TASK MANAGEMENT WEB APP

FUNCTIONAL REQUIREMENTS	
SOFTWARE	Using React for the Client UI and Express for the Server with node.js and MongoDB to manage the databse. Create and manage components and their state for each UI, ensure all the fetch functions are correctly linked with the routers in the server. Set up middleware to manage roles and responsibilities including user registration and login, handling JWT's and any other restrictions we may need to incur, set appropriate controllers and schema models and interface with MongoDB. Deploy on Heroku or C-Panel.
EXPECTED USERS OF THE SOFTWARE	Individuals, Teams and Team Managers.
CORE APP REQUIREMENTS	Allow individuals to create, edit, delete and display their own tasks as well as any of the tasks of team members who have approved their team member account. Team leaders will also need to able to globally add new tasks that are broadcast to the entire team as well as being individually populated for each team member. Team members will also need to be updated on each individual team member's task status in terms of progress, deadlines, challenges, overdue tasks and completed task. Needs be designed for mobile first and then upwards.
FUTURE ADDITIONS	Allow the inclusion of QR Scanners and Check_Out Facilities for online purchases. This will be managed once the need arises.
NON-FUNCTIONAL REQUIREMENTS	
USEABILITY	Upon launch, new user registration page, login-in, app deployment that opens on existing tasks on the database. Each task must have the CRUD capabilities attached to it individually in lieu of a DOM event (click, button, dropdown etc). Task creation, deletion and editing must happen on the main user page same on a conditional form (modal) which updates the original form state. In real-time using use-effect hooks. Users will also be granted certain privileges based on their respective company roles eg. Team leaders vs team members. Team leaders will have global CRUD access. App will be available and dynamic on all devices.
RELIABILITY	All software will be backed-up and updated on a regular basis. All database software will be kept safe in MongoDB Atlas. Depending on deployment, all PasS providers provide a multitude of back-up server functions for the eventualities of downtime ensuring that if one server crashes the app traffic is diverted to other servers to ensure that the app is always up. Deployment will only be done once all the user tests have been carried out on the app extensively.
PERFORMANCE	All media to be kept at minimal size and best practice formats like WebP. Once the application is built incorporate Next.js to start improving on Server side rendering and optimised SEO. Check Google console for continued app and page performances.
SECURITY	Ensure all JWT, User registration and login details as well as any other valuable API key's are kept in ENV files safe from the console in the front end. Heroku and MongoDB manage the security on the deployment and database end.