

SHOPSMART: YOUR DIGITAL GROCERY EXPERIENCE

Introduction:

ShopSmart is a full-featured online grocery web application built with React.js on the frontend and Node.js with MongoDB on the backend. It enables users to browse, search, and order products seamlessly, while providing sellers and admins robust tools for product management and order tracking. The platform is designed for reliability, scalability, and security.

Key Features

For Users

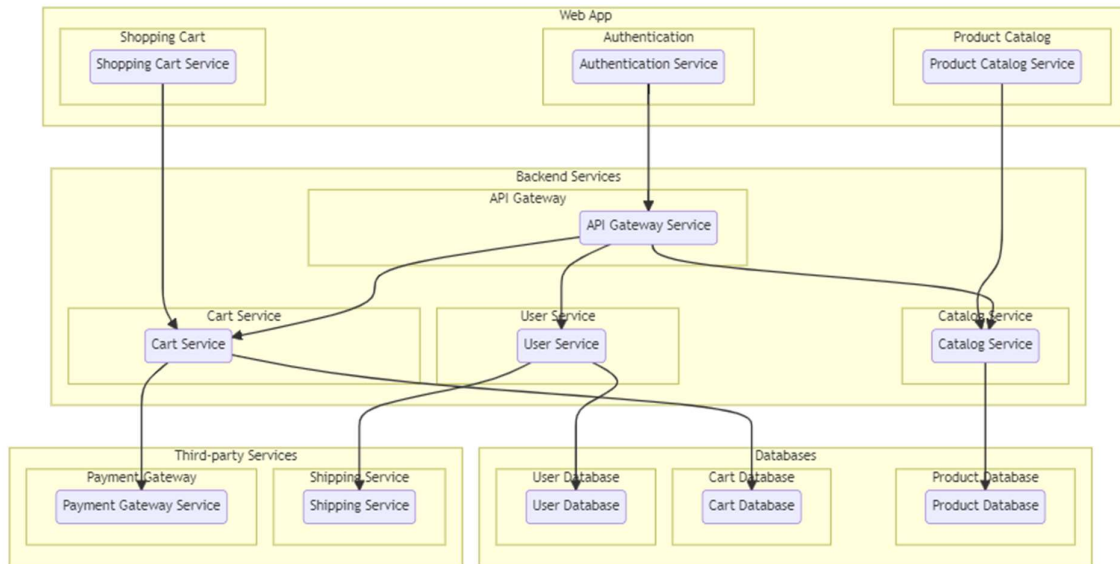
- **Product Search & Browse:** Easily browse through categorized products.
- **Add to Cart & Order:** Manage cart, place orders, and track their status.
- **Wishlist & Reviews:** Add products to a wishlist and leave reviews.
- **Manage Address:** Add or update delivery addresses.
- **Secure Payments:** Integrated payment options including gateways and COD.

For Admins

- **Dashboard:** Centralized admin dashboard for managing all operations.
- **Product Management:** Add/edit/delete categories and products.
- **Order Management:** View all orders, update status, and manage shipping.
- **User Management:** View users, handle roles and feedback.

Architecture

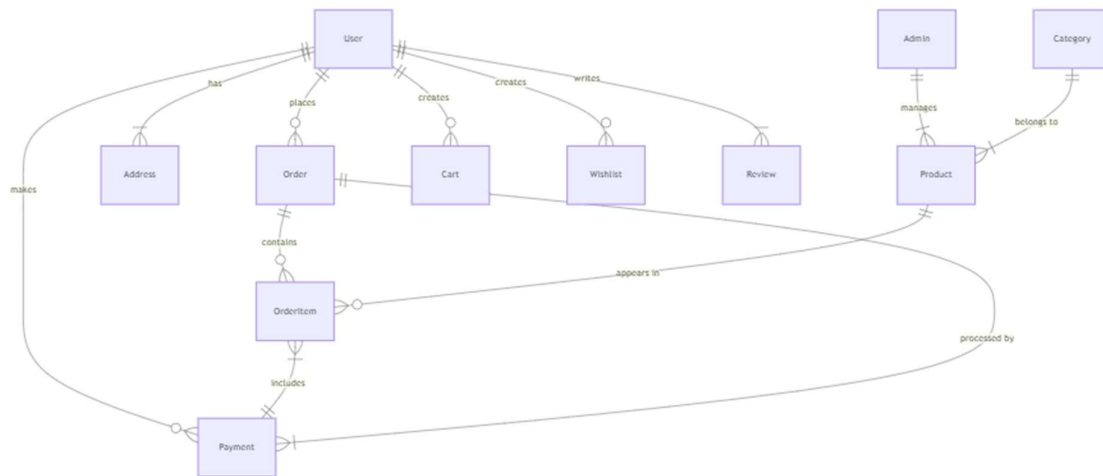
Technical Architecture:



The technical architecture of an flower and gift delivery 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.

The backend includes microservices for user, cart, and product management. These services connect to separate databases and interact with third-party systems. An API gateway handles client requests and directs them to appropriate services, ensuring modularity and scalability.

ER Diagram:



The Entity-Relationship (ER) diagram for an flower and gift delivery app visually represents the relationships between different entities involved in the system, such as users, products, orders, and reviews. It illustrates how these entities are related to each other and helps in understanding the overall database structure and data flow within the application.

The ER diagram also highlights key actions like placing orders, managing carts, writing reviews, and making payments. It shows how users interact with products and how admins manage categories and inventory. This visual structure supports efficient database design, ensuring smooth data management, consistency, and easier backend development throughout the application.

Project Structure:

Pre-requisites

To run and develop this MERN (MongoDB, Express.js, React.js, Node.js) stack-based project, the following tools and software must be installed and set up properly.

1. Node.js & npm (Node Package Manager)

Purpose:

Used to run JavaScript code on the server and manage project dependencies.

How to Install:

1. Download from: <https://nodejs.org>
2. Install the **LTS (Long-Term Support)** version.
3. After installation, check versions in terminal:

```
node -v
```

```
npm -v
```

2. MongoDB (Database)

Purpose:

Used to store all backend data like users, products, orders, and reviews.

Options:

- **Local MongoDB:** Install it directly on your system.
- **MongoDB Atlas (Cloud):** Use a free cloud-hosted database.

How to Use Locally:

1. Download from:
<https://www.mongodb.com/try/download/community>
2. Install and start the MongoDB server: mongod

To Connect (as shown in backend code):

```
const db = 'mongodb://127.0.0.1:27017/grocery';
```

3. Express.js

Purpose:

Express.js is a web application framework for Node.js that helps build fast, scalable, and minimal backend web applications and APIs.

Key Features:

- Lightweight and flexible
- Supports routing, middleware, and HTTP methods
- Easy integration with MongoDB using Mongoose
- Perfect for building RESTful APIs

4. React.js

Definition:

React.js is a **JavaScript library** developed by Facebook for building dynamic and interactive user interfaces (UIs), especially for single-page applications (SPAs).

Key Features:

- Component-based architecture
- Fast rendering using virtual DOM
- One-way data binding
- Reusable UI components

Usage in Project:

- Used to build the entire frontend of the app
- Manages pages like Home, Login, Register, Product List, Cart, Orders, Admin Panel, etc.
- Interacts with backend APIs using axios or fetch

5. Visual Studio Code (VS Code)

Purpose:

Used as the main code editor for writing and debugging frontend/backend code.

How to Install:

- Download from: <https://code.visualstudio.com>
- Install essential extensions like:
 - ESLint
 - Prettier
 - MongoDB (optional)

6. Git & GitHub (for version control and cloning)

Purpose:

Used to clone the project and manage code versions.

Steps to Clone the Project:

1. Install Git: <https://git-scm.com/downloads>
2. Clone the project:

```
git clone <your-project-git-url>
cd your-project-folder
```

7. Install Project Dependencies

Each project folder (client and server) has its own package.json. You must install dependencies in both.

Step 1 – Install backend packages:

```
cd server
npm install
```

Step 2 – Install frontend packages:

cd client

npm install

6. Run the Project

Start Backend:

cd server

node index.js // or nodemon index.js (if installed)

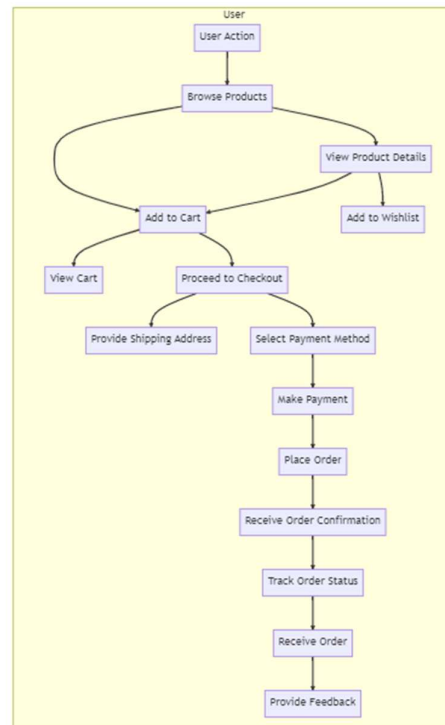
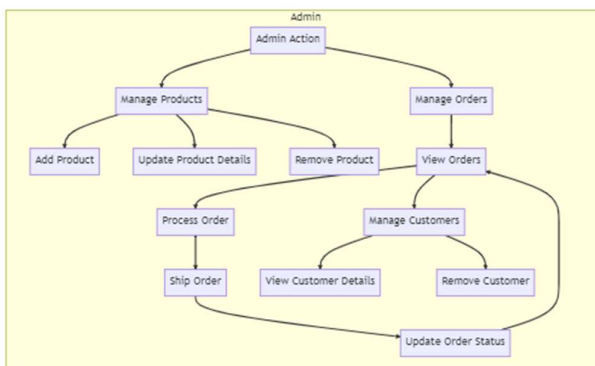
Start Frontend:

cd client

npm start

Role Based Access

Roles of Admin and User can be defined for an online grocery web application.



Admin Role

- Responsibilities: The admin role has full control and administrative privileges over the system.
- Permissions:
 - Manage: Admins can add, edit, and delete shop information along with products.
 - Manage product bookings: Admins can view and manage all product bookings made by users and agents, including canceling or modifying product bookings.
 - Manage users: Admins can create, edit, and delete user accounts, as well as manage their roles and permissions.
 - Generate reports: Admins have access to generate reports and analytics on product booking details, booking counts, and sales reports.

User Role

- Responsibilities: Users are the customers of the online shopping web application who can search for products, and make product bookings.
- Permissions:
 - View products: Users can search for products, based on interest.
 - Product bookings: Users can select products that are available and complete the order process.
 - Manage product booking: Users can view their own product order bookings, modify booking details, track booking details, and cancel their bookings
 - Manage cart: Users can view their cart details and modify them if needed.

The image displays two side-by-side screenshots of the Visual Studio Code Explorer sidebar, illustrating the file structure of a project. The left screenshot shows a project with a folder named 'GROCERY' containing subfolders 'grocery-shop', 'frontend', 'src', 'Pages', and 'Home'. The right screenshot shows a similar structure but with more files visible, including 'categoryController.js', 'orderController.js', 'productController.js', 'reviewsController.js', 'userController.js', 'middleware', 'models', 'public/uploads', 'routes', 'utils', and 'index.js'.

Project Flow

Frontend Development

Frontend development involves building the user interface (UI) and implementing the visual elements of the online shopping web application. It focuses on creating an intuitive and engaging user experience that allows users to interact with the application seamlessly.

Setup the frontend development and to connect node.js with MongoDB Database

```
PS C:\Users\dkaru\OneDrive\Documents\grocery> cd grocery-shop
PS C:\Users\dkaru\OneDrive\Documents\grocery\grocery-shop> npm start

> grocery-shop@1.0.0 start
> node backend/index.js

Loaded MONGODB_URI: mongodb+srv://vamsi:vamsi143@cluster0.s6mqhgm.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
URI mongodb+srv://vamsi:vamsi143@cluster0.s6mqhgm.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
Server Running At http://localhost:4000
App Is Connected To Database Successfully....!!
```

User Interface (UI) Design

The User Interface (UI) Design of the ShopSmart website focuses on providing a clean, intuitive, and user-friendly experience. The design includes a well-structured layout with clearly defined sections such as home, product listing, cart, and order history. Navigation is simple, allowing users to quickly browse products, view details, and manage their orders without confusion. Visual hierarchy and consistent styling help users understand actions and content easily.

Color schemes, icons, and buttons are chosen for clarity and accessibility, ensuring the platform is usable for all. The responsive design adapts seamlessly to different screen sizes, providing a consistent experience across devices.

Responsive Design

The responsive design of the **ShopSmart** website ensures that the user interface adjusts seamlessly across various devices, including desktops, tablets, and smartphones. This adaptability improves accessibility and enhances the user experience by maintaining layout consistency and usability, regardless of screen size.

Key elements such as the navigation bar, product cards, cart, and order sections are designed using flexible grids and media queries. This approach ensures that text remains readable, buttons are easily clickable, and images are properly scaled. The responsive design not only boosts customer satisfaction but also makes the platform more accessible and professional across all platforms.

Product Catalog

The **Product Catalog** on the ShopSmart website is designed to offer a clean and organized view of all available grocery items. Products are displayed in structured categories with clear images, names, prices, and essential details, allowing customers to browse and select items with ease.

Search and filter options enhance user convenience by helping them quickly find specific products based on name, category, or price range. Each product card includes an “Add to Cart” button for a smooth shopping experience. The catalog design ensures quick load times, clarity, and accessibility across all devices through responsive layout techniques.

Additionally, the Product Catalog integrates seamlessly with the backend database, ensuring real-time updates on product availability, stock status, and pricing. Admins can easily manage listings—adding, editing, or removing products—via the admin dashboard, keeping the catalog accurate and up to date. The design focuses on simplicity and efficiency, enabling users to make informed purchasing decisions quickly. Visual consistency, intuitive navigation, and mobile responsiveness make the

catalog user-friendly for all shoppers, enhancing overall engagement and satisfaction on the ShopSmart platform.

Shopping Cart and Checkout Process

The **Shopping Cart** feature in the ShopSmart web application offers a streamlined and efficient way for users to manage their selected products. Once a user adds items to their cart, they can view a detailed list displaying product names, quantities, prices, and total costs. The cart also provides easy options to update item quantities or remove unwanted products. All actions are updated dynamically, ensuring users see real-time changes without needing to refresh the page. This interactive cart experience enhances usability and ensures transparency during the purchase journey.

The **Checkout Process** is designed to be simple, secure, and user-friendly. After finalizing the items in the cart, users can proceed to checkout, where they are prompted to provide essential details such as shipping address and preferred payment method. ShopSmart supports multiple payment options, including Cash on Delivery and digital transactions, allowing flexibility based on user preferences. All transactions are handled securely, and users receive a confirmation once the order is successfully placed. The entire flow is designed to minimize clicks and reduce friction, ensuring a smooth checkout experience even for first-time users.

To further improve user trust and convenience, the system also allows users to view past orders and track the status of ongoing deliveries. Any errors during the checkout process—such as invalid payment inputs or missing address fields—are highlighted clearly, allowing for quick correction. Overall, the shopping cart and checkout system aim to deliver a hassle-free, intuitive experience that supports the core goal of the ShopSmart platform: convenient and secure online grocery shopping.

User Authentication and Account Management

User Authentication in the ShopSmart web application ensures that only registered users can access personalized features such as managing their cart, placing orders, and viewing order history. During the login or registration process, the app securely handles credentials using authentication mechanisms like JSON Web Tokens (JWT), which provide session-based access and protect sensitive information. Unauthorized users are restricted from accessing admin or user-specific routes, enhancing data security and privacy.

Account Management allows users to update their profile information, change passwords, and manage their previous orders with ease. This functionality is designed to offer flexibility and control over user data, ensuring a personalized shopping experience.

Payment Integration

Payment Integration in the ShopSmart web application enables secure and smooth transactions during the checkout process. The system is designed to support both online payment methods like UPI or card payments (via third-party gateways such as Stripe or Razorpay) and Cash on Delivery (COD), depending on user preference. APIs handle payment requests, ensuring proper transaction validation, confirmation, and error handling in case of payment failure.

Security is a core aspect of this module. Sensitive details such as card information are never stored on the server, and all transactions occur over encrypted channels to comply with industry standards. Successful payment updates the order status and triggers backend processes like inventory updates and order confirmation.

Backend Development

Backend Development in the ShopSmart grocery web application is responsible for handling business logic, managing database operations, and ensuring secure communication between the frontend and server. The backend is built using **Node.js** with **Express.js** as the framework, offering a scalable and efficient environment for developing RESTful APIs. These APIs manage essential functions such as user authentication, product management, cart operations, order processing, and payment integration. The backend interacts with a **MongoDB Atlas** database to store and retrieve structured data related to users, products, orders, and inventory.

Security and performance are prioritized through the use of middleware like `express.json`, `cors`, and `jsonwebtoken` for data handling, CORS support, and route protection. Proper error handling ensures smooth operation and clear feedback to the frontend. Mongoose is used for defining schemas and models, simplifying interactions with MongoDB. This backend architecture supports modular code, easier maintenance, and scalability, making it adaptable for future feature enhancements.

```
grocery-shop > backend > JS index.js > ...
35 //Config Cloudinary
36 cloudinary.config({
37   cloud_name: process.env.CLOUD_NAME,
38   api_key: process.env.CLOUD_API_KEY,
39   api_secret: process.env.CLOUD_API_SECRET_KEY,
40 });
41
42 app.listen(process.env.PORT, "localhost", () => {
43   console.log(`Server Running At http://localhost:${process.env.PORT}`);
44 });
45
46 //Load Route
47 app.use("/api/user", userRoutes);
48 app.use("/api/product", productRoute);
49 app.use("/api/category", categoryRoute);
50
51 //Access Front End Static Files
52 app.use(express.static(path.join(__dirname, "../frontend/build")));
53
54 //Access Front End All URL
55 app.get("/*", (req, res) => {
56   res.sendFile(path.resolve(__dirname, "../frontend/build/index.html"));
57 });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Integration

Frontend-Backend Integration in the ShopSmart grocery web application is a crucial phase that brings together the user-facing interface and the server-side logic to create a cohesive and fully functional system. This process involves establishing a smooth and secure communication channel between the React.js frontend and the Node.js/Express backend through RESTful API calls. The integration ensures that user actions on the frontend, such as logging in, browsing products, adding items to the cart, or placing orders, are seamlessly connected to backend operations like data fetching, order validation, and database updates.

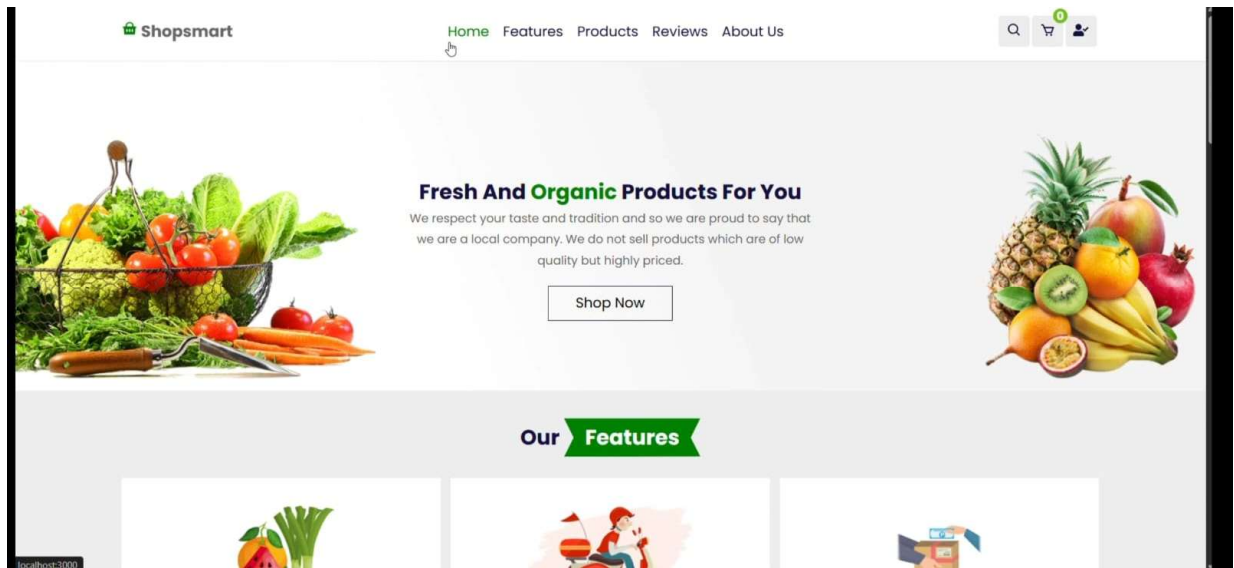
API endpoints created on the backend handle various tasks such as retrieving product details, processing user logins, handling cart operations, and placing orders. The frontend uses libraries like Axios or Fetch API to make HTTP requests to these endpoints. For example, when a user selects a product, the frontend sends a GET request to the backend API to fetch data from the MongoDB Atlas database. Similarly, when a user places an order, a POST request is sent with all necessary order details, which the backend then stores in the database and returns a confirmation response.

Authentication tokens (JWTs) are often used to ensure that only authorized users can access protected routes, such as viewing personal orders or adding products as an admin. These tokens are stored securely in the frontend and attached to every protected request header for verification on the backend. Additionally, error messages from the backend are handled gracefully in the frontend to inform users about issues like invalid credentials or failed transactions.

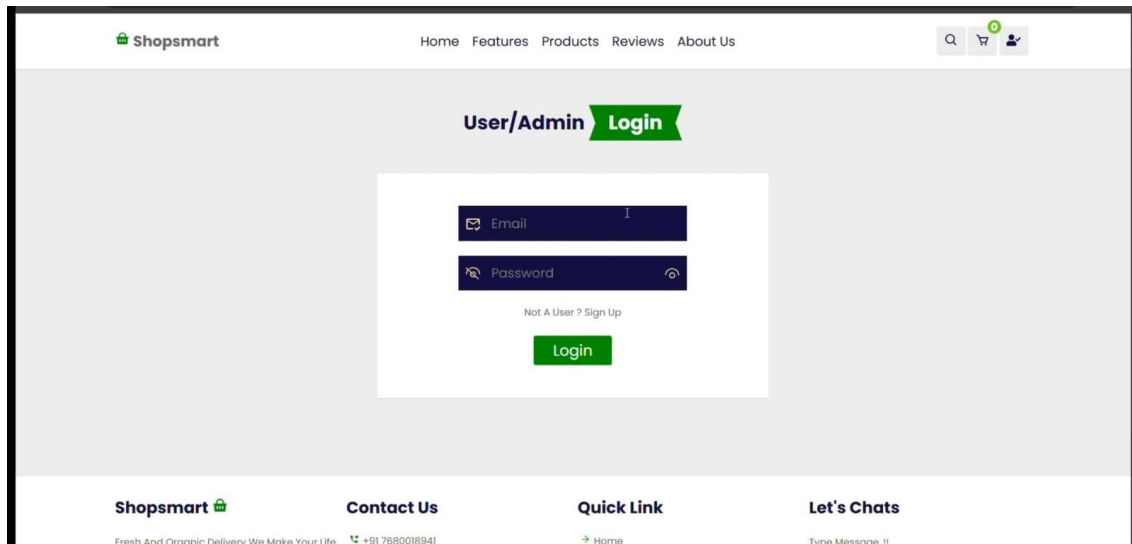
This tight integration between the frontend and backend allows the ShopSmart app to deliver dynamic content, process user interactions in real time, and maintain synchronization between the UI and backend logic. It also enhances scalability, maintainability, and the overall user experience by creating a responsive and reliable system for online grocery shopping.

Final Deployment

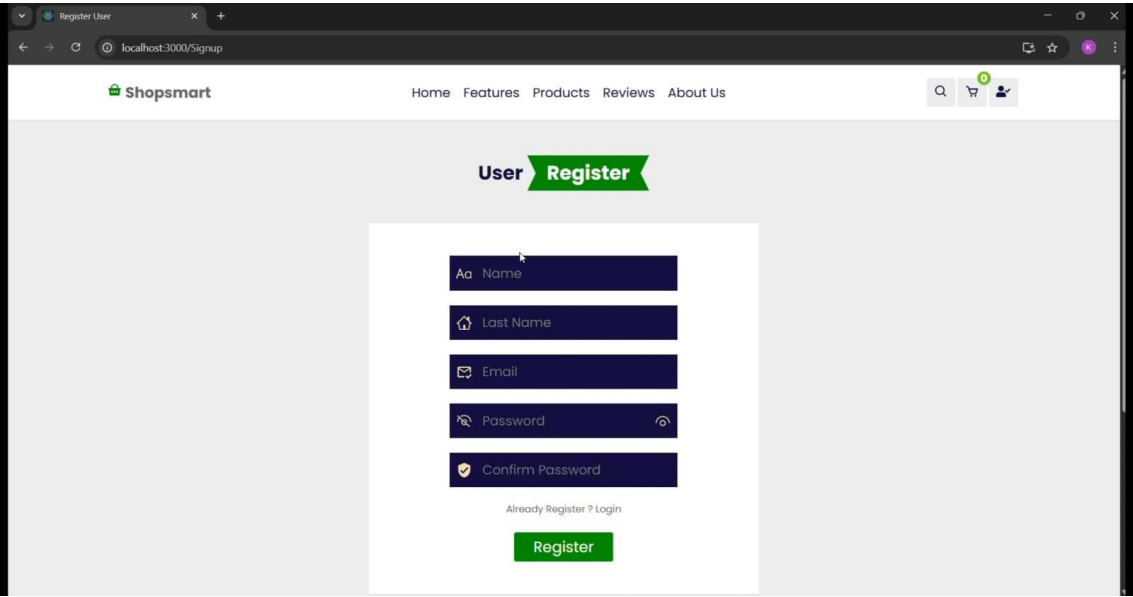
Home Page:



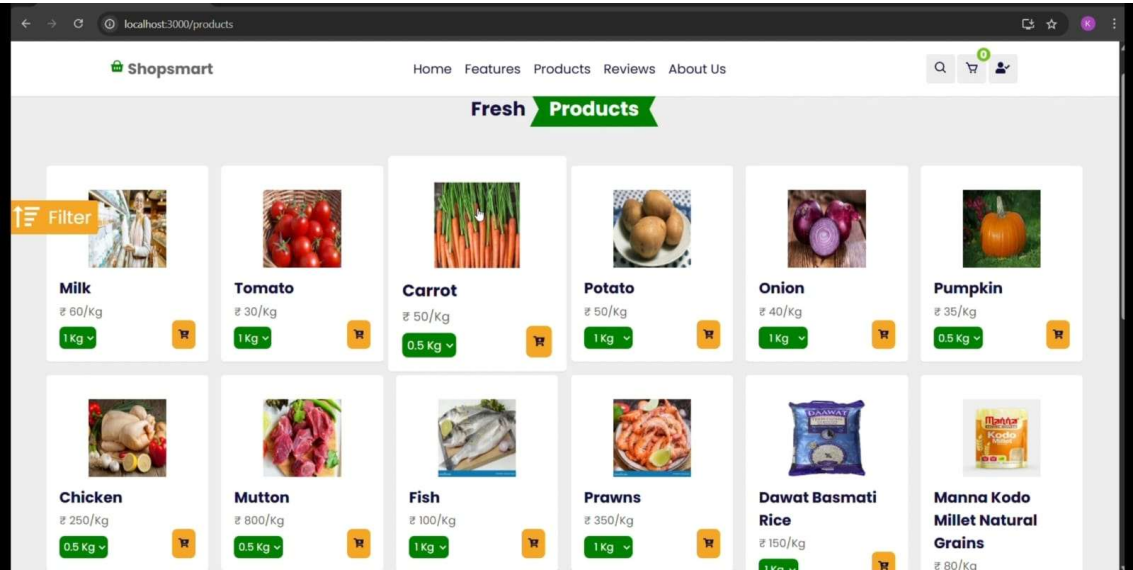
Login Page:



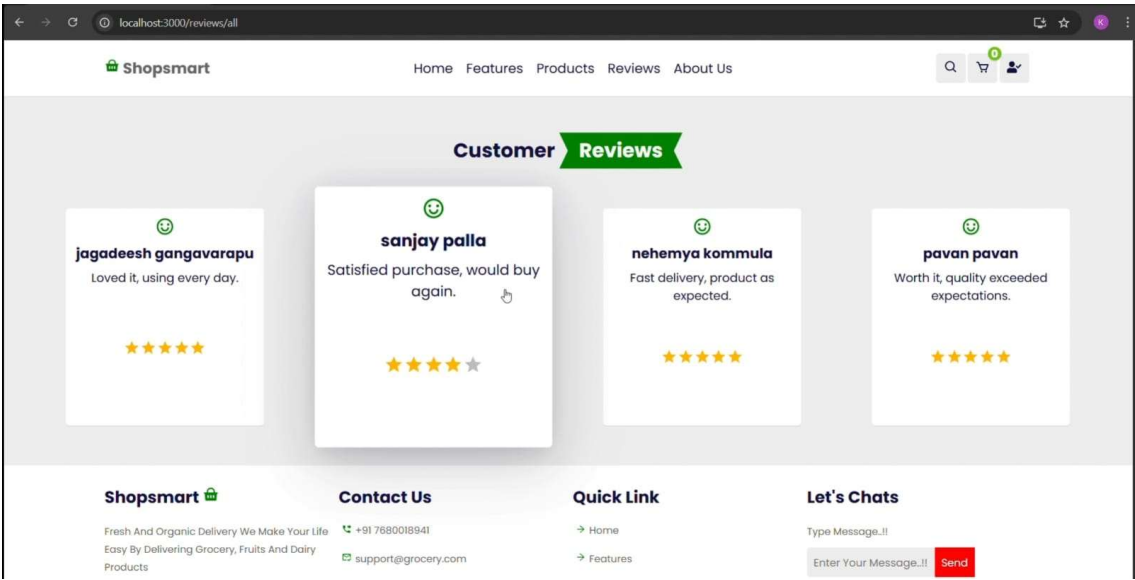
Registration Page:



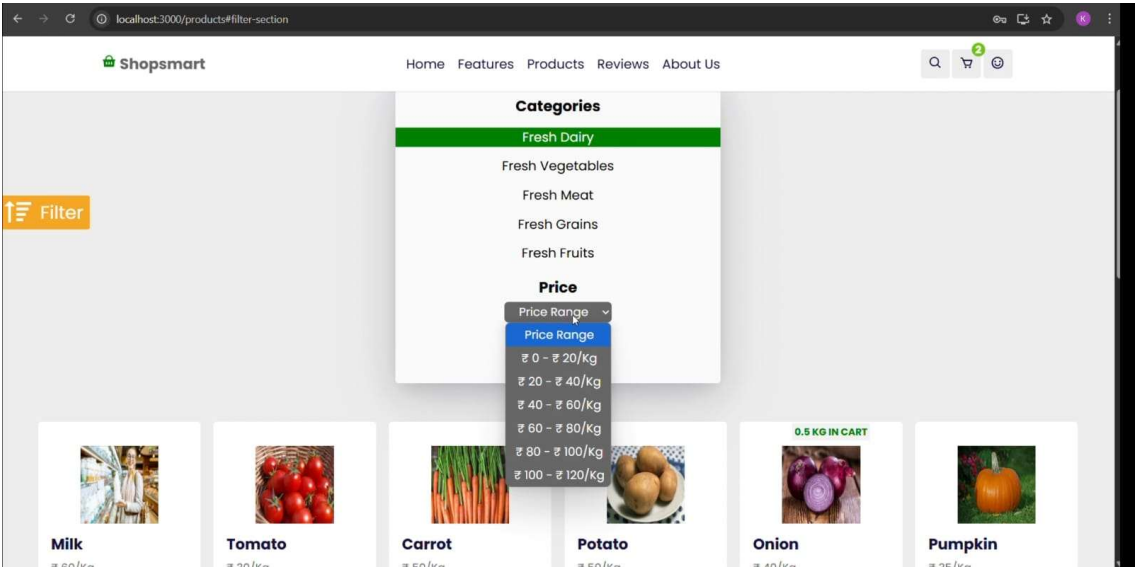
Products Page:



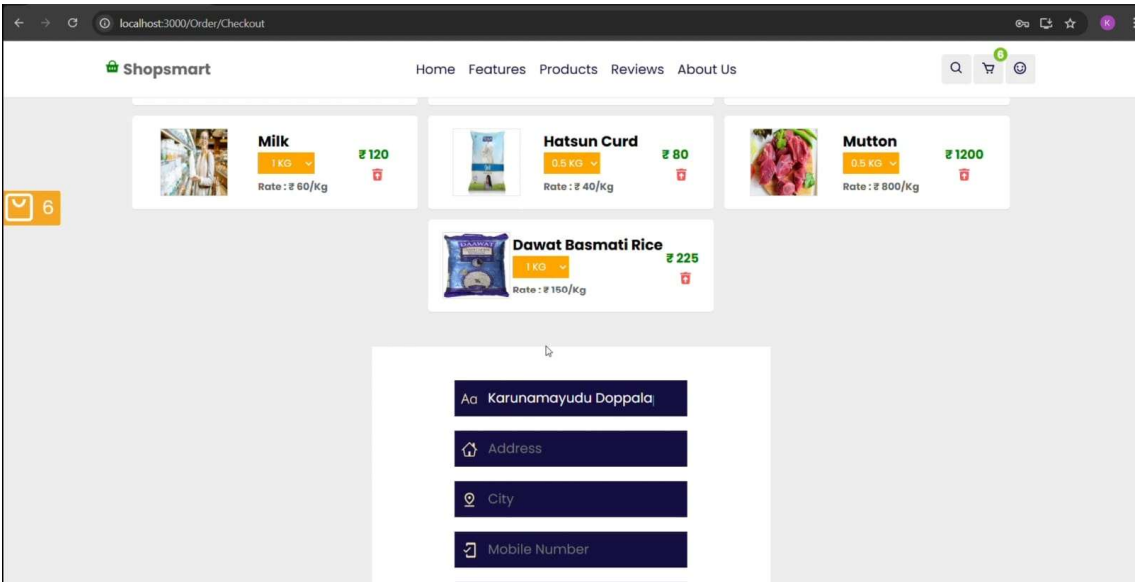
Reviews Page:



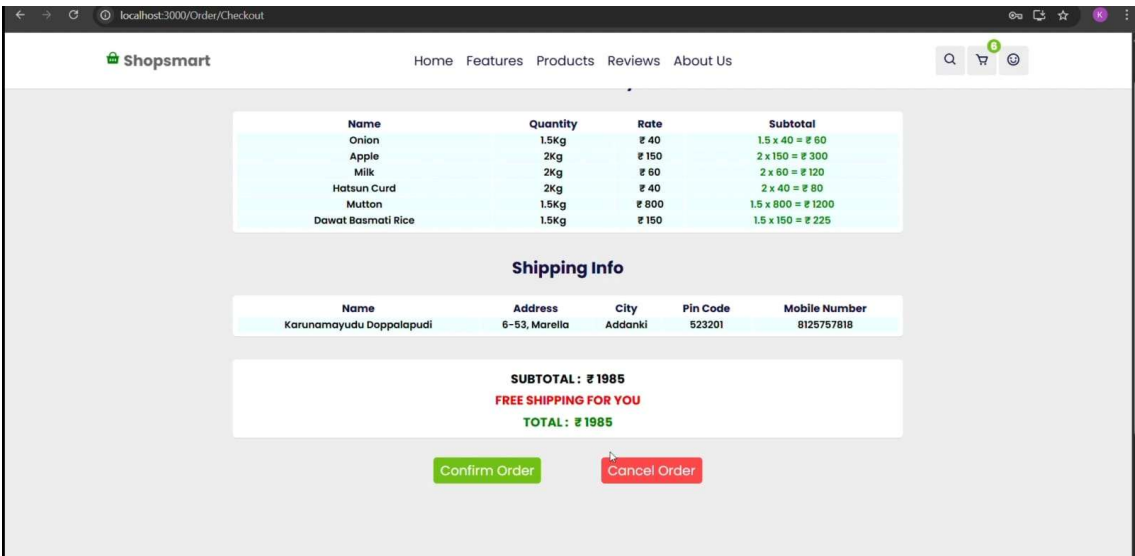
Filter Groceries page through Categories and Price:



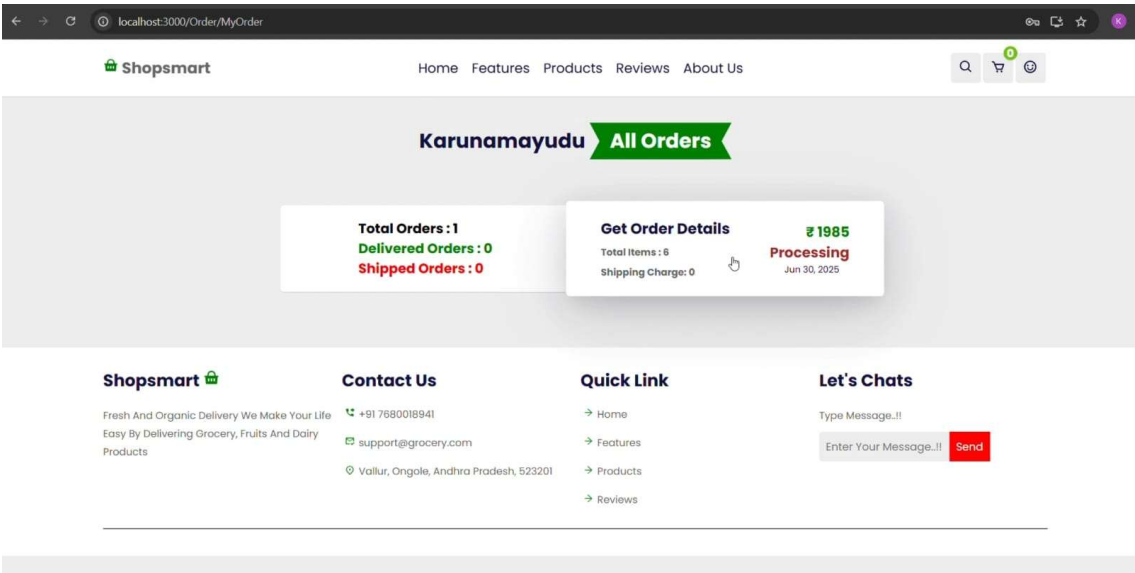
Checkout Page:



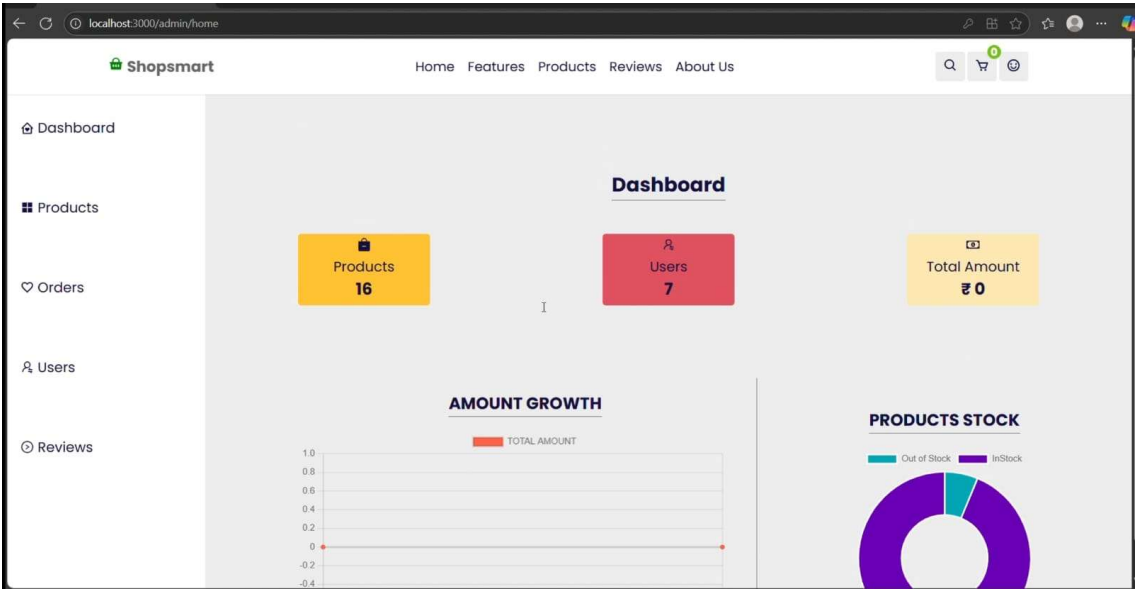
Groceries Check and Confirm Order Page:



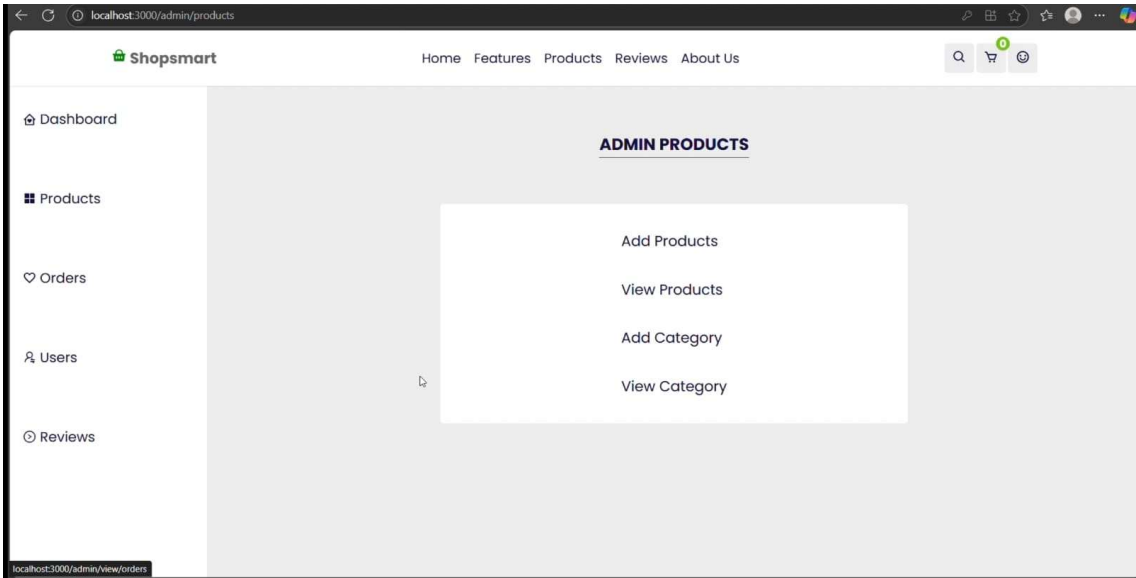
MyOrder Page:



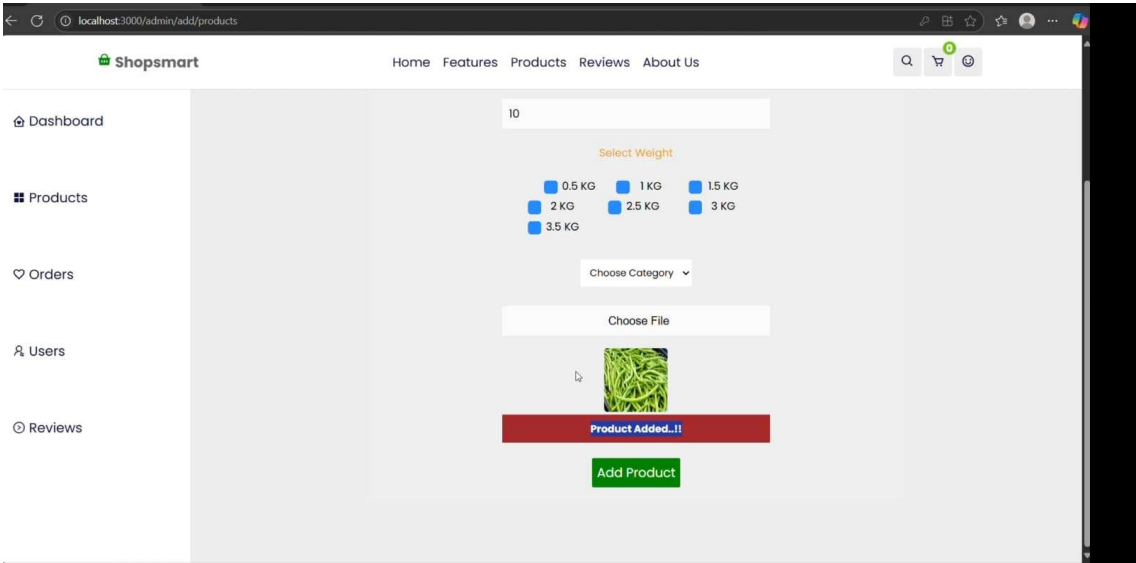
Admin Dashboard:



Admin Products Page:



Admin Add Product Page:



Admin View Product Page:

Shopsmart

HomeFeaturesProductsReviewsAbout Us







Dashboard

Products

Orders

Users

Reviews

6862363fab37578d2918dce8	Mutton		₹ 800/Kg	Meat	18.5	IN STOCK		
6862368fab37578d2918dcf2	Fish		₹ 100/Kg	Meat	30	IN STOCK		
686236d7ab37578d2918dcfd	Prawns		₹ 350/Kg	Meat	39.5	IN STOCK		
686237d7ab37578d2918dd11	Dawat Basmati Rice		₹ 150/Kg	Grains	15.5	IN STOCK		
686238e9ab37578d2918dd40	Manna Kodo Millet Natural Grains		₹ 80/Kg	Grains	30	IN STOCK		
68623df5ab37578d2918df16	Apple		₹ 150/Kg	Fruits	48	IN STOCK		

Admin Users Page:

Shopsmart

HomeFeaturesProductsReviewsAbout Us

Dashboard

Products

Orders

Users

Reviews

ALL USERS

Id	Name	Last Name	Email	Role	Actions
6862455eab37578d2918dfa0	Karunamayudu	Doppalapudi	karunamayudu@gmail.com	User	
686244caab37578d2918df81	karuna	doppalapudi	karuna@gmail.com	User	
68623a3aab37578d2918dd82	jagadeesh	gangavarapu	jagadeesh@gmail.com	User	
686239f9ab37578d2918dd73	sanjay	palla	sanjay@gmail.com	User	
6862398aab37578d2918dd5a	nehemya	kommula	nehemya@gmail.com	User	
686175a0e6d7eee39945bf7f	pavan	pavan	pavan@gmail.com	User	
68615143b9a177ad3268ea4	obul	undela	obul@gmail.com	Admin	

Admin Orders Page:

Shopsmart

HomeFeaturesProductsReviewsAbout Us

Q

0

Dashboard

Products

Orders

Users

Reviews

ALL ORDERS

Order id	Name	Items	Amount	Status	Date	Actions
#68624646ab37578d2918dfee	Karunamayudu Doppalapudi	6	₹ 1985	Shipped	Jun 30, 2025	✓
#68623bb8ab37578d2918dde2	nehemya kommula	3	₹ 220	Shipped	Jun 30, 2025	✓
#68623b30ab37578d2918ddbc	sanjay palla	1	₹ 175	Shipped	Jun 30, 2025	✓
#68623afcab37578d2918dda7	sanjay palla	1	₹ 450	Shipped	Jun 30, 2025	✓
#686175f6e6d7eee39945bf9e	pavan pavan	1	₹ 30	Shipped	Jun 29, 2025	✓

localhost:3000/admin/view/orders

About Us Page:

Shopsmart

HomeFeaturesProductsReviewsAbout Us

Q

0

About Us

Welcome to **ShopSmart** – your one-stop digital grocery store! We're dedicated to making grocery shopping fast, simple, and convenient. With a wide variety of fresh fruits, vegetables, dairy, and household essentials, ShopSmart brings everything you need right to your doorstep. We aim to save your time while ensuring quality and affordability. Enjoy a smooth online shopping experience anytime, anywhere.

Shopsmart

Fresh And Organic Delivery We Make Your Life Easy By Delivering Grocery, Fruits And Dairy Products

Contact Us

+91 7680018941
support@grocery.com
Vallur, Ongole, Andhra Pradesh, 523201

Quick Link

→ Home
→ Features
→ Products

Let's Chats

Type Message..!!
Enter Your Message..!! Send

Conclusion:

This project outlines the development of a comprehensive grocery shopping web application with distinct user roles for customers and administrators. Each milestone—project setup, backend development, database design, frontend implementation, and final integration—contributes to a well-structured and scalable system. The integrated functionalities ensure smooth product management, secure user authentication, role-specific access, and a responsive shopping experience for end-users.