Frontend development with react.js
Project documentation format

Introduction
Project Title: Inventory Management System
Team Members:
1.Kaviya. M as team leader
2.Karen Josette as code executer
3.Merlin as documentation
4.Kiruthiga.S as demo video creator
Project Overview
Purpose:
The purpose of this project is to design and implement an Inventory Management System to manage stock levels, sales, and product information efficiently. It helps businesses streamline operations, reduce costs, and improve customer satisfaction.
Features:
* Product catalog with SKU, name, price, and category.
* Add, update, and delete inventory items.
* Cart and sales simulation.
* Search and filter functionalities.
* Persistent storage using local or backend database.
Architecture
Component Structure:
Header: * Navigation bar with links (Home, Cart, Inventory, Sales, Add Product).
Home Page: * Overview and introduction.
Inventory Page: * Displays product list with actions (Add, Edit, Remove).
Add Product Page: * Form to add or update products.
Cart/Sales Page: * Handles sales simulation and checkout process.
State Management:
* Local state is used for form handling.
* Global state (via Context API/Redux) manages inventory and cart data.
Routing:
* React Router used for page navigation between Home, Inventory, Sales, Cart, and Add Product.
Setup Instructions
Prerequisites:
* Node.js
* npm or yarn
Installation:
1. Clone the repository.
2. Navigate to the project directory.
3. Run npm install to install dependencies.
4. Configure environment variables if needed.
Folder Structure
Client:

components/* – Reusable UI components.
pages/* – Page-level components (Home, Inventory, Cart, Add Product).
assets/* – Images, logos, and styles.
context/* – State management (inventory, cart).
Utilities:

* Helper functions for price calculation, filtering, and validation.
* Custom hooks for local storage persistence.

Running the Application

Run the following command in the client directory:
bash

```
npm start
```

Component Documentation
Key Components:
Inventory Table: * Displays product list and actions.
Product Form: * Form for adding and editing products.
Cart: * Displays items in the cart and handles checkout.
Header: * Navigation bar component.
Reusable Components:
Button: * Customizable button component.
Modal: * Reusable modal for confirmations.
Input Field: * Reusable input with validation.
State Management
Global State:

* Inventory data, cart items, and sales records.

Local State:

* Input form fields and modal visibility.

User Interface
The UI is designed to be clean and responsive.

* Home page displays an overview and images (logo, banner).
* Inventory page shows a product list with management options.
* Cart page simulates checkout.

Styling
CSS Frameworks/Libraries:

* Tailwind CSS
* Shaden/Ui for UI components

Theming:

* Custom theme with primary and secondary colors.

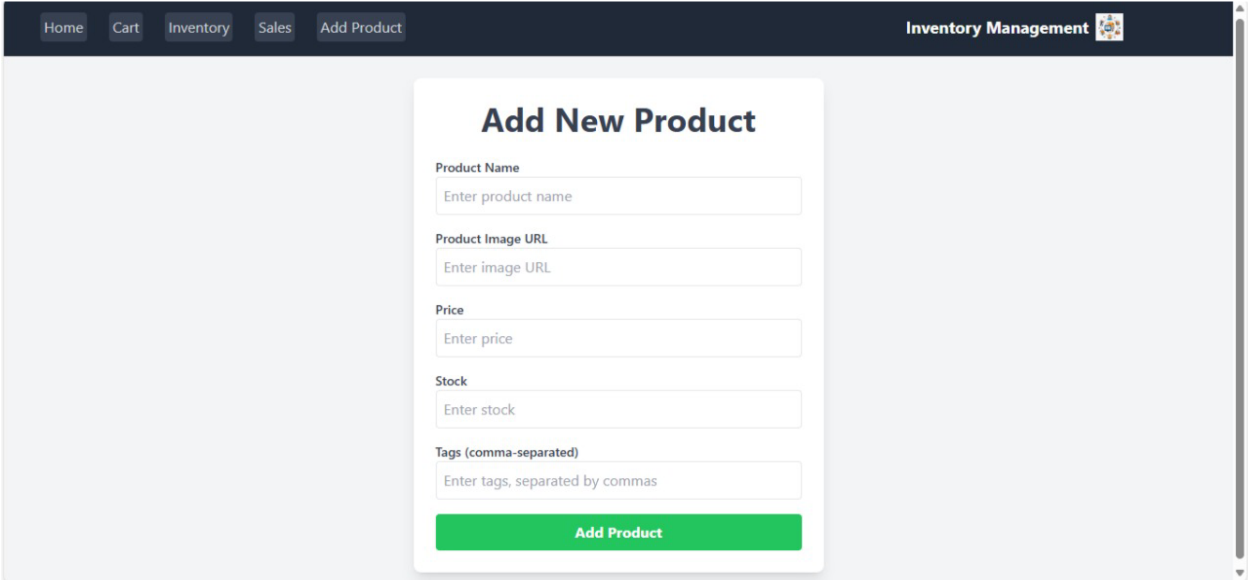
Testing
Testing Strategy:

* Unit testing with Jest.
* Component testing with React Testing Library.
* End-to-end testing with Cypress.

Code Coverage:

* Jest coverage reports for unit and integration tests.

Demo images



Add New Product

Product Name

Enter product name

Product Image URL

Enter image URL

Price

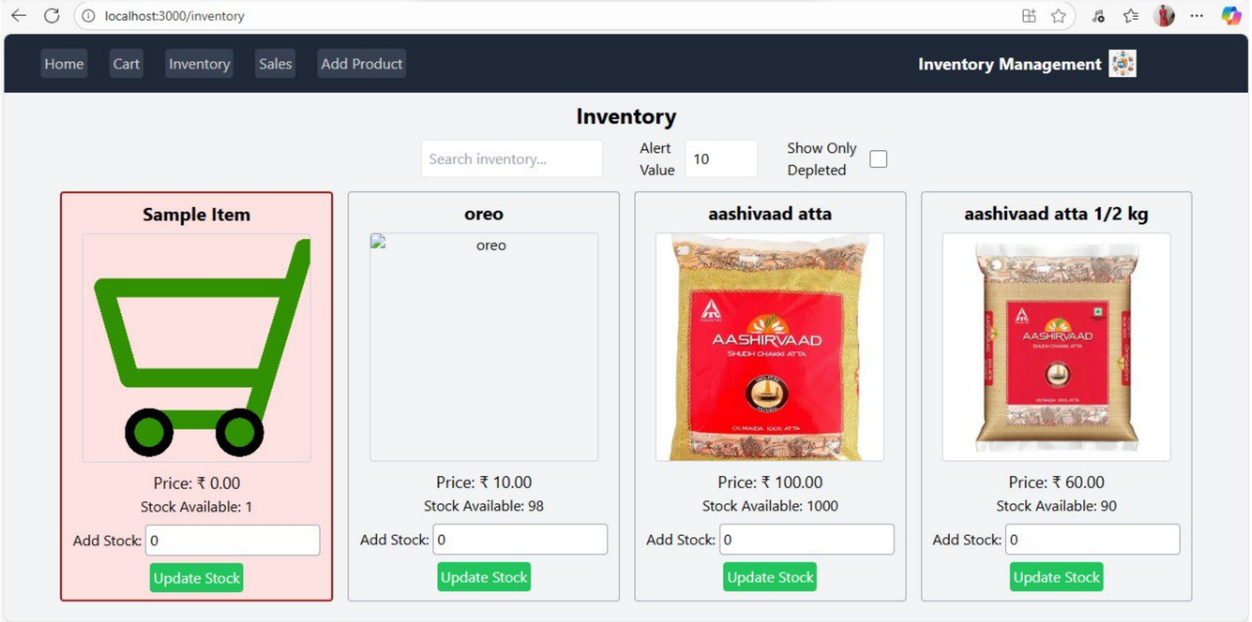
Enter price

Stock

Enter stock

Tags (comma-separated)

Enter tags, separated by commas



Inventory

Search inventory...

Alert Value 10

Show Only Depleted □

Sample Item	oreo	aashivaad atta	aashivaad atta 1/2 kg
Price: ₹ 0.00	Price: ₹ 10.00	Price: ₹ 100.00	Price: ₹ 60.00
Stock Available: 1	Stock Available: 98	Stock Available: 1000	Stock Available: 90
Add Stock: 0	Add Stock: 0	Add Stock: 0	Add Stock: 0

Known Issues

* Limited backend integration in prototype mode.
* Cart state resets on full page refresh without backend.

Future Enhancements

* Integration with a real backend API.
* Role-based access (admin, staff).
* Advanced reporting and analytics.
* Barcode scanning support.