Expense Tracker Project - Detailed Q&A

Q1. What is the Expense Tracker project?

Expense Tracker is a web application that helps users keep track of their income and expenses. Users can add transactions, categorize them, and visualize financial summaries. It's built using the MERN stack (MongoDB, Express.js, React.js, Node.js).

Q2. What technologies did you use?

Frontend: React.js, Context API, Axios, TailwindCSS or Styled Components

Backend: Node.js + Express.js

Database: MongoDB (Mongoose)

Auth: JWT + bcrypt

Charts: Recharts or Chart.js

Deployment: Render (backend), Vercel or Netlify (frontend), MongoDB Atlas

Q3. What features does your app have?

- User Registration/Login (JWT)
- Add/Edit/Delete transactions
- Categorize income/expenses
- View summaries and charts
- Protected dashboard
- Responsive UI

Q4. How does login/logout work?

When a user logs in, backend sends a JWT token. This token is stored in localStorage. It is sent with every protected API call. Logout removes the token from localStorage.

Q5. How is the frontend built?

React with components for forms, dashboard, lists, etc. Context API stores global states. Axios handles API calls. React Router is used for navigation.

Expense Tracker Project - Detailed Q&A

Q6. How does your backend work?

Built with Node + Express. Defines REST APIs to register/login users and manage transactions. JWT middleware protects routes.

Q7. What is your database structure?

User Schema: username, email, hashed password, createdAt

Transaction Schema: userId, title, amount, type, category, date

Q8. How do you protect user data?

Passwords are hashed using bcrypt. JWT protects routes. Inputs are validated on both frontend and backend.

Q9. How do you show the data visually?

Used Chart.js or Recharts to show pie charts (income vs expense) and bar charts (monthly spending).

Q10. Where did you host this project?

Frontend: Vercel or Netlify

Backend: Render

Database: MongoDB Atlas

.env file used to secure secrets

Q11. What features will you add in the future?

- Budget tracking
- Recurring transactions
- PDF/CSV export
- Dark mode
- PWA/mobile version

Expense Tracker Project - Detailed Q&A

Q12. Why did you choose the MERN stack?

MERN uses JavaScript everywhere, is developer-friendly, and has strong community support. It's great for building full-stack web apps.

Q13. What challenges did you face?

Understanding JWT, using Context API properly, connecting frontend and backend, handling CORS, and deploying the project.

Q14. How does Context API help in your project?

Allows global data sharing like user info and transactions without passing props. Simplifies state management.

Q15. What is JWT and how do you use it?

JWT is a token that contains user info. Backend creates it on login. Frontend stores and sends it to access protected routes.

Q16. What is middleware in Express?

Middleware runs before route handlers. I used it to check JWT in protected routes.

Q17. How did you manage API calls?

Used Axios to call backend APIs with JWT token in headers. Also handled errors and loading states.

Q18. What are some UI/UX decisions?

Clean, simple UI with Tailwind or Styled Components. Forms are easy to use. Toast messages give feedback.

Q19. How did you structure your folders?

Separated components, context, routes, controllers, and models for better organization.

Expense Tracker Project - Detailed Q&A

Q20. How did you handle validation?

Frontend: simple if-checks. Backend: express-validator to check email format, required fields, and password length.

Q21. How does MongoDB help?

Flexible NoSQL database. Easy to store and query user transactions. Mongoose helps define schemas.

Q22. Difference between income and expense?

Each transaction has a 'type' field. I use filtering and reducing to calculate totals and balance.

Q23. What are protected routes?

Routes that require JWT token. React redirects unauthenticated users. Express middleware checks tokens.

Q24. How did you test your APIs?

Used Postman to send requests to register, login, add/edit/delete transactions. Tested token-based auth.

Q25. How do you handle CORS?

Used the CORS package in Express. Allowed frontend URL to access backend resources.

Q26. What did you learn from this project?

Full-stack development, JWT auth, MongoDB, deploying apps, solving bugs, building complete projects confidently.