**STORE MANAGER: KEEP TRACK OF INVENTORY PROJECT DOCUMENTATION**

# 1. Introduction

1. **Project Title**: S**tore Manager: Keep track of Inventory**
2. **Team Members:**

Project Title : S**tore Manager: Keep track of Inventory**

Team ID : NM2025TMID41374

TREM LEADER : BHARATH B (bharathsarvin35@gmail.com)

ROLE : CODING AND DEVALOPMENT

TEAM MEMBER : SAMPATHSANKAR R (sampathsankar26@gmail.com)

ROLE : CODING AND DEVALOPMENT

TEAM MEMBER : SANJAY K (ssanjay28674@gmail.com)

ROLE : DEMO VIDEO

TEAM MEMBER : MATHANSUNDHAR K(mathanrockey32@gmail.com)

ROLE : DOCUMENT CREATER

# 2. Project Overview

## Purpose

* The purpose of this project is to create a simple web-based Inventory Management System that helps manage products, track inventory, and record sales.

## Features

* Home Page (store.html) to view and Purchase Products.
* Add new products (add.html)
* Manage and view stock list (inventory.html)
* Record sales Details (sales.html)
* Custom CSS for user interface styling.
* JavaScript for dynamic functionality and local-Storage handling

# 3. Architecture

## Component Structure

The system consists of four main HTML pages:

1. **store.html**–Display all products for browsing and purchase.
2. **cart.html** – Shows products added to the shopping cart and allows checkout
3. **add.html** - Page for adding new products to inventory.
4. **inventory.html** - Displays current stock levels of the Products.
5. **sales.html** - Used for recording sales.

## State Management

* Data flow is handled via simple JavaScript and localStorage to persist items temporarily.

## Routing

* Navigation between pages is achieved via HTML links. No advanced routing library is used.

# 4. Setup Instructions

## Prerequisites

1. A modern web browser (Chrome, Firefox, Edge)
2. Visual Studio Code (VS Code)
3. VS Code Live Server extension installed

## Installation

1. Clone the repository from GitHub:  
git clone https://github.com/sathyaraj97/Store-Manager-Keep-Track-of-Inventory-NM2025TMID47063.git  
2. Open the project folder in VS Code.  
3. Install the Live Server extension in VS Code.  
4. Right-click on store.html and choose 'Open with Live Server'.  
5. The application will run locally in your browser.

* Download the project folder.
* Ensure all HTML, CSS, and JS files are in the same directory.
* Open store.html in the browser to start.

# 5. Folder Structure

* Project Folder:  
  - store.html (Products Catalog)

- cart.html (checkout-cart)  
- add.html (Add Products)  
- inventory.html (Inventory List)  
- sales.html (Sales Records)  
- assets/ (images, CSS, JS)

# 6. Running the Application

1. **Option 1 – Run via Live Server in VS Code -- (Recommended)**

* Open the project folder in **VS Code**.
* Right-click on **store.html** and select **“Open with Live Server”**.
* The application will run in your default browser. From the Home Page, you can navigate to **add.html**, **inventory.html**, or **sales.html** as required.

1. **Option 2 – Run directly in a Web Browser -- (Alternative)**

* Open the project folder on your computer.
* Double-click on **store.html** to open it in your default web browser.
* Navigate to other pages (**add.html**, **inventory.html**, **sales.html**) using the navigation bar.

# 7. Component Documentation

## store.html

* Provides navigation links to all other modules (cart.html, inventory.html, sales.html, and add.html).
* Displays the **Products Catalog** with a search feature (search-Box).
* Includes a **footer section** with app information, social links, and useful links.
* Uses store.css for styling and store.js for product loading and interactivity.

## cart.html

* Displays the list of products currently added to the shopping cart (cart-Container).
* Shows a **cart summary** (cart-Summary) with total items and pricing details.
* Provides two main actions:
  + **Checkout Cart** → Processes the order (via checkout-Cart() in cart.js).
  + **Clear Cart** → Empties the cart (via clear-Cart() in cart.js).
* Uses cart.css for styling the cart layout and buttons.
* Uses cart.js to manage cart operations (add, remove, checkout, and clear).
* Integrated with the **navbar** for navigation between Home, Cart, Inventory, Sales, and Add Product.

## add.html

1. Provides a **form** to input new product details including:

* Product Name
* Product Image URL
* Price
* Stock Quantity
* tags (comma-separated)

1. Uses **form validation** with required attributes to ensure all fields are filled.
2. Shows a **success message** (✅ Product added successfully!) after submission.
3. Uses add.css for styling and add.js for handling form submission, validation, and updating inventory data.
4. Integrated with navigation bar (links to Home, Cart, Inventory, Sales, Add Product).

## inventory.html

* Displays a **grid of products** (inventory-Grid) with current stock details.
* Provides **search functionality** (search-Input) to quickly find products.
* Includes **stock alert control** (alert-Value) to highlight items below a certain threshold.
* Has a **checkbox option** (show-Depleted) to filter and show only depleted/out-of-stock items.
* Uses inventory.css for styling and inventory.js for dynamic loading, filtering, and stock alert functionality.

## sales.html

* Displays a section (sales-Container) for showing all recorded sales.
* Provides a **centralized view of transactions**, allowing tracking of sold products.
* Updates inventory quantities after sales are recorded, ensuring stock is kept accurate.
* Uses sales.css for layout and styling.
* Uses sales.js to dynamically fetch, display, and manage sales data.
* Integrated with the **navbar** for navigation between Home, Cart, Inventory, Sales, and Add Product.

# 8. State Management

* The project uses JavaScript variables and local-Storage for temporary data storage.
* Actions performed on add.html (adding products)or cart.html (Check-out/removing items from the cart) automatically update and reflect changes onsales.html (recording sales), inventory.html(Stock Updated) and store.html.(indicated out of stock if the stock is Not Available).

# 9. User Interface

* The UI consists of HTML forms, tables, and navigation menus. It is styled with basic CSS for clarity and usability.

# 10. Styling

* The project uses custom CSS styling. Additional libraries or frameworks can be added in the future.

# 11. Testing

* Manual testing has been performed:
* Adding products and verifying they appear in Both inventory and Products catalog. (Using JS)
* Recording sales and ensuring stock is reduced. (Using JS)
* Navigation between pages works correctly.
* Sales Records are recorded with correct Date and Time, Quantity, Cart-Details, Each Product amount And Total Amount. (Using JS)

# 12. Screenshots or Demo

* Screenshots of the UI or Demo video will be added via a Google Drive Link.

# 13. Known Issues

* Data does not persist after browser cache is cleared.
* No authentication system implemented.
* Limited reporting capabilities.

# 14. Future Enhancements

* Add a backend with a real database for persistent storage.
* Implement user authentication and roles.
* Generate detailed reports and analytics.
* Improve UI with frameworks like Bootstrap or Tailwind CSS.
* Add search and filtering options in inventory.