

GROCERY WEB APP

1.INTRODUCTION:

PROJECT TITLE:SHOPSMART-ONLINNE GROCERY SHOPPING APP

Welcome to our Grocery Web App, your one-stop shop for all your grocery need With our user-friendly interface and wide selection of highquality 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

2.TEAM MEMBERS:

- 1.K.PUJITHA (23MH5A0510)
- 2.E.TANUJA (22MH1A05A1)
- 3.P.SAI KIRAN (22MH1A0552)
- 4.Nikesh Appari (22MH1A0583)

3.PROJECT OVERVIEW

Purpose:

ShopSmart is an online grocery web application that allows users to browse, search, and purchase groceries from the comfort of their homes. The goal is to simplify grocery shopping with a user-friendly interface and fast delivery service.

Features:

- User registration and login
- Product catalog with categories
- Cart and checkout system
- Admin panel to manage products and orders
- Order history tracking

4.ARCHITECTURE:

Frontend:

Built using React.js, the frontend includes pages for home, login, product listing, cart, and admin dashboard. Uses React Router for navigation and Axios for API requests.

Backend:

Developed with Node.js and Express.js, it handles user authentication, product management, and order processing.

Database:

Uses MongoDB to store data such as user info, product details, orders, and admin credentials. Mongoose is used as the ODM.

5.SET UP INSTRUCTIONS:

Prerequisites:

- Node.js
- MongoDB
- npm or yarn

Installation:

1. Clone the repository:

```
git clone https://github.com/yourusername/shopsmart.git  
cd shopsmart
```

2. Install dependencies:

```
cd client
```

```
npm install
```

```
cd ../server
```

```
npm install
```

3. Set up environment variables (.env file):

```
MONGO_URI=your_mongodb_connection_string
```

```
JWT_SECRET=your_jwt_secret
```

```
PORT=5000
```

6.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.

7. 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.

8. 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.

```
> db > JS connect.js

const mongoose = require("mongoose");
const db= 'mongodb://127.0.0.1:27017/grocery'
// Connect to MongoDB using the connection string

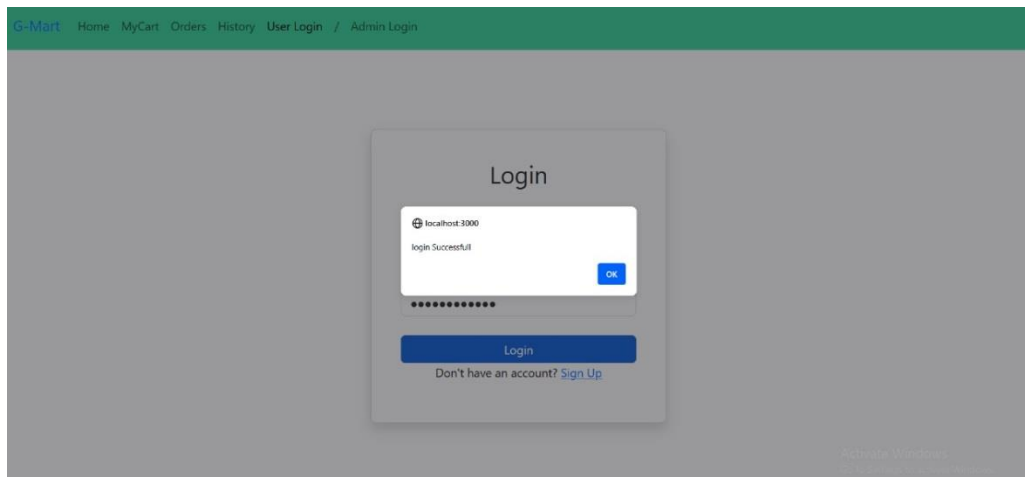
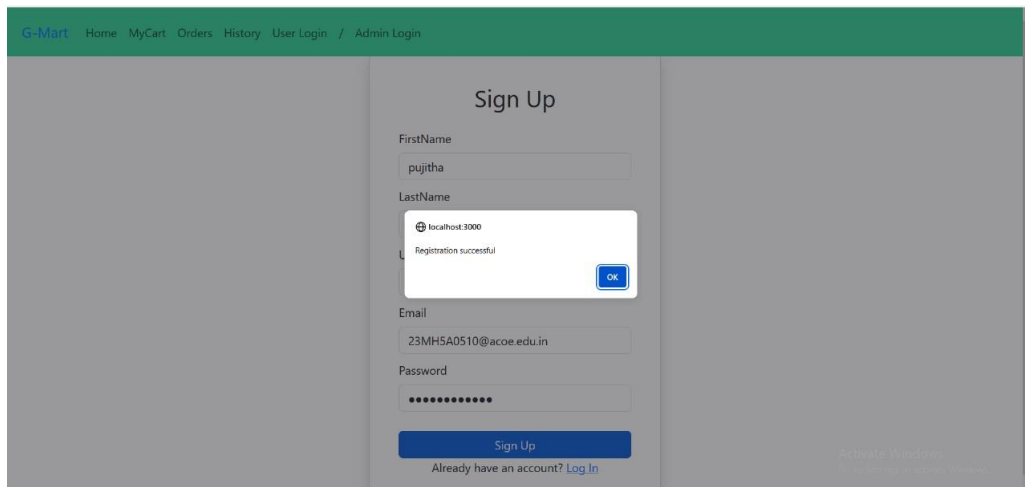
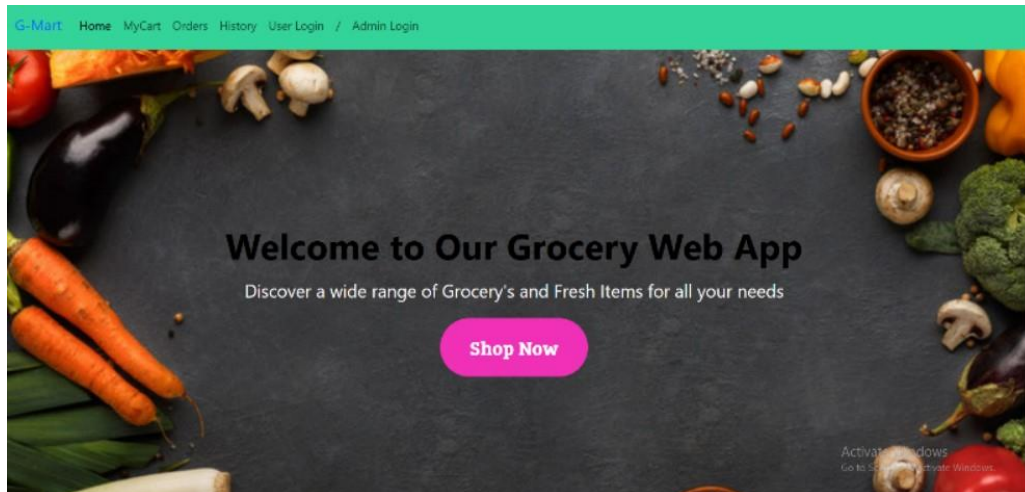
mongoose.connect(db, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log(`Connection successful`);
}).catch((e) => {
  console.log(`No connection: ${e}`);
});

const userSchema = new mongoose.Schema({
  firstname: { type: String},
  lastname: { type: String },
  username: { type: String, unique: true },
  email: { type: String},
  password: { type: String }
});

const adminSchema = 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 },
  description: { type: String, }
});

const productSchema = new mongoose.Schema({
  productname: { type: String, required: true },
  description: { type: String, required: true },
  price: { type: Number, required: true }
});
```



Products Page:

[G-Mart](#) [Home](#) [MyCart](#) [Orders](#) [History](#) [Logout](#)


Search By Product Name

Search by product name

Filter By Category


all

Products




carrot
\$80

[Buy Now](#) [Add to Cart](#)




apple
\$100

[Buy Now](#) [Add to Cart](#)




banana
\$40

[Buy Now](#) [Add to Cart](#)




milk
\$60

[Buy Now](#) [Add to Cart](#)




chicken
\$100

[Buy Now](#) [Add to Cart](#)




cashew
\$100

[Buy Now](#) [Add to Cart](#)




fish
\$400

[Buy Now](#) [Add to Cart](#)




kemin
\$90

[Buy Now](#) [Add to Cart](#)




SAMOSAs

[Buy Now](#) [Add to Cart](#)




PIZZA

[Buy Now](#) [Add to Cart](#)




cookies

[Buy Now](#) [Add to Cart](#)




tomato

[Buy Now](#) [Add to Cart](#)



onion

[Buy Now](#) [Add to Cart](#)



Almond

[Buy Now](#) [Add to Cart](#)

[G-Mart](#) [Home](#) [MyCart](#) [Orders](#) [History](#) [Logout](#)


My Orders

Order ID: 685d450403c5ec6c52444297
Name: pujitha k
Phone: 0987654321
Date: 2025-06-26T13:03:00.606Z
Price: 200
Status: Pending
Payment Method: cod

Activate Windows
Go to Settings to activate Windows.


[G-Mart](#) [Home](#) [MyCart](#) [Orders](#) [History](#) [Logout](#)

My Cart




apple
\$100

[Remove from Cart](#) [Buy this](#)



banana
\$40

[Remove from Cart](#) [Buy this](#)



chicken
\$300

[Remove from Cart](#) [Buy this](#)

ADMIN LOGIN:

Grocery Web App

DashboardUsersProductsAdd productOrdersLogout

Dashboard

Product Count

16 Products

View Products

User Count

4 Users

View Users

Order Count

3 Orders

View Orders

Add Product

Add

Activate Windows
Go to Settings to activate Windows.

Grocery Web App

DashboardUsersProductsAdd productOrdersLogout

Add Product






Product Name	Rating	Price
chilli	4.8	30
Image URL	Category	Count in Stock
+ziRjoZIDrwATvKywAAAAA==	Vegetables	40
Description		
green chillies		
Add Product		

Activate Windows
Go to Settings to activate Windows.

Grocery Web App

DashboardUsersProductsAdd productOrdersLogout

Users

sl/no	Userid	User name	Email	Operation
1	685d27e150dda0b28ccb9db2	kalyani	kalyani@gmail.com	 view
2	685d2e6c50dda0b28ccb9ded	pujitha	kamepallipujitha02@gmail.com	 view
3	685d2f4950dda0b28ccb9e13	tanuja	tanuja@gmail.com	 view
4	685d43ca03c5ec6c52444283	pujitha kamepalli	23MH5A0510@acoe.edu.in	 view
5	685d461b03c5ec6c524442aa	sai kiran	saikiran@gmail.com	 view

Activate Windows
Go to Settings to activate Windows.

Orders

Order ID: 685d2ee150dda0b28ccb9e01
Fullname: pujitha k
Phone: 0987654321
Product ID: 685d26dc50dda0b28ccb9dc1
Quantity: 10
Total price: 800
Payment Method: cod
Address: surampalem
Created At: 2025-06-26T11:31:30.624Z
Status: Confirmed

[Update Status](#)

Order ID: 685d2f2d50dda0b28ccb9e0e
Fullname: kalyani k
Phone: 1234567890
Product ID: 685d2dd250dda0b28ccb9de6
Quantity: 2
Total price: 140
Payment Method: cod
Address: kandukur
Created At: 2025-06-26T11:29:49.437Z
Status: Pending

[Update Status](#)

Activate Windows
Go to Settings to activate Windows.