STORE MANAGER: KEEP TRACK OF INVENTARY

Project Documentation

Project Title: Store Manager: Keep Track of Inventory

Team ID :NM2025TMID37147

Team Leader: Nasreen banu .S & nazvivo2006@gmail.com

Team Members: Nasreen fathima. A & nasreen12062006@gmail.com

Team Members: Swetha.Y & swethay2007@gmail.com

Team Members: Gayathri .G & gayathri133gayuuu@gmail.com

INTRODUCTION

INTRODUCTION

Inventory management is a crucial part of any retail or product-based business, regardless of its size. While large companies use complex ERP systems to manage their stock and sales, many small shops and businesses still rely on outdated methods such as handwritten notes, Excel sheets, or basic billing tools. These manual approaches are time-consuming, error-prone, and lack real-time updates or meaningful data tracking. Problems such as inaccurate stock levels, missed restocking opportunities, and loss of sales history are common in such systems. To address these challenges, there is a need for a simple, effective, and easy-to-use digital tool that helps small business owners manage their inventory and track sales without the overhead of backend servers or expensive software. The project titled **"Store Manager: Keep Track of Inventory"** is designed as a lightweight, frontend-based inventory management application that fulfills this exact need. Built using **ReactJS** and relying on **Chrome Local Storage** for data handling, the system provides an efficient and user-friendly platform to manage products, monitor stock, and record sales with minimal technical setup.

This project is divided into five main tabs: **Home, Cart, Sales, Inventory**, and **Add Product**. The Home tab displays all available products with options to "Add to Cart" or "Remove from Cart" and includes a search bar for quick filtering. The Cart tab collects selected products and allows the user to either **Checkout**, which updates the stock and records the sale, or **Clear** the cart without processing. Completed transactions are recorded and displayed in the **Sales** tab, showing key details like product names, quantities, total sale value, and the date and time of the transaction. The **Inventory** tab provides features to update stock levels and set alert thresholds for low stock items. Products that fall below the set alert value are highlighted in red, and users can filter the view to show only depleted items. The **Add Product** tab allows for adding new products by entering details like name, image URL, price, stock quantity, and tags. All of this data is stored in the browser's **Local Storage**, which means the application works without any backend database or server setup and can retain data across browser sessions.

The use of ReactJS makes the interface dynamic, modular, and responsive, allowing users to interact with the system in real time. Each component of the application is designed with clarity and simplicity, ensuring even users with little technical knowledge can operate it effectively. Moreover, the application can function offline since all operations are handled on the client side, making it ideal for small businesses that may not always have internet access. While the current version uses Local Storage for simplicity, the structure of the project allows for easy future upgrades to integrate with backend technologies like **MongoDB** or **Firebase** if needed. The goal of this project is not only to demonstrate technical skills in frontend development using ReactJS but also to provide a **practical, working solution** that addresses a real-world problem. By streamlining sales tracking and inventory control into a single, browser-based tool, the system enhances efficiency, reduces manual errors, and gives small business owners better visibility into their operations. This project showcases how modern frontend technologies can be used to build functional applications that make a tangible difference in day-to-day business tasks.

SOURCE CODE

Cart.jsx :

```
import CartItem from "./CartItem";

import { useCart } from "../../context/CartContext";

import { useCartDispatch } from "../../context/CartContext";

import { useInventoryDispatch } from "../../context/InventoryContext";

import { useSalesDispatch } from "../../context/SalesContext";
```

```
export default function Cart() {
  const inventoryDispatch = useInventoryDispatch();
  const cartItemsFromContext = useCart();
  const cartDispatch = useCartDispatch();
  const saleDispatch = useSalesDispatch();

  let count = 0;
  let cartValue = 0;

  if (cartItemsFromContext.length > 0) {
    cartItemsFromContext.forEach((item) => {
      cartValue += item.price * item.quantity;
      count += item.quantity;
    });
  }

  return (
    <div className="min-h-screen flex flex-col items-center bg-sky-200 p-6 font-sans">
      {/* Heading with emoji and uppercase */}
      <h1 className="text-3xl font-extrabold uppercase text-black mb-4 flex items-center gap-2">
        🛒 Your Cart
      </h1>

      {count === 0 ? (
        <p className="text-lg text-gray-700">Your cart is empty.</p>
      ) : (
        <div className="w-full max-w-3xl">
          <p className="mb-4 text-lg">
            No. of products in cart: <b>{count}</b> | Total Cart Value:{" "}
            <b>{cartValue.toFixed(2)}</b>
          </p>
```

```
<div className="mb-4 flex gap-2 flex-wrap">
 <button
  onClick={() => {
   cartItemsFromContext.forEach((cartItem) => {
    inventoryDispatch({
     type: "STOCK_SOLD",
     productName: cartItem.productName,
     stock: cartItem.quantity,
    });
   });
   saleDispatch({
    type: "NEW_SALE",
    saleValue: cartValue,
    products: cartItemsFromContext,
   });
   cartDispatch({
    type: "EMPTY_CART",
   });
   alert(
    "Checkout successful! Inventory has been updated."
   );
  }}
  className="bg-green-500 hover:bg-green-600 text-white font-bold rounded p-2 transition"
 >
  Checkout Cart
 </button>

 <button
  onClick={() => {
   cartDispatch({
    type: "EMPTY_CART",
   });
```

```
          }}
          className="bg-red-500 hover:bg-red-600 text-white font-bold rounded p-2 transition"
         >
          Clear Cart
         </button>
        </div>

        <div className="flex flex-wrap justify-center gap-4">
         {cartItemsFromContext.map((product) => (
          <CartItem key={product.productName} product={product} />
         ))}
        </div>
      </div>
    )}
   </div>
  );
}
```

Product.jxs :

```
import { useState } from "react";
import { useInventoryDispatch } from "../../context/InventoryContext";

const Product = ({ product, alertValue }) => {
   const [addStock, setAddStock] = useState(0)
   const inventoryDispatch = useInventoryDispatch()

   return (
      <div
          className={`border px-3 py-2 rounded text-center flex flex-col items-center ${product.stock < alertValue ? "border-red-800 bg-red-100 border-2" : "border-gray-400"
           }`}
      >
```

```
<h1 className="font-bold text-xl">{product.productName}</h1>

<div className="w-[250px] h-[250px] overflow-hidden border border-gray-300 rounded mt-2">

  <img

    src={product.imageUrl}

    alt={product.productName}

    className="w-full h-full object-cover"

  />

</div>

<p className="text-lg mt-2">Price: ₹ {product.price.toFixed(2)}</p>

<div>

  <p>Stock Available: {product.stock}</p>

</div>

<div className="mt-2">

  Add Stock:{" "}

  <input

    onChange={(e) => {

      setAddStock(e.target.value);

    }}

    className="border border-gray-400 rounded p-1"

    value={addStock}

    type="number"

    name="new-stock-qty"

    id="new-stock-qty"

  />

</div>

<div>

  <button

    onClick={() => {

      setAddStock(0);

      inventoryDispatch({

        type: "STOCK_ADDED",

        productName: product.productName,
                    stock: addStock,
                  });
                }}
                className="bg-green-500 hover:bg-green-600 rounded p-1 text-white mt-2"
              >
                Update Stock
              </button>
            </div>
          </div>
        );
      };

export default Product;
```

AddProduct.jxs:

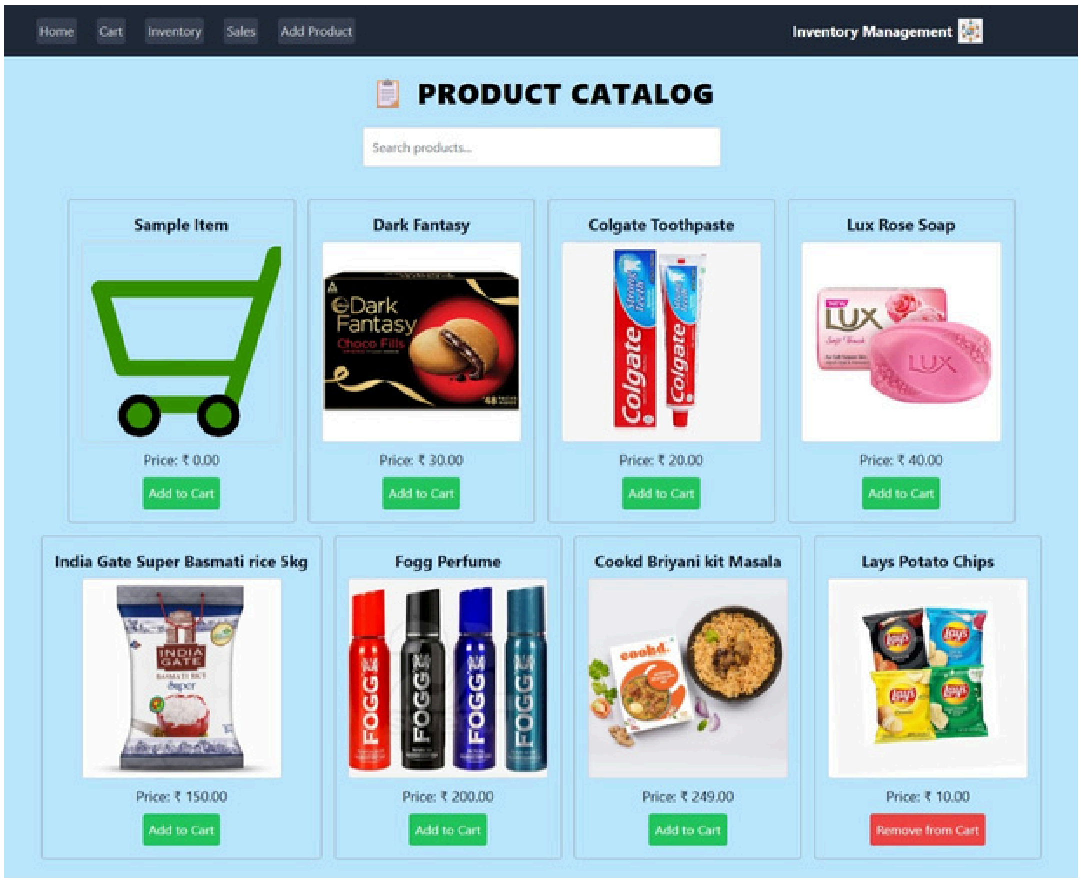
```
import { useInventoryDispatch } from "../context/InventoryContext";


export default function AddProduct() {
  const dispatchToInventory = useInventoryDispatch();


  const onAddProduct = (e) => {
    e.preventDefault();


    const newProduct = {
      productName: e.target.productName.value,
      imageUrl: e.target.imageUrl.value,
      price: parseFloat(e.target.price.value),
      tags: e.target.tags.value.split(",").map((tag) => tag.trim()),
      stock: e.target.stock.value,
    };


    dispatchToInventory({
```

```
    type: "NEW_PRODUCT",

    ...newProduct,

   });


   e.target.reset();

   alert("Product added successfully!");

 };


 return (

   <div className="min-h-screen flex justify-center items-start bg-sky-200 p-6 font-sans">

     <div className="w-full max-w-md bg-white shadow-lg rounded-lg p-6 mt-6">

       {/* Heading with emoji */}

       <h1 className="uppercase font-extrabold text-3xl text-black text-center mb-6 flex items-center justify-center gap-2">

         ➕ Add New Product

       </h1>


       {/* Form */}

       <form onSubmit={onAddProduct} className="space-y-4">

         <div>

           <label

             htmlFor="productName"

             className="block text-sm font-medium text-gray-700"

           >

             Product Name

           </label>

           <input

             id="productName"

             type="text"

             className="w-full p-2 border rounded focus:ring focus:ring-green-300 focus:outline-none"

             placeholder="Enter product name"

             required

           />
```

```
</div>


<div>
  <label
    htmlFor="imageUrl"
    className="block text-sm font-medium text-gray-700"
  >
    Product Image URL
  </label>
  <input
    id="imageUrl"
    type="text"
    className="w-full p-2 border rounded focus:ring focus:ring-green-300 focus:outline-none"
    placeholder="Enter image URL"
    required
  />
</div>


<div>
  <label
    htmlFor="price"
    className="block text-sm font-medium text-gray-700"
  >
    Price
  </label>
  <input
    id="price"
    type="number"
    step="0.01"
    className="w-full p-2 border rounded focus:ring focus:ring-green-300 focus:outline-none"
    placeholder="Enter price"
    required
  />
</div>

<div>
  <label
    htmlFor="stock"
    className="block text-sm font-medium text-gray-700"
  >
    Stock
  </label>
  <input
    id="stock"
    type="number"
    step="0.01"
    className="w-full p-2 border rounded focus:ring focus:ring-green-300 focus:outline-none"
    placeholder="Enter stock"
    required
  />
</div>

<div>
  <label
    htmlFor="tags"
    className="block text-sm font-medium text-gray-700"
  >
    Tags (comma-separated)
  </label>
  <input
    id="tags"
    type="text"
    className="w-full p-2 border rounded focus:ring focus:ring-green-300 focus:outline-none"
    placeholder="Enter tags, separated by commas"
```

```
          required
         />
        </div>

        <div className="text-center">
         <button
          type="submit"
          className="w-full bg-green-500 text-white font-bold py-2 px-4 rounded hover:bg-green-600 transition"
         >
          Add Product
         </button>
        </div>
       </form>
      </div>
     </div>
    );
   }
```

SCREENSHOTS



PRODUCT CATALOG

Sample Item
Price: ₹ 0.00

Dark Fantasy
Price: ₹ 30.00

Colgate Toothpaste
Price: ₹ 20.00

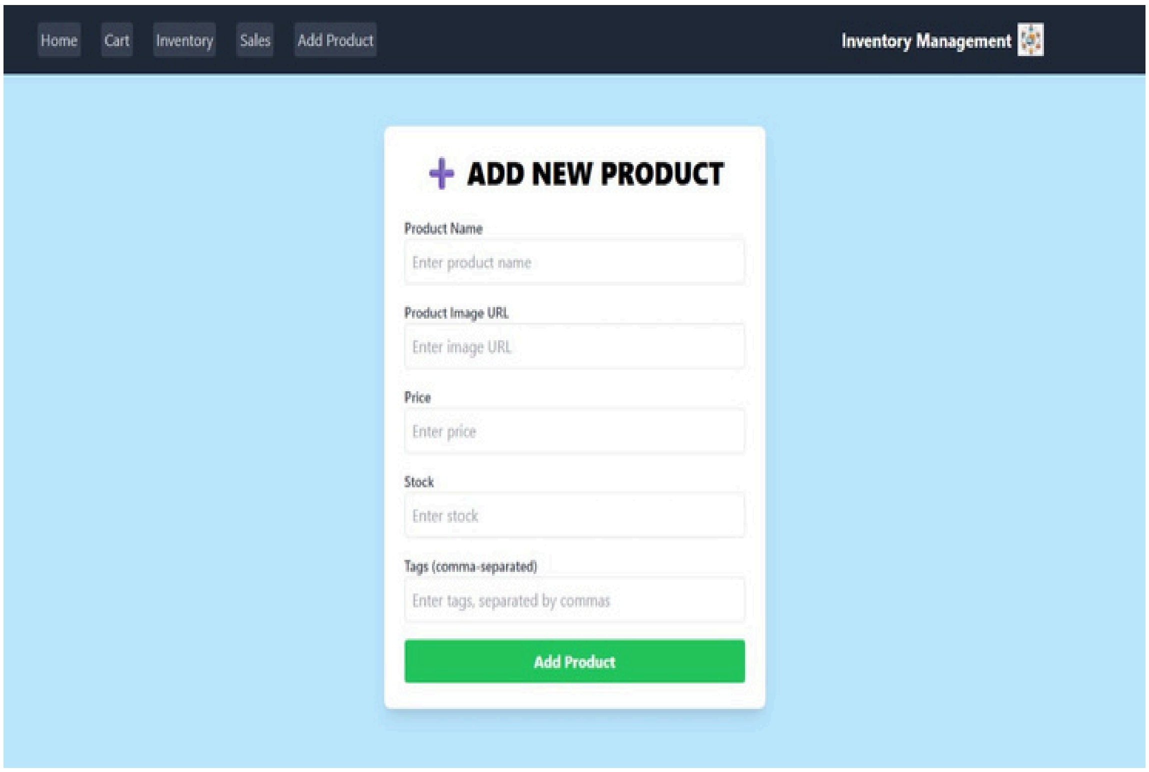
Lux Rose Soap
Price: ₹ 40.00

India Gate Super Basmati rice 5kg
Price: ₹ 150.00

Fogg Perfume
Price: ₹ 200.00

Cookd Briyani kit Masala
Price: ₹ 249.00

Lays Potato Chips
Price: ₹ 10.00



ADD NEW PRODUCT

Product Name

Enter product name

Product Image URL

Enter image URL

Price

Enter price

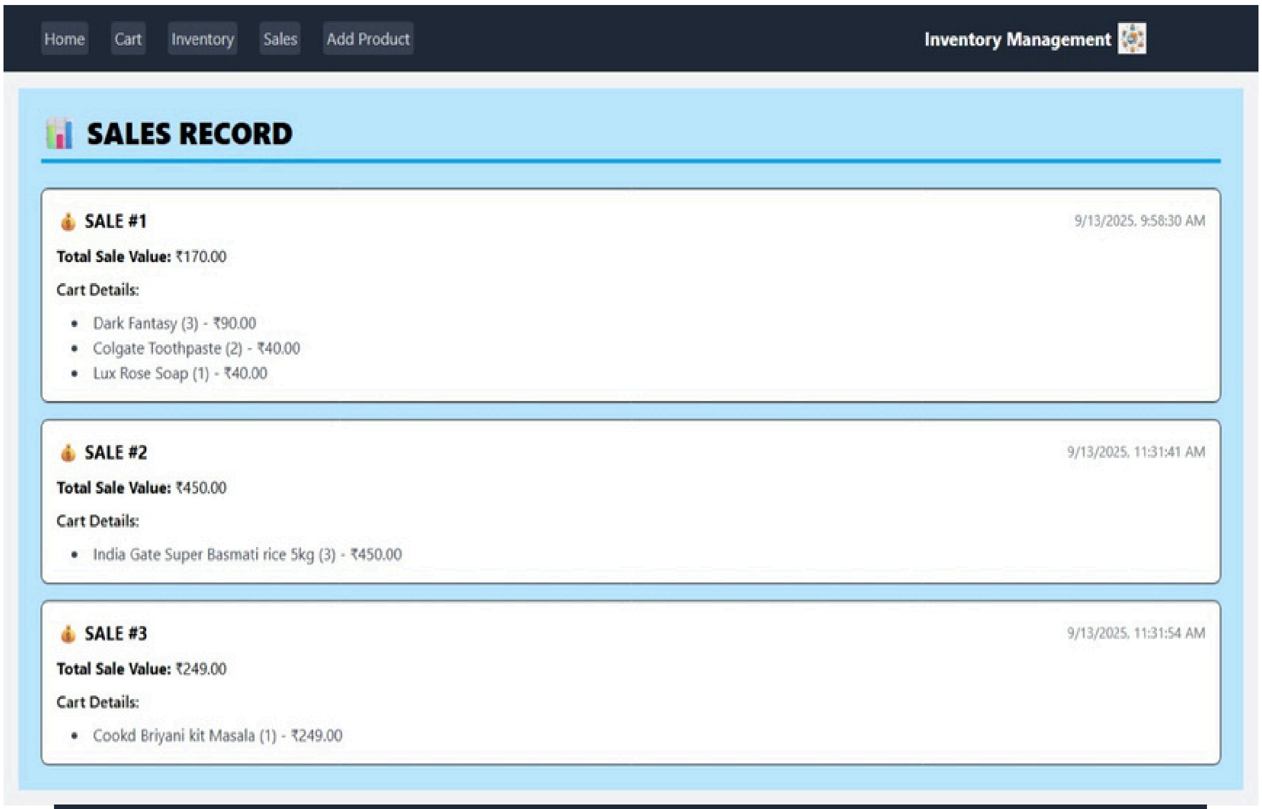
Stock

Enter stock

Tags (comma-separated)

Enter tags, separated by commas

Add Product



Home Cart Inventory Sales Add Product   Inventory Management

SALES RECORD

SALE #1   9/13/2025, 9:58:30 AM

Total Sale Value: ₹170.00

Cart Details:

- Dark Fantasy (3) - ₹90.00
- Colgate Toothpaste (2) - ₹40.00
- Lux Rose Soap (1) - ₹40.00

SALE #2   9/13/2025, 11:31:41 AM

Total Sale Value: ₹450.00

Cart Details:

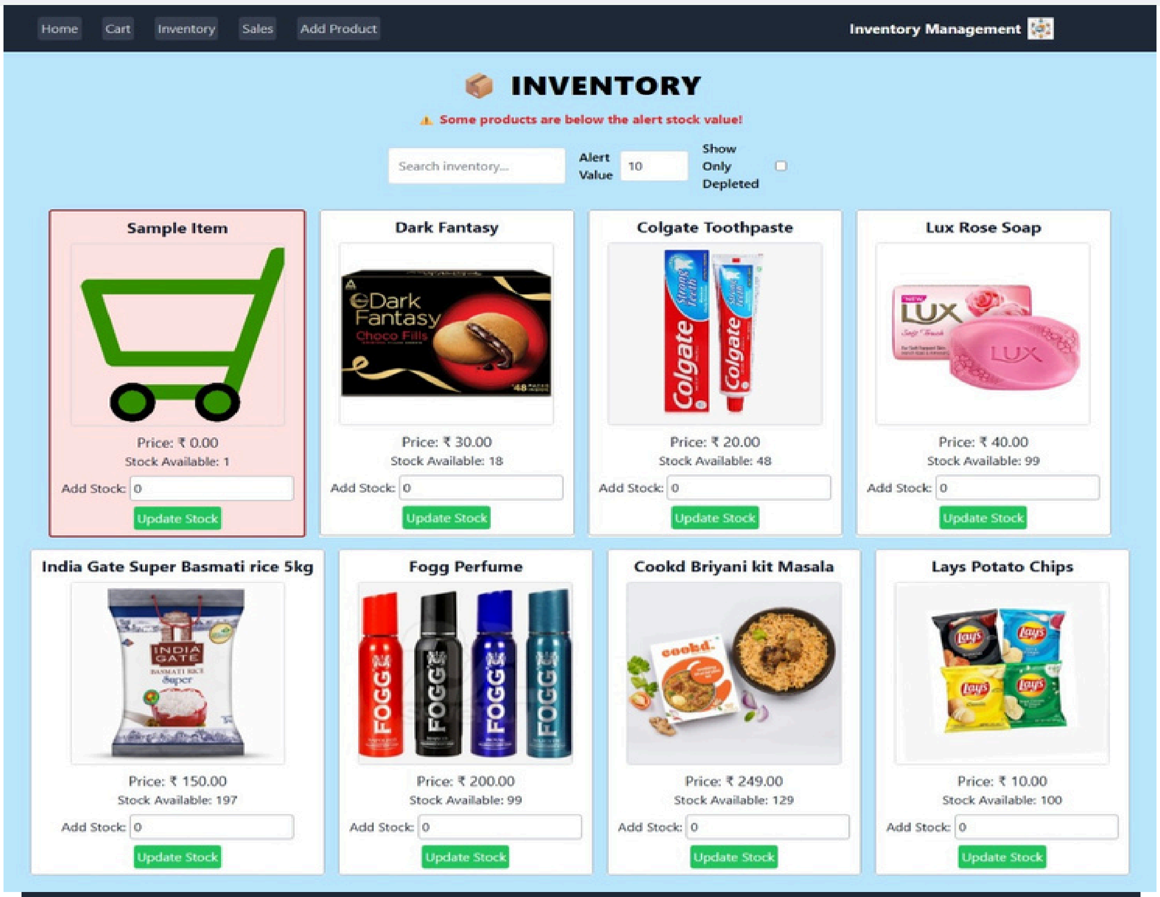
- India Gate Super Basmati rice 5kg (3) - ₹450.00

SALE #3   9/13/2025, 11:31:54 AM

Total Sale Value: ₹249.00

Cart Details:

- Cookd Briyani kit Masala (1) - ₹249.00



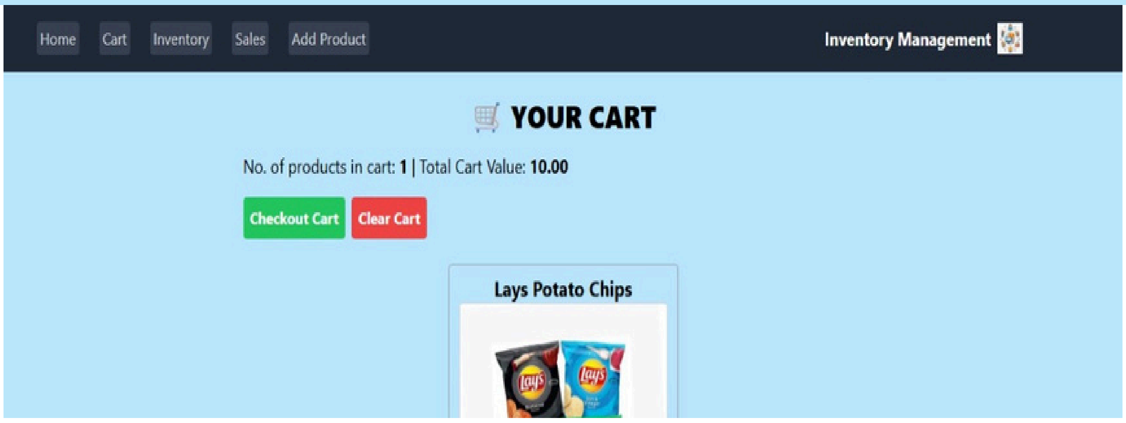
Home Cart Inventory Sales Add Product   Inventory Management

INVENTORY

⚠ Some products are below the alert stock value!

Search inventory... Alert Value 10 Show Only Depleted ☐

Product	Price	Stock Available	Add Stock
Sample Item	₹ 0.00	1	0
Dark Fantasy	₹ 30.00	18	0
Colgate Toothpaste	₹ 20.00	48	0
Lux Rose Soap	₹ 40.00	99	0
India Gate Super Basmati rice 5kg	₹ 150.00	197	0
Fogg Perfume	₹ 200.00	99	0
Cookd Briyani kit Masala	₹ 249.00	129	0
Lays Potato Chips	₹ 10.00	100	0



Home Cart Inventory Sales Add Product   Inventory Management

YOUR CART

No. of products in cart: **1** | Total Cart Value: **10.00**

Lays Potato Chips

DEMO LINK: https://drive.google.com/drive/folders/1A3GO9_-DmiKrYBurz2QMGT7_FiXO0DSn?usp=sharing

Local: http://localhost:3001

On Your Network: http://192.168.1.156:3001