# FULLSTACK MERN PROJECT REPORT

# PROJECT TITLE : SHOPSMART YOUR DIGITAL GROCERY STORE EXPERIENCE

## TEAM ID: LTVIP2025TMID20506

### Team Leader : Mohammad Asif Baig

### Team member : Mirza Mohammad Abbas

### Team member : Misala Charan Kumar

### Team member : Mogalla Pavan Venkata Kumar

**Team Member and Role:-**

| **Name** | **Role** |
| --- | --- |
| Mohammad Asif Baig | Team Lead, Backend Integration, Architecture |
| Mirza Mohammad Abbas | UI Development, React Components |
| Misala Charan Kumar | Project Coordination, API Integration, Deployment |
| Mogalla Pavan Venkata Kumar | Testing, Documentation, Support Modules |

# Introduction:

"Welcome to our Shopsmart Web App, your one-stop shop for all your grocery needs! With our user-friendly interface and wide selection of high-quality products, we aim to make your grocery shopping experience convenient and enjoyable. Whether you're looking for fresh produce, pantry staples, or household essentials, our app has you covered. Explore our virtual aisles, add items to your cart with ease, and have your groceries delivered right to your doorstep. Experience the future of grocery shopping with our Grocery Web App today!"

**Project Overview:**

**📝 Purpose:-**

The purpose of ShopSmart: Your Digital Grocery Store Experience is to provide a seamless and convenient online shopping platform for a wide range of users, including tech-savvy individuals, homemakers, and everyday consumers. The project aims to make grocery shopping more accessible and efficient by offering a user-friendly web application where customers can explore products, manage their purchases, and enjoy a hassle-free digital experience. It also focuses on building trust by ensuring secure transactions and safeguarding user data.

**⭐ Features:-**

* User-Friendly Navigation: Intuitive and responsive design allows users to easily browse product categories and find what they need.
* Product Details and Cart Management: View product information, add items to the cart, and modify them before checkout.
* Secure Checkout Process: Seamless and protected payment system to complete purchases confidently.
* Seller Dashboard: Sellers can manage product listings, update inventory, and handle customer orders efficiently.
* Admin Panel: Administrators can process payments, monitor user activity, and respond to customer inquiries.
* Data Privacy and Security: Emphasis on securing customer data and maintaining confidentiality of personal information.
* Wide Product Selection: Includes essentials across various categories to cater to different types of customers.
* Optimized Shopping Experience: The app is built to ensure speed, reliability, and customer satisfaction.

**Technical Architecture:-**



The technical architecture of an grocery-webapp app typically involves a client-server model, where the frontend represents the client and the backend serves as the server. The frontend is responsible for user interface, interaction, and presentation, while the backend handles data storage, business logic, and integration with external services like payment gateways and databases. Communication between the frontend and backend is typically facilitated through APIs, enabling seamless data exchange and functionality.

**ER-Diagram:**

****

The technical architecture of an grocery-webapp app typically involves a client-server model, where the frontend represents the client and the backend serves as the server. The frontend is responsible for user interface, interaction, and presentation, while the backend handles data storage, business logic, and integration with external services like payment gateways and databases. Communication between the frontend and backend is typically facilitated through APIs, enabling seamless data exchange and functionality.

**Key Features:**

**Product Catalog:** Our grocery-webapp app provides an extensive product catalog with various categories and subcategories. Users can easily search, browse, and filter products based on their preferences, making it effortless to find the desired items.

**Shopping Cart and Checkout:** The app includes a shopping cart feature that enables users to add products, review their cart, and proceed to checkout. The checkout process offers multiple payment options, ensuring a smooth and secure transaction experience.

**Product Reviews and Ratings:** Customers can provide feedback and rate products, helping other users make informed purchasing decisions. This feature fosters a sense of community and trust among users.

**Order Tracking:** Once an order is placed, users can track its status in real-time. They receive updates on order processing, shipping, and delivery, providing transparency and peace of mind.

**Admin Dashboard:** For administrators, our grocery-webapp app offers a comprehensive dashboard to manage products, inventory, orders, and customer information. It provides insights into sales performance, stock levels, and customer analytics, enabling efficient business operations.

**Order Management:** The app manages the order lifecycle, including order placement, tracking, and status updates. Users can view their order history, track shipments, and request returns or cancellations.

**Search and Filtering:** Users can search for products using keywords and apply filters to narrow down the search results based on criteria such as price range, brand, or customer ratings.

**Shopping Cart and Checkout**: The app includes a shopping cart feature that enables users to add products, review their cart, and proceed to checkout. The checkout process offers multiple payment options, ensuring a smooth and secure transaction experience.

**Product Reviews and Ratings:** Customers can provide feedback and rate products, helping other users make informed purchasing decisions. This feature fosters a sense of community and trust among users.

**Order Tracking:** Once an order is placed, users can track its status in real-time. They receive updates on order processing, shipping, and delivery, providing transparency and peace of mind.

**Admin Dashboard:** For administrators, our grocery-webapp app offers a comprehensive dashboard to manage products, inventory, orders, and customer information. It provides insights into sales performance, stock levels, and customer analytics, enabling efficient business operations.

**Order Management**: The app manages the order lifecycle, including order placement, tracking, and status updates. Users can view their order history, track shipments, and request returns or cancellations.

**Search and Filtering:** Users can search for products using keywords and apply filters to narrow down the search results based on criteria such as price range, brand, or customer ratings.

## PRE REQUISITES:

To develop a full-stack Ecommerce App for Furniture Tool using React js, Node.js,Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: <https://nodejs.org/en/download/>

• Installation instructions:<https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download:<https://www.mongodb.com/try/download/community>

• Installation instructions:<https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

• Installation: Open your command prompt or terminal and run the following

command: **npm install express**

**React js: React** is a JavaScript library for building client-side applications.

And Creating Single Page Web-Appliaction

**Getting Started**

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

**Quik Start**

npm create vite@latest

cd my-app

npm install

npm run dev

If you've previously installed create-react-app globally via npm install -g create-react-app, we recommend you uninstall the package using npm uninstall -g create-react-app or yarn global remove create-react-app to ensure that npx always uses the latest version.

**Create a new React project:**

• Choose or create a directory where you want to set up your React project.

• Open your terminal or command prompt.

• Navigate to the selected directory using the cd command.

• Create a new React project by running the following command: npx create-react-app your-app-name.Wait for the project to be created:

• This command will generate the basic project structure and install the necessary dependencies

**Navigate into the project directory:**

• After the project creation is complete, navigate into the project directory by running the following command**: cd your-app-name**

**Start the development server:**

• To launch the development server and see your React app in the browser, run the following command:  **npm run dev**

• The npm start will compile your app and start the development server.

• Open your web browser and navigate to <https://localhost:5173> to see your React app.

You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more ad- vanced configurations and features by referring to the official React documentation: <https://react.dev/>

**HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Library:** Utilize React to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• Git: Download and installation instructions can be found at:<https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

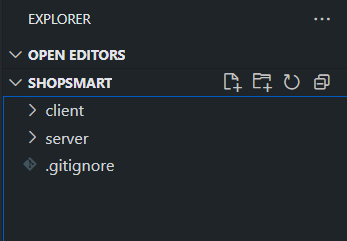
• Visual Studio Code: Download from <https://code.visualstudio.com/download>

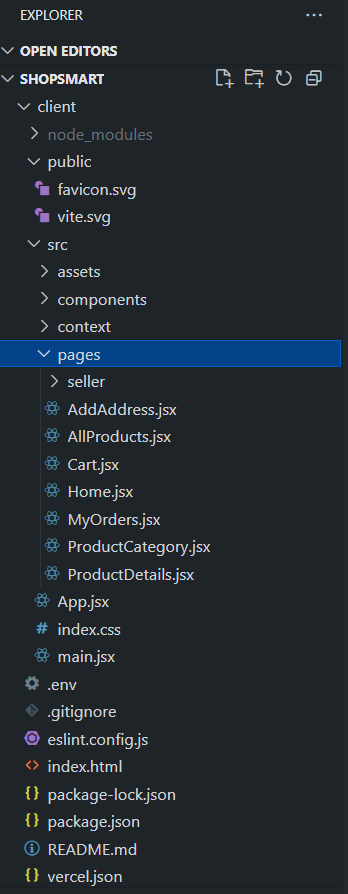
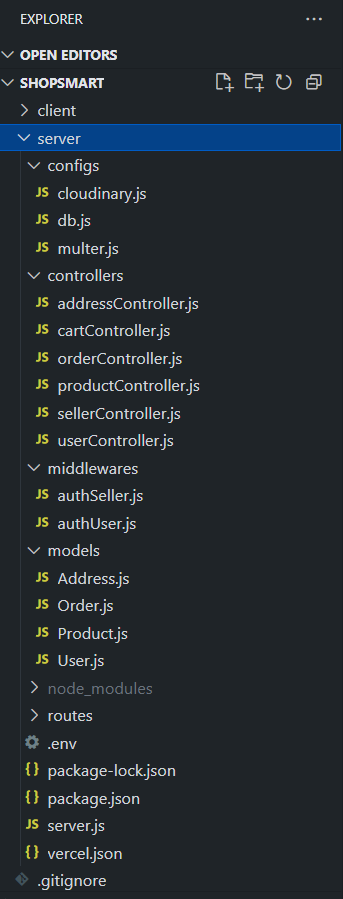
• Sublime Text: Download from

<https://www.sublimetext.com/download>

• WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

**Project Structure:-**

****

** **

**Roles and Responsibility**

**User:-**

* Registration and Authentication: Users are responsible for creating an account on the platform and securely logging in to access its features.
* Browsing and Shopping: Users can browse products, add them to their cart, and proceed to checkout for purchasing.
* Payment: Users are responsible for making payments for their orders using the available payment methods.
* Order Management: Users can view their order history, track their deliveries, and manage their account details.
* Feedback and Reviews: Users can provide feedback on products and services and leave reviews to help other users make informed decisions.
* Compliance: Users are expected to adhere to the platform's terms and conditions and privacy policy.

**Admin:-**

* User Management: Admins can manage user accounts, including creating, updating, and deleting accounts as necessary.
* Product Management: Admins are responsible for managing the platform's product listings, including adding new products, updating existing ones, and removing outdated products.
* Order Management: Admins can view and manage all orders placed on the platform, including processing payments, tracking deliveries, and handling returns or refunds.
* Content Management: Admins can manage the platform's content, including creating and updating informational pages, blog posts, and other content.
* Analytics and Reporting: Admins can generate reports and analyze data to gain insights into the platform's performance and user behavior.
* Compliance and Security: Admins are responsible for ensuring that the platform complies with relevant laws and regulations and that user data is kept secure.
* Customer Support: Admins can provide support to users, including responding to inquiries, resolving issues, and handling complaints.
* Marketing and Promotion: Admins can create and manage marketing campaigns and promotions to attract and retain users.

**Admin & User Flow:**

****

The project flow for a grocery-web app involves user actions such as browsing products, adding items to the cart, proceeding to checkout, providing shipping details, selecting payment methods, making payments, and receiving order confirmation. Admin

actions include managing products, viewing and processing orders, managing customers, and updating product details.

**PROJECT FLOW:-**

Before starting to work on this project, let’s see the demo.

Demo link:-

<https://drive.google.com/file/d/1OA5mkJzda3LZc8Uo7BAI8qZHcH3wHiuy/view?usp=sharing>

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

* Node.js.
* MongoDB.
* Create-react-app.

**2. Create project folders and files:**

* Client folders.
* Server folders.

**3. Install Packages:**

**Frontend npm Packages**

* Axios.
* React-Router –dom.
* Bootstrap.
* React-Bootstrap.
* React-icons.

**Backend npm Packages**

* Express.
* Mongoose.
* Cors.

**Milestone 2: Backend Development:**

* **Setup express server**

1. Create index.js file in the server (backend folder).
2. Create a .env file and define port number to access it globally.
3. Configure the server by adding cors, body-parser.

* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Define API Routes:**
  + Create separate route files for different API functionalities such as users orders, and authentication.
  + Define the necessary routes for listing products, handling user registration and  login,managing orders, etc.
  + Implement route handlers using Express.js to handle requests and interact with the database.
* **Implement Data Models:**
  + Define Mongoose schemas for the different data entities like products, users,  and orders.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.
* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Error Handling:**

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

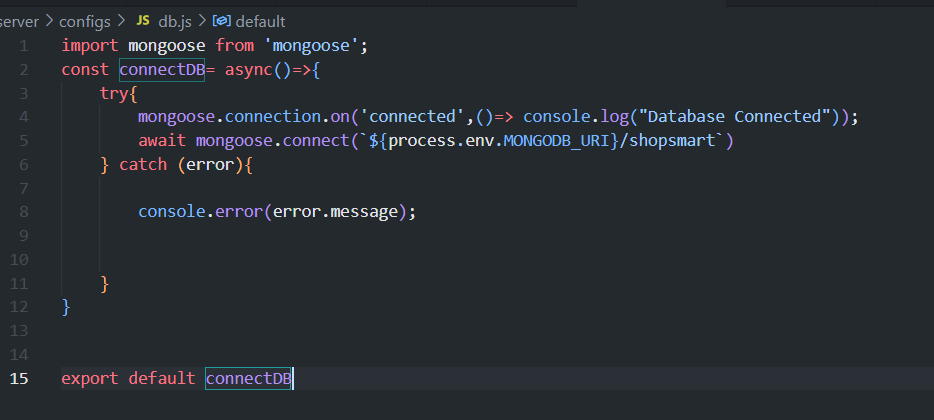
**Milestone 3: Database:**

**1. Configure MongoDB:**

* Install Mongoose.
* Create database connection.
* Create Schemas & Models.

**2. Connect database to backend:**

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



**3. Configure Schema:**

Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The schemas are looks like for the Application.

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

    firstname: { type: String},

    lastname: { type: String },

    username: { type: String, unique: true },

    email: { type: String},

    password: { type: String }

});

// category schema

const categorySchema = new mongoose.Schema({

    category: { type: String, required: true, unique: true, },

    description: { type: String, }

});

const productSchema = new mongoose.Schema({

    productname: { type: String, required: true },

    description: { type: String, required: true },

    price: { type: Number, required: true },

    image: { type: String, required: true },

    category: { type: String, ref: 'Category', required: true },

    countInStock: { type: Number, required: true, min: 0 },

    rating: { type: Number, required: true },

    dateCreated: { type: Date, default: Date.now }

});

const addToCartSchema = new mongoose.Schema({

    userId: { type: String, required: true },

    productId: { type: String, required: true },

    quantity: { type: Number, minimum: 1, required: true, default: 1 },

});

const orderSchema = new mongoose.Schema({

    firstname: { type: String, required: true },

    lastname: { type: String, required: true },

    user: { type: String, ref: 'User', required: true },

    phone: { type: String, required: true },

    productId: { type: String, required: true },

    productName: { type: String, required: true },

    quantity: { type: String, default: 1 },

    price: { type: String, required: true },

    status: { type: String, enum: ['Pending', 'Confirmed', 'Shipped', 'Delivered', 'Canceled',], default: 'Pending' },

    paymentMethod: { type: String, required: true },

    address: { type: String, required: true },

    createdAt: { type: Date, default: Date.now }

});

const models = {

    Users: mongoose.model('User', userSchema),

    Category: mongoose.model('Category', categorySchema),

    Product: mongoose.model('Product', productSchema),

    AddToCart: mongoose.model('AddToCart', addToCartSchema),

    Order: mongoose.model('Order', orderSchema),

};

module.exports = models;

**Milestone 4: Frontend Development:**

**1. Setup React Application:**

• Create React application.

• Configure Routing.

• Install required libraries.

**2. Design UI components:**

• Create Components.

• Implement layout and styling.

• Add navigation.

**3. Implement frontend logic:**

• Integration with API endpoints.

• Implement data binding.

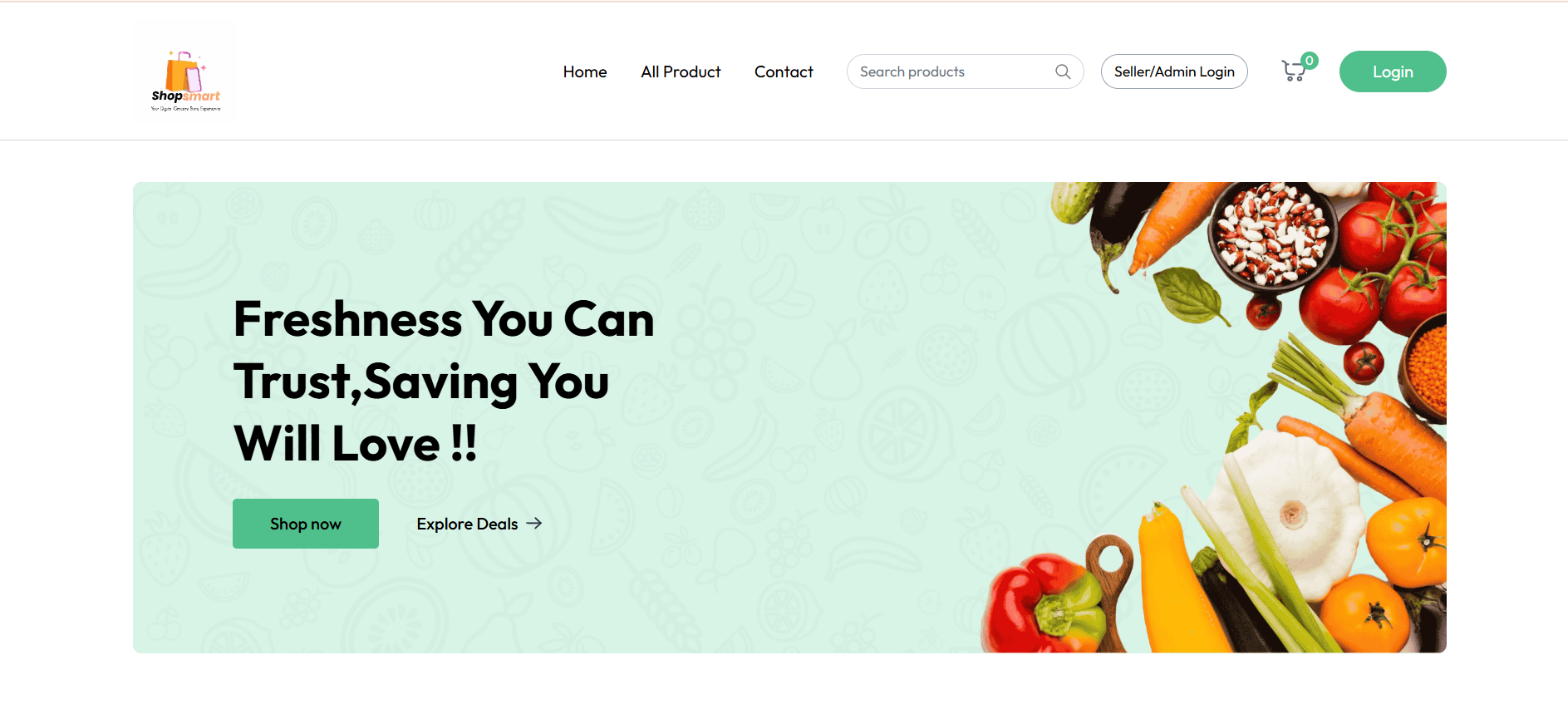
Reference:-

<https://drive.google.com/file/d/1jzebl3jn6a37_ntwPmjg5K2yzpKPbF0a/view?usp=sharing>

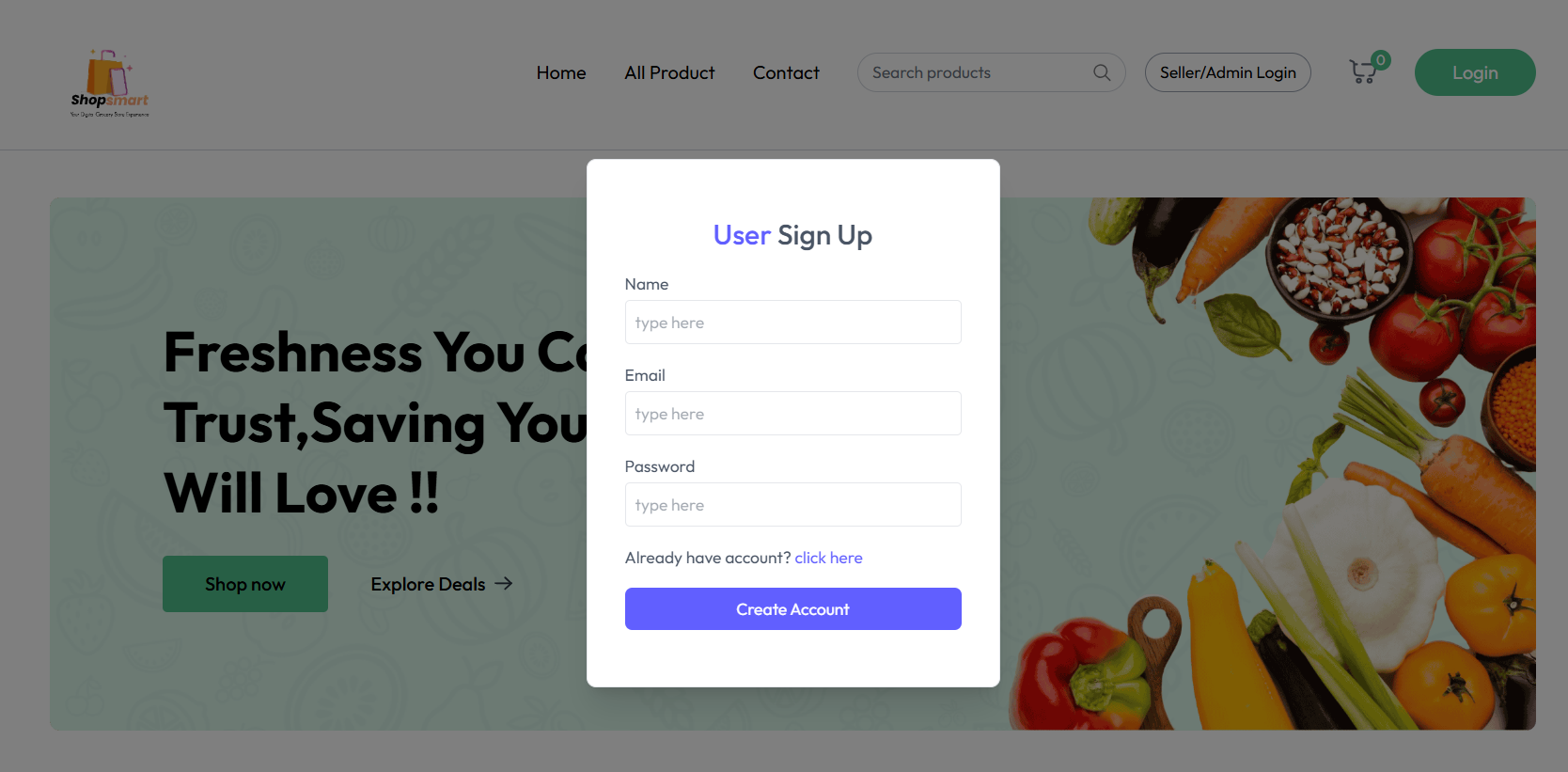
**Milestone 5: Project Implementation:**

Finally, after finishing coding the projects we run the whole project to test it’s working process and look for bugs. Now, let’s have a final look at the working of our Darshan Ease.

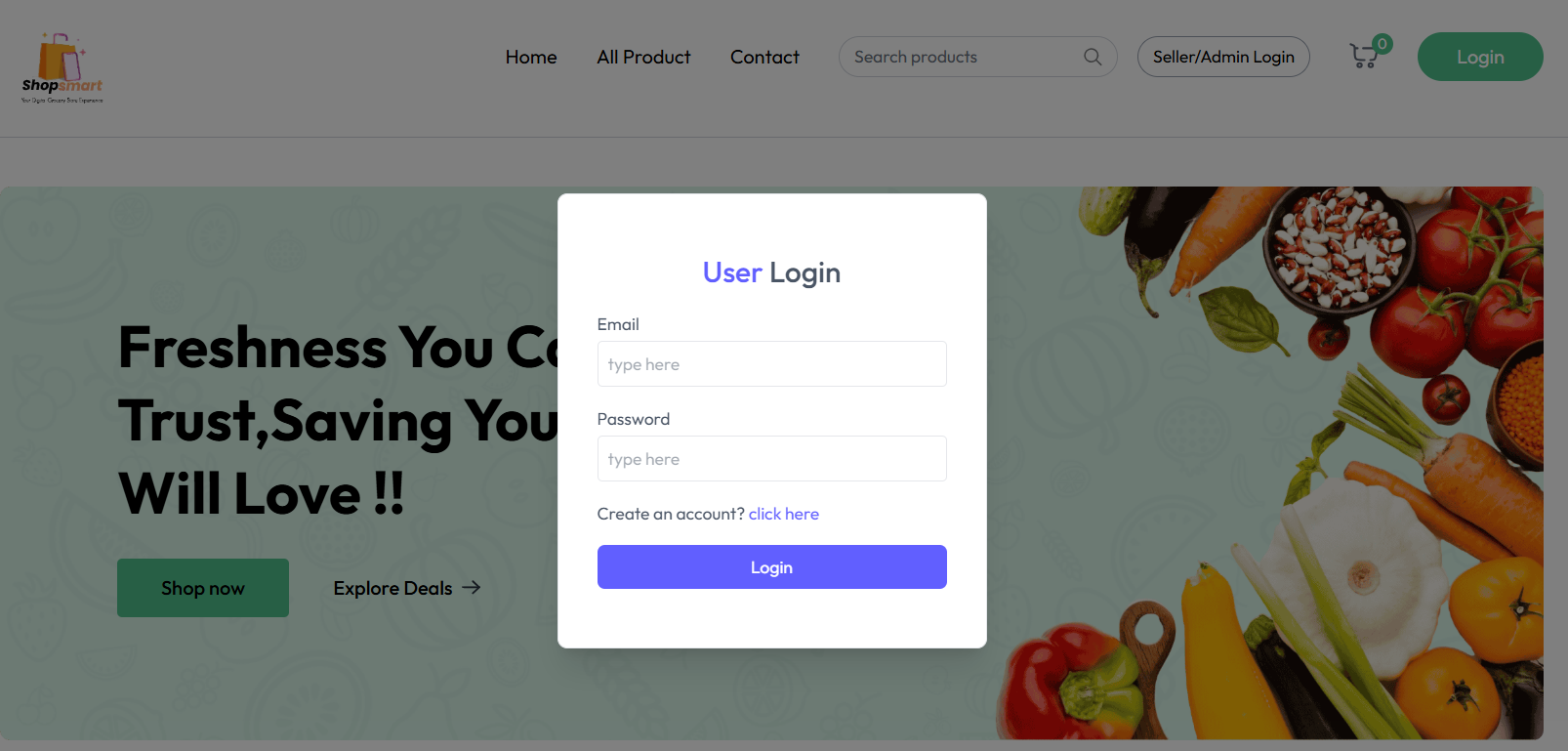
Landing page:-



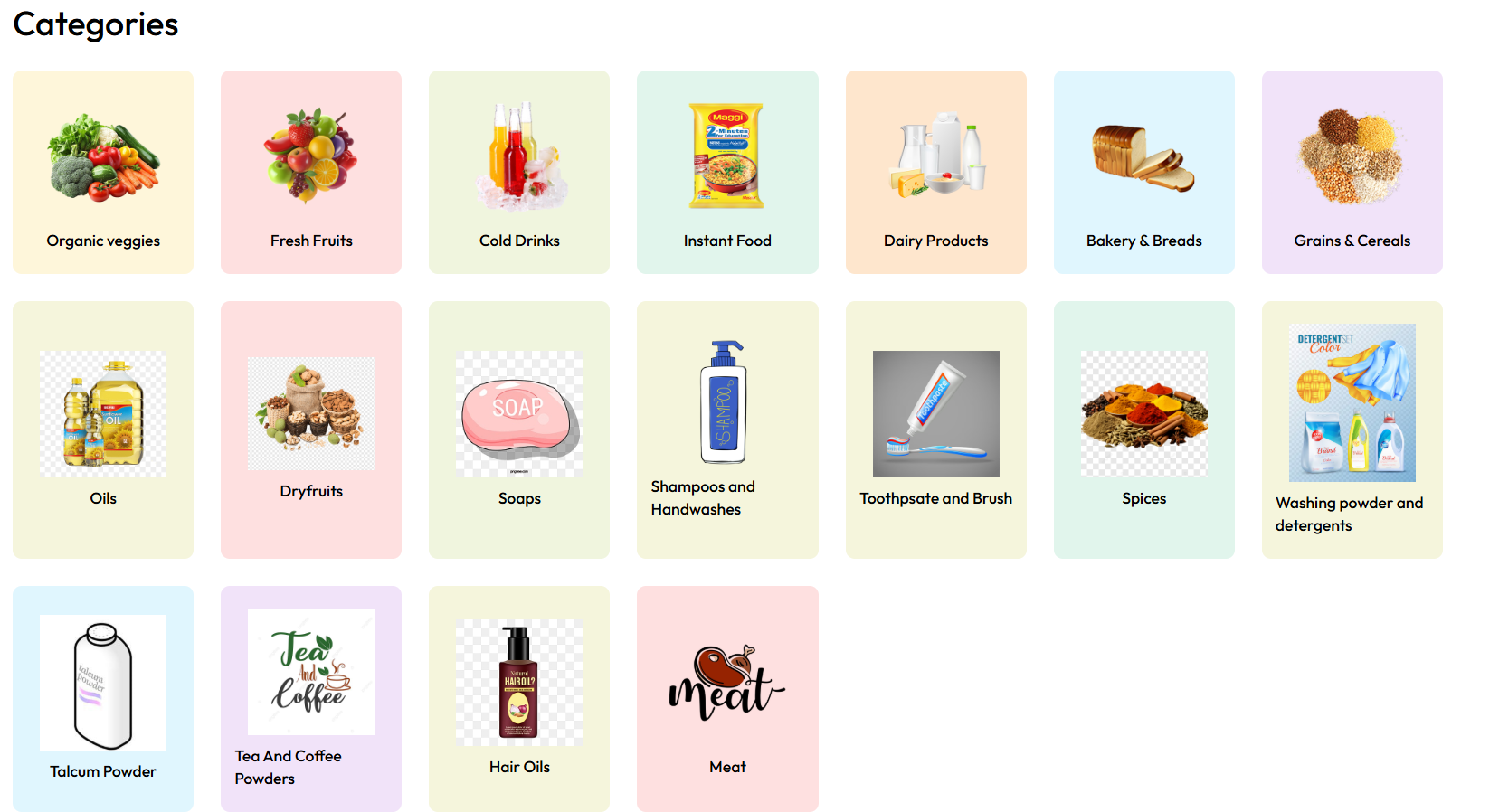
Register Page:-



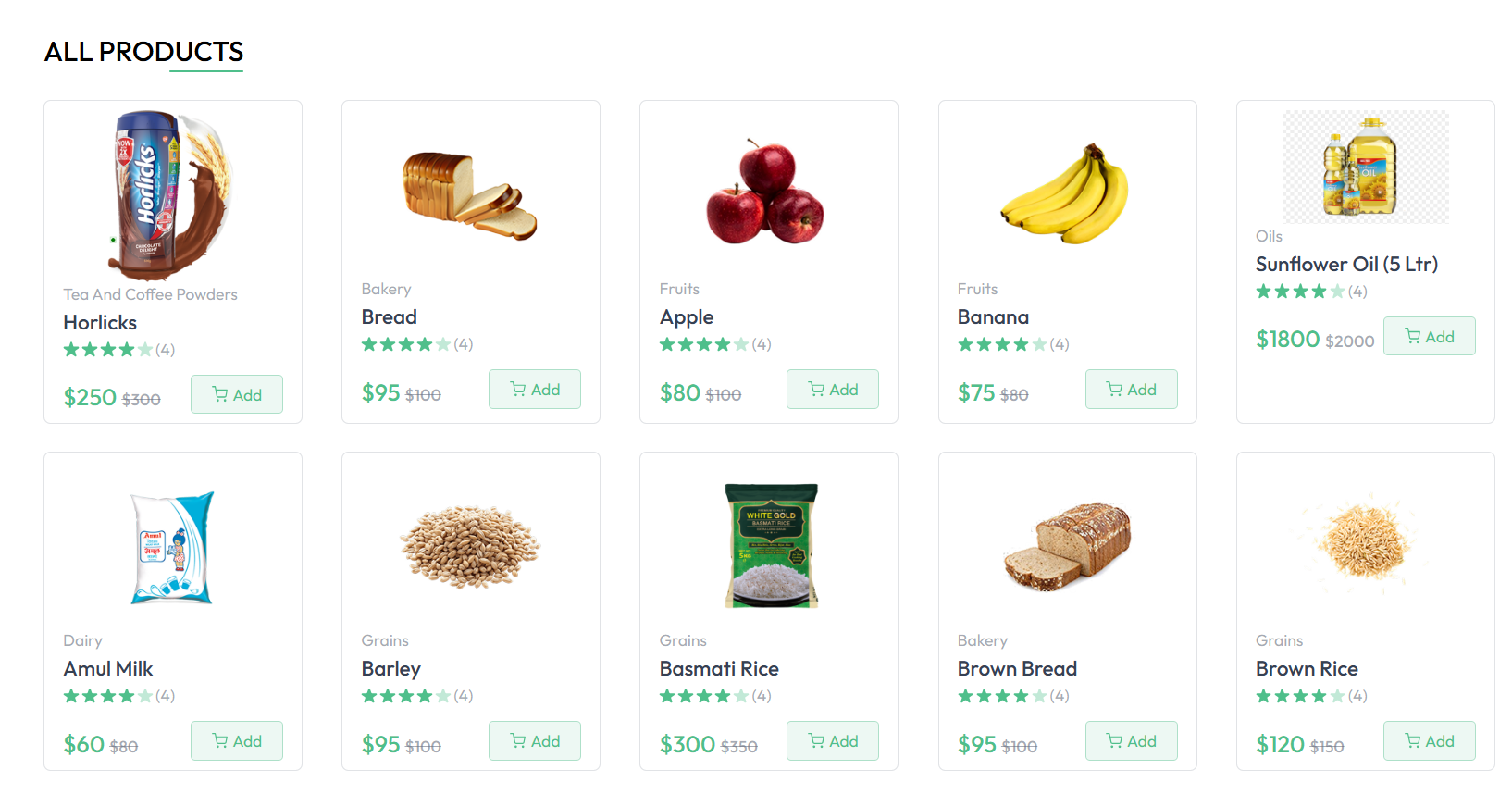
Login Page:-

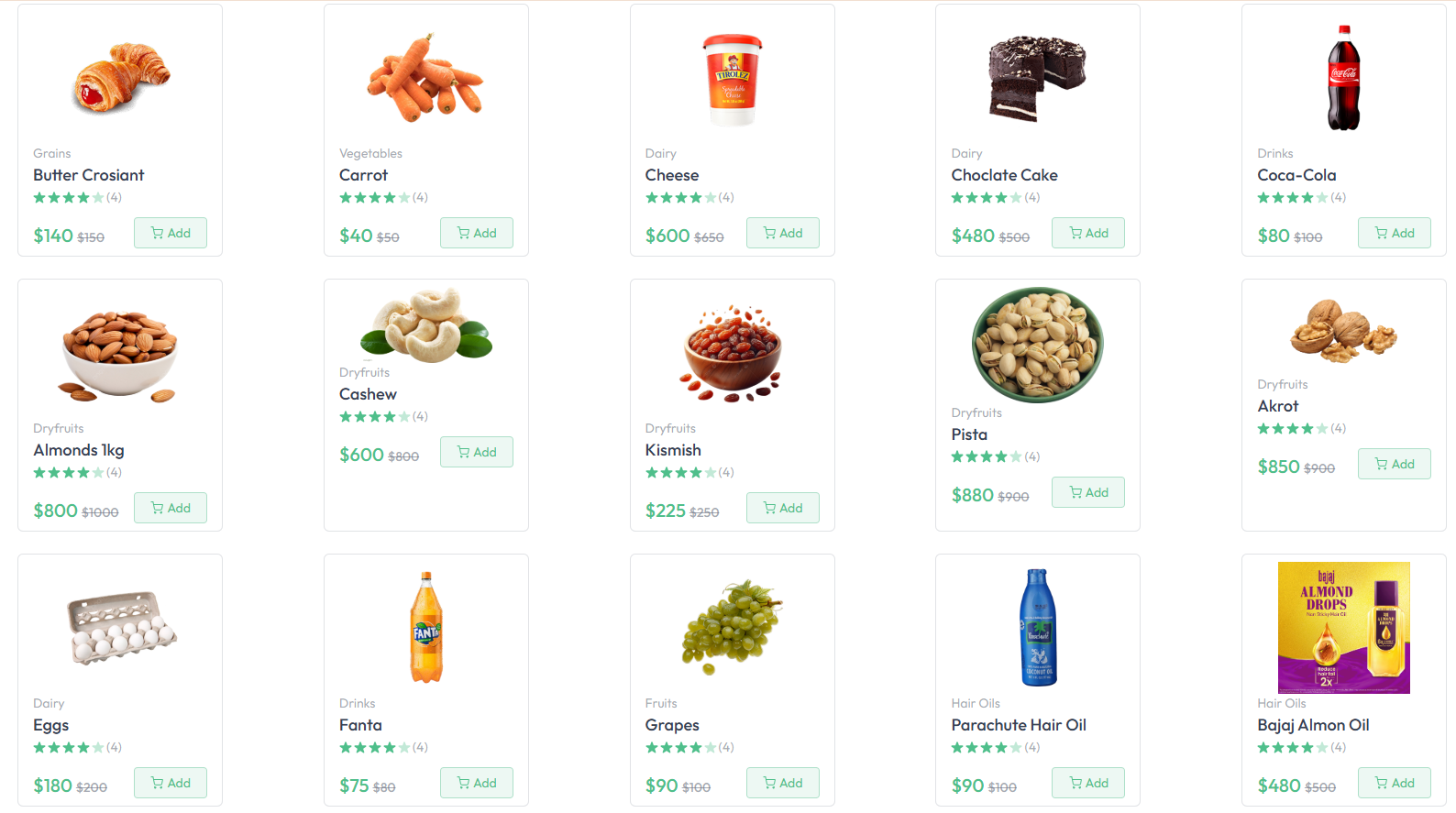


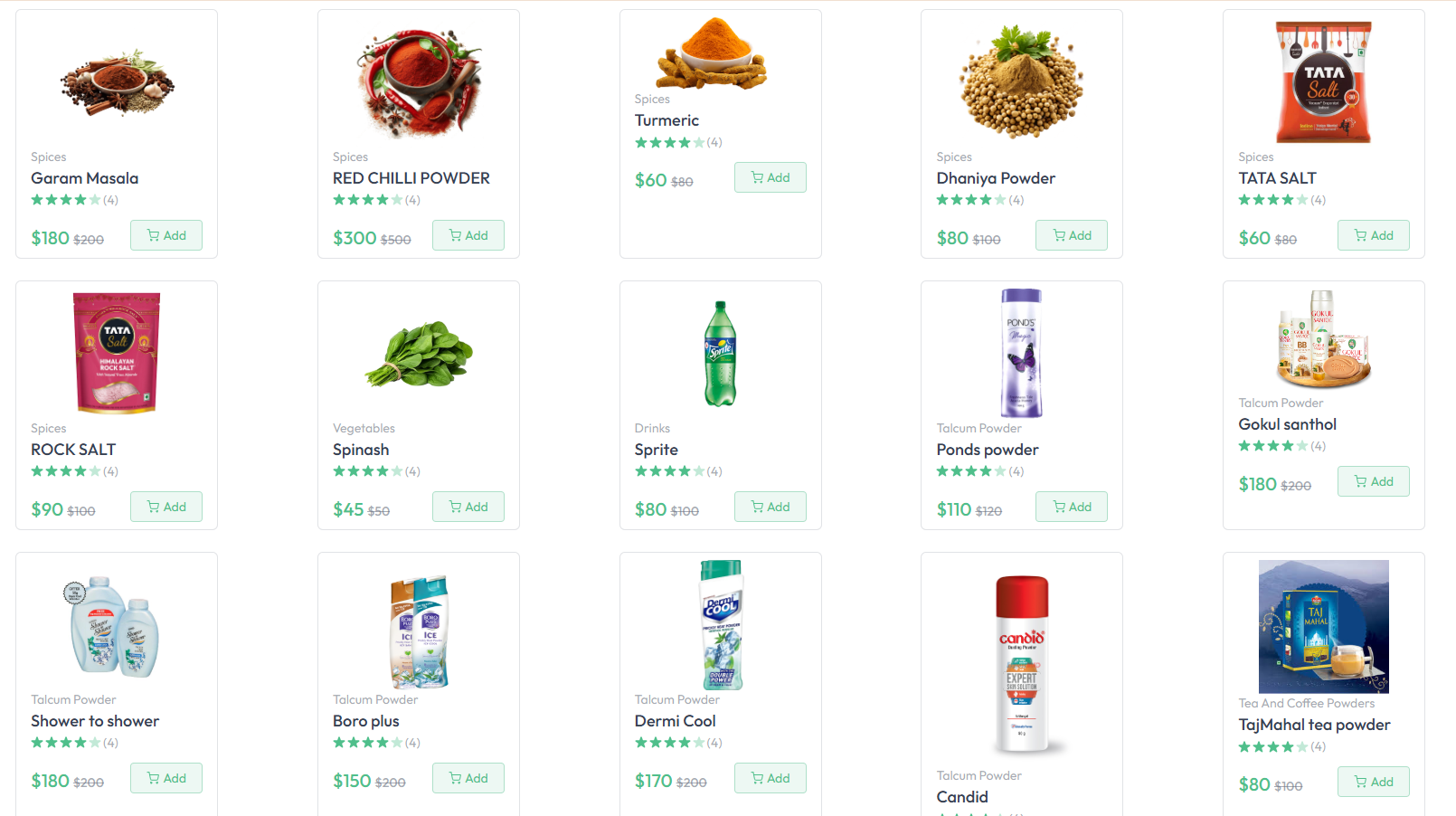
Category Page:-

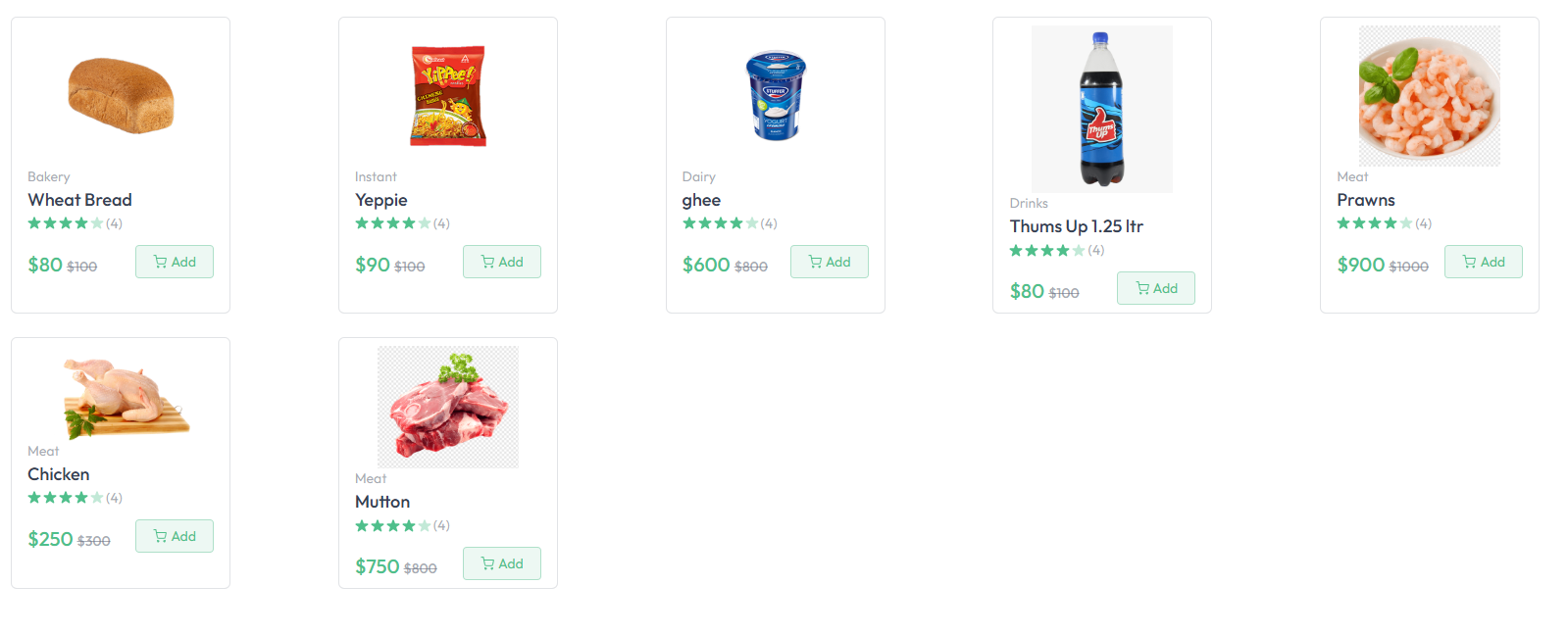


All Products Page:-



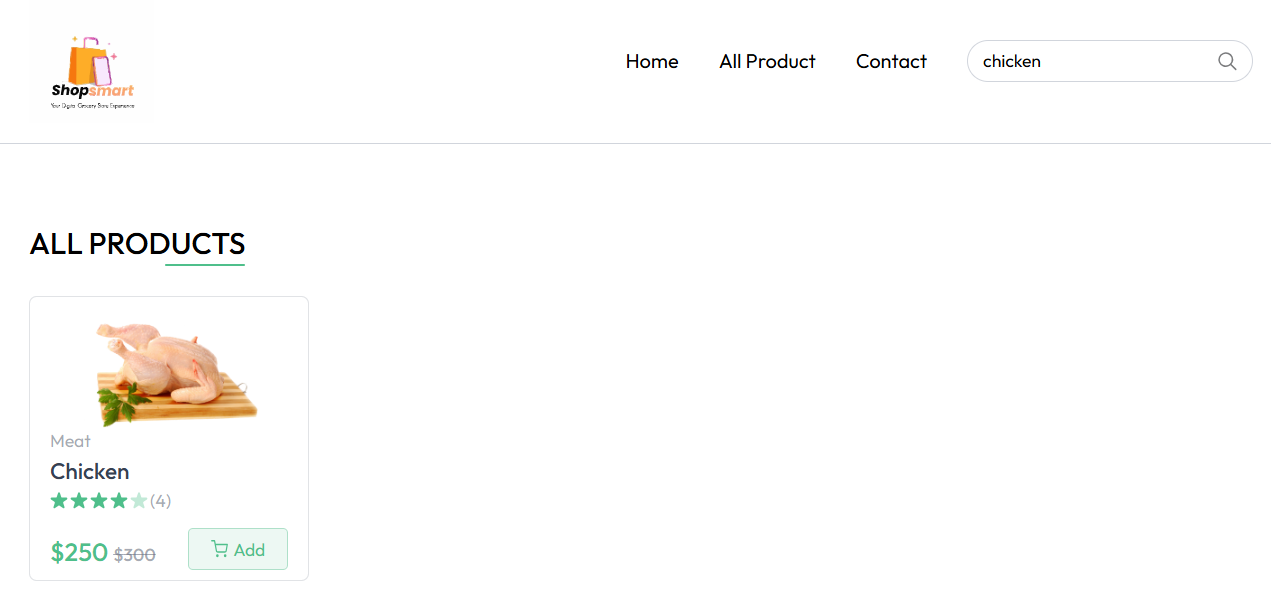




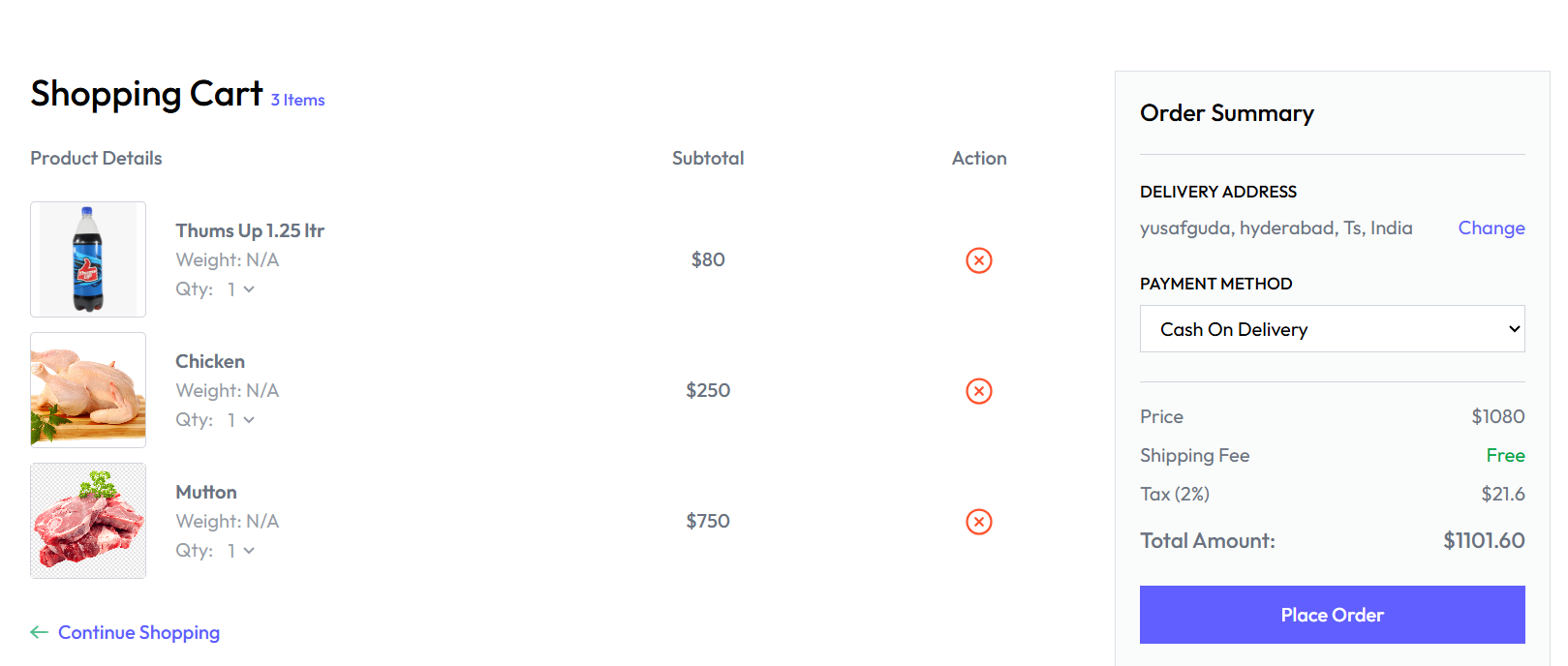


**Search Feature:-**

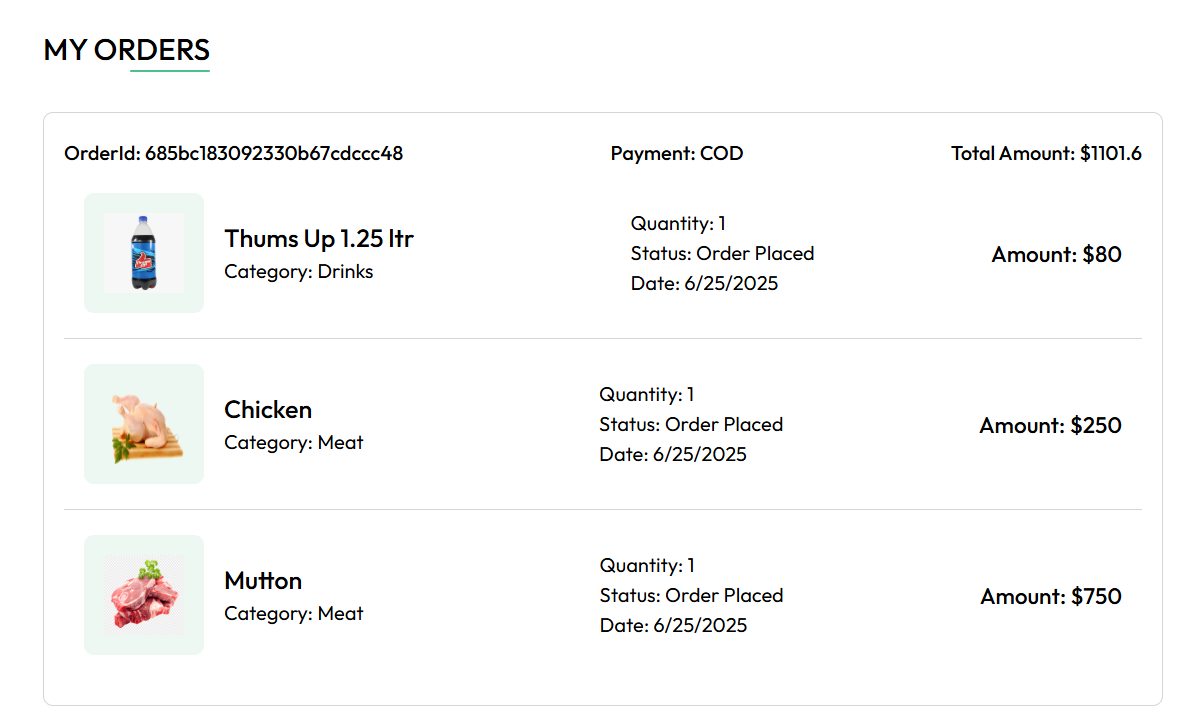
Search product through search bar and it filters and show you the product



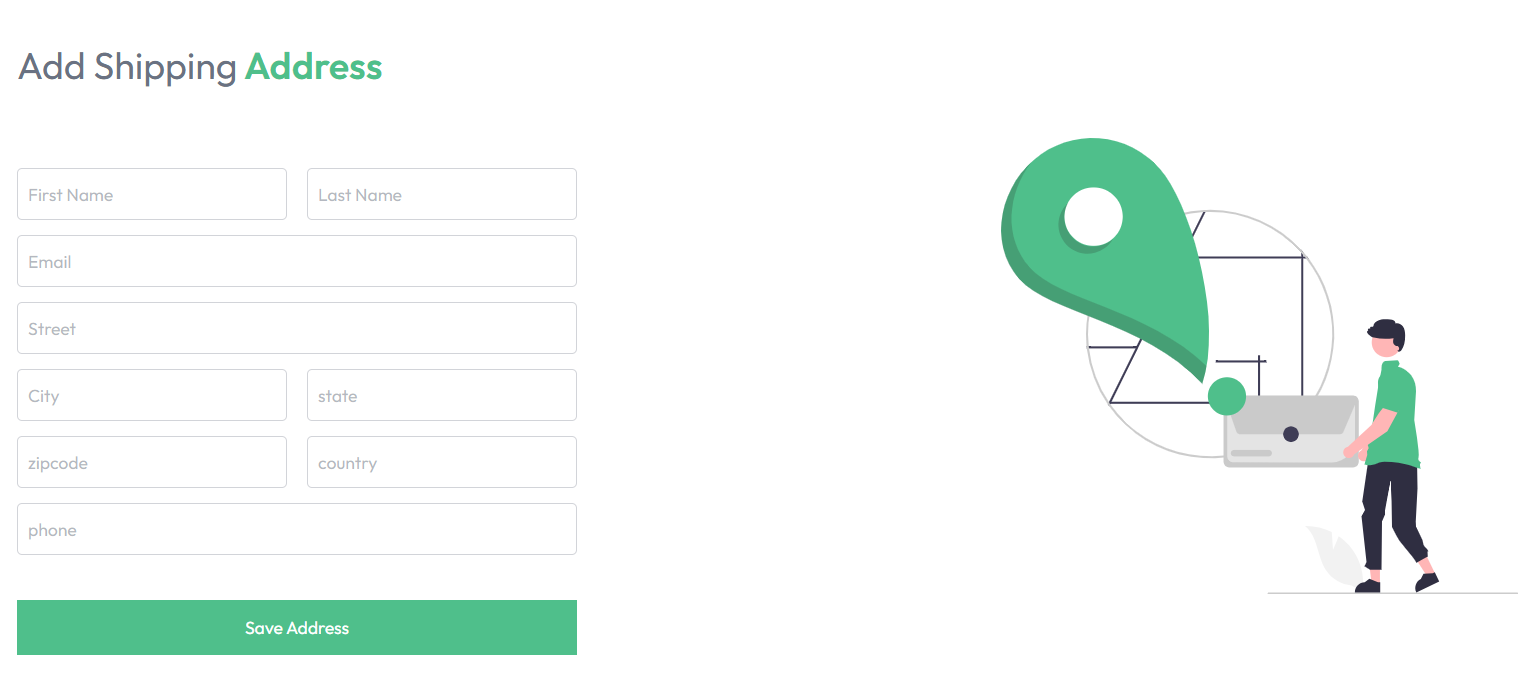
My Cart:-



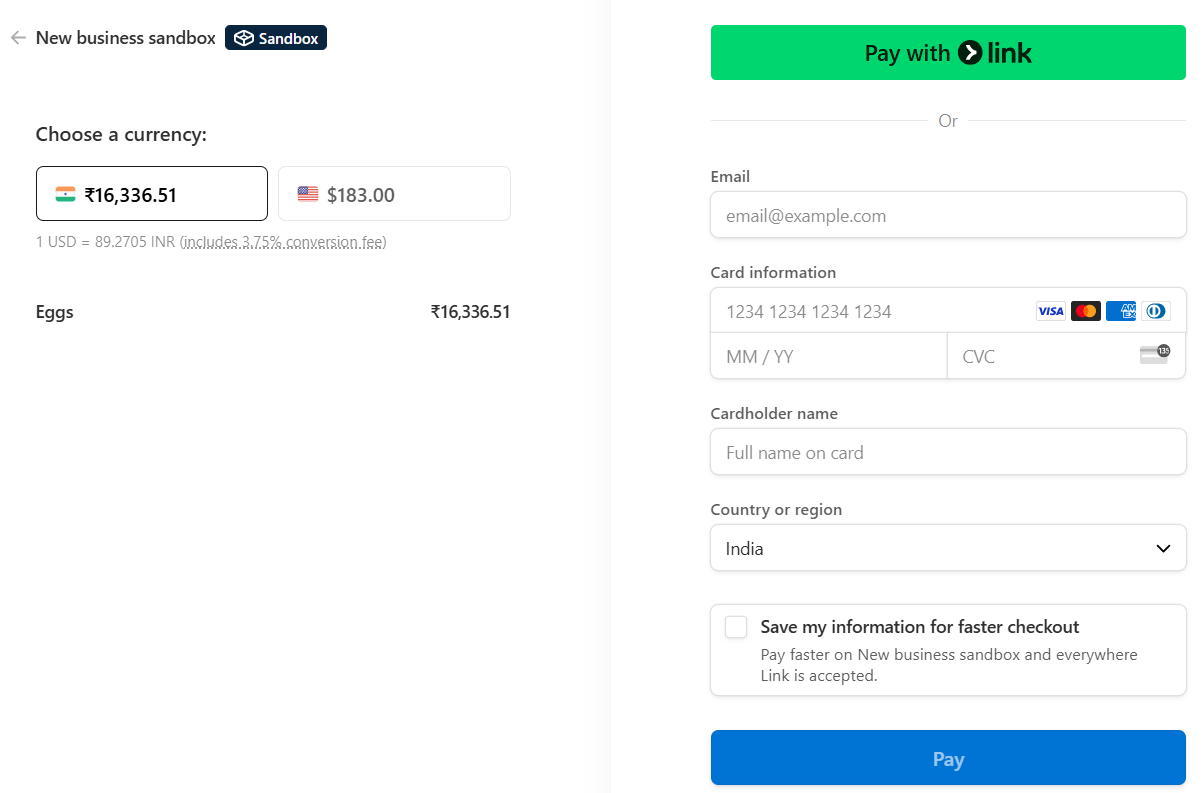
My Orders Page:-



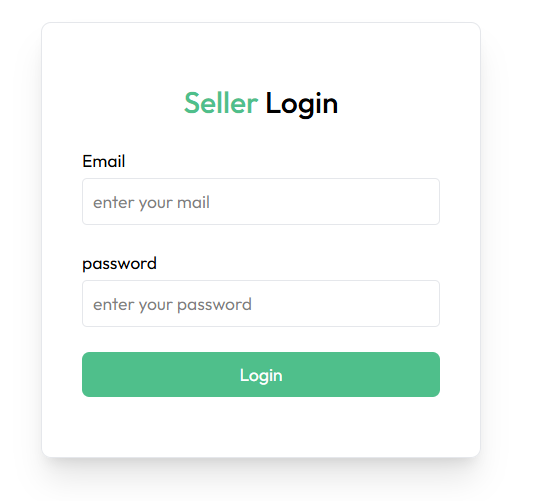
Address Page:-



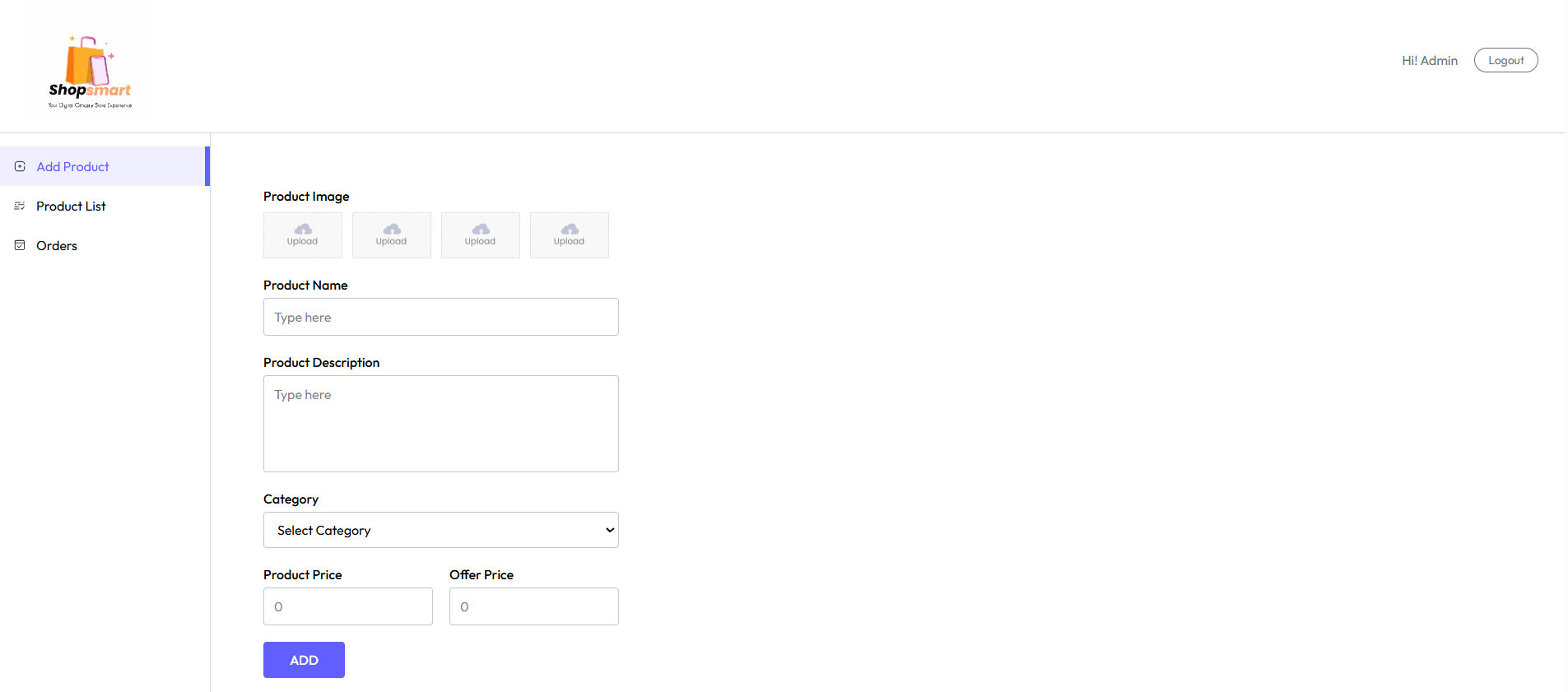
Payment Page (online payment:Stripe):-



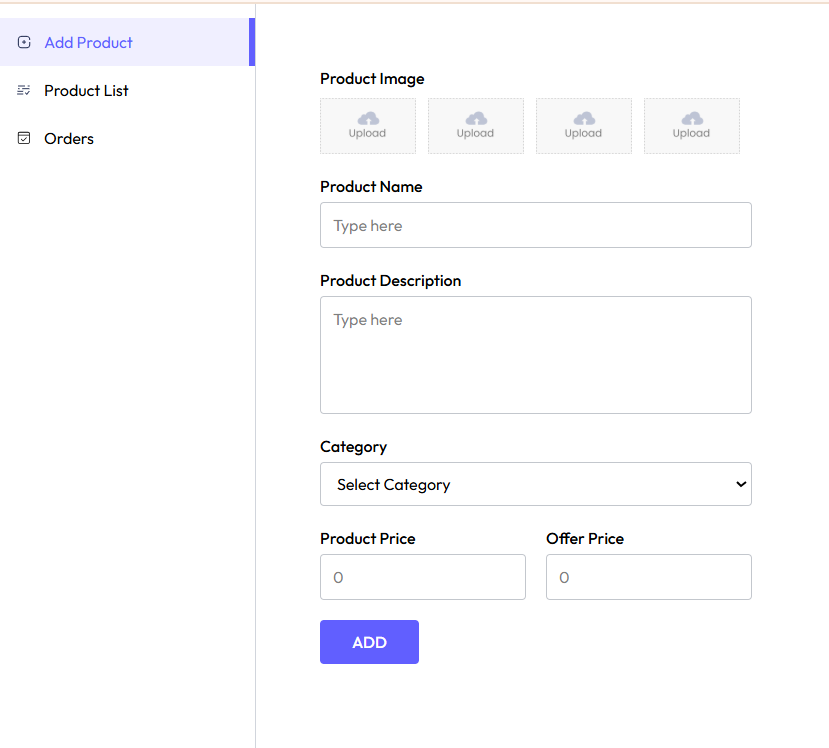
Admin Login Page:-



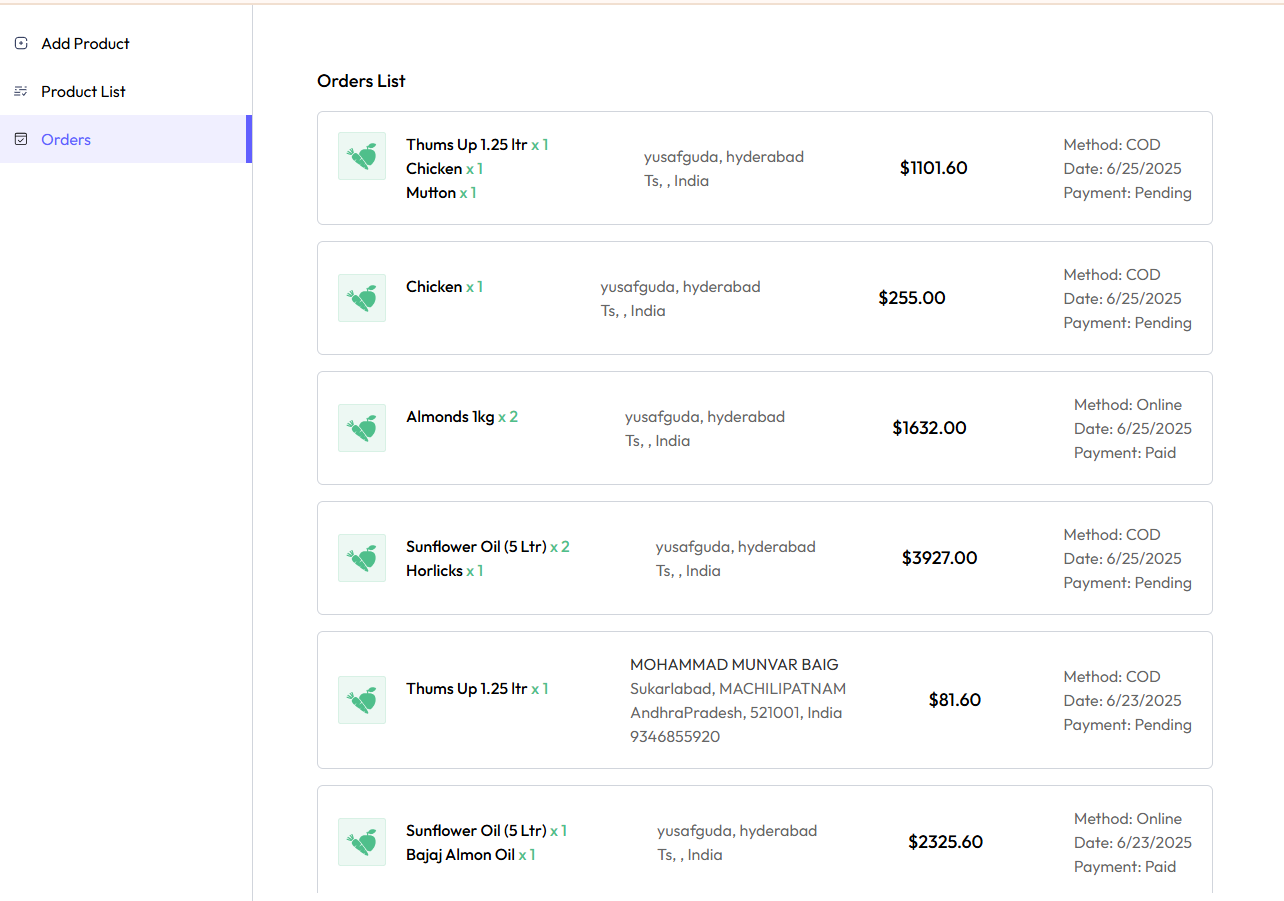
Admin Dashboard Page:



Add Product page:-



Admin Orders Page:-



The demo of the app is available at:-

<https://drive.google.com/file/d/1OA5mkJzda3LZc8Uo7BAI8qZHcH3wHiuy/view?usp=sharing>

**The demo code and overall project at :-**

<https://drive.google.com/file/d/1jzebl3jn6a37_ntwPmjg5K2yzpKPbF0a/view?usp=sharing>

**Future Enhancements:-**

1. Multi-Vendor Marketplace  
   Enable multiple sellers to register and manage their own shops, expanding the platform’s scalability.
2. Product Recommendation System  
   Use machine learning to suggest products based on user behavior, preferences, and purchase history.
3. Coupons and Promo Codes  
   Add support for discount coupons and promotional codes to boost sales and user engagement.
4. Order Tracking System  
   Allow users to track their orders in real-time with status updates and estimated delivery times.
5. PWA Support  
   Convert the web app into a Progressive Web App (PWA) for offline use and mobile installation.
6. Review and Rating System  
   Let customers review and rate products, enhancing trust and product feedback.
7. Chatbot Integration  
   Implement an AI-powered chatbot to assist users with queries, orders, and support.
8. Admin Analytics Dashboard  
   Add a visual dashboard with graphs and statistics to help admins track sales, orders, and user activity.

**\*\* Thank You \*\***