Project Documentation

StoreManagement

1. Introduction

• \*\*Project Title: Store Management

• \*\*Team ID:

• \*\*Team Leader: K.hemamalini -mgc7hemamalini @gmail.com

• \*\*Team Members and their role:\*\*

Hemamalini.K.Wesite creator

Vanitha.D-Document creator

Hemalatha.P- Voiceover

Sanjitha.S.- Web designer

2. Project Overview

The Store Management System is a web-based application developed as part of the Naan Mudhalvan initiative. The primary objective of this project is to provide a simple, efficient, and user-friendly solution for managing products, sales, and stock in small to medium-scale businesses.

Traditional St management methods, such as manual registers or spreadsheets, are often error-prone, time-consuming, and inefficient. This system addresses those challenges by automating the process of tracking inventory, managing product details, updating stock, recording sales, and generating alerts when items are running low.

The application is built using React.js for the frontend and styled with TailwindCSS for responsive and modern UI design. State management is handled using React Context API and Reducers, with data persistence provided through localStorage to ensure that the data remains available even after a page refresh.

Key modules of the system include:

Product Catalog – Displays all products with details like name, price, and available stock.

Cart Management – Allows users to add, remove, and update quantities of products before checkout.

StoreManagement – Enables administrators to add new products, update stock, and monitor low-stock alerts.

Sales Tracking – Maintains a record of completed sales with date, products sold, and total transaction value.

Search & Alerts – Provides quick search functionality and notifies users when stock falls below a specified level.

This project not only enhances efficiency in day-to-day Storeoperations but also reduces the chances of human error, thereby ensuring better decision-making for business owners. It is designed to be scalable and can be further enhanced to integrate advanced features such as barcode scanning, cloud-based storage, and detailed analytics dashboards.

3. Architecture

Project Architecture

The architecture of the StoreManagement System is designed to provide a modular, scalable, and user-friendly application. It follows a frontend-centric model and data persistence via localStorage.

Architecture Overview

Frontend: React.js with TailwindCSS for building a responsive and interactive user interface.

Persistence Layer: LocalStorage is used to store inventory, cart, and sales data, ensuring persistence across sessions.

Version Control: Git & GitHub for source code management and collaboration.

Layered Breakdown

1. Frontend (UI Layer)

Built using React.js components (Inventory, Cart, Sales, Product, etc.).

Styled with TailwindCSS for responsiveness and clean design.

Handles user interactions (adding to cart, updating stock, searching products).

1. Application Logic (State Management Layer)

Context + Reducers manage global states:

StoreContext → Products & Stock.

Cart Context → Items in cart & checkout process.

Sales Context → History of sales records.

LocalStorage Integration ensures data persists even after page reloads.

1. Data Layer

LocalStorage acts as the primary storage for:

StoreData (products, stock levels).

Cart Data (items, quantities, totals).

Sales Records (transaction history).

Technology Stack

Frontend: React.js, TailwindCSS

Persistence: LocalStorage

Version Control: Git, GitHub

4.Setup Instructions

Prerequisites

Make sure the following tools are installed on your system before setting up the project:

Node.js & npm – JavaScript runtime and package manager.

Download Node.js

Git – Version control system.

Download Git

React.js – JavaScript library for building UI.

TailwindCSS – For styling and responsiveness.

Code Editor – Visual Studio Code (recommended).

Download VS Code

Installation Steps

* Open the folder and change directory

Cd inventory-management

* Install Dependencies

Npm install

* Run the Development Server

Npm start

The app will start at: <http://localhost:3000>

* Project Structure (basic)

Src/

├── components/ # Reusable UI components

├── context/ # Context API files for Inventory, Cart, Sales

├── pages/ # Page components (Inventory, Cart, Sales, AddProduct)

├── hooks/ # Custom hooks (if any)

├── App.js # Main application file

└── index.js # Entry point

✅ After completing these steps, your StoreManagement System will be running locally.

5.Folder Structure

6. Running the Application

Once the setup and installation are complete, you can run the application locally using the following steps:

Frontend

Cd inventory-management

Npm start

This will start the React development server.

By default, the app runs at: <http://localhost:3000>

Open your browser and visit:

👉 <http://localhost:3000>

You should see the application homepage with the StoreDashboard.

7. User Interface

•Home page:

•Add product page:

• Storepage:

* Cart page:
* Sales page:

The StoreManagement System provides the following key features:

1. StoreManagement

Maintain a catalog of products with details such as name, price, image, tags, and available stock.

Update stock levels automatically after each sale or manually when new stock is added.

1. Product Catalog

Display all available products with their details.

Search functionality to quickly find products by name.

1. Cart Functionality

Add products to the cart for sales transactions.

Update quantities (increment/decrement) or remove items from the cart.

Display total value of items in the cart.

1. Checkout Process

On checkout, stock levels are updated automatically.

The cart is cleared, and a sales record is created.

1. Sales Records

Maintain a detailed history of all sales with date, items sold, and total value.

Display sales records in a clean, sorted layout (latest to oldest).

1. Add New Products

Easily add new products to the Storeby providing product details (name, price, stock, tags, and image URL).

1. Depleting Stock Alerts

Highlight products with low stock in red for quick identification.

Alert threshold can be customized based on requirements.

1. Responsive Design & Styling

Built with TailwindCSS for a modern and mobile-friendly interface.

Ensures a consistent look across devices.

1. Data Persistence

Uses localStorage to persist inventory, cart, and sales data across sessions.

1. Error Handling & Validation

Prevents negative stock updates.

Restricts checkout when the cart is empty.

Validates input fields when adding or updating products.

Thank you...!