**Dashboard Project Documentation**

**Table of Contents**

### [Introduction](https://chat.openai.com/c/bf8c8c96-55db-4840-926e-97fb0bf8317d#introduction)

1. [Design Choices](https://chat.openai.com/c/bf8c8c96-55db-4840-926e-97fb0bf8317d#design-choices)
2. [Challenges and Solutions](https://chat.openai.com/c/bf8c8c96-55db-4840-926e-97fb0bf8317d#challenges-and-solutions)
3. [Screenshots and Functionality](https://chat.openai.com/c/bf8c8c96-55db-4840-926e-97fb0bf8317d#screenshots-and-functionality)

**Introduction**

The dashboard project is designed to provide users with tools for managing inventory and analyzing revenue data. It offers a user-friendly interface with features like inventory management, revenue analysis, low inventory alerts, and product registration.

**Design Choices**

**Frontend Framework**

We chose to build the frontend of the dashboard using React.js due to its component-based architecture, which makes it easy to manage complex user interfaces. Additionally, we utilized popular libraries such as React Router for navigation and Chart.js for data visualization.

**Styling**

For styling, we opted for a modern and user-friendly design. We used CSS for basic styling and incorporated a CSS-in-JS library for more complex styling needs. The color scheme is designed to be intuitive, with different colors representing key data points and actions.

**Data Management**

We manage our data using state management libraries like Redux to efficiently handle data flow within the application. We also fetch data from a mock API for demonstration purposes. In a real-world scenario, this would be replaced with actual API calls.

**Challenges and Solutions**

**Data Integration**

**Challenge**: Integrating data from various sources and ensuring it's up-to-date was a challenge. We needed to keep the inventory and sales data synchronized.

**Solution**: We implemented a system to update inventory levels automatically when sales occurred. Low inventory alerts were triggered when the current stock fell below a predefined threshold.

**Chart Rendering**

**Challenge**: Rendering interactive charts with real-time data can be complex.

**Solution**: We used the Chart.js library to create dynamic and responsive charts. Data is updated and rendered in real-time as users interact with the application.

**Screenshots and Functionality**

**Inventory Management Page**

* List view of all products with current inventory status.
* Options to sort, filter, and search products.
* Functionality to update inventory levels.
* Low inventory alerts and forecasting to indicate when restocking is necessary.

**Revenue Analysis Page**

* Displays total orders and total sales.
* Allows filtering of revenue data by category.
* Presents revenue trends using interactive charts.

**Product Registration Page**

* Form to add new products to the inventory.
* Fields for product name, description, price, and initial stock level.
* Option to upload product images.
* Successful submission updates the inventory and is reflected across the dashboard.