Sub: Complete question bank for Backpack 02 with React (30 Q&A), MongoDB (20 Q&A), Node.js (20 Q&A), and Express.js (20 Q&A).

React (30 Q&A)

1. What are React lifecycle methods?

Answer: They are special methods in class components that run during different stages of a component's life, such as mounting, updating, and unmounting.

2. What is componentDidMount used for?

Answer: It is called after the component is mounted and is commonly used for API calls or DOM manipulations.

3. What is the difference between componentDidUpdate and shouldComponentUpdate?

Answer: componentDidUpdate runs after an update is committed, while shouldComponentUpdate decides whether the component should re-render.

4. What is getDerivedStateFromProps?

Answer: A static method used to update state based on props changes before rendering.

5. How does componentWillUnmount help?

Answer: It runs before a component is removed from the DOM, often used for cleanup like clearing timers or removing event listeners.

6. What replaces lifecycle methods in functional components?

Answer: Hooks like useEffect replace lifecycle methods for functional components.

7. What is React Router?

Answer: A library that enables routing in React applications without page reloads.

8. How to create navigation links in React Router?

Answer: By using the Link or NavLink component from 'react-router-dom'.

9. What is a dynamic route?

Answer: A route that uses parameters in the URL, e.g., /user/:id.

10. How do you navigate programmatically?

Answer: By using the useNavigate hook in React Router v6.

11. What is a nested route?

Answer: A route defined inside another route to create hierarchical navigation.

12. What is the difference between BrowserRouter and HashRouter?

Answer: BrowserRouter uses HTML5 history API, while HashRouter uses URL hash for navigation.

13. What is state in React?

Answer: An object that stores data specific to a component and can change over time.

14. How to update state in class components?

Answer: By using the setState method.

15. How is state managed in functional components?

Answer: By using the useState hook.

16. What is lifting state up?

Answer: Sharing state between components by moving it to their closest common ancestor.

17. What is Redux?

Answer: A state management library that provides a global store for predictable state updates.

18. What is Context API?

Answer: A React feature that allows passing data deeply without prop drilling.

19. How do you create controlled components in React?

Answer: By storing form input values in component state and updating via onChange events.

20. What are uncontrolled components?

Answer: Components where form data is handled by the DOM instead of React state.

21. How to handle form submission in React?

Answer: By using an onSubmit event handler that prevents default behavior.

22. What is form validation in React?

Answer: Ensuring user input meets certain conditions before processing it.

23. How do you handle multiple inputs in a form?

Answer: By using one onChange handler with input name attributes.

24. What is useForm in React Hook Form?

Answer: A hook that simplifies form state management and validation.

25. How do you protect routes in React?

Answer: By creating private routes that require authentication before rendering.

26. What is JWT in authentication?

Answer: JSON Web Token is a secure way to transmit information between parties for authentication.

27. How do you store authentication tokens?

Answer: Commonly in localStorage or HTTP-only cookies.

28. What is useAuth hook?

Answer: A custom hook to manage authentication state in React.

29. What is OAuth?

Answer: An open standard for access delegation, commonly used for social login.

30. How do you handle logout in React?

Answer: By clearing tokens and resetting authentication state.

MongoDB (20 Q&A)

31. What does CRUD stand for in MongoDB?

Answer: Create, Read, Update, Delete - basic database operations.

32. How to insert data in MongoDB?

Answer: Using insertOne() or insertMany() methods.

33. How to read data in MongoDB?

Answer: Using the find() method with optional filters.

34. How to update documents in MongoDB?

Answer: Using updateOne() or updateMany() methods.

35. How to delete documents in MongoDB?

Answer: Using deleteOne() or deleteMany() methods.

36. What is a filter query in MongoDB?

Answer: A condition used inside find() to match documents.

37. How to use $gt in MongoDB?

Answer: It matches documents where a field's value is greater than a given number.

38. What does $in operator do?

Answer: Matches documents where a field's value is in a given array.

39. What is projection in MongoDB?

Answer: Specifying which fields to include or exclude in the output.

40. What is sorting in MongoDB?

Answer: Arranging documents in ascending or descending order using sort().

41. Which package is used to connect MongoDB with Node.js?

Answer: The 'mongodb' or 'mongoose' package.

42. How to connect MongoDB in Node.js?

Answer: By using MongoClient.connect() or mongoose.connect().

43. What is mongoose?

Answer: An ODM (Object Data Modeling) library for MongoDB and Node.js.

44. How to define a schema in mongoose?

Answer: By using new mongoose.Schema() with field definitions.

45. What is a model in mongoose?

Answer: A compiled version of the schema used for CRUD operations.

46. What is indexing in MongoDB?

Answer: A way to improve query performance by creating a data structure that stores field values.

47. How to create an index in MongoDB?

Answer: Using createIndex() method on a collection.

48. What is a compound index?

Answer: An index on multiple fields in a collection.

49. What is a unique index?

Answer: An index that ensures all values in a field are unique.

50. What is the default index in MongoDB?

Answer: The _id field has a default unique index.

Node.js (20 Q&A)

51. What is Node.js?

Answer: A JavaScript runtime built on Chrome's V8 engine for running JS outside the browser.

52. How to create a simple server in Node.js?

Answer: By using the built-in 'http' module and calling http.createServer().

53. What is the default port for HTTP?

Answer: Port 80 for HTTP and port 443 for HTTPS.

54. How to send a response in Node.js?

Answer: By using res.write() and res.end() methods.

55. What is nodemon?

Answer: A tool that restarts the Node.js server automatically when file changes are detected.

56. Which module is used for file operations in Node.js?

Answer: The built-in 'fs' module.

57. How to read a file in Node.js?

Answer: By using fs.readFile() or fs.readFileSync().

58. How to write to a file in Node.js?

Answer: By using fs.writeFile() or fs.writeFileSync().

59. What is the difference between sync and async file methods?

Answer: Sync methods block code execution, async methods do not.

60. How to delete a file in Node.js?

Answer: By using fs.unlink() method.

61. What is middleware in Node.js?

Answer: Functions that process requests before they reach the final route handler.

62. How is middleware added in Express?

Answer: By using app.use() or specifying middleware in route handlers.

63. What is the purpose of body-parser?

Answer: To parse incoming request bodies in middleware.

64. What is next() in middleware?

Answer: A function that passes control to the next middleware in the stack.

65. What is CORS middleware?

Answer: Middleware that enables cross-origin resource sharing.

66. Is Node.js single-threaded?

Answer: Yes, Node.js uses a single-threaded event loop for handling requests.

67. How does Node.js handle multiple requests?

Answer: Through non-blocking asynchronous operations and callbacks.

68. What is the event loop?

Answer: A mechanism that handles asynchronous callbacks in Node.js.

69. What is libuv in Node.js?

Answer: A library that handles async I/O operations for Node.js.

70. Why is Node.js good for I/O heavy applications?

Answer: Because it uses non-blocking I/O and event-driven architecture.

Express.js (20 Q&A)

71. What is Express.js?

Answer: A minimal and flexible Node.js framework for building web applications.

72. How to handle GET requests in Express?

Answer: By using app.get(path, callback).

73. How to handle POST requests in Express?

Answer: By using app.post(path, callback).

74. How to access query parameters?

Answer: By using req.query object.

75. How to access route parameters?

Answer: By using req.params object.

76. How to handle errors in Express?

Answer: By using middleware with four parameters: (err, req, res, next).

77. What is the default error handler in Express?

Answer: A built-in handler that catches unhandled errors and sends a response.

78. How to create custom error middleware?

Answer: By defining a function with (err, req, res, next) and using app.use().

79. What is 404 handling?

Answer: Handling requests for resources that do not exist.

80. What does next(err) do?

Answer: Passes the error to the next error handling middleware.

81. How to connect Express with MongoDB?

Answer: By using mongoose.connect() or MongoClient.connect().

82. Where should the DB connection code be placed?

Answer: Usually in a separate config file or at the start of the server.

83. How to perform CRUD with MongoDB in Express?

Answer: By defining routes that use MongoDB methods inside route handlers.

84. What is async/await in DB operations?

Answer: A syntax to handle asynchronous operations in a cleaner way.

85. Why use environment variables for DB URI?

Answer: To keep sensitive credentials secure and configurable.

86. What is scaffolding in Express?

Answer: Creating a project structure quickly using a tool or template.

87. Which tool is used for scaffolding Express apps?

Answer: The 'express-generator' package.

88. How to install express-generator?

Answer: By running npm install -g express-generator.

89. How to create a new app with express-generator?

Answer: By running express appName and installing dependencies.

90. Why is scaffolding useful?

Answer: It saves time and enforces a standard project structure.