Store Manager - Project Documentation

# Introduction

Project Title: Store Manager

Team Members: Haresh(Captain), Hafeezur Rahman, Janarthanan, Karol

# Project Overview

Purpose: The Store Manager project is designed to manage store operations efficiently. It enables users to purchase items, track inventory, view sales information, manage a cart, and add new products.

Features:   
- Home Page  
- Cart  
- Inventory  
- Sales Information  
- Add Product

# Architecture

Component Structure: The application is built using React.js with multiple reusable components. Major components include Home, Cart, Inventory, Sales, and AddProduct.

State Management: State is managed using React's useState and Context API to ensure smooth data flow.

Routing: React Router is used for navigating between Home, Cart, Inventory, Sales, and Add Product pages.

# Setup Instructions

Prerequisites: Node.js and npm must be installed.  
Installation:  
1. Clone the repository.  
2. Navigate to the project folder.  
3. Run `npm install` to install dependencies.  
4. Run `npm start` to start the development server.

# Folder Structure

The project follows a standard React application structure:  
- public/  
- src/  
 - components/  
 - pages/  
 - assets/  
- package.json  
- tailwind.config.js

# Running the Application

To run the application locally:  
1. Open terminal in the project folder.  
2. Run `npm start`.

# Component Documentation

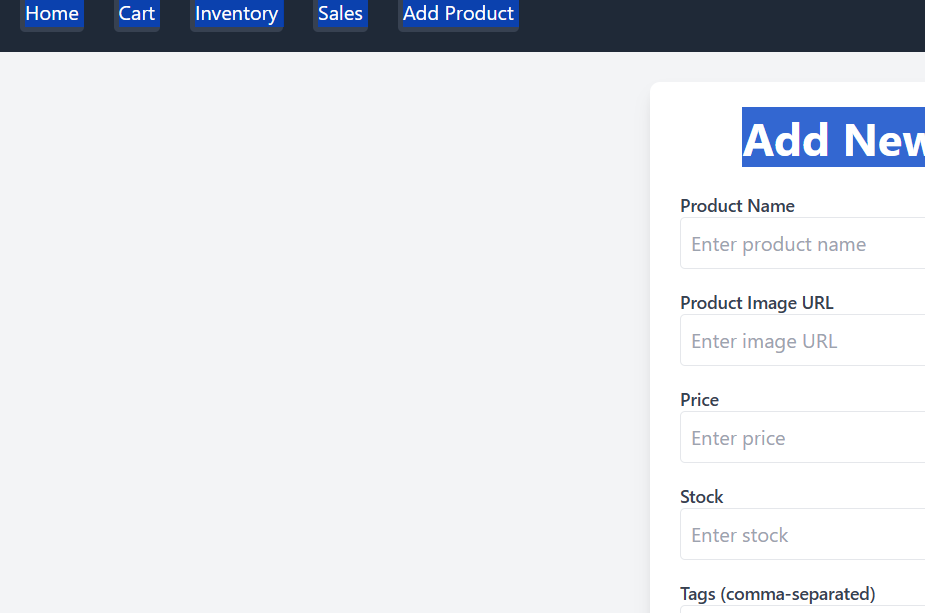
Key Components:  
- Home: Landing page with navigation.  
- Cart: Displays items added for purchase.  
- Inventory: Shows available stock.  
- Sales: Displays sales data.  
- AddProduct: Form to add new products.

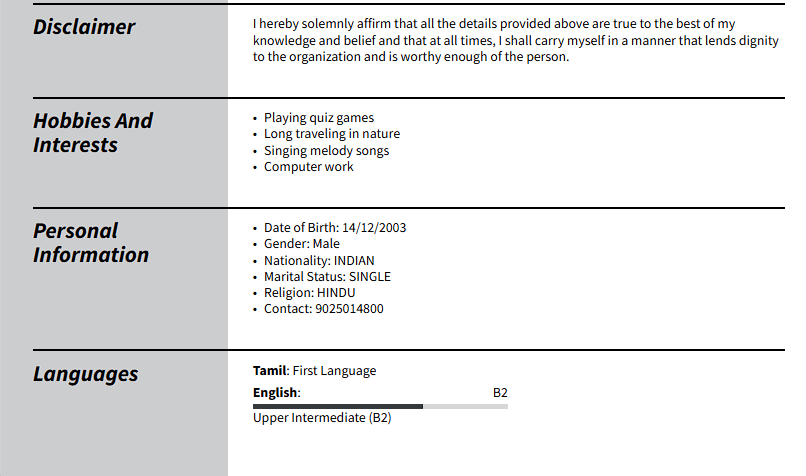
# State Management

Global State: Managed via Context API.  
Local State: Managed with useState hooks within components.

# User Interface

The UI is built with React components and styled using Tailwind CSS. Below are screenshots of the application:





# Styling

The project uses Tailwind CSS for styling, ensuring a responsive and modern UI.

# Testing

Testing Strategy: Components are tested manually. Future improvements may include unit testing with Jest and React Testing Library.  
Code Coverage: Not implemented in this version.

# Known Issues

- Limited error handling.  
- No backend integration for persistent data storage.

# Future Enhancements

- Implement authentication for users.  
- Add backend with database integration.  
- Improve reporting and analytics.  
- Enhance UI with animations and advanced styling.