**Project Documentation**

**1.Introduction**

**Project title**: Store Manager-Keep Track of Inventory

**Team ID**: NM2025TMID39982

**Team Leader**:

Name: Swedha D

**Mail ID**: [sac2427bca2892@ssacollegechennai.com](mailto:sac2427bca2892@ssacollegechennai.com)

**Team Members**:

1.Name: Sruthika D

Mail ID: [sac2427bca5329@ssaccollegechennai.com](mailto:sac2427bca5329@ssaccollegechennai.com)

2. Name: Sharmila S

Mail ID: [sac2427bca5284@ssaccollegechennai.com](mailto:sac2427bca5284@ssaccollegechennai.com)

3.Name: Supriya R

Mail ID: [sac2427bca5283@ssacollegechennai.com](mailto:sac2427bca5283@ssacollegechennai.com)

**2.Project Overview**

**Purpose:**

The goal of Store Manager is to help businesses keep track of their product inventory efficiently. The application simplifies stock management by allowing users to add, edit, and remove items, while also providing insights on availability and low-stock alerts.

**Features:**

* Home, sales, inventory, and add product.
* Add, update, and delete products.
* Search and filter inventory by category, name, or stock status.
* Low-stock alerts and notifications.
* Dashboard showing stock summary.
* Responsive and user-friendly interface.

**3.Architecture:**

**Component Structure**:

* Dashboard Component - Displays inventory summary and alerts.
* Product List Component - Lists all
* products in stock with filtering and sorting options.
* Product Form Component Handles adding and editing products.
* Search Bar Component - Provides product search functionality.
* Notification Component - Displays alerts for low-stock or out-of-stock items.

**State Management:**

* **Context API** is used for global state management (managing product list, stock levels, alerts).
* Local state within components for handling form inputs.

**Routing:**

* **React Router** used for navigation between pages:
* / - Dashboard
* /Products - Product List
* /Add -Add Product Form
* /edit/:id -Edit Product Form

**4.Setup Instructions**

**Prerequisites:**

* Node.js
* npm. or yarn package manager

**Installation:**

* **Clone the repository:**

git clone

<https://github.com/zo5NuNTr/inventory.git>

/store manager

cd inventory

1. **Install dependencies:**

npm install

1. **Configure environment variables (if needed).**
2. **Start the development server:**

npm start

**5.Folder Structure**

store manager/

│── public/

│── sac/

│ ├── components/

│ │ ├── Dashboard’s

│ │ ├── ProductList.jsx

│ │ ├── ProductForm.jsx

│ │ ├── SearchBar.jsx

│ │ └── Notification’s

│ ├── context/

│ │ └── InventoryContext.js

│ ├── pages/

│ │ ├── Home.jsx

│ │ ├── Products.jsx

│ │ └── EditProduct.jsx

│ ├── utils/

│ │ └── helpers.js

│ ├── App.js

│ └── index.js

│── package. json

**6.Running the Application**

**Frontend:**

**cd inventory**

**npm install**

npm start

**7.Component Documentation**

**Key Components**:

* **Dashboard**: Displays stock summary and alerts.
* **Product List**: Lists products with search and filter options.
* **Product Form**: Adds or edits product details.
* **Notification**: Alerts users about low stock.

**Reusable Components**:

* **Button Component** – Styled button for consistency**.**
* **Modal Component** – Used for confirmation dialogs.

**8.Authentication**:

* JWT-based authentication for secure login.
* Middleware protects private routes.

**9.State Management**

**Global State:**

Managed with Context API to store inventory data, handle product updates, and generate alerts.

**Local State**:

Used in forms to manage temporary input values before saving to global state.

**10.User Interface**

* Dashboard showing total products, low-stock items, and out-of-stock count.
* Product list with search, filter, and pagination.
* Forms for adding/editing products with validation.

**11.Styling**

* **CSS Frameworks/Libraries:** Tailwind CSS for styling.
* **Theming:** Consistent color scheme with light and dark modes.

**12.Testing**

**Testing Strategy:**

* Unit testing for components using **Jest.**
* Integration testing for state management and routing.
* End-to-end testing using **React Testing Library**.

**Code Coverage:**

* Jest coverage reports generated for all components.

**13.Screenshots or Demo**

**Home Page**

**Sales Page**

**No sales record, yet no product has been added**

**Add Product**

**Step 1: Go to chrome**

**Step 2: Type the product name, choose the desired product**

**Choose open image in new tab**

**Copy the image address, In the product image url**

**Choose copy image address**

**Go to home page, and check whether the product has been added**

**With the same steps, I have added four more products**

**Adding product to the cart**

**The product has been added to the cart successfully**

**Sales record has been added successfully**

**Since, the product has been added successfully**

**14.Known Issues**

* No backend integration yet (data reset on refresh).
* Limited product filtering options.

**15.** **Future Enhancements**

* Integration with backend API and database.
* Barcode scanner support for adding products.
* Export inventory data to CSV/PDF.
* Role-based authentication (Admin, Staff).
* Mobile app version.