Explanation of Technology Choices

Node.js

- **Why**: I use Node.js because its event-driven architecture handles concurrent API requests efficiently, which is essential for a task management system with real-time updates. The npm ecosystem provides libraries like express and mysql2 that speed up development.
- **Advantages**: Non-blocking I/O reduces server load, and its JavaScript base aligns with my frontend (Next.js) for a unified language.

MySQL

- **Why**: I chose MySQL for its relational database capabilities, ensuring data integrity with foreign keys (e.g., linking tasks to users). It supports complex queries needed for filtering tasks by status or assignee.
- **Advantages**: Reliable, widely used, and supports indexing for performance optimization, which I've implemented to address slow queries.

Next.js

- **Why**: I use Next.js for its SSR and static generation, providing fast initial loads and better user experience. Its file-based routing and API routes simplify the architecture, integrating seamlessly with my Node.js backend.
- **Advantages**: Built-in optimizations (e.g., automatic code splitting) and easy deployment make it ideal for a scalable web app.

JWT Authentication

- **Why**: I implemented JWT for authentication because it offers a stateless, secure way to verify users across requests. It fits my need for role-based access (admin, manager, user) and integrates well with Node.js.
- **Advantages**: Tokens are compact, reducing overhead, and jwtDecode allows client-side token parsing, enhancing flexibility.