1.Introduction :

Project Title : store manager : keep track of inventory

Team Members:

1. M.KAMASRI

2. R.ASWIN

3. R.ABIRAMI

4. K.BHUVANESHWARI

5. M.BOOMIKA

Team Members Roles :

1. M.KAMASRI – Coding

2. R.ASWIN – Coding Development

3. R.ABIRAMI- Document

4. K.BHUVANESHWARI –Document

5. M.BOOMIKA – Demo Video

2.Project Overview:

Purpose:

Store Manager is a React-based frontend application designed to help store owners easily manage and track their inventory. The platform allows adding, editing, and removing items, monitoring stock levels, and visualizing product data in a clean dashboard.

Key Features

Add/edit/delete inventory items

Real-time stock updates

Product search and filtering

Category-based product organization

Dashboard with inventory statistics

Responsive design

3.Architecture:

Component Structure

src/

├── components/

│ ├── InventoryList/

│ ├── InventoryItem/

│ ├── Header/

│ ├── Footer/

│ └── Modal/

├── pages/

│ ├── Dashboard.jsx

│ ├── Inventory.jsx

│ └── NotFound.jsx

Dashboard: Main overview of statistics and quick actions

InventoryList: Displays all items

InventoryItem: Displays individual item details

Modal: Reusable for forms (add/edit item)

State Management

Global State: Managed using Context API with useReducer for inventory data

Local State: Managed via useState for form inputs, modal visibility, etc.

Routing

Using React Router v6 for page navigation:

<Route path="/" element={<Dashboard />} />

<Route path="/inventory" element={<Inventory />} />

<Route path="\*" element={<NotFound />} />

4. Setup Instructions:

✅ Prerequisites

Node.js >= 18.x

npm or yarn

Git

Installation

git clone https://github.com/yourusername/store-manager.git

cd store-manager

npm install

Environment Variables

Create a .env file in the root if needed:

REACT\_APP\_API\_URL=http://localhost:4000/api

5. Folder Structure:

Client (React App)

src/

├── assets/ # Static files (images, logos)

├── components/ # Reusable and feature-specific components

├── context/ # Global state using Context API

├── hooks/ # Custom hooks (e.g., useInventory)

├── pages/ # Route-level components

├── utils/ # Helper functions (e.g., formatDate)

├── App.jsx

└── index.js

6. Running the Application:

To start the development server:

npm start

This runs the app in development mode on http://localhost:300

7. Component Documentation:

Key Components

Component Description Props

InventoryList Renders a list of inventory items items, onDelete, onEditInventoryItem Represents an individual item item, onEdit, onDeleteModal Reusable modal for forms isOpen, onClose, childrenDashboardCard Shows stats like total items title, value, icon

Reusable Components

Modal

Button

Input

Dropdown

8.State Management:

Global State

Inventory data is stored globally using Context API and useReducer.

Example:

const [state, dispatch] = useReducer(inventoryReducer, initialState);

Local State

Used for form fields, modal toggle, loading states:

const [modalOpen, setModalOpen] = useState(false);

9.User Interface:

Screenshots or GIFs Here

Inventory Page

Add/Edit Modal

Dashboard with Summary Cards

Mobile Responsive View

10. Styling:

CSS Frameworks/Libraries

Tailwind CSS for utility-first styling

Heroicons for iconography

Theming

Custom color palette defined in tailwind.config.js

Light/Dark mode (optional toggle implemented)

11. Testing:

Testing Strategy

Unit Testing: Components tested with Jest and React Testing Library

Integration Testing: Simulated user flows (e.g., adding an item)

E2E Testing (optional): Planned using Cypress

Code Coverage

Run coverage report:

npm test -- --coverage

12.Screenshots or Demo:

Live Demo: <https://your-demo-link.com>

13. Known Issues:

No backend authentication implemented yet

Search bar filters only by name (not by category or stock status)

Limited accessibility testing

14. Future Enhancements:

User authentication & role-based access

Improved filtering & sorting

Export inventory to CSV

Integration with barcode scanner

PWA support