Project Titles

GrocerX: Your Digital Grocery Store Experience

Team Members:

Here List team members and their roles:

1.L.Rohith(Full Stack Developer): Combines both frontend and backend responsibilities, ensuring smooth communication between the two. This role also handles bug fixing, feature integration, and overall system performance.

2.K.Tarun Kumar(Frontend Developer): Responsible for designing the user interface using React.js. This role focuses on ensuring a responsive, user-friendly design, as well as integrating the frontend with backend APIs.

3.K.Akash(Backend Developer): Develops the backend server using Node.js and Express.js, ensuring the creation of secure, scalable RESTful APIs, as well as handling authentication, data processing, and business logic.

4.K.Yuva Gandhi Karthik (Database Administration): Manages the MongoDB database, focusing on schema design, data integrity, and database optimization to ensure efficient data storage and retrieval.

GrocerX web

Introduction:

"Welcome to our GrocerX Web , 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!"

Description:

At GrocerX, we're committed to bringing you the freshest groceries and everyday essentials, all in one place — to make shopping easy and delightful.

Since 2005, our mission has been simple: to serve every household with fresh produce, essential pantry items, and unmatched quality. Every item we offer is carefully selected to meet the highest standards and earn your trust.

From daily essentials to specialty items, we offer everything you need — when you need it. Let GrocerX bring the joy of premium groceries straight to your home

Scenario Based Case Study:

Meet Hema, a busy professional with a hectic schedule who values convenience and efficiency in her daily life. Priya loves to cook and prefers using fresh ingredients in her meals. However, her tight schedule often makes it challenging for her to find the time to visit grocery stores regularly.

- Hema's Solution: The GrocerX Web .

Priya discovers the GrocerX Web , a one-stop solution for all her grocery needs. The app offers a wide selection of high-quality products, including fresh produce, pantry staples, and household essentials, all available at her fingertips.

User Registration and Authentication:Hema registers an account on the app, providing her basic details and preferences. She logs in securely using her credentials, ensuring her privacy and security.

Product Listings:Hema browses through the app's extensive list of products, organized neatly into categories for easy navigation. She can quickly find what she's looking for and add items to her virtual cart with a simple tap.

Personalized Recommendations:The app uses Hema's purchase history and preferences to provide personalized recommendations, helping her discover new products and brands that align with her tastes.

Convenient Delivery Options:Hema can choose to have her groceries delivered to her doorstep at a time that suits her schedule. She can also select express delivery for urgent needs.

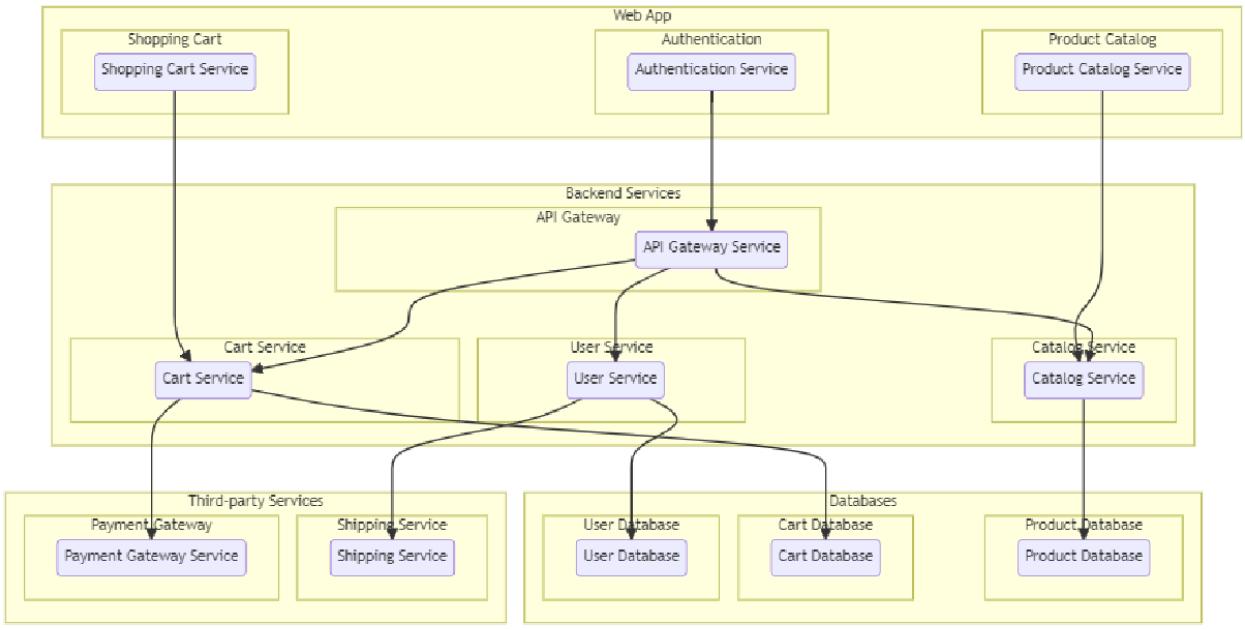
Secure Payment Gateway:The app integrates with a secure payment gateway, allowing Priya to pay for her groceries online using various payment methods, including credit/debit cards and digital wallets.

Order Tracking:Hema receives a confirmation of her order along with a tracking link that allows her to monitor the status of her delivery in real-time.

Customer Support:The app provides excellent customer support, with a dedicated team available to assist Hema with any queries or concerns she may have.

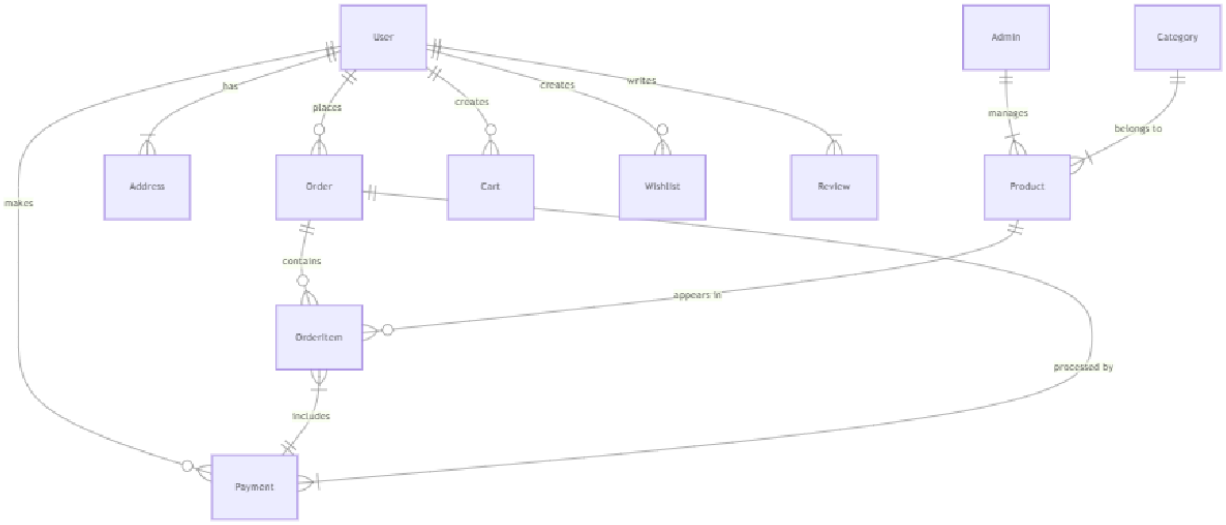
Priya's Experience: Thanks to the GrocerX Web , Hema can now enjoy the convenience of having fresh, high-quality groceries delivered right to her doorstep, saving her time and effort. She can focus on what matters most to her—cooking delicious meals for herself and her loved ones.

Technical Architecture:-



The technical architecture of an grocerX-web 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 grocerX-web 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 grocerX -web 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.

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

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 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

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.

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

Roles and Responsibility

User:-

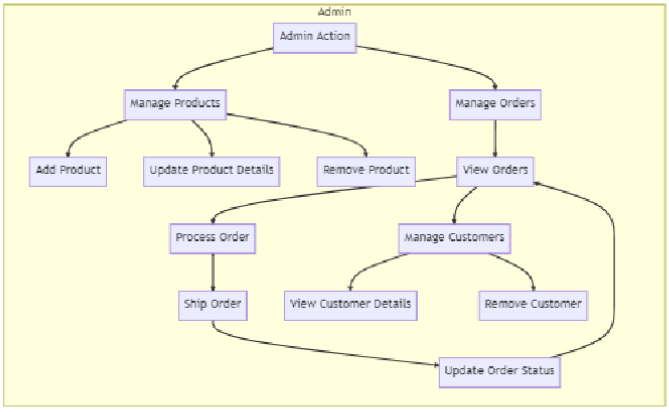
- Registration and Authentication: Users are responsible for creating an account on the platform and securely logging in to access its features.
- Browsing and Shopping: Users can browse products, add them to their cart, and proceed to checkout for purchasing.
- Payment: Users are responsible for making 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.

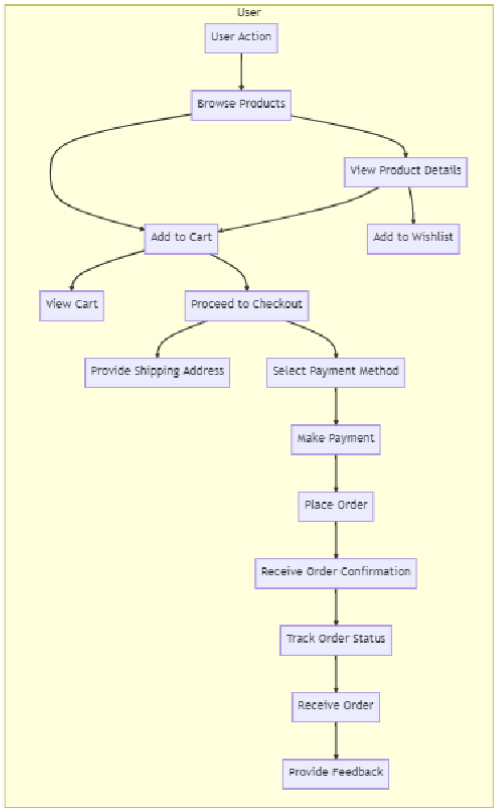
Admin:-

- 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 grocerX-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:-

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.

```
const mongoose = require("mongoose");

const db= 'mongodb://127.0.0.1:27017/grocery'
// Connect to MongoDB using the connection string

mongoose.connect(db, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log(`Connection successful`);
}).catch((e) => {
  console.log(`No connection: ${e}`);
});
```

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.

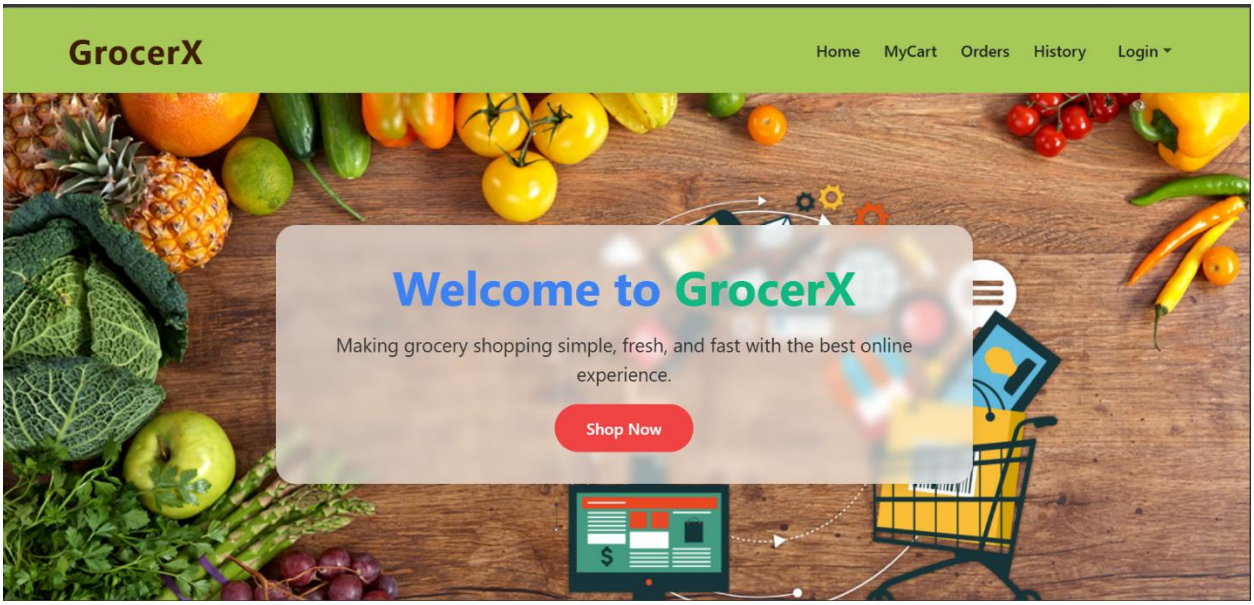
3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

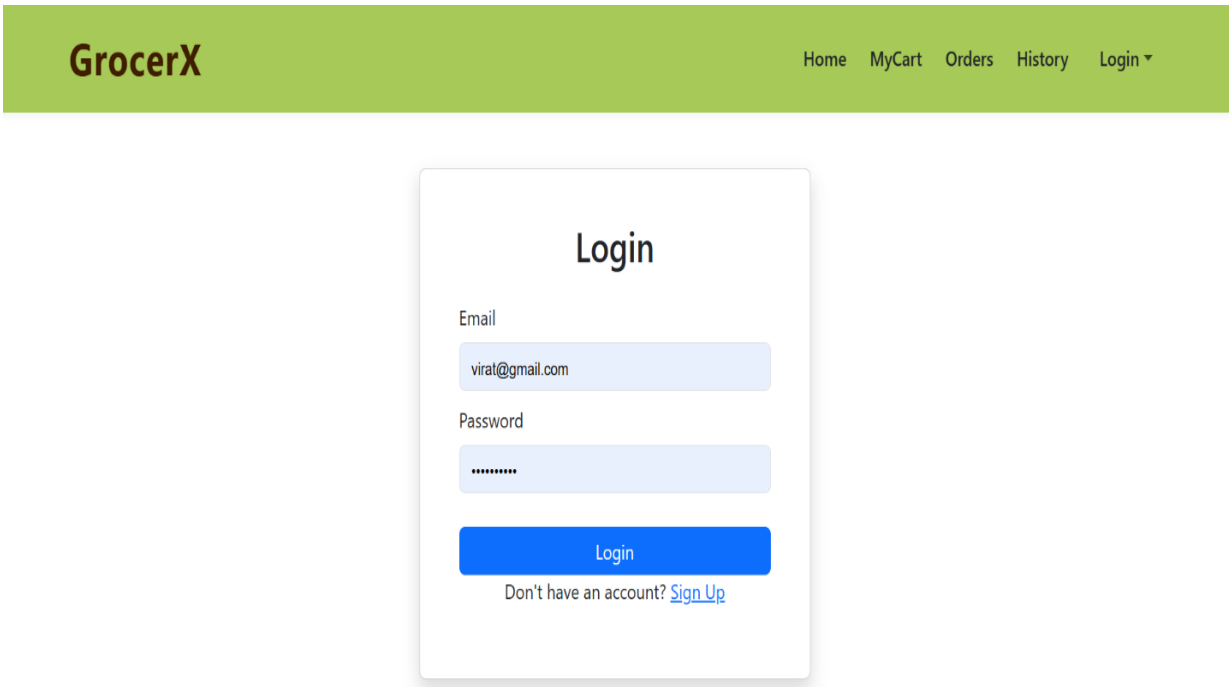
Milestone 5: Project Implementation:

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 GrocerX Web.

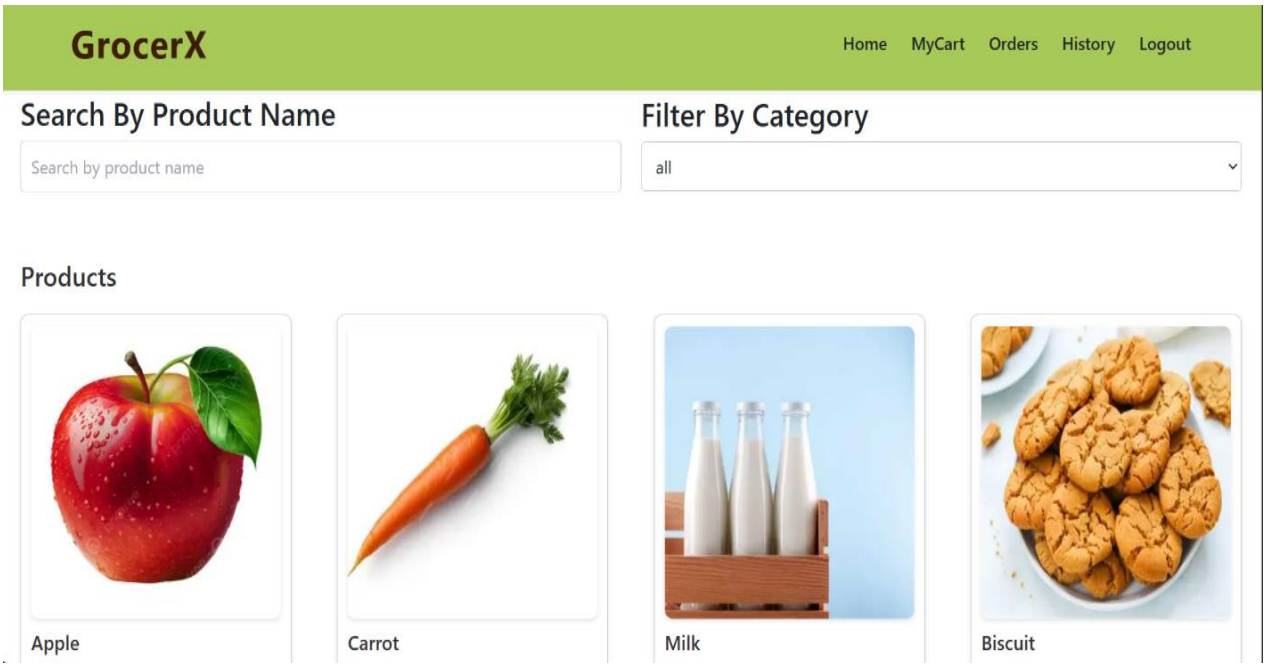
Landing page:-

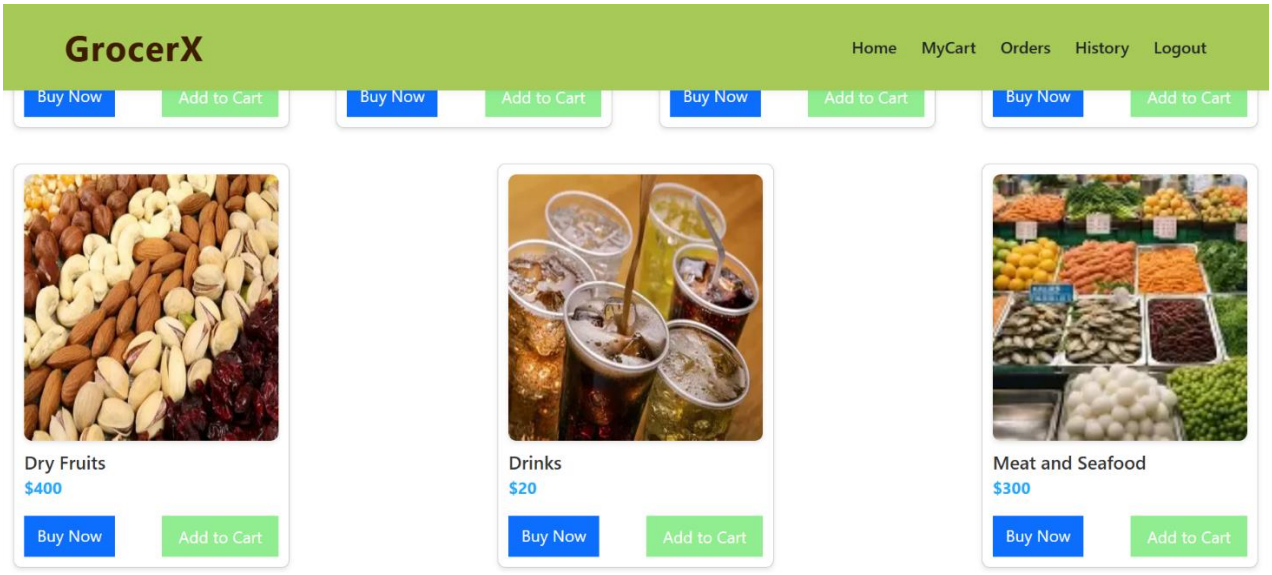


Login Page:-

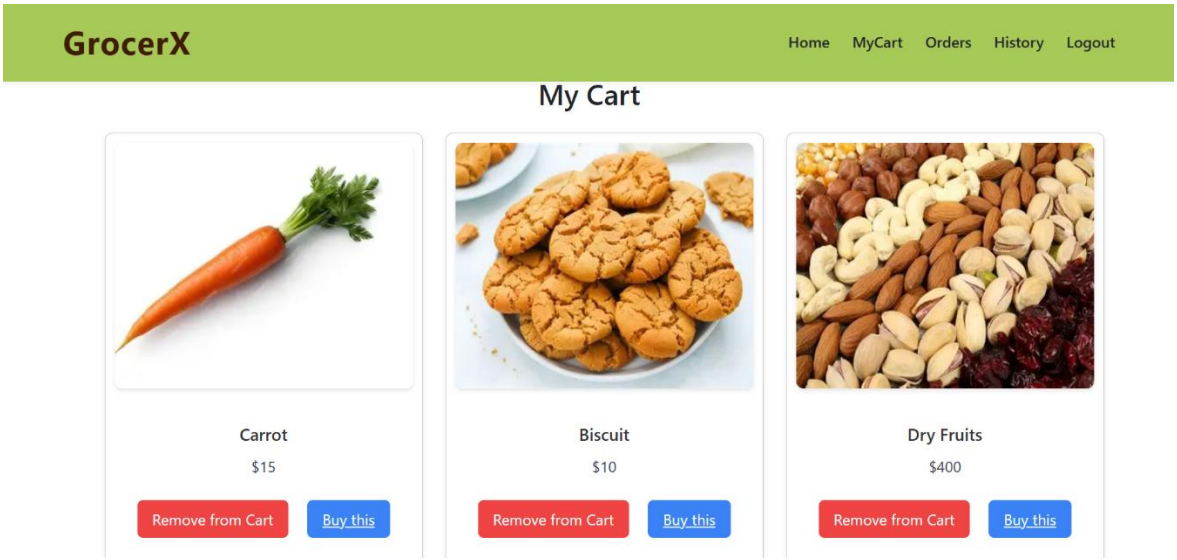


Items Page:-

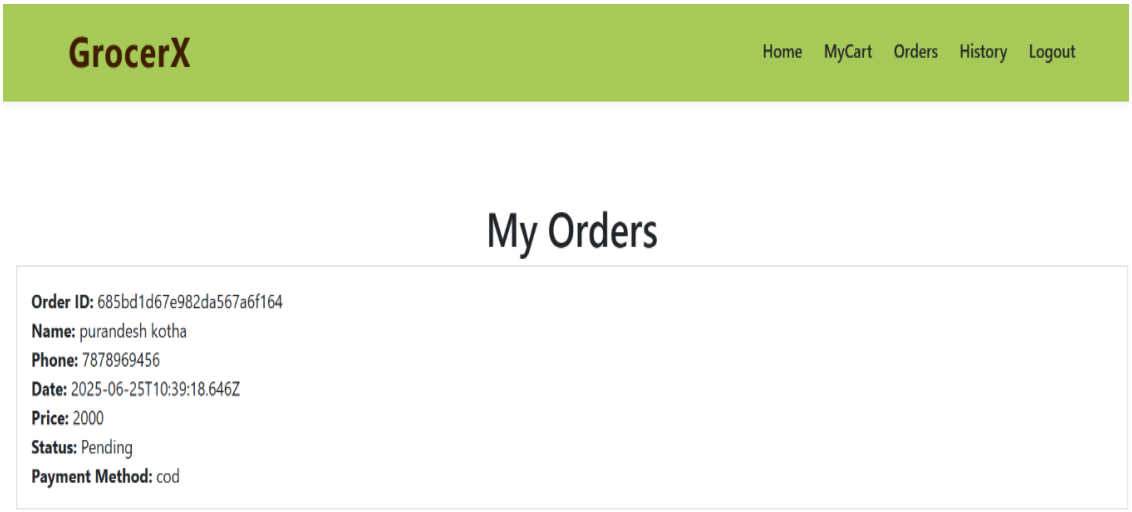

GrocerX
Home
MyCart
Orders
History
Logout
Search By Product Name
Search by product name
Filter By Category
all
Products
Apple
Carrot
Milk
Biscuit


GrocerX
Home
MyCart
Orders
History
Logout
Buy Now
Add to Cart
Dry Fruits
$400
Buy Now
Add to Cart
Drinks
$20
Buy Now
Add to Cart
Meat and Seafood
$300
Buy Now
Add to Cart

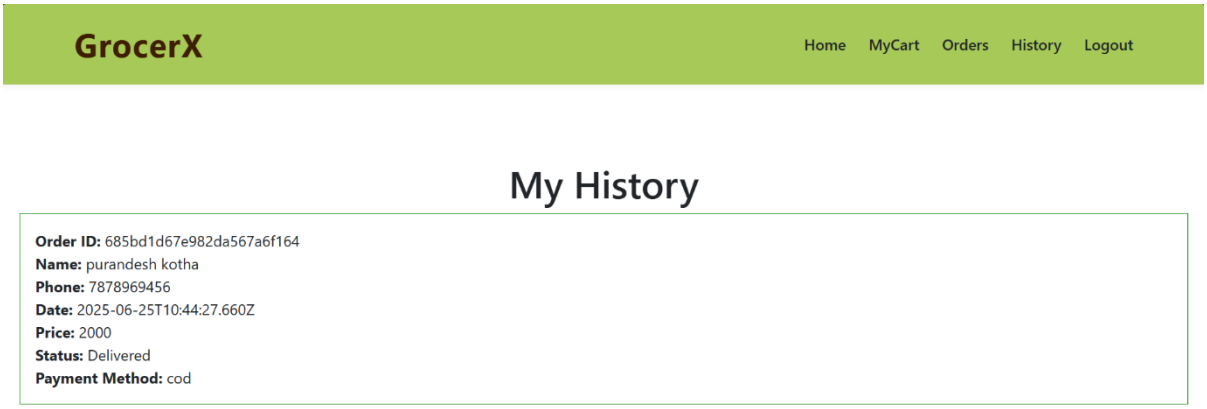
My Cart:-



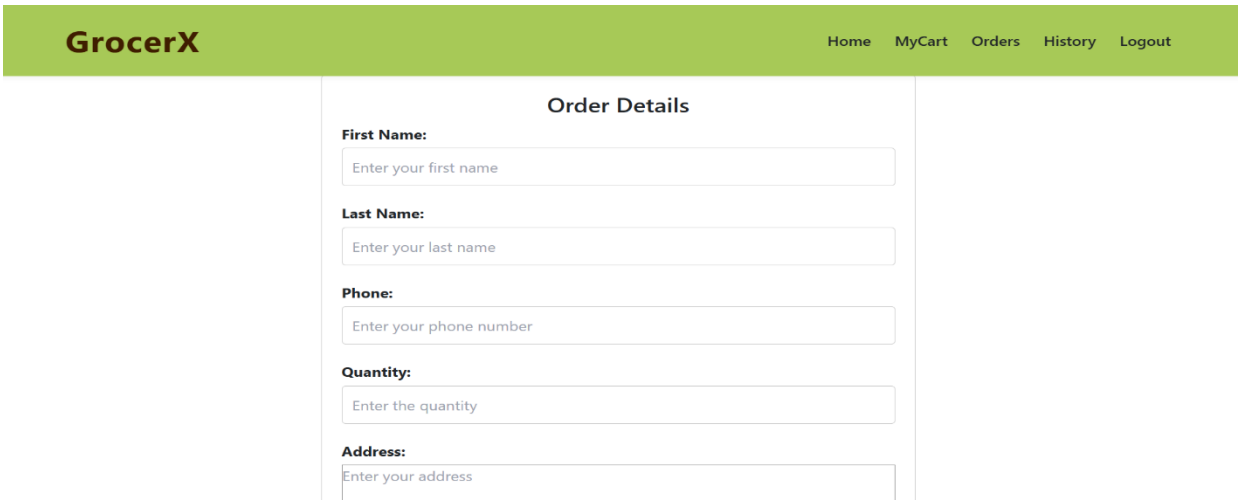
My Orders Page:-



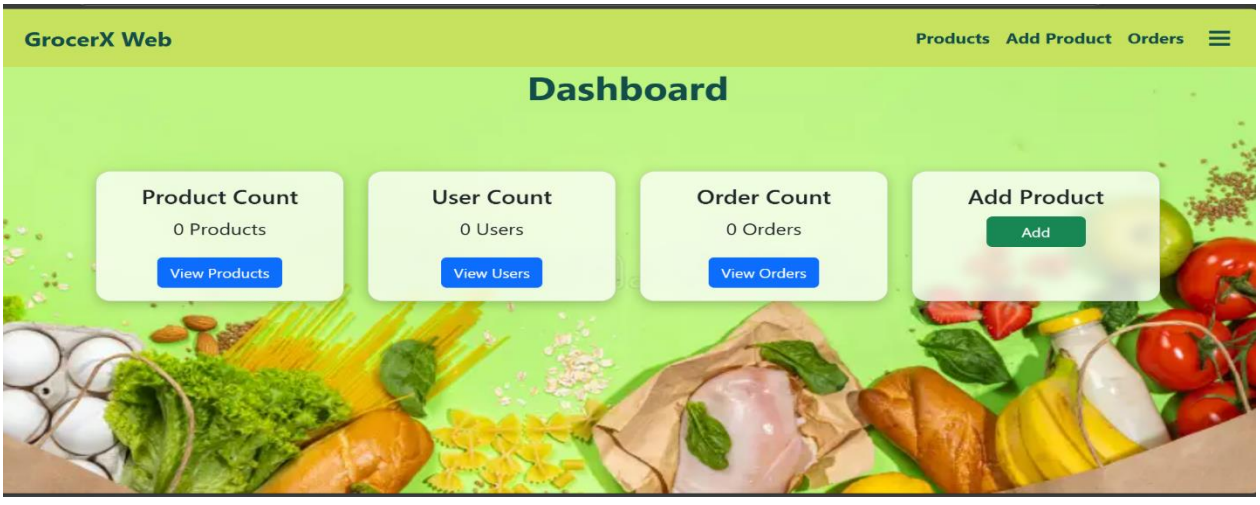
My History Page:-



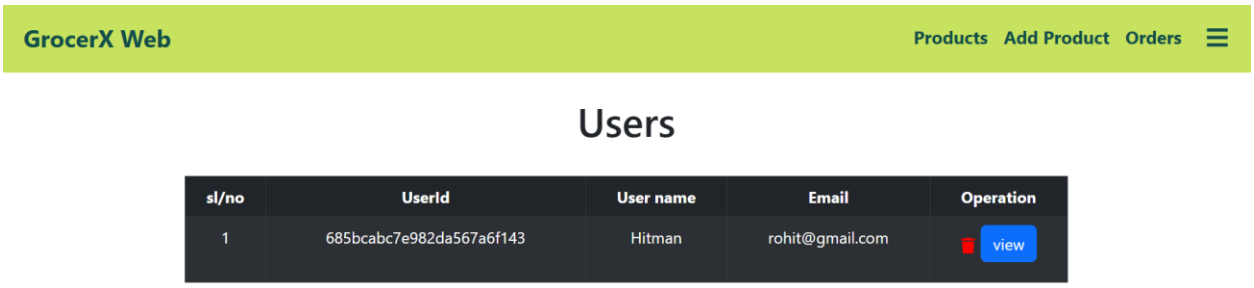
Place Order Page:-



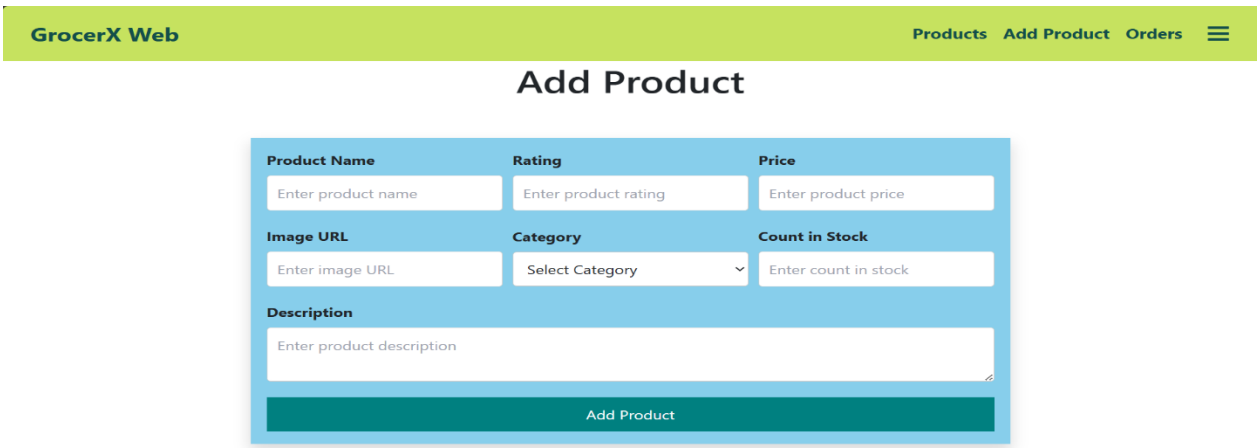
Admin Dashboard Page:



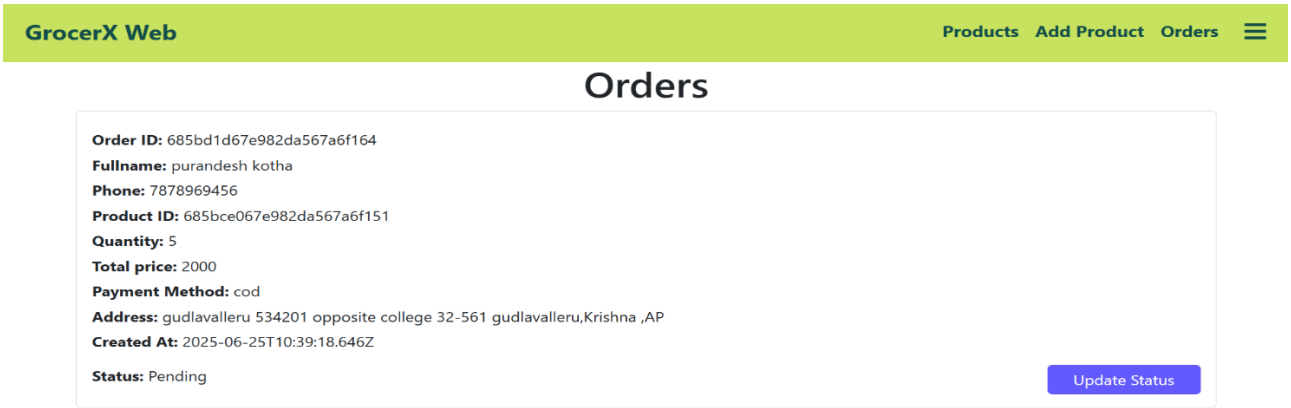
Users Page:-



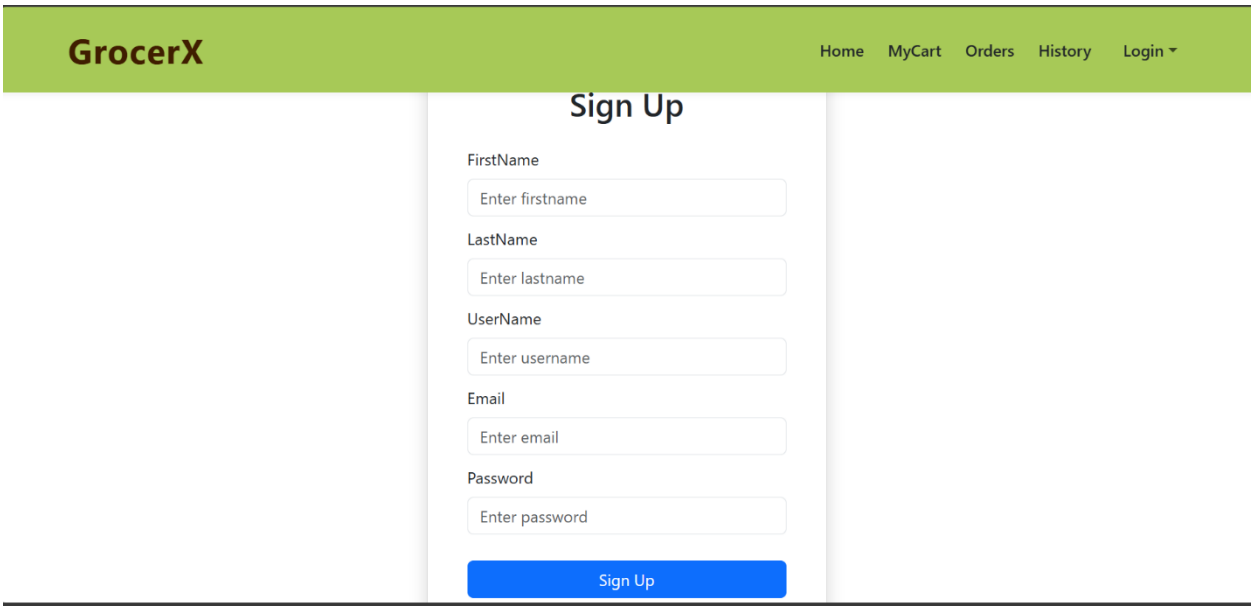
Add Product page:-



Admin Orders Page:-



Sign up:


GrocerX
Home
MyCart
Orders
History
Login
Sign Up
FirstName
Enter firstname
LastName
Enter lastname
UserName
Enter username
Email
Enter email
Password
Enter password
Sign Up