# Inventory Management System

## Introduction

Project Title: Inventory Management System

Team Members & Roles:

* S. Samuel Jebaraj – Project Lead & JavaScript Developer
* S. Monish – UI/UX Designer
* J.P. Mowrish – HTML Developer
* P. Sakthi – Testing & Debugger
* R. Rajith – CSS Developer

## Project Overview

Purpose: The Inventory Management System is designed to manage products efficiently across different sections like inventory, sales, and store updates. It helps track stock levels, manage sales, and maintain consistency across all views.

Features:

* Add, update, and delete products.
* Real-time inventory tracking.
* Sales record management.
* Automatic reflection of changes between inventory.html, store.html, sales.html, and cart.html.
* Data stored in localStorage for persistence.

## Architecture

Component Structure (HTML/CSS/JS):

* inventory.html – Displays current stock items.
* store.html – Shows available items for purchase.
* sales.html – Records and tracks completed sales.
* cart.html – Manages customer purchase details.
* add.html – For adding new items to the inventory.

State Management: Uses JavaScript variables and localStorage for data persistence. Updates on one page reflect in related sections dynamically.

Routing: Handled through multiple static HTML files linked together.

## Setup Instructions

Prerequisites:

* Web browser (Chrome, Edge, or Firefox).
* Text editor (VS Code recommended).

Installation:

* Clone or download the project folder.
* Open index.html in a browser to start the application.

## Folder Structure

* /assets → CSS styles, images.
* /js → Contains JavaScript logic for state management and interactions.
* /pages → HTML files (inventory, store, add, sales, cart).

## Running the Application

Simply open index.html or inventory.html in a browser. No server is required since it’s a frontend-only system.

## Component Documentation

* Inventory Page – Displays available stock with CRUD options.
* Store Page – Lists items for customers.
* Sales Page – Records and shows sales transactions.
* Add Page – Input form to add new products.

## State Management

Global State: Managed through localStorage.

Local State: Handled via JavaScript variables inside individual HTML scripts.

## User Interface

Clean and responsive layout using CSS. Tables and forms for better data entry and viewing.

## Styling

Custom CSS for UI design. Responsive layouts for desktop and mobile.

## Testing

Manual testing of each page for add, update, delete operations.

Verified data consistency across inventory and sales pages.

## Known Issues

Data is stored only in browser localStorage (not in a database).

Refreshing clears temporary states unless stored.

## Future Enhancements

* Add backend integration (Node.js / Database).
* Implement user authentication.
* Generate sales and stock reports.
* Add search and filter functionality.