**Frontend Development with React.js**

1. **Introduction**

* **Project Title:**Store Manager: Keep Track of Inventory
* **Team ID: NM2025TMID40705**
* **Team Leader**: YUVAN RAJ .M/yuvanraj2024@gmail.com
* **Team Member:**
* **VIGNESH**
* **SURIYAPRAKESH.M**
* **THOUSEEF**

1. **Project Overview**

* **Purpose:** Store Inventory Manager enables store managers to efficiently track inventory levels, monitor sales, manage restocking, and generate reports to optimize stock management and reduce losses.
* **Features:**
* Inventory Management
* Stock Updates
* Checkout at Cart
* Products can be added to cart
* Adding New Products to Inventory
* Alert View for Depleting Stock
* Search Functionality for Products
* All sale records are stored with sale value

1. **Architecture**

* **Frontend:**React.js with Tailwind CSS for responsive user interfaces.
* **Backend:**None (client-side only application).
* **Database:**Browser LocalStorage to store inventory data, cart items, and sales records.

1. **Setup Instructions**

* **Prerequisites:**
* Node.js and npm
* Git
* React.js
* Visual Studio Code
* **Installation Steps:**

Clone the repository

* git clone <repository-url></repository-url>

### Install dependencies

* cd inventory-management-frontend
* npm install
* npm start

### 5. Folder Structure

Store-Inventory-Manager/

|-- client/ # React frontend |

|-- components/ |

|-- AddProduct.jsx | |

|-- Cart/ | |

|-- Cart.jsx | |

|-- CartItem.jsx | |

|-- Inventory/ | |

|-- Inventory.jsx | |

|-- Product.jsx |

|-- NavBar.jsx | |

|-- Product/ | |

|-- Product.jsx | |

|-- ProductCatalog.jsx | |

|-- ProductList.jsx |

|-- Sales/ | |

|-- Sales.jsx | |

|-- SaleRecord.jsx |

|-- context/ | |

|-- CartContext.js | |

|-- InventoryContext.js | |

|-- SalesContext.js | |

-- hooks/ | |

|-- useLocalStorage.js |

|-- pages/ | |

|-- NotFound.jsx |

|-- App.css |

|-- App.js |

|-- App.test.js |

|-- index.css |

|-- index.js |

|-- reportWebVitals.js |

|-- setupTests.js |

|-- .gitignore |

|-- package.json |

|-- package-lock.json |

|-- README.md |

|-- tailwind.config.js

|-- server/ # Node.js backend (placeholder, not implemented)

### 6. Running the Application

* **Frontend:** cd client npm start
* **Backend:** N/A
* **Access:** Visit <http://localhost:3000>

### 7. API Documentation

* N/A (no backend APIs; all logic is client-side)

### 8. Authentication

* None (no user authentication implemented)

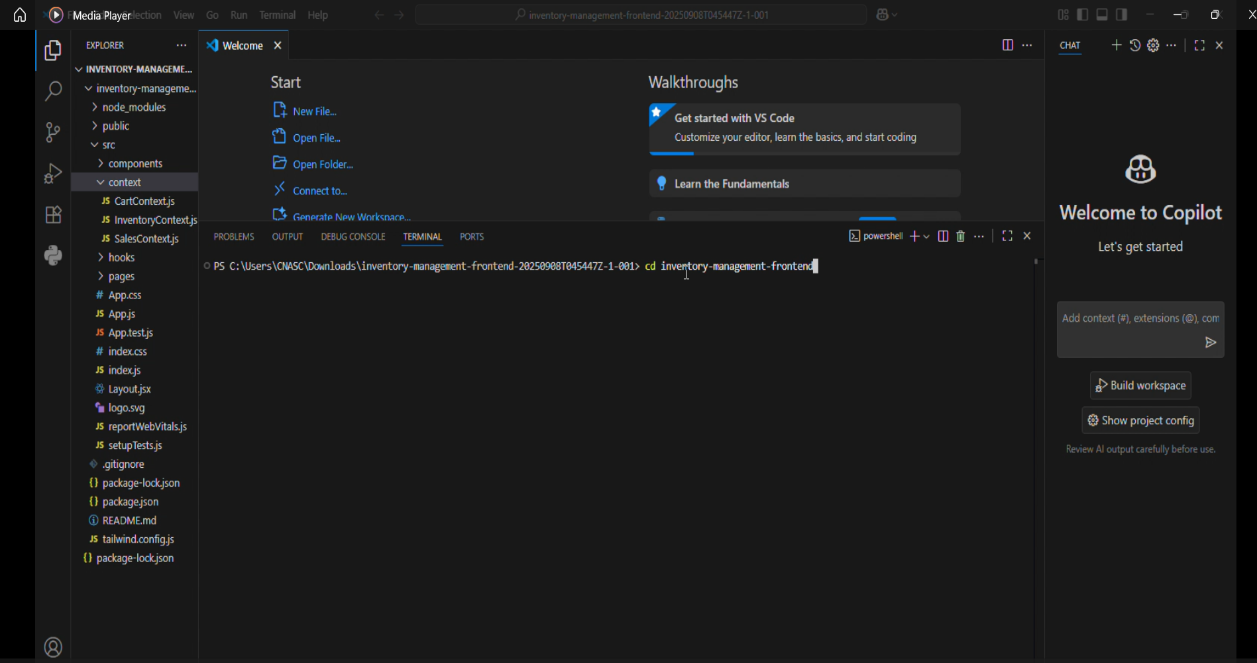
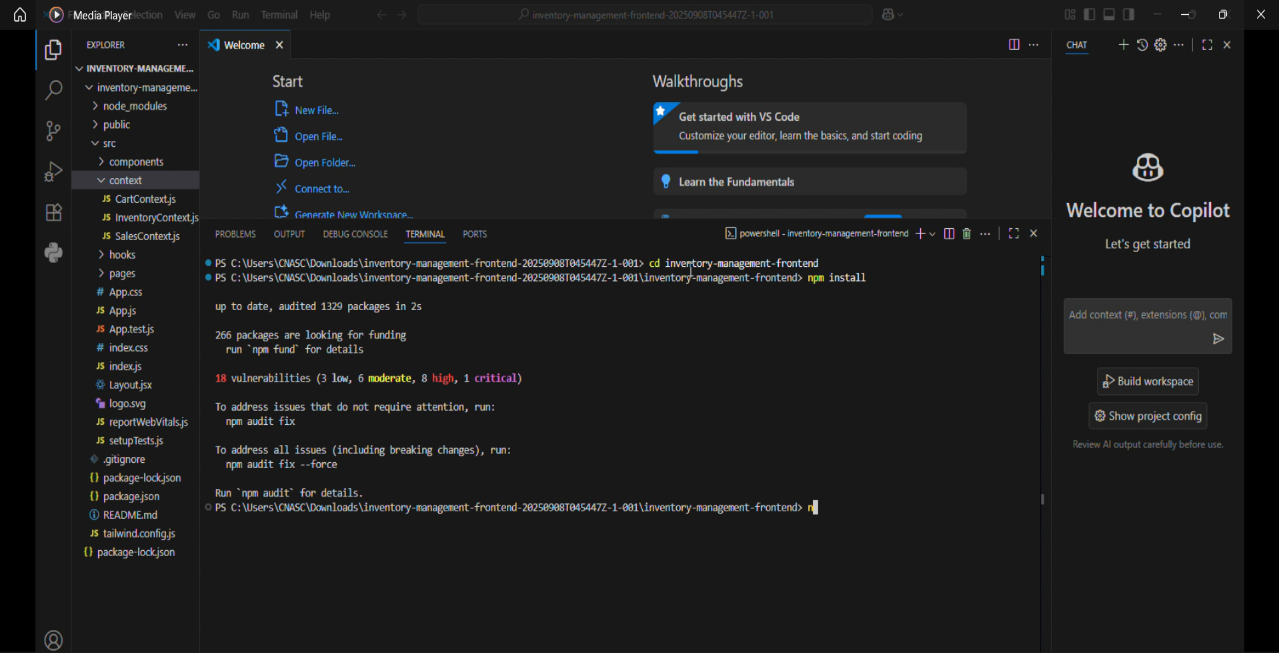
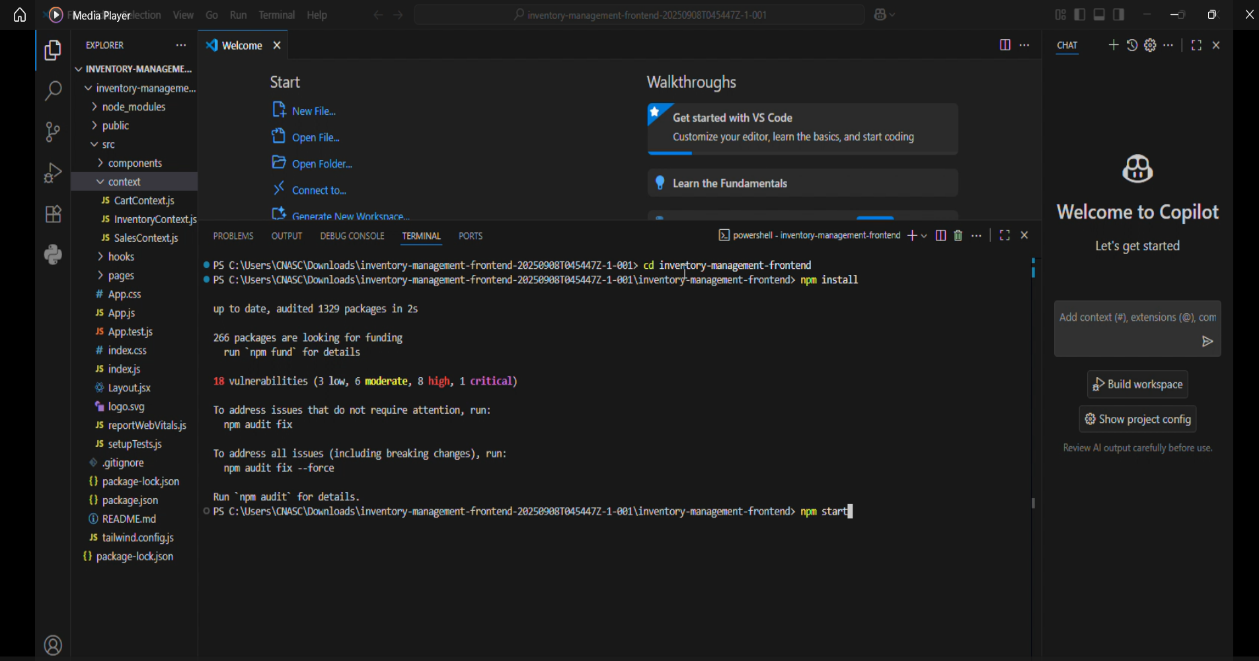
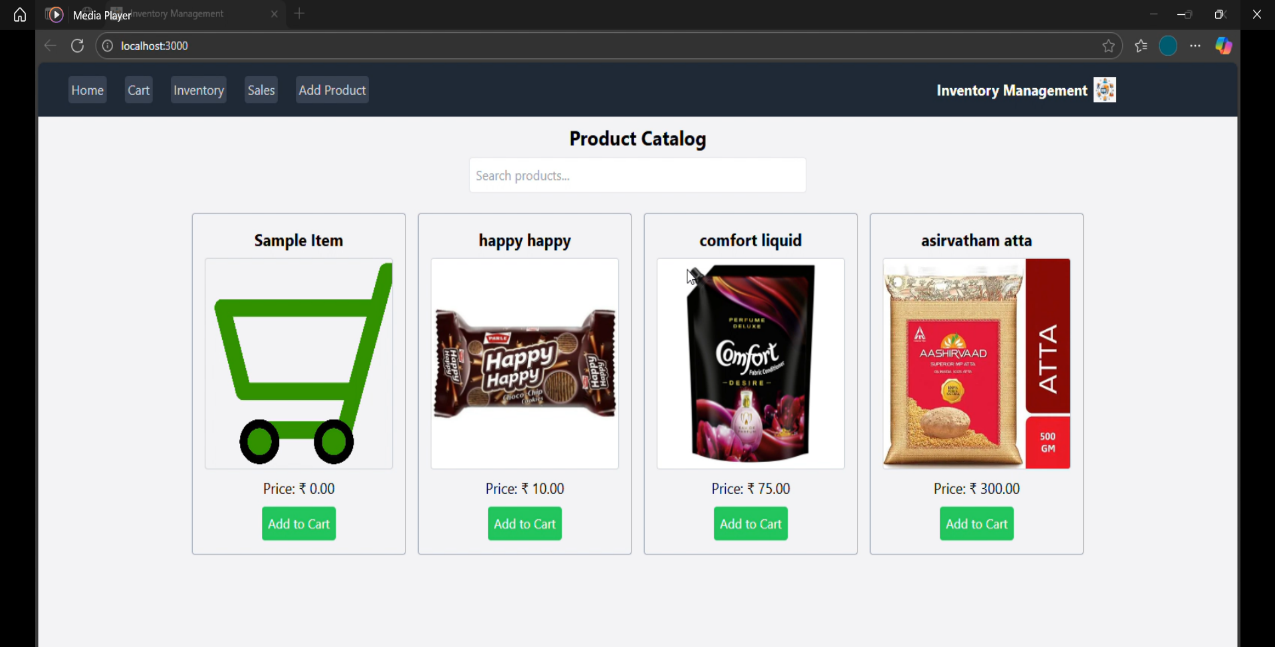
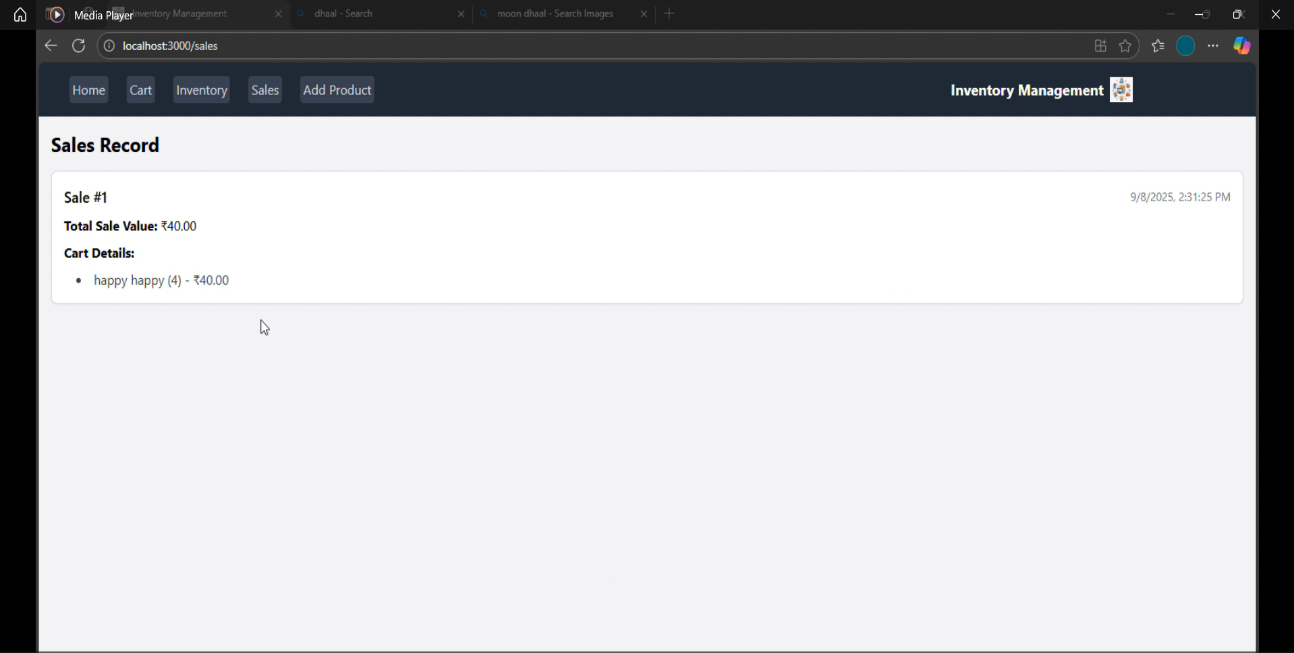
9. **User Interface**

* Product Catalog (Landing Page)
* Add Product Page
* Inventory View
* Cart Page
* Sales Records Page

### 10. Testing

* Unit testing with Jest and React Testing Library
* Manual testing during development
* Tools: Chrome Dev Tools

**11. Demo link**

* [Click](https://drive.google.com/file/d/1_edilwPwaOcH3DpbvTvPeSmDE_jNiZuq/view?usp=drivesdk) for the demo video.  
  
* 
* 
* 
* 

### 12. Known Issues

* **LocalStorage Dependency:** The application relies on browser LocalStorage for data persistence, which may lead to data loss if the browser cache is cleared or if storage limits are exceeded.
* **No Backend Integration:** The absence of a backend server limits scalability, data security, and multi-user support, as all data is stored client-side.
* **Duplicate Product Components:** The presence of Product.jsx in both components/Inventory/ and components/Product/ may cause confusion in maintenance and could lead to code redundancy.
* **Limited Error Handling:** The application lacks robust error handling for edge cases, such as invalid input in the Add Product form or cart quantity updates.
* **No Authentication:** Without user authentication, the application cannot restrict access or manage user-specific data, limiting its use in multi-user environments.

### 13. Future Enhancements

* **Backend Integration:** Implement a Node.js/Express.js backend with MongoDB to enable server-side data storage, multi-user support, and secure data persistence.
* **Authentication System:** Add JWT-based authentication to support user roles (e.g., admin, manager) and secure access to inventory and sales data.
* **Enhanced Error Handling:** Introduce comprehensive error handling for user inputs and state management to improve reliability and user experience.
* **Search and Filter Improvements:** Enhance the Product Catalog with advanced search and filter options (e.g., by category, price, or stock level) to improve usability.
* **Real-Time Notifications:** Implement real-time low-stock alerts and restock notifications using WebSocket or polling mechanisms.
* **Consolidate Product Components:** Refactor Product.jsx to eliminate duplication and centralize product-related logic for better maintainability.
* **Mobile Optimization:** Optimize the UI for mobile devices by enhancing responsive design and touch interactions using Tailwind CSS.
* **Automated Testing:** Expand test coverage with additional unit and integration tests for critical components like CartContext.js, InventoryContext.js, and SalesContext.js to ensure robustness.