Full Stack position home assignment:
Build Tracker Application

Overview
Please create a full-stack application called "Build Tracker" to manage and display software build records.
This project is designed to highlight your expertise in both frontend and backend development.
The backend will be crafted using Node.js, featuring a SQLite database for data storage, and will interface with the frontend through GraphQL.
The frontend will be developed with Vue.js, employing TypeScript to enhance its functionality and structure.

Backend Specification

Technology Stack

- **Runtime Environment:** Node.js
- **Database:** SQLite
- **API:** GraphQL

Database Fields
Database schema should include following list of fields:

- BuildId: Unique identifier for the build.
- StartTime: Timestamp when the build process started.
- EndTime: Timestamp when the build process ended.
- Caption: Short description or title of the build.
- Command: The command line used to initiate the build.
- Status: Status of the build.
- ErrorsNumber: Count of errors generated during the build.
- WarningsNumber: Count of warnings generated during the build.

GraphQL API
Queries:

- getAllBuilds: Retrieves all builds from the database.
- getBuildById(id: Int): Retrieves a specific build by its BuildId.

Frontend Specification

Technology Stack

- **Framework:** Vue
- **Language:** JS/TypeScript
- **CSS Framework**: Tailwind
- **State Management:** Vuex (optional, if needed)
- **Communication:** Apollo Client (for GraphQL

** Detailed Figma **

Overview
Construct a table with pagination to display build data, which is retrieved from a GraphQL server.

Features:

- **Build List View**:

Construct a table to showcase all builds, sourcing data from the backend using GraphQL. The table should encompass columns for each database field specified. Integrate pagination to facilitate easy navigation through the data.

User Interface Requirements:

- **Responsive Design:** Design the application to be fully functional and visually appealing on both desktop and mobile platforms.
- **Styling:** Employ Tailwind CSS for the application's styling to ensure a modern and cohesive aesthetic.

Documentation

- **README File:** Include comprehensive setup and deployment instructions, a brief description of the project.
- **Code Comments:** Ensure the codebase is well-documented to explain the functionality of major components.

Evaluation Criteria
- **Code Quality:** Cleanliness, organization, and adherence to best practices.
- **Functionality:** All features are implemented and work as described.
- **Design and UX:** Usability and visual appeal of the application.
- **Documentation and Testing:** Comprehensive documentation and test coverage.

Submission Guidelines:
Please submit the project by emailing a zipped archive of the project files. Ensure the zip file contains all necessary build scripts and detailed instructions for running the application.

Good luck!