**Frontend Development with React.js**

**Project Documentation format**

1. **Introduction**
   * Project Title: Inventory Management Frontend
   * Team Members:Aalbin Joseph S-Teamleader, Abhishek A-team member, Chethan S-Team member, Gokul R-Team member
2. **Project Overview**
   * Purpose: The Inventory Management Frontend provides an interface for managing products, sales, and carts. It enables users to add products, track inventory, manage sales records, and handle cart operations.  
       
     Features:  
     - Product management (Add and view products)  
     - Sales tracking with detailed records  
     - Cart functionality for product checkout  
     - Context API-based global state management  
     - Responsive design with Tailwind CSS
   * **Features**: Highlight the key features and functionalities of the frontend.
3. **Architecture**
   * Component Structure:  
     - Components: AddProduct, NavBar, Cart, CartItem, SaleRecord  
     - Layout component manages overall structure  
     - Components interact through contexts (InventoryContext, CartContext, SalesContext)  
       
     State Management:  
     - Context API used for global state (inventory, cart, and sales)  
       
     Routing:  
     - Basic routing handled in App.js (React Router can be extended if needed)
   * **State Management**: Describe the state management approach used (e.g., Context API, Redux).
   * **Routing**: Explain the routing structure if using react-router or another routing library.
4. **Setup Instructions**
   * Prerequisites:  
     - Node.js and npm installed  
       
     Installation:  
     1. Clone the repository  
      git clone <repo-url>  
     2. Navigate to project directory  
      cd inventory-management-frontend  
     3. Install dependencies  
      npm install  
     4. Start the development server  
      npm start
   * **Installation**: Provide a step-by-step guide to clone the repository, install dependencies, and configure environment variables.
5. **Folder Structure**
   * Client (React App):  
     - /public → Static assets (logos, manifest, icons)  
     - /src  
      • /components → UI components (Cart, Sales, AddProduct, NavBar)  
      • /context → Context providers (InventoryContext, CartContext, SalesContext)  
      • App.js, Layout.jsx → Main layout and app entry  
      • index.js → React entry point  
      • setupTests.js, App.test.js → Testing configuration
   * **Utilities**: Explain any helper functions, utility classes, or custom hooks used in the project.
6. **Running the Application**
   * Provide commands to start the frontend server locally.
     + Run the application with:  
         
       npm start
7. **Component Documentation**
   * Key Components:  
     - AddProduct: Form for adding products to inventory  
     - Cart: Displays products added to the cart  
     - CartItem: Individual cart item with controls  
     - SaleRecord: Displays details of a sale transaction  
     - NavBar: Navigation bar across pages  
       
     Reusable Components:  
     - CartItem (used in Cart)
   * **Reusable Components**: Detail any reusable components and their configurations.
8. **State Management**
   * Global State:  
     - Managed via Context API (Inventory, Cart, Sales)  
       
     Local State:  
     - Managed with React useState hooks inside functional components
   * **Local State**: Explain the handling of local states within components.
9. **User Interface**
   * UI is styled with Tailwind CSS, ensuring responsiveness. Includes forms, tables, and navigation UI.
10. **Styling**

* CSS Framework:  
  - Tailwind CSS  
    
  Theming:  
  - Custom color palette defined in tailwind.config.js
* **Theming**: Explain if theming or custom design systems are implemented.

1. **Testing**

* Testing Strategy:  
  - Jest and React Testing Library used for unit and integration tests  
    
  Code Coverage:  
  - setupTests.js configures testing environment  
  - App.test.js includes sample tests
* **Code Coverage**: Explain any tools or techniques used for ensuring adequate test coverage.

1. **Screenshots or Demo**

* UI is styled with Tailwind CSS, ensuring responsiveness. Includes forms, tables, and navigation UI.

1. No major known issues at present. Future testing may reveal optimizations for large inventories.

* Document any known bugs or issues that users or developers should be aware of.

1. - Add authentication and user roles  
   - Enhance reporting with charts and analytics  
   - Implement search and filtering in inventory  
   - Integrate with backend APIs for persistent storage

* Outline potential future features or improvements, such as new components, animations, or enhanced styling.