Project Overview Document

Comprehensive Task Management Application: Design, Implementation, and Future Enhancements

1. What was your requirements gathering and design process? Was it useful/successful?

The requirements gathering involved identifying key features essential to task management, such as task CRUD operations, user authentication, and priority tracking. Additional needs for user experience enhancements, like visualization and form validation, were also identified. This process was successful as it enabled a clear roadmap for both backend and frontend development, creating a comprehensive application that balances functionality with user-friendly design.

2. Give a high-level overview of your application and its features.

- This task management application enables users to create, update, and track tasks with a variety of functionalities, including:
- User Authentication with JWT for secure access.
- Task Management with CRUD operations for tasks, priority categorization, and task counts.
- Data Visualization for displaying task counts and priority breakdown.
- Calendar Integration for viewing and adding events.
- User Interface with responsive design powered by Material-UI.

3. Where does its data come from (external/internal APIs)?

The data comes from an internal API developed in the backend. This RESTful API, built with Node.js, Express, and Sequelize, interacts with a MySQL database to handle user and task data.

4. How is this data processed and displayed?

Backend: Task and user data are processed with Sequelize models and business logic in controllers, secured by JWT-based authentication.

Frontend: Data is fetched via Axios and displayed in components built with Material-UI. Visualizations for task insights are rendered using Nivo charts for a visually engaging experience.

5. How can the user interact with your application?

Users can interact with the application by:

- Navigating the sidebar and header to access task and calendar views.
- Performing Actions like adding, updating, and deleting tasks.
- Viewing Visualizations on task counts and priorities.
- Managing Profile with options to view and update user details.

6. How have you structured/broken up your components/code?

Backend: The code is modularized into controllers, routes, models, and utilities for a clear separation of concerns, making it easy to maintain.

Frontend: Components are organized into components for reusable items, pages for high-level views, and services for API interactions, keeping the code manageable and modular.

7. What kinds of React hooks have you used (eg. state, context, effect, navigate)? How?

useState: Manages local state for components, like form inputs and task details.

useEffect: Handles side effects, such as fetching tasks or user data on component load.

useContext: Shares state, like user authentication data, across components for streamlined access.

8. Have you created and used any custom hooks?

Custom hooks weren't explicitly mentioned, but they would be useful for repetitive logic like authentication checks and data fetching.

9. What external tools/libraries have you used (eg. bootstrap/axios/MUI)? How? Why?

Material-UI: Provides a consistent and visually appealing user interface aligned with Material Design.

Axios: Simplifies HTTP requests to the backend, with easy-to-use promise-based syntax.

Nivo: Enhances user experience by visualizing task data through customizable and responsive charts.

Formik & Yup: Manages form states and validation, improving form reliability and user experience.

10. How might you extend the features of your application in the future?

Future enhancements could include:

- Password Recovery through email-based resets.
- Customization Options for themes and layouts.
- Reminders and Notifications to prompt users about task deadlines.
- Collaboration Features to allow shared task management among users.
- Advanced Analytics and Reporting for detailed productivity insights.