API to fetch a list of users from MongoDB and display them in a React table.

5. **Simple Calculator**

   Build a React component that performs basic arithmetic operations (add, subtract, multiply, divide).

6. **CRUD API for Books**

   Create RESTful APIs using Express and MongoDB to perform CRUD operations on a "Books" collection.

7. **Weather App**

Integrate a public weather API with a React frontend to display the current weather based on user input (city name).

8. **Login Form Validation**

   Design a React login form with client-side validation for email and password fields.

9. **File Upload Endpoint**

   Write an Express route to handle single file uploads and store metadata in MongoDB.

10. **Display a List of Posts**

    Fetch a list of posts from a mock API (or MongoDB) and render them in a React component with pagination.

11. **Search Filter**

    Create a React component to filter a list of items (e.g., products) based on user input.

12. **Simple Chat Interface**

    Build a React UI for a chat interface with a static list of messages (no backend required).

13. **REST API for Comments**

    Implement an Express API to add and retrieve comments for a specific post, stored in MongoDB.

14. **Dynamic Dropdown**

    Create a React dropdown that fetches options from an Express API connected to MongoDB.

15. **Basic Authentication Middleware**

    Write an Express middleware to check if a user is authenticated using a hardcoded token.

Intermediate-Level Questions (16-35)

1. **Task Manager with Categories**

Extend the to-do list app to include task categories, with MongoDB storing tasks and categories in separate collections.

2. **User Authentication System**

   Implement signup, login, and logout functionality using JWT, Express, and MongoDB, with a React frontend.

3. **Real-Time Notifications**

   Build a system using Socket.io, Node.js, and React to display real-time notifications when a new item is added.

4. **E-commerce Product Listing**

   Create a React page to display products from MongoDB via an Express API, with sorting and filtering options.

5. **Blog Application**

   Develop a full-stack blog app with React for the frontend and Express/MongoDB for creating and listing posts.

6. **Rate Limiting API**

   Add rate-limiting to an Express API endpoint using middleware to restrict requests per user.

7. **Infinite Scroll**

   Implement infinite scrolling in React to load more items from an Express API as the user scrolls.

8. **Form with File Upload**

   Build a React form to upload an image with metadata (e.g., title, description) to MongoDB via Express.

9. **Role-Based Access Control**

   Extend the authentication system to include user roles (admin, user) with restricted API access.

10. **Shopping Cart**

    Create a React shopping cart component that stores items in local storage and syncs with a MongoDB backend.

11. **Autocomplete Search**

    Build a React search bar with autocomplete suggestions fetched from an Express API.

12. **API Pagination**

    Modify an Express API to return paginated results from MongoDB with limit and offset parameters.

13. **Real-Time Chat Application**

    Implement a basic chat app using Socket.io, React, and Node.js, with messages stored in MongoDB.

14. **Dashboard with Charts**

    Use a charting library (e.g., Chart.js) in React to display data fetched from an Express API.

15. **URL Shortener**

    Create a full-stack app to shorten URLs, store them in MongoDB, and redirect using Express.

16. **Multi-Step Form**

    Build a multi-step registration form in React with validation and final submission to an Express API.

17. **Event Scheduler**

    Design a React app to schedule events, with CRUD operations handled by Express and MongoDB.

18. **API Error Handling**

    Enhance an Express API with custom error handling middleware and meaningful error responses.

19. **React Context for State Management**

    Refactor a React app to use Context API for managing global state (e.g., user data).

20. **File Download Endpoint**

Write an Express route to allow users to download a file stored on the server or in MongoDB.

Advanced-Level Questions (36-50)

1. **Social Media Feed**

   Build a React feed component with infinite scroll, likes, and comments, backed by Express and MongoDB.

2. **Real-Time Collaborative Editor**

   Create a collaborative text editor using React, Socket.io, and Node.js (no persistence required).

3. **API with Aggregation**

   Write an Express API that uses MongoDB aggregation to return grouped data (e.g., sales by month).

4. **Microservices Architecture**

   Split a monolithic MERN app into two services: one for users and one for products, with separate Express servers.

5. **Custom React Hook**

   Build a custom React hook to fetch and cache data from an Express API.

6. **Secure File Upload**

   Enhance the file upload system with size limits, file type validation, and storage in MongoDB GridFS.

7. **Full-Text Search**

   Implement a search feature in MongoDB with text indexing and display results in React.

8. **Rate Limiter with Redis**

   Integrate Redis with Express to implement a distributed rate limiter for API requests.

9. **React Drag-and-Drop**

Create a React component for dragging and dropping tasks between columns, with updates saved to MongoDB.

10. **API Caching**

    Add caching to an Express API using an in-memory store (e.g., Redis or a simple object) to improve performance.

11. **User Profile with Image**

    Build a full-stack feature to upload and display a user profile picture, stored in MongoDB.

12. **Debounced Search**

    Implement a debounced search input in React that queries an Express API after a delay.

13. **Task Queue System**

    Use a task queue (e.g., Bull with Redis) in Node.js to process background jobs triggered by an API.

14. **React Lazy Loading**

    Refactor a React app to lazy-load components and routes, fetching data from an Express API.

15. **Multi-Tenant Application**

    Design a MERN app where each tenant (e.g., company) has isolated data in MongoDB, with tenant-based routing in Express.