# Grocery web App

# Introduction:

"Welcome to our modern Grocery Web Platform—a smart and efficient way to shop for all your household needs. With an intuitive interface and an expansive catalogue of high-quality items, this web app transforms traditional shopping into a seamless digital experience. Whether it’s fresh fruits, pantry supplies, or daily essentials, users can explore, choose, and receive doorstep deliveries—making life simpler and smarter."

**Description:**

This Grocery Web App is a complete digital shopping experience designed to meet the daily needs of users in the most convenient way possible. It provides a broad selection of grocery items including farm-fresh produce, packaged goods, and household essentials.

With a user-focused layout and smooth navigation, the platform ensures a hassle-free browsing and buying process. Customers can effortlessly search for products, view detailed information, apply filters based on dietary needs (e.g., organic, gluten-free, vegan), and receive personalized suggestions based on past purchases.

The app also includes scheduled delivery features, ensuring users can receive their groceries at the time that works best for them. By removing the need for physical store visits, this platform saves time while maintaining the joy of choosing quality products from anywhere.

**Scenario Based Case Study:**

Scenario-Based Case Study:

Meet Rashmika, a working mother who juggles her professional and personal responsibilities daily. She prefers cooking fresh meals for her family but rarely finds time to visit a supermarket.

Her solution? The Grocery Web App.

Rashmika signs up with just a few clicks and is instantly able to explore neatly categorized product lists. The app learns her preferences, helping her quickly find the items she needs most. With flexible delivery options, she schedules groceries to arrive at times that suit her family routine.

She pays through the secure payment system and gets real-time updates about her delivery. The platform even provides suggestions for new healthy food items she might like.

Thanks to the Grocery Web App, Rashmika now enjoys stress-free grocery shopping and spends more quality time with her loved ones.

**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>

**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:-**

The development of the Grocery Web App is organized into five key milestones, each focusing on a specific phase of the project lifecycle—from setup to deployment. Below is the complete project workflow:

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

* **Install essential tools**: Node.js, MongoDB, React environment.
* **Initialize project structure**: Create separate folders for client (frontend) and server (backend).
* **Install frontend packages**:
  + axios – for API calls
  + react-router-dom – for routing
  + react-icons, bootstrap, react-bootstrap – for UI/UX design
* **Install backend packages**:
  + express, mongoose, cors, dotenv – for API creation and database connection

**🔸 Milestone 2: Backend Development**

* **Set up Express server** with middleware (cors, body-parser)
* **Implement user authentication** using custom routes for login and registration
* **Create modular route files**: for users, products, and orders
* **Build RESTful APIs**: for CRUD operations across users, products, cart, and order management
* **Use Mongoose** to define data schemas and models
* **Secure routes** using middleware for protected user/admin access
* **Handle errors gracefully** using custom error handling middleware

**🔸 Milestone 3: Database Integration**

* **Establish MongoDB connection** using Mongoose
* **Create and organize schemas**:
  + User schema with authentication
  + Product and category schemas
  + Cart and order schemas
* **Verify data structure** aligns with the application ER diagram
* **Seed test data** for development and testing
* **Ensure schema relationships** are working (e.g., references for foreign keys)

**🔸 Milestone 4: Frontend Development**

* **Initialize React app** using vite or create-react-app
* **Design reusable components**: Header, Footer, Product Cards, Cart Page, Login/Register
* **Implement routing** for user pages (Home, Products, Cart, Orders, etc.)
* **Connect frontend with backend APIs** using axios
* **Add dynamic data rendering**: Fetch and display products, user history, order status
* **Style using Bootstrap & custom CSS** for a responsive layout

**🔸 Milestone 5: Project Implementation**

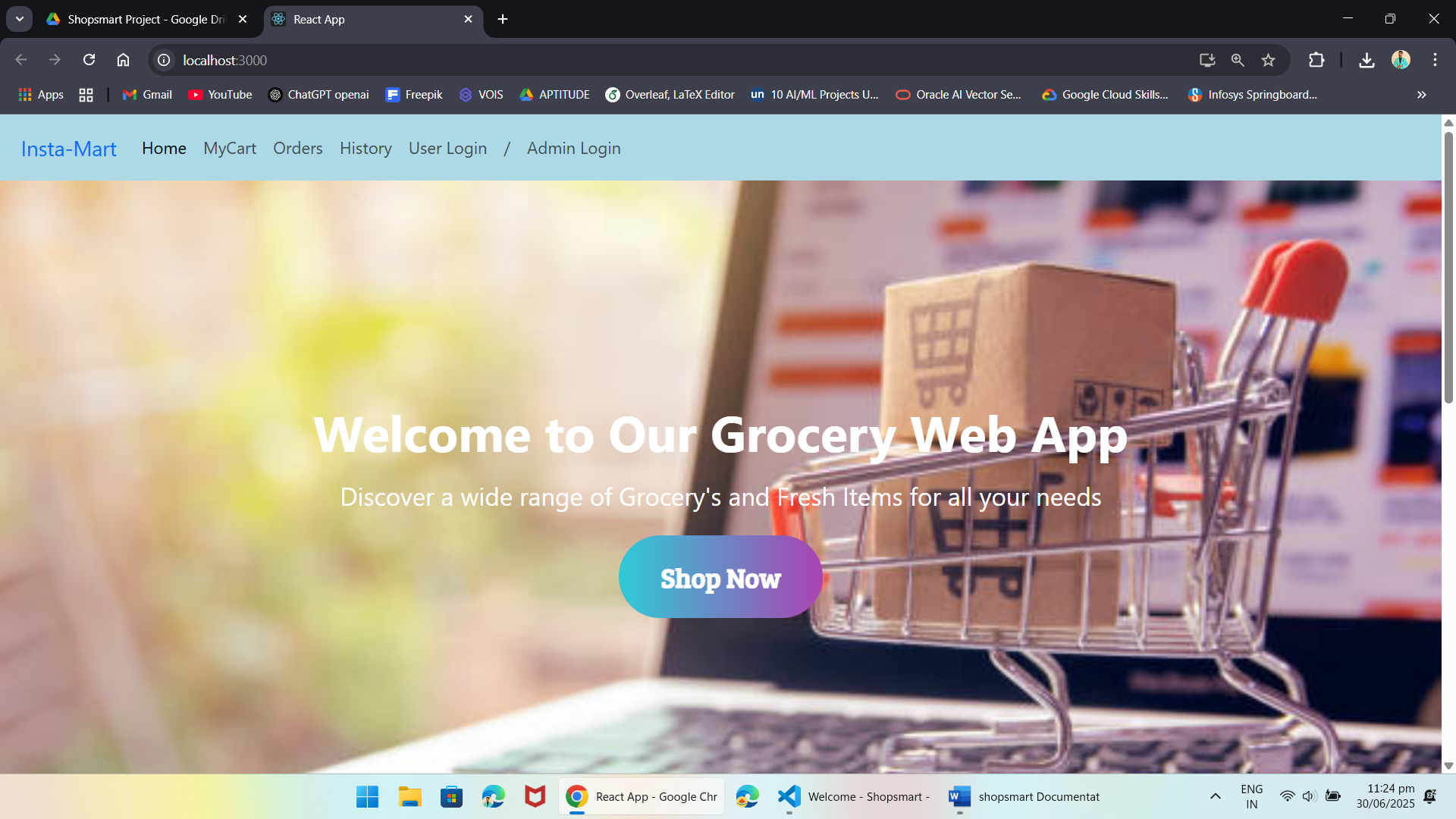
Finally, after finishing coding the projects we run the whole project to test its working process and look for bugs. Now, let’s have a final look at the working of our project.

🎯 **Demo Link**:  
[*https://drive.google.com/drive/folders/1DliXM-cn6b-A78V5RNYv2qEb77qnBRRV?usp=sharing*](https://drive.google.com/drive/folders/1DliXM-cn6b-A78V5RNYv2qEb77qnBRRV?usp=sharing)

📁 **Code Repository**:  
<https://github.com/AkashMunnangi/shopsmart/>

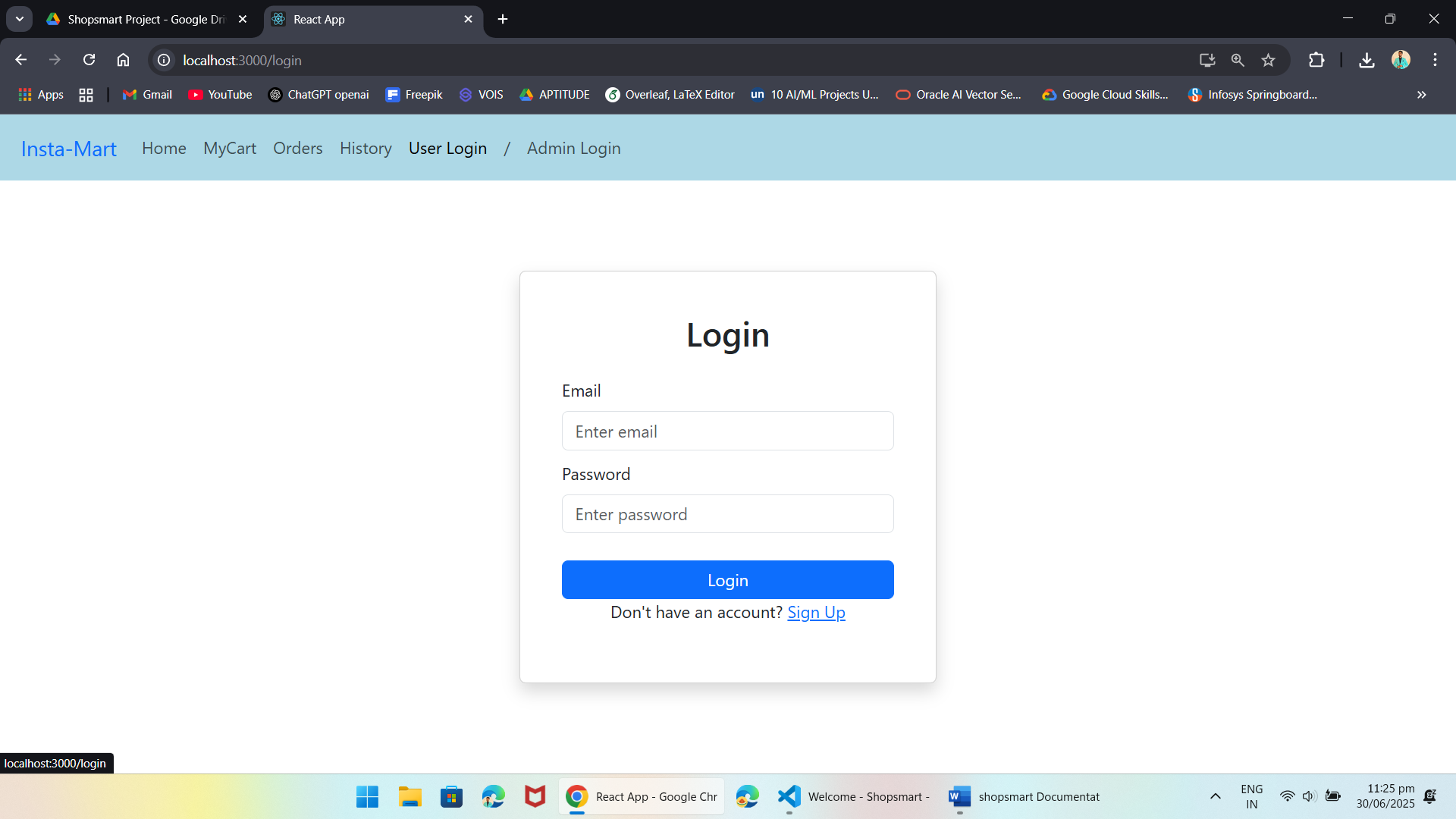
**Output:**

Landing page:

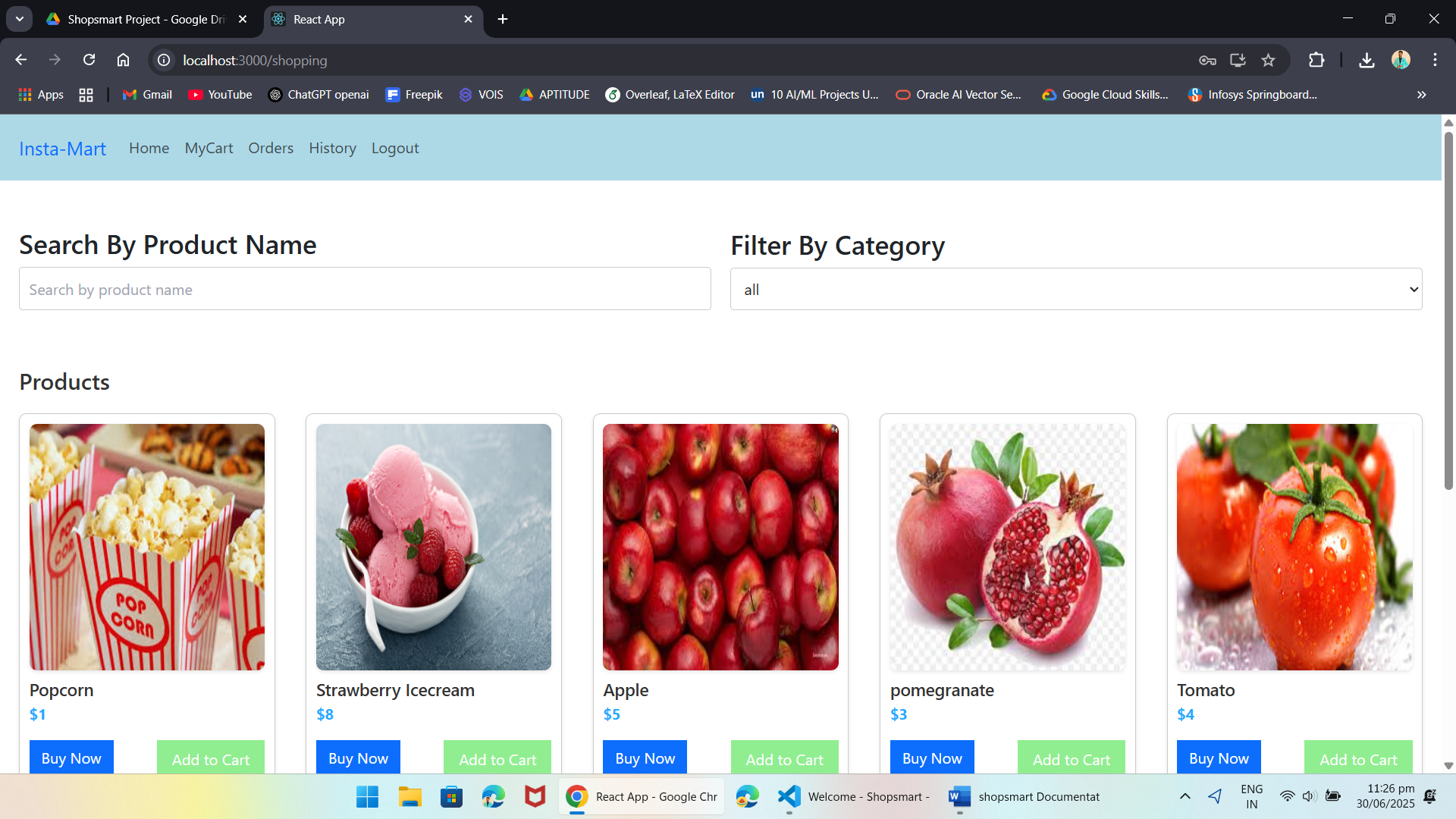


***USER***

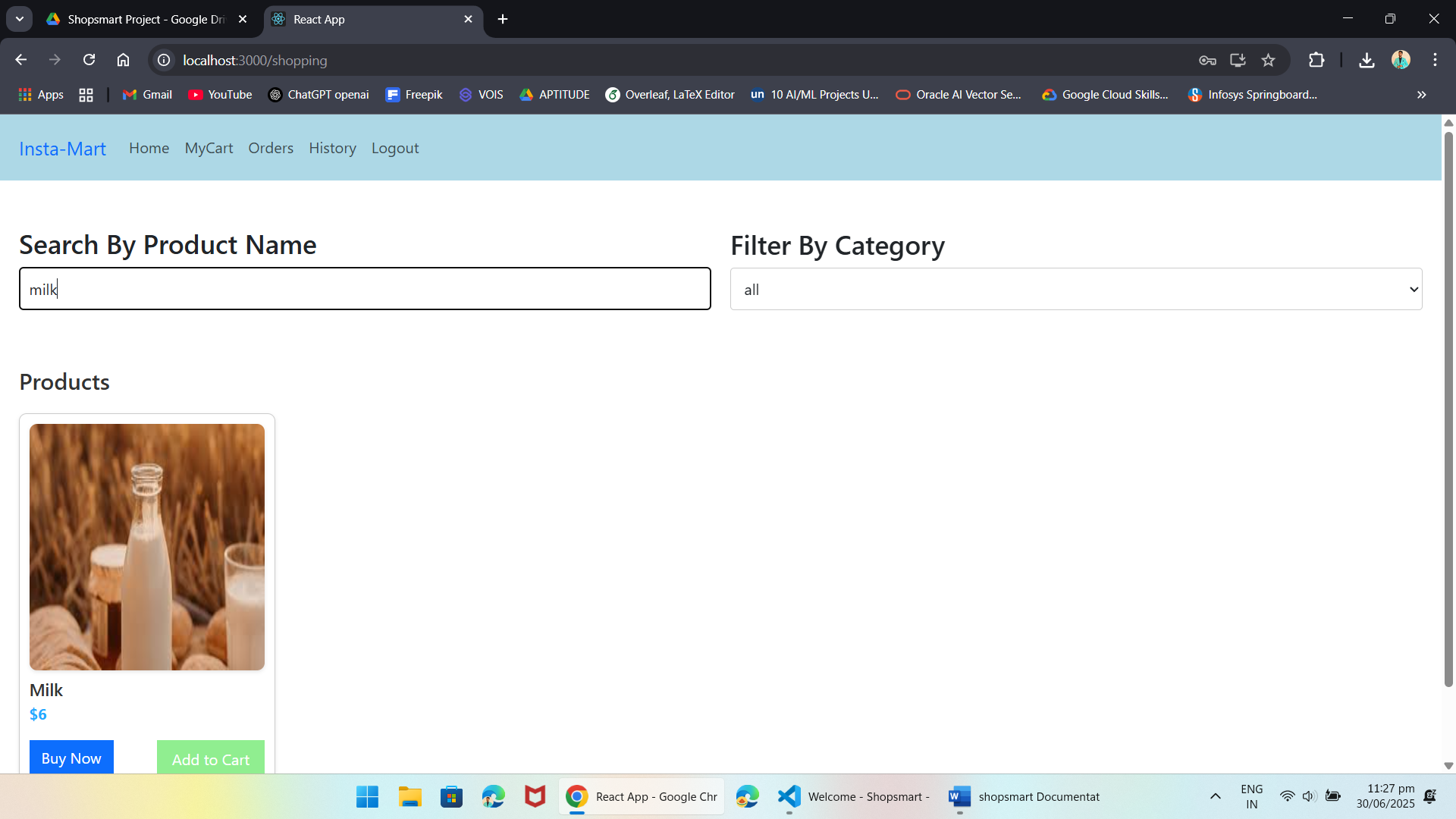
Login Page:



Items Page:



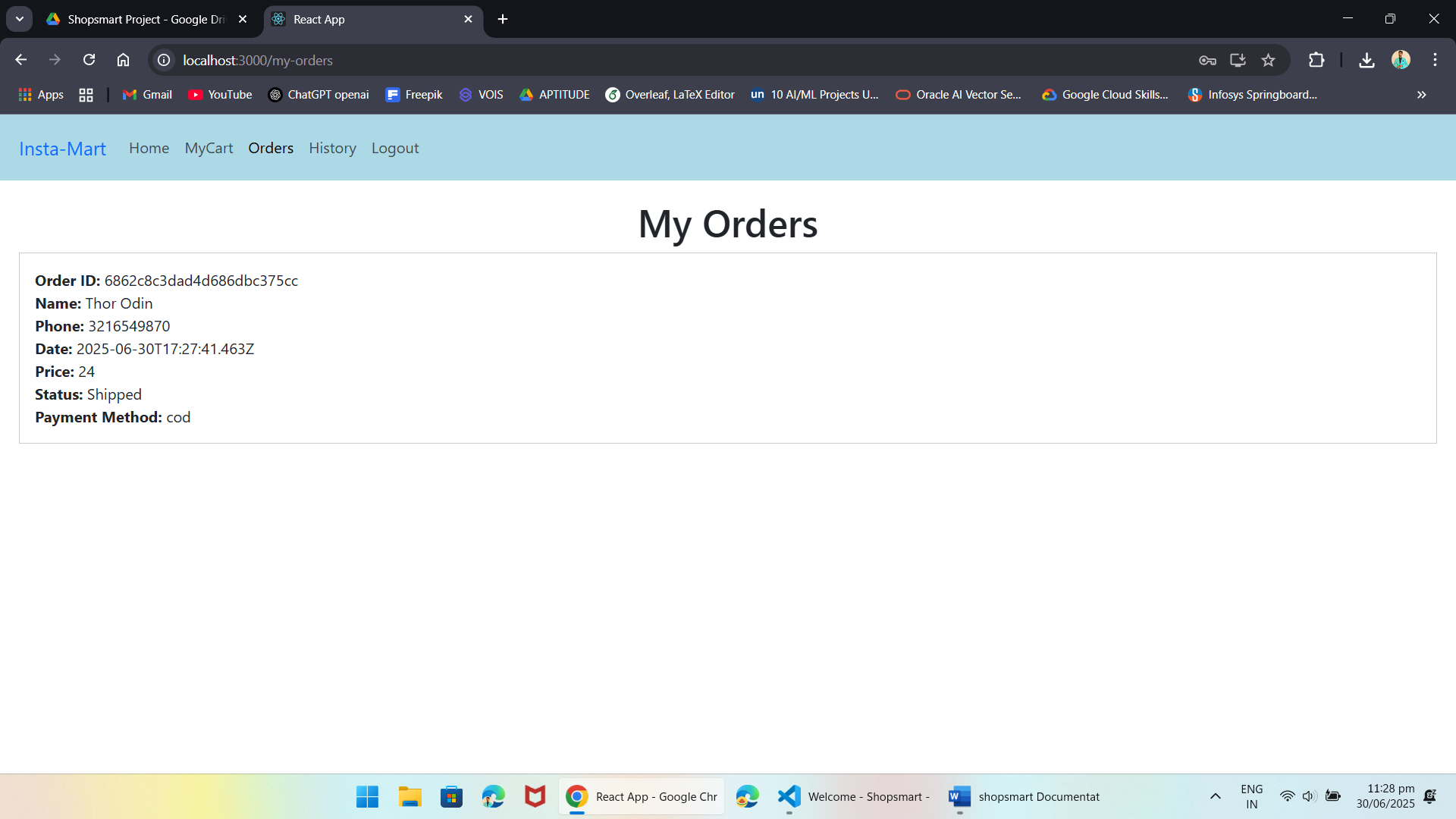
Searching Products:



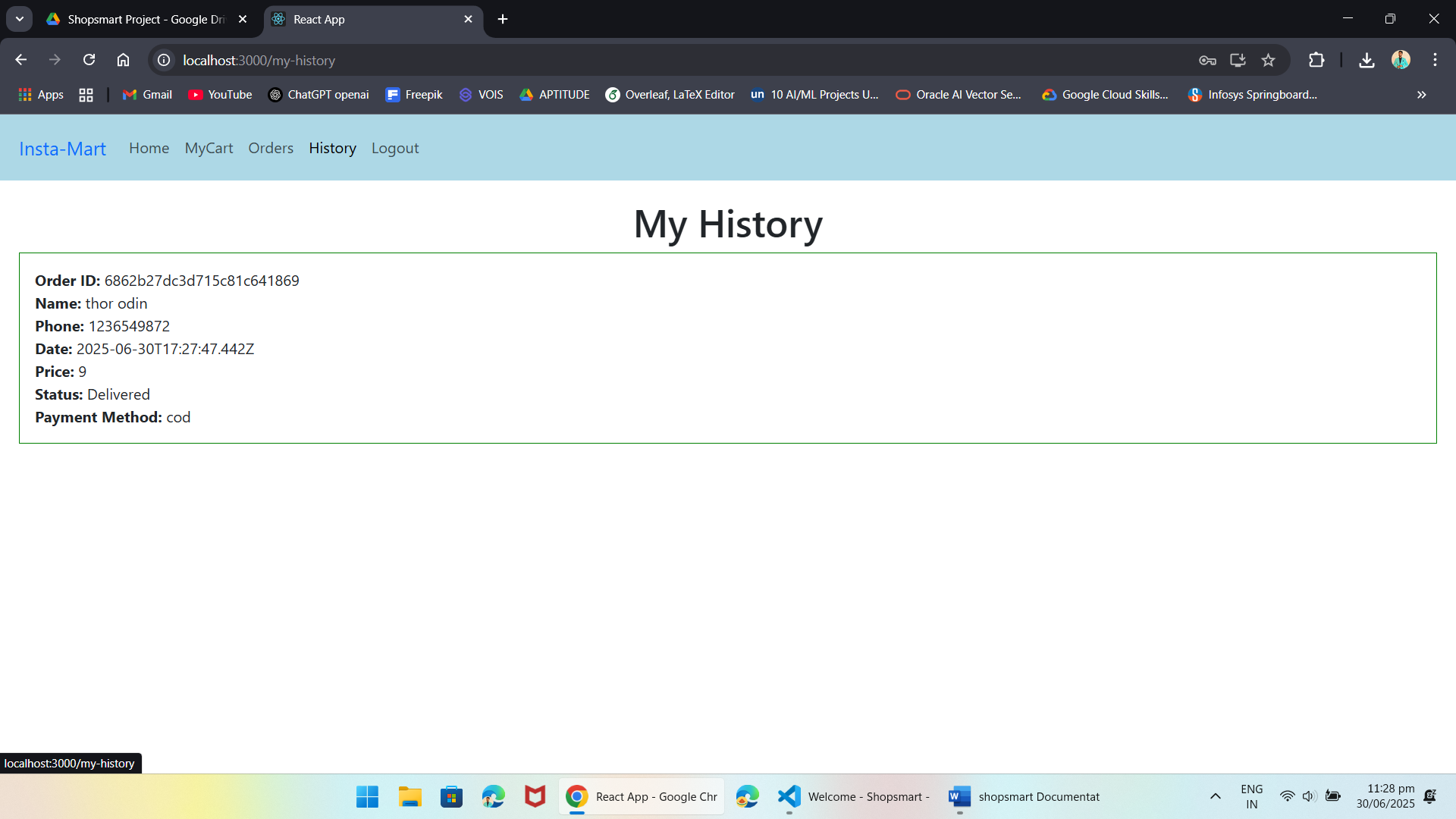
My Cart:



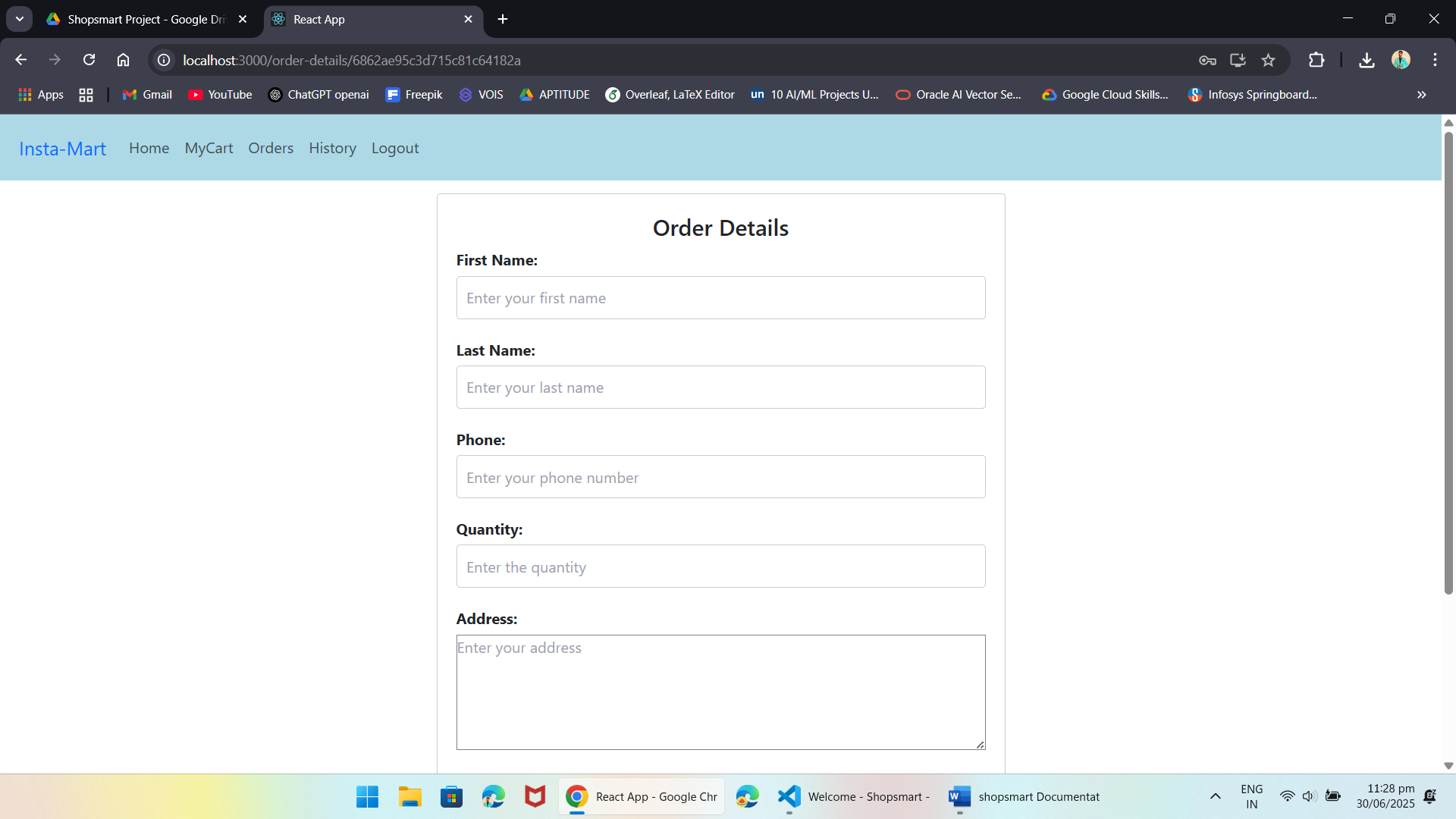
My Orders Page:



My History Page:

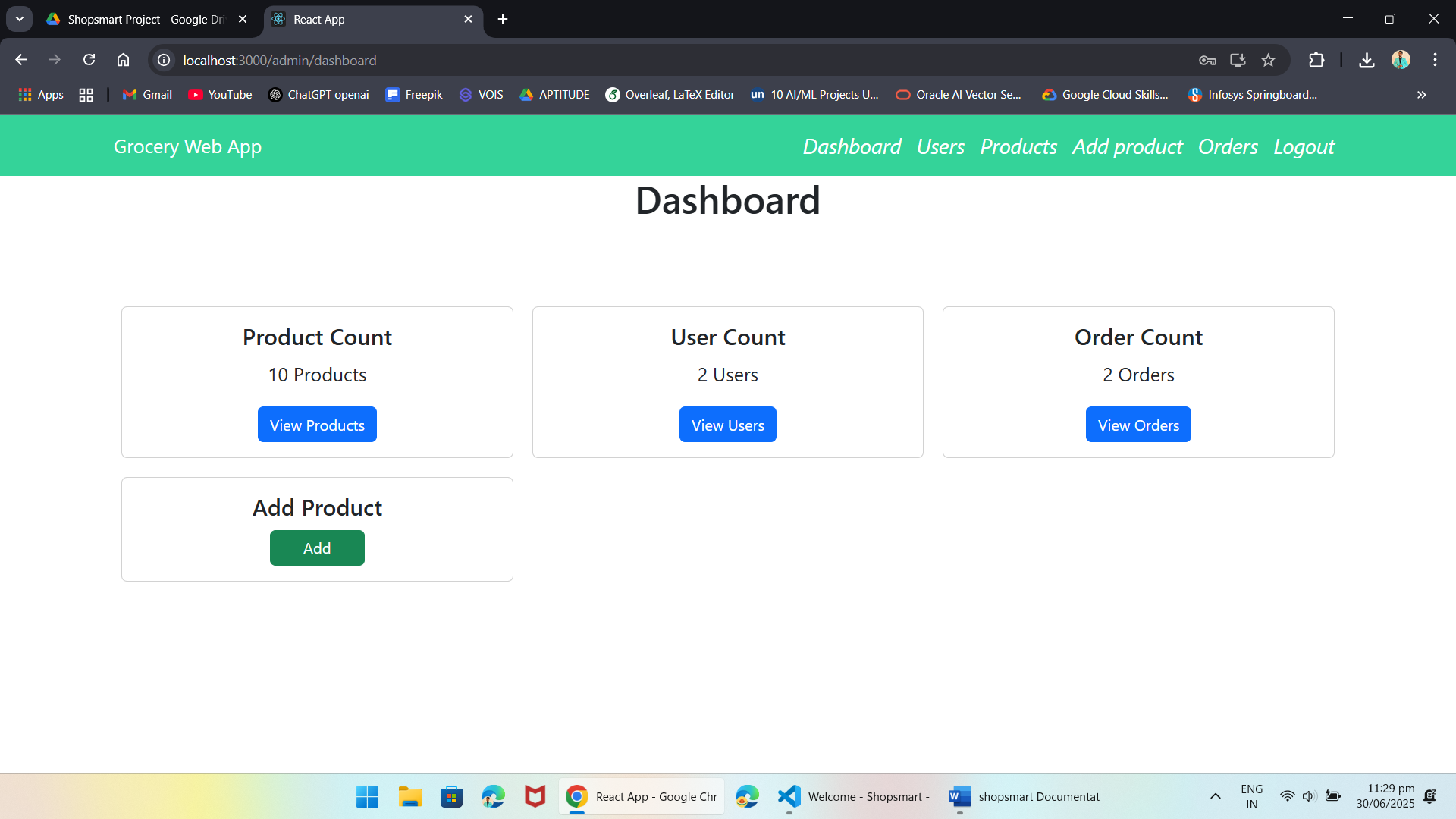


Place Order Page:

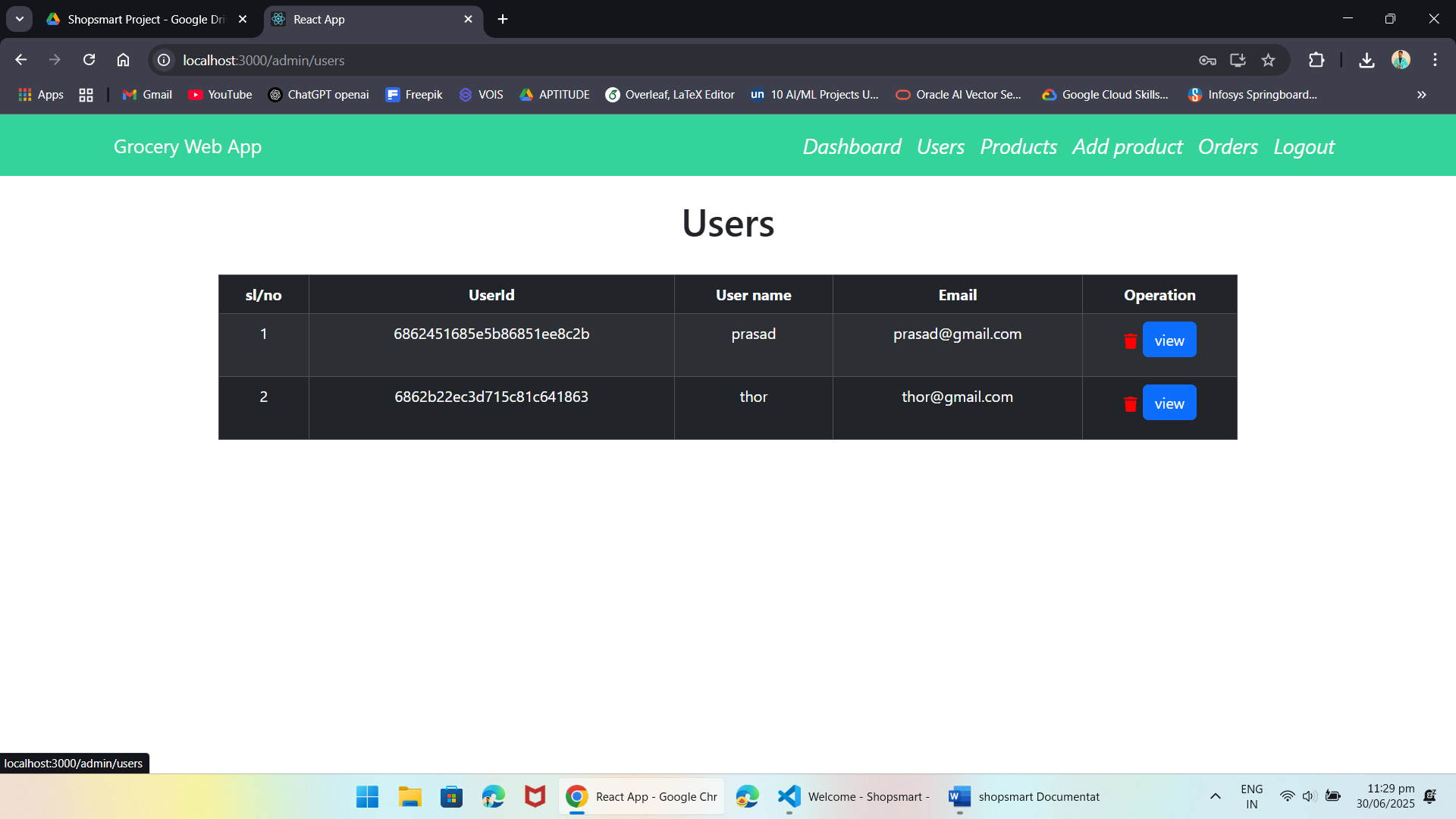


**Admin**

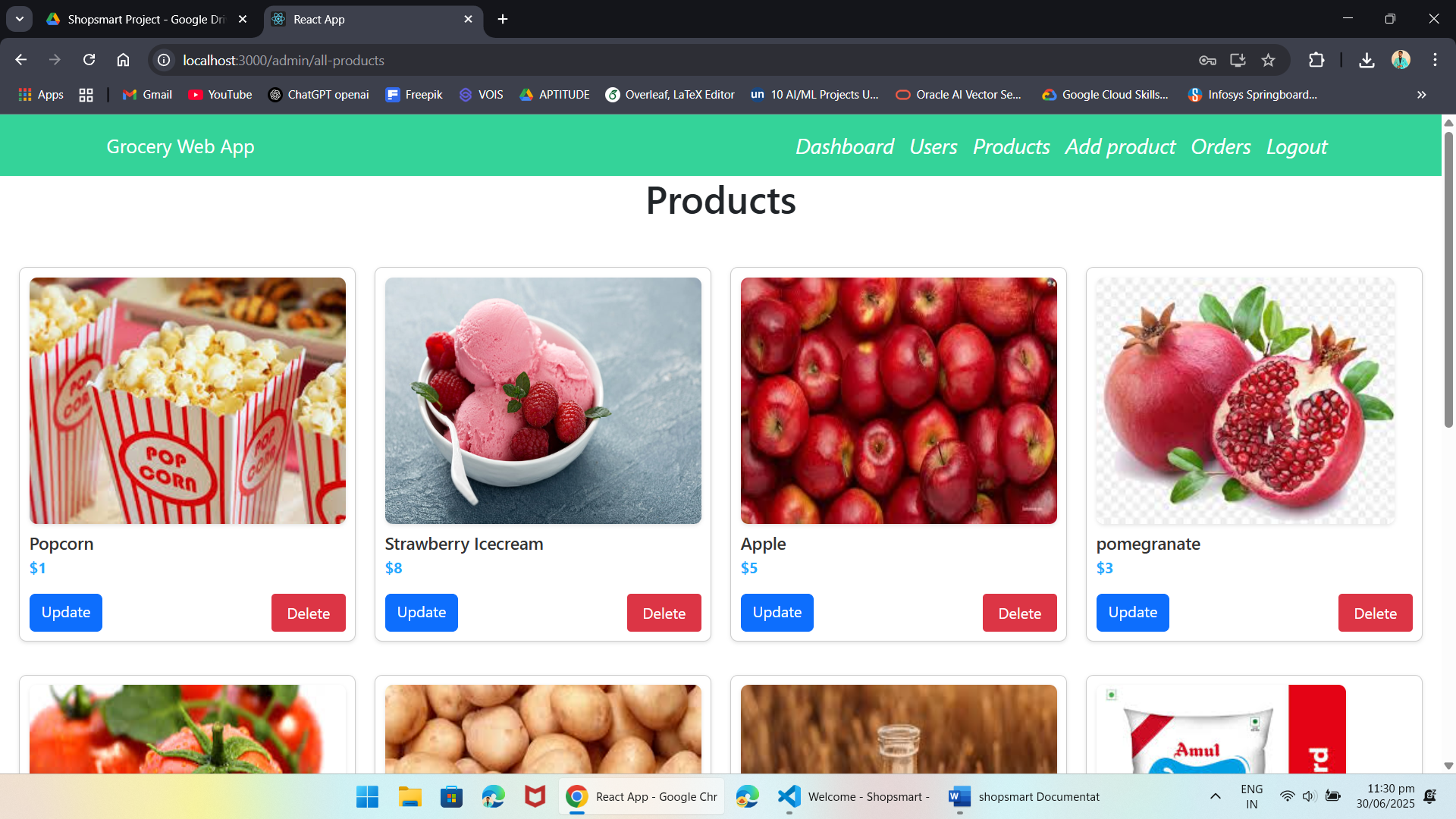
Dashboard Page:



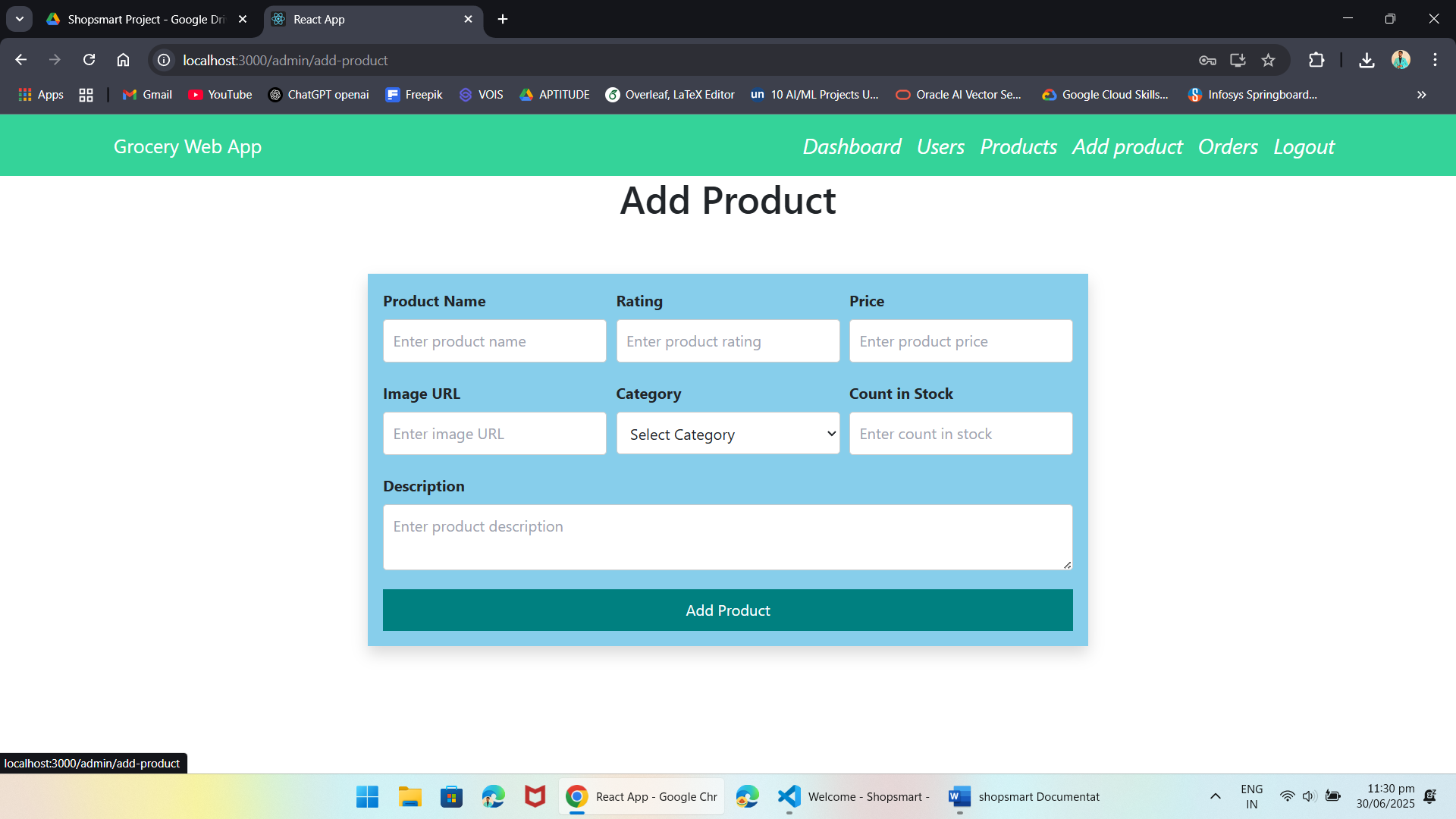
Users Page:



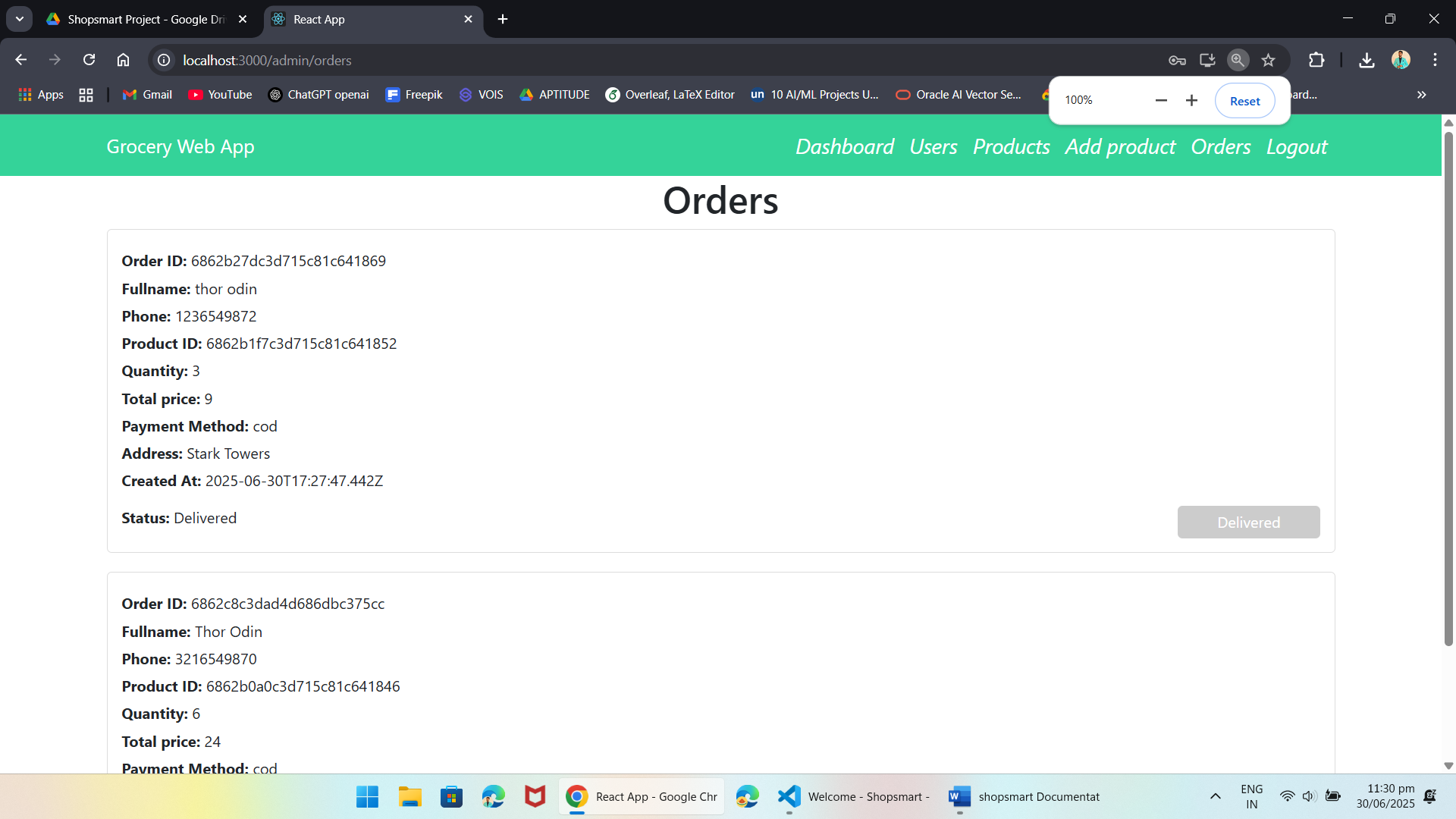
Add Product page:



Add products:



Admin Orders Page:



The demo of the app is available at:

https://drive.google.com/drive/folders/1DliXM-cn6b-A78V5RNYv2qEb77qnBRRV?usp=sharing

Akash Munnangi

Ph. 7382331404