

Full Stack Development with MERN Project Documentation

Introduction:Full Stack Development with MERN Project Documentation

Introduction:

- **Project Title:** ShopSmart: Your Digital Grocery Store Experience
- **Team ID:** LTVIP2025TMID54445
- **Team Size:** 4

Team Leader : Seelam Deepthi

Team member: Devarampati Venkata Pradeep

Team member: Devarakonda Swathi Sri

Team member: Devana Sivalakshimi

Welcome to our Grocery 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!

Description:

Our Grocery Web App is more than just a convenient way to shop for groceries—it's a comprehensive solution designed to enhance every aspect of your shopping experience. Imagine a virtual supermarket at your fingertips, offering an extensive

selection of high-quality products that cater to your every need.

From fresh produce sourced directly from local farms to pantry staples and household essentials, our app provides everything you need to keep your kitchen well-stocked. Our user-friendly interface ensures that you can navigate our virtual aisles effortlessly, easily finding what you're looking for and adding items to your cart with a simple click.

But the convenience doesn't stop there. With our app, you can schedule deliveries at your preferred time, ensuring that your groceries arrive when it's most convenient for you. Say goodbye to long lines and tedious grocery trips—our app brings the supermarket to you, saving you time and hassle.

We understand that everyone's dietary preferences and requirements are different, which is why our app offers a wide range of options, including organic, gluten-free, and vegan products. You can also discover new and exciting products through our app, making grocery shopping an adventure rather than a chore.

Our commitment to quality extends beyond our products to our customer service. Our dedicated team is always available to assist you with any questions or concerns you may have, ensuring that your shopping experience is nothing short of exceptional.

Experience the future of grocery shopping with our Grocery Web App. Download it today and discover the convenience of fresh, high-quality groceries delivered right to your doorstep.

Scenario Based Case Study:

Meet priya, 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.

- **Priya's Solution:** The Grocery Web App.

Priya discovers the Grocery Web App, 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: Priya 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: Priya 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 Priya's purchase history and preferences to provide personalized recommendations, helping her discover new products and brands that align with her tastes.

Convenient Delivery Options: Priya 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.

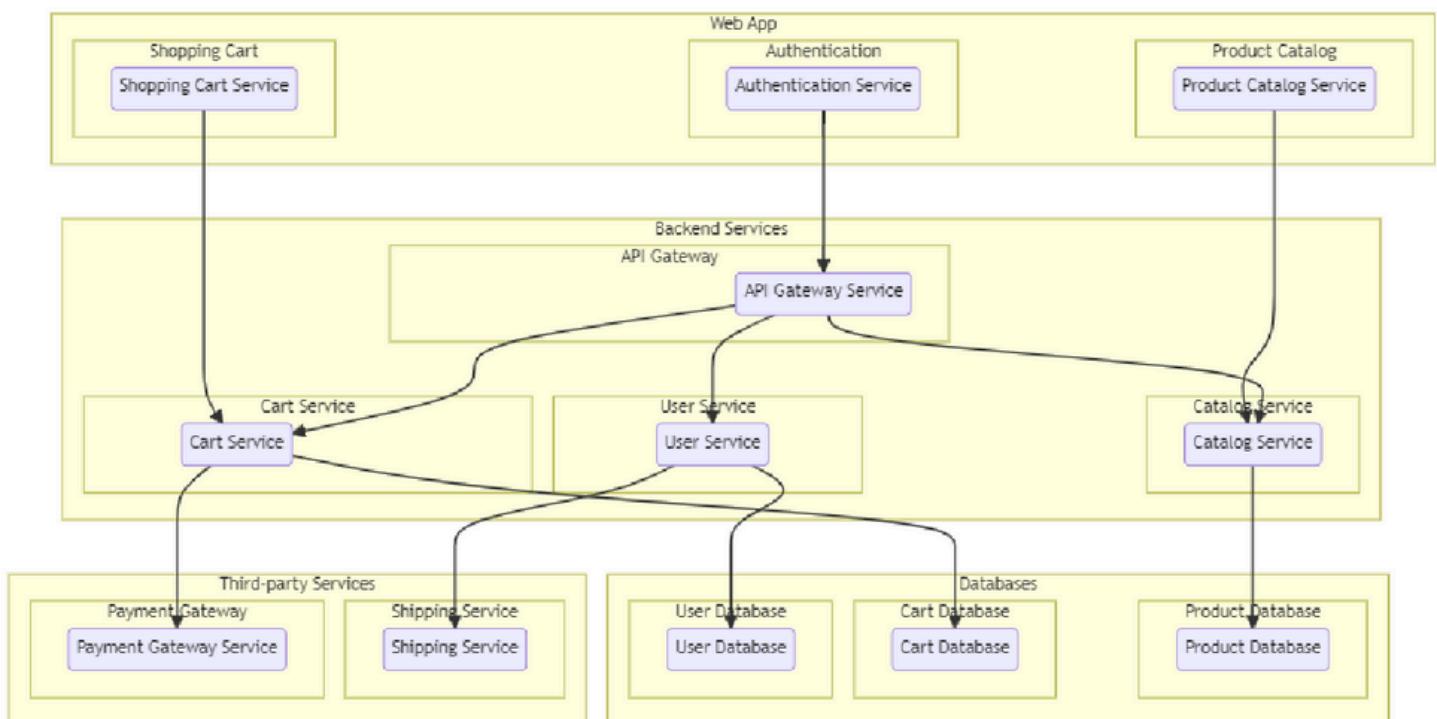
Secure Payment Gateway: The app integrates with a secure payment gateway, allowing Priya to pay for her groceries online using various payment methods, including credit/debit cards and digital wallets.

Order Tracking: Priya 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 Priya with any queries or concerns she may have.

Priya's Experience: Thanks to the Grocery Web App, Priya 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.

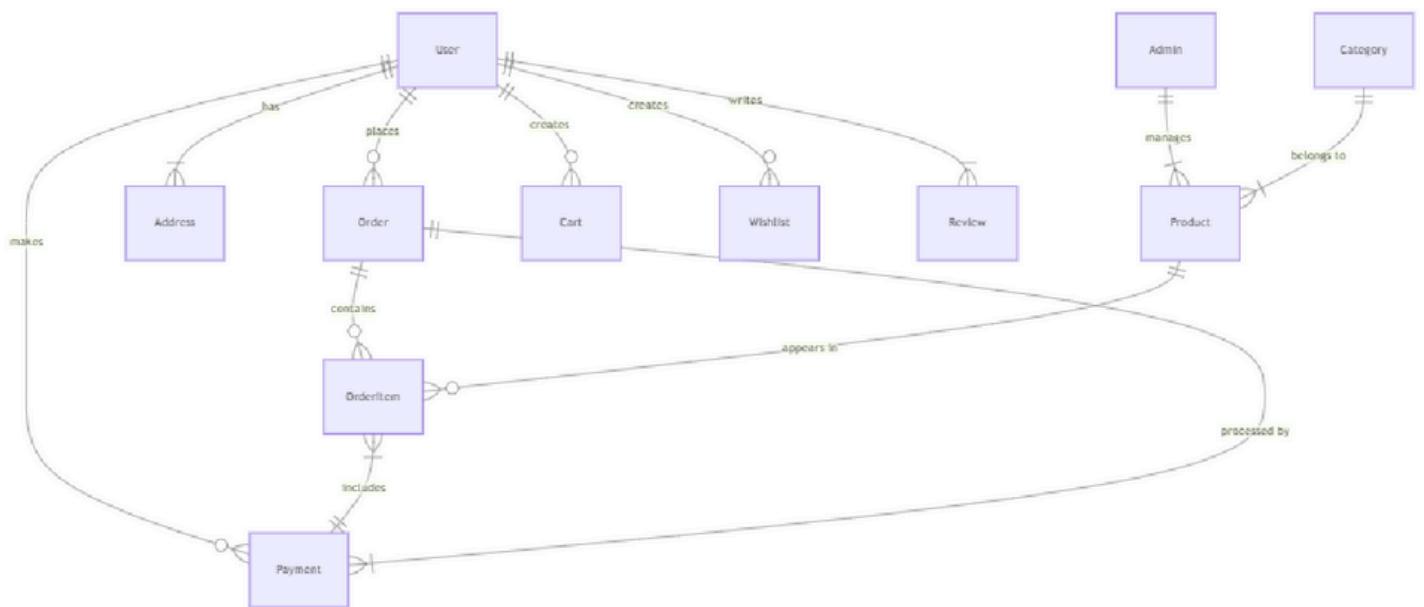
Technical Architecture:-



The technical architecture of a 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 a 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-Appiaction

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 advanced 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.

- **VisualStudioCode:** 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.

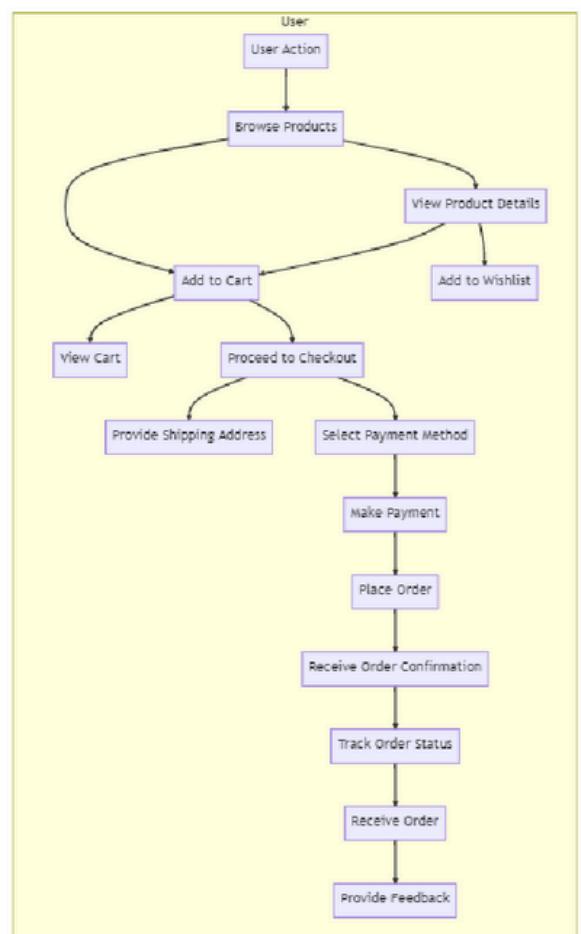
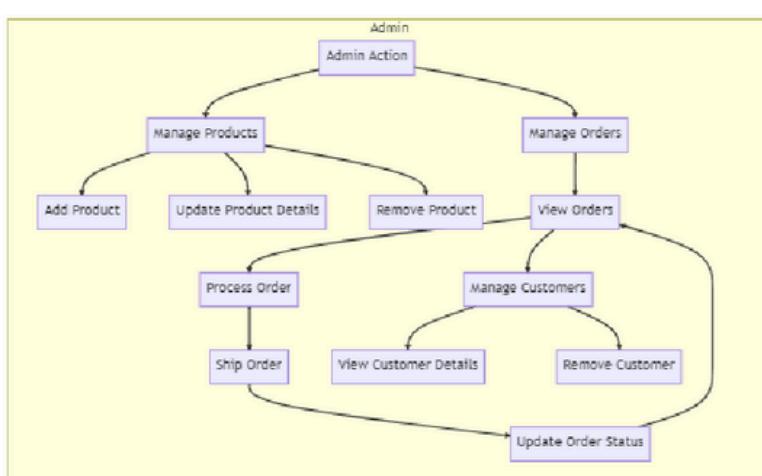
Admin:-

- 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:-

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

Demo link:-

https://drive.google.com/file/d/1HLowclqs2d8lxTprS2jqPmR4AOnUW8xD/view?usp=drive_link

Use the code in:-

https://drive.google.com/drive/folders/1RP-29p9mf-bbLAK5r7S4HSF_Itai0i1G?usp=drive_link

or follow the videos below for better understanding.

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.

Reference Link:-

https://drive.google.com/file/d/1Acv3Lx3PtJcOYkUjREWAzloCi6w96Tl/view?usp=drive_link

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.

Reference Link:-

https://drive.google.com/file/d/18-gnCVAZdojBhl7QMO_yShH1JL5A6v32/view?usp=drive_link

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.

```
1 const mongoose = require('mongoose');
2
3 mongoose.connect(
4   'mongodb+srv://shopsmart:test123@cluster0.jsjtxes.mongodb.net/grocerydb?retryWrites=true&w=majority&appN
5   {
6     useNewUrlParser: true,
7     useUnifiedTopology: true
8   }
9 )
10 Tabnine | Edit | Test | Explain | Document
11 .then(() => console.log('✅ Connected to MongoDB Atlas'))
12 Tabnine | Edit | Test | Explain | Document
13 .catch((err) => console.error('✗ MongoDB connection error:', err));
14
```

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, required: true },
  lastname: { type: String, required: true },
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

// 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, max: 255 },
  rating: { type: Number, required: true },
  unit: { type: String, default: 'Kg' }, // Optional field for unit
  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 paymentSchema = new mongoose.Schema({
  user: { type: String, ref: 'User', required: true },
  name: { type: String, required: true },
  order: { type: String, ref: 'Order', required: true },
  amount: { type: Number, required: true },
  paymentMethod: { type: String, required: true },
  deliveryStatus: { type: String, required: true },
  status: { type: String, enum: ['Pending', 'Success', 'Failed'], default: 'Pending' },
  createdAt: { type: Date, default: Date.now }
});

const feedbackSchema = new mongoose.Schema({
  user: { type: String, required: true },
  message: { 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),
  Payment: mongoose.model('Payment', paymentSchema),
  Feedback: mongoose.model('Feedback', feedbackSchema)
};

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:-

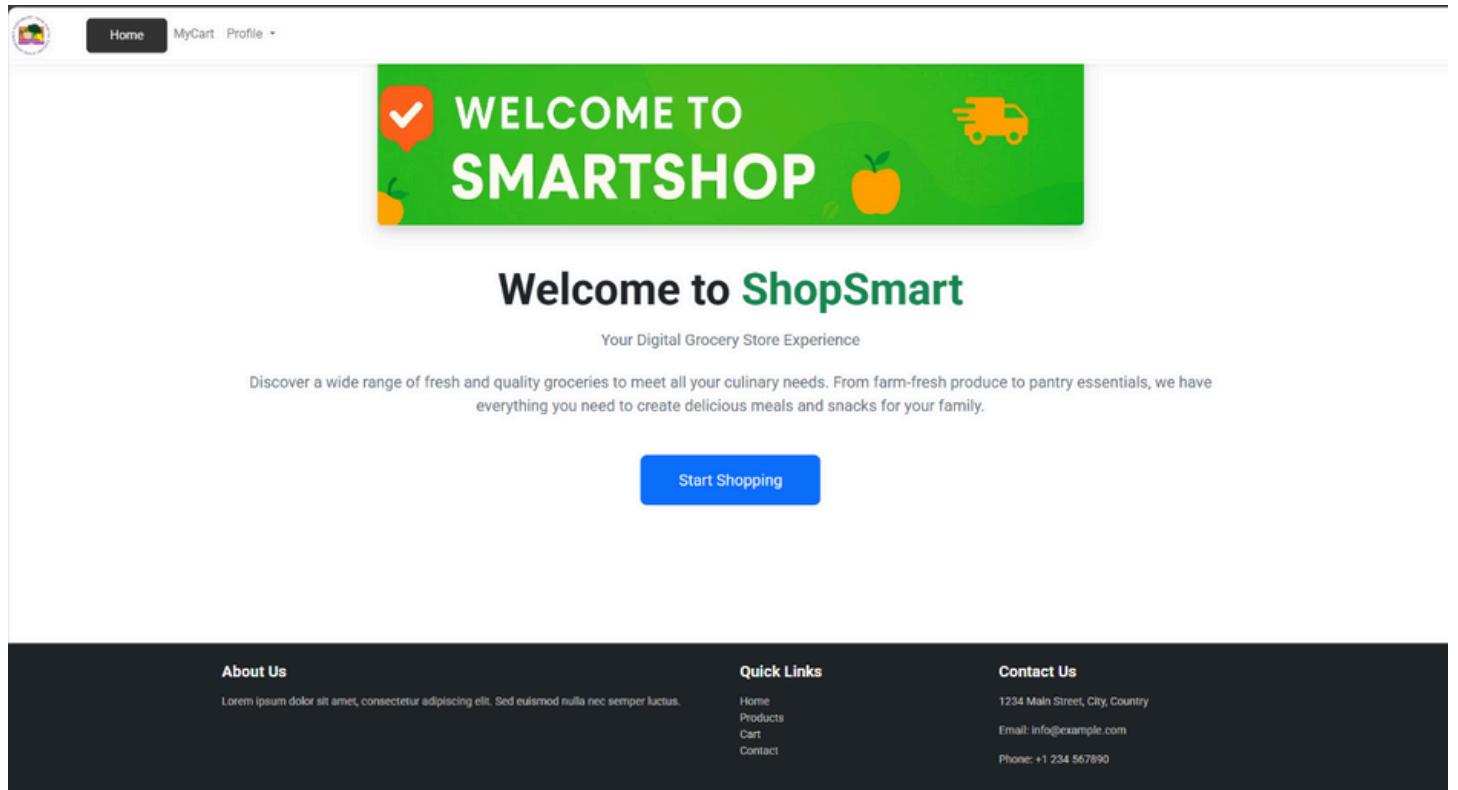
https://drive.google.com/file/d/1MFQ7NIQ02ynLioEMySPpEzdX-c0i6Wer/view?usp=drive_link

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:-



The screenshot shows the homepage of a grocery delivery service named ShopSmart. At the top, there is a navigation bar with icons for Home, MyCart, and Profile. Below the navigation is a large green banner with the text "WELCOME TO SMARTSHOP" in white, flanked by a checkmark icon and a delivery truck icon. The main heading "Welcome to ShopSmart" is displayed in a large, bold, black font. Below it, a subtitle "Your Digital Grocery Store Experience" is shown in a smaller, gray font. A descriptive paragraph encourages users to discover a wide range of fresh produce and pantry essentials. A prominent blue "Start Shopping" button is located below the text. At the bottom of the page, there are three columns: "About Us" (with placeholder text), "Quick Links" (listing Home, Products, Cart, and Contact), and "Contact Us" (with address, email, and phone number).

About Us
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod nulla nec semper luctus.

Quick Links
Home
Products
Cart
Contact

Contact Us
1234 Main Street, City, Country
Email: info@example.com
Phone: +1 234 567890

Login Page:-



Home MyCart Profile ▾



Login

Email

Enter email

Password

Enter password

Sign In

[Don't have an account? click here](#)

Items Page:-



Home MyCart Profile ▾

Logout

Find Your Groceries

Search products...



Categories ▾



Available Products



Tomatoes
Fresh red tomatoes.

₹30/Kg



Bananas
Ripe yellow bananas.

₹40/Dozen



Apples
Kashmir apples

₹100/Dozen



Brown Bread
Healthy multi-grain brown bread

₹35/Pack

My Cart:-

The screenshot shows a user interface for a shopping cart. At the top, there is a navigation bar with icons for Home, MyCart (which is highlighted and has a green notification badge with the number '3' above it), Profile, and Logout. The main title 'My Cart' is centered above three product cards. Each card contains an image of the product, its name, its price, and two buttons: 'Buy this product' and a red 'X' button.

Product	Image	Name	Price	Action Buttons
Apples		Apples	Price: ₹100	Buy this product X
Croissant		Croissant	Price: ₹50	Buy this product X
Rice		Rice	Price: ₹60	Buy this product X

My Orders Page:-



Home MyCart Profile ▾

Logout

My Orders

Name: akshay

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Order Canceled

My History Page:-



Home MyCart Profile ▾

Logout

History

Product Name: Grapes

Quantity: 2

Total Price: 100

Payment Method: cash-on-delivery

Address: andhrapradesh

Created At: Jun 26, 2025

Status: Delivered

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Place Order Page:-



Order Details

Firstname

Lastname

Phone

Quantity

Address

Payment method:

[Confirm Order](#)

Admin Dashboard Page:



Profile ▾ [Logout](#)

Grocery



Dashboard

[Dashboard](#)

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

Users

Total users: 4

[View Users](#)

Orders

Total orders: 5

[View Orders](#)

Add Products

Click to add new products

[Add Products](#)

Add Category

Click to add new category

[Add Category](#)

Users Page:-



Profile ▾ [Logout](#)

Grocery



Users

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

User Id: 68565e8b5b8a9f2270576ee1

First Name: Admin

Last Name: User

Username: admin123

Email: admin@gmail.com

User Id: 6858ff136cb82a281fedd7f0

First Name: Akshay

Last Name: Chatla

Username: akshay828

Email: akshaychatla99@gmail.com

User Id: 685a3817186404bc217e3388

First Name: test

Last Name: 123

Username: test123

Email: test@gmail.com

User Id: 685cf00a3fd383cef55adbfa

First Name: test

Last Name: 1

Username: test1

Email: test1@gmail.com

Admin's Add Products page:-

The screenshot shows the 'Add Products' section of the admin interface. On the left, there's a sidebar with a profile icon, 'Profile' and 'Logout' buttons at the top, and a navigation menu with 'Grocery' selected. Under 'Grocery', the 'Add Products' option is highlighted with a dark background. The main content area has a title 'Add Products'. It contains several input fields: 'Productname' (placeholder 'Enter productname'), 'Price' (placeholder 'Enter price'), 'Image' (placeholder 'Enter image url'), 'Category' (placeholder 'Enter category'), 'Stock' (placeholder 'Enter stock'), 'Rating' (placeholder 'Enter rating'), 'Unit' (placeholder 'Unit' with 'Kg' selected), and a large 'Description' text area (placeholder 'Enter description'). At the bottom is a grey button labeled 'Add Product'.

Admin Orders Page:-

The screenshot shows the 'Orders' section of the admin interface. The sidebar is identical to the previous screenshot. The main area displays a table of orders. The first order row contains the following information: Order ID: 685ce38f3fd383cef55adb0b, Fullname: Akshay Chatla, Phone: 07075768029, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-406 sri lanka colony vuyyuru, andhra pradesh, Created At: Jun 26, 2025, Status: Delivered, and a green 'Delivered' button. The second order row contains similar information: Order ID: 685ceee53fd383cef55adbcb, Fullname: Akshay 12345, Phone: 1234567891, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-221, andhra pradesh, and a green 'Delivered' button.

The demo of the app is available at:-

[Demo video.mp4](#)

• **Project Title:** ShopSmart: Your Digital Grocery Store Experience

• **Team Members:**

Team Leader: Seelam Deepthi

Team member: Naresh Perumalla

Team member: Nelavalli Venkata Bhaskara Devi Phanindra

Team member: Palaparthi Babu

Welcome to our Grocery 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!

Description:

Our Grocery Web App is more than just a convenient way to shop for groceries—it's a comprehensive solution designed to enhance every aspect of your shopping experience. Imagine a virtual supermarket at your fingertips, offering an extensive selection of high-quality products that cater to your every need.

From fresh produce sourced directly from local farms to pantry staples and household essentials, our app provides everything you need to keep your kitchen well-stocked. Our user-friendly interface ensures that you can navigate our virtual aisles effortlessly, easily finding what you're looking for and adding items to your cart with a simple click.

But the convenience doesn't stop there. With our app, you can schedule deliveries at your preferred time, ensuring that your groceries arrive when it's most convenient for you. Say goodbye to long lines and tedious grocery trips—our app brings the supermarket to you, saving you time and hassle.

We understand that everyone's dietary preferences and requirements are different, which is why our app offers a wide range of options, including organic, gluten-free, and vegan products. You can also discover new and exciting products through our app, making grocery shopping an adventure rather than a chore.

Our commitment to quality extends beyond our products to our customer service. Our dedicated team is always available to assist you with any questions or concerns you may have, ensuring that your shopping experience is nothing short of exceptional.

Experience the future of grocery shopping with our Grocery Web App. Download it today and discover the convenience of fresh, high-quality groceries delivered right to your doorstep.

Scenario Based Case Study:

Meet priya, 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.

- **Priya's Solution:** The Grocery Web App.

Priya discovers the Grocery Web App, 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: Priya 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: Priya 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 Priya's purchase history and preferences to provide personalized recommendations, helping her discover new products and brands that align with her tastes.

Convenient Delivery Options: Priya 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.

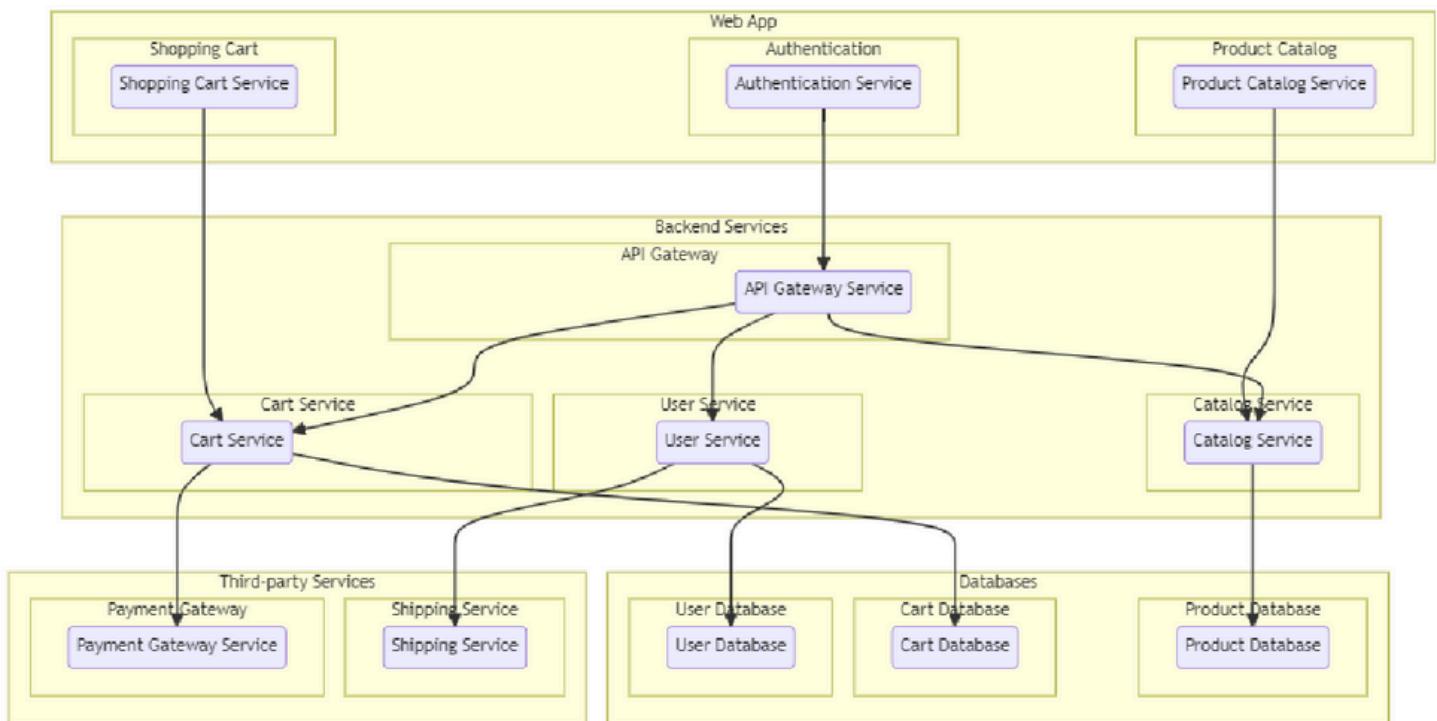
Secure Payment Gateway: The app integrates with a secure payment gateway, allowing Priya to pay for her groceries online using various payment methods, including credit/debit cards and digital wallets.

Order Tracking: Priya 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 Priya with any queries or concerns she may have.

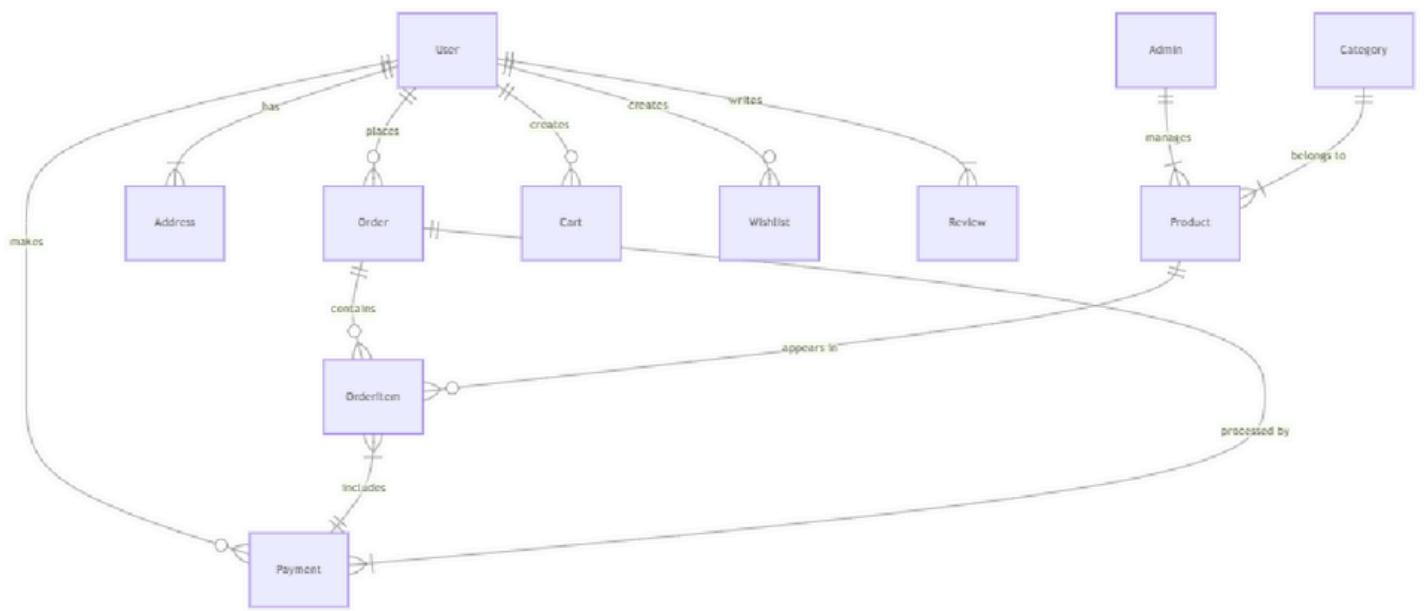
Priya's Experience: Thanks to the Grocery Web App, Priya can now enjoy the convenience of having fresh, high-quality groceries delivered right to her doorstep, saving her time and effort. She can focus on what matters most to her—cooking delicious meals for herself and her loved ones.

Technical Architecture:-



The technical architecture of a 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 a 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-Appiaction

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 direotory 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 advanced 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.

- **VisualStudioCode:** 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:-

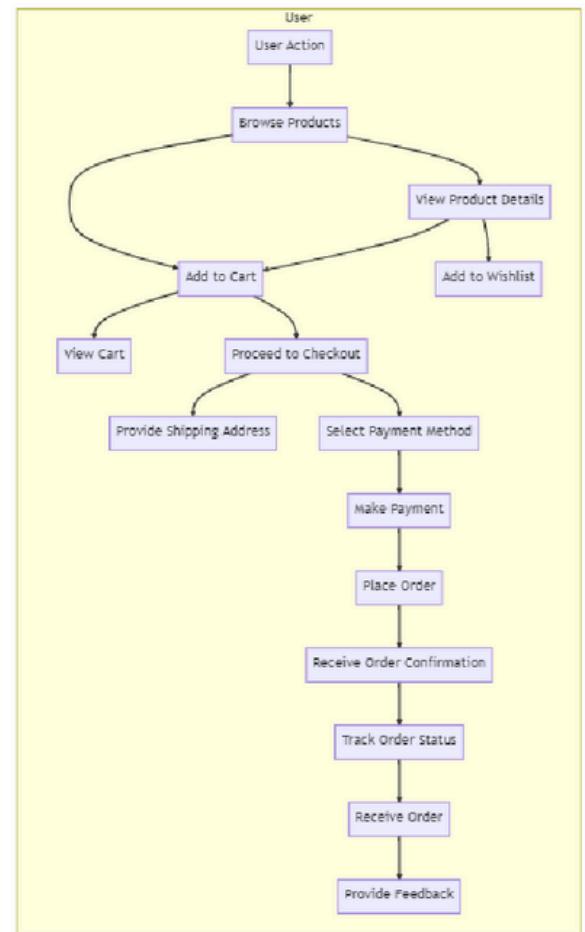
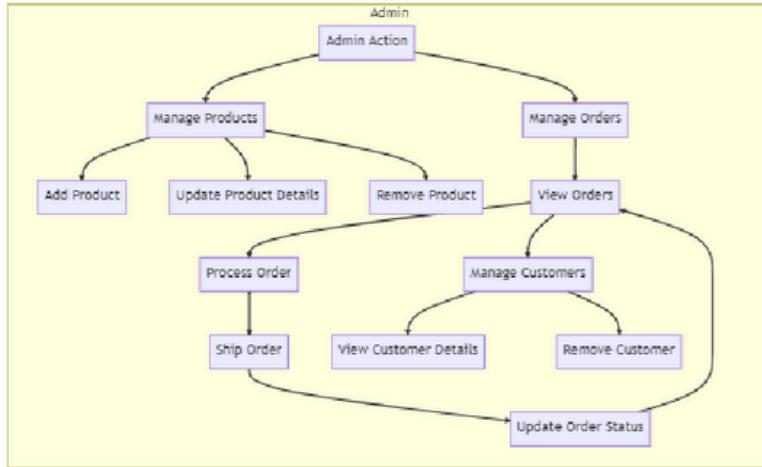
- Registration and Authentication: Users are responsible for creating an account on the platform and securely logging in to access its features.
- Browsing and Shopping: Users can browse products, add them to their cart, and proceed to checkout for purchasing.
- Payment: Users are responsible for making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

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

Demo link:-

https://drive.google.com/file/d/1HLowclqs2d8lxTprS2jqPmR4AOnUW8xD/view?usp=drive_link

Use the code in:-

https://drive.google.com/drive/folders/1RP-29p9mf-bbLAK5r7S4HSF_Itai0i1G?usp=drive_link

or follow the videos below for better understanding.

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.

Reference Link:-

https://drive.google.com/file/d/1Acv3Lx3PtJcOYkUjREWAzloCi6w96Tl/view?usp=drive_link

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.

Reference Link:-

https://drive.google.com/file/d/18-gnCVAZdojBhl7QMO_yShH1JL5A6v32/view?usp=drive_link

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.

```

1 const mongoose = require('mongoose');
2
3 mongoose.connect(
4   'mongodb+srv://shopsmart:test123@cluster0.jsjtxes.mongodb.net/grocerydb?retryWrites=true&w=majority&appN
5   {
6     useNewUrlParser: true,
7     useUnifiedTopology: true
8   }
9 )
Tabnine | Edit | Test | Explain | Document
10 .then(() => console.log('✅ Connected to MongoDB Atlas'))
Tabnine | Edit | Test | Explain | Document
11 .catch((err) => console.error('✗ MongoDB connection error:', err));
12

```

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, required: true },
  lastname: { type: String, required: true },
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

// 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, max: 255 },
  rating: { type: Number, required: true },
  unit: { type: String, default: 'Kg' }, // Optional field for unit
  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 paymentSchema = new mongoose.Schema({
  user: { type: String, ref: 'User', required: true },
  name: { type: String, required: true },
  order: { type: String, ref: 'Order', required: true },
  amount: { type: Number, required: true },
  paymentMethod: { type: String, required: true },
  deliveryStatus: { type: String, required: true },
  status: { type: String, enum: ['Pending', 'Success', 'Failed'], default: 'Pending' },
  createdAt: { type: Date, default: Date.now }
});

const feedbackSchema = new mongoose.Schema({
  user: { type: String, required: true },
  message: { 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),
  Payment: mongoose.model('Payment', paymentSchema),
  Feedback: mongoose.model('Feedback', feedbackSchema)
};

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:-

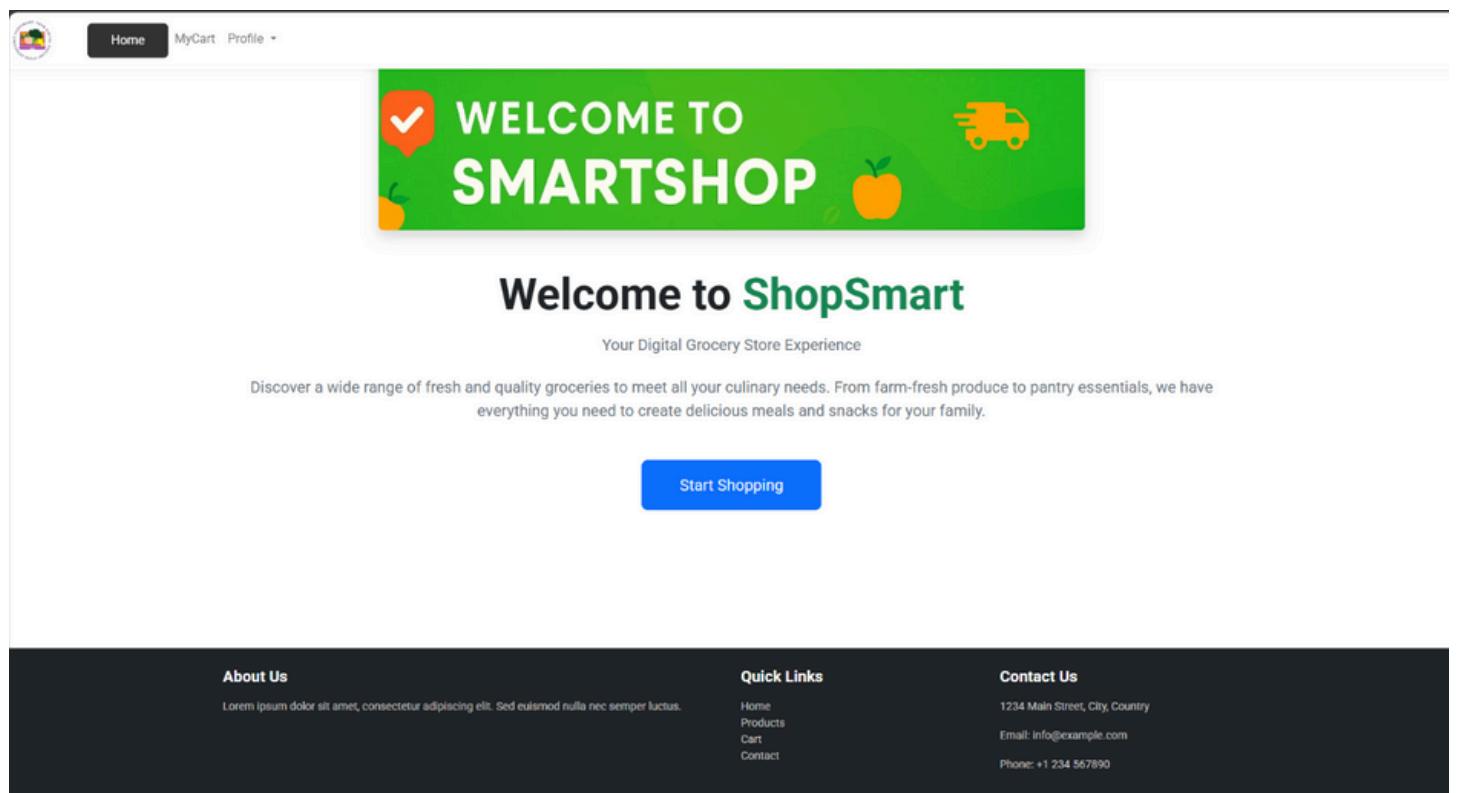
https://drive.google.com/file/d/1MFQ7NIQ02ynLioEMySPpEzdX-c0i6Wer/view?usp=drive_link

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:-



The screenshot shows the landing page of the ShopSmart digital grocery store. The top navigation bar includes a logo, 'Home', 'MyCart', and 'Profile'. The main header features a green banner with the text 'WELCOME TO SMARTSHOP' and icons for a checkmark, a delivery truck, and fruits. Below the banner, the page title is 'Welcome to ShopSmart' with the subtitle 'Your Digital Grocery Store Experience'. A descriptive paragraph highlights the variety of fresh and quality groceries available. A large blue 'Start Shopping' button is centered. At the bottom, there are three columns: 'About Us' (with placeholder text), 'Quick Links' (with links to Home, Products, Cart, and Contact), and 'Contact Us' (with address, email, and phone number).

Login Page:-



Home MyCart Profile ▾



Login

Email

Enter email

Password

Enter password

Sign In

[Don't have an account? click here](#)

Items Page:-



Home MyCart Profile ▾

Logout

Find Your Groceries

Search products...



Categories ▾



Available Products



Tomatoes

Fresh red tomatoes

₹30/Kg



Bananas

Ripe yellow bananas

₹40/Dozen



Apples

Kashmir apples

₹100/Dozen



Brown Bread

Healthy multi-grain brown bread

₹35/Pack

My Cart:-

The screenshot shows a user interface for a shopping cart. At the top, there is a navigation bar with icons for Home, MyCart (which is highlighted), Profile, and Logout. A small notification badge with the number '3' is visible above the MyCart button. The main title 'My Cart' is centered at the top, accompanied by a shopping cart icon.

The cart contains three items, each displayed in a separate card:

- Apples**
Price: ₹100
[Buy this product](#) X
- Croissant**
Price: ₹50
[Buy this product](#) X
- Rice**
Price: ₹60
[Buy this product](#) X

My Orders Page:-



Home MyCart Profile ▾

Logout

My Orders

Name: akshay

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Order Canceled

My History Page:-



Home MyCart Profile ▾

Logout

History

Product Name: Grapes

Quantity: 2

Total Price: 100

Payment Method: cash-on-delivery

Address: andhrapradesh

Created At: Jun 26, 2025

Status: Delivered

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Place Order Page:-



Order Details

Firstname

Lastname

Phone

Quantity

Address

Payment method:

[Confirm Order](#)

Admin Dashboard Page:



Profile ▾ [Logout](#)

Grocery



Dashboard

[Dashboard](#)

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

Users

Total users: 4

[View Users](#)

Orders

Total orders: 5

[View Orders](#)

Add Products

Click to add new products

[Add Products](#)

Add Category

Click to add new category

[Add Category](#)

Users Page:-



Profile ▾ [Logout](#)

Grocery



Users

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

User Id: 68565e8b5b8a9f2270576ee1

First Name: Admin

Last Name: User

Username: admin123

Email: admin@gmail.com

User Id: 6858ff136cb82a281fedd7f0

First Name: Akshay

Last Name: Chatla

Username: akshay828

Email: akshaychatla99@gmail.com

User Id: 685a3817186404bc217e3388

First Name: test

Last Name: 123

Username: test123

Email: test@gmail.com

User Id: 685cf00a3fd383cef55adbfa

First Name: test

Last Name: 1

Username: test1

Email: test1@gmail.com

Admin's Add Products page:-

The screenshot shows the 'Add Products' section of the admin interface. On the left, there's a sidebar with a profile icon, 'Profile' and 'Logout' buttons at the top, and a navigation menu with options: Dashboard, Users, Products, Add Products (which is highlighted), Add Category, Orders, and Feedback. The main area has a title 'Add Products'. It contains several input fields: 'Productname' (placeholder 'Enter productname'), 'Price' (placeholder 'Enter price'), 'Image' (placeholder 'Enter image url'), 'Category' (placeholder 'Enter category'), 'Stock' (placeholder 'Enter stock'), 'Rating' (placeholder 'Enter rating'), 'Unit' (placeholder 'Unit' with 'Kg' selected), and a large 'Description' text area (placeholder 'Enter description'). At the bottom is a grey button labeled 'Add Product'.

Admin Orders Page:-

The screenshot shows the 'Orders' section of the admin interface. The sidebar is identical to the previous one. The main area displays a table of orders. The first order row contains: Order ID: 685ce38f3fd383cef55adb0b, Fullname: Akshay Chatla, Phone: 07075768029, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-406 sri lanka colony vuyyuru, andhra pradesh, Created At: Jun 26, 2025, Status: Delivered, and a green 'Delivered' button. The second order row contains: Order ID: 685ceee53fd383cef55adbcb, Fullname: Akshay 12345, Phone: 1234567891, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-221, andhra pradesh, and a grey 'Pending' button.

The demo of the app is available at:-

[Demo video.mp4](#)

Full Stack Development with MERN Project Documentation

Introduction:

- **Project Title:** ShopSmart: Your Digital Grocery Store Experience

- **Team Members:**

Team Leader: Palagani Veda Sri

Team member: Naresh Perumalla

Team member: Nelavalli Venkata Bhaskara Devi Phanindra

Team member: Palaparthi Babu

Welcome to our Grocery 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!

Description:

Our Grocery Web App is more than just a convenient way to shop for groceries—it's a comprehensive solution designed to enhance every aspect of your shopping experience. Imagine a

virtual supermarket at your fingertips, offering an extensive selection of high-quality products that cater to your every need.

From fresh produce sourced directly from local farms to pantry staples and household essentials, our app provides everything you need to keep your kitchen well-stocked. Our user-friendly interface ensures that you can navigate our virtual aisles effortlessly, easily finding what you're looking for and adding items to your cart with a simple click.

But the convenience doesn't stop there. With our app, you can schedule deliveries at your preferred time, ensuring that your groceries arrive when it's most convenient for you. Say goodbye to long lines and tedious grocery trips—our app brings the supermarket to you, saving you time and hassle.

We understand that everyone's dietary preferences and requirements are different, which is why our app offers a wide range of options, including organic, gluten-free, and vegan products. You can also discover new and exciting products through our app, making grocery shopping an adventure rather than a chore.

Our commitment to quality extends beyond our products to our customer service. Our dedicated team is always available to assist you with any questions or concerns you may have, ensuring that your shopping experience is nothing short of exceptional.

Experience the future of grocery shopping with our Grocery Web App. Download it today and discover the convenience of fresh, high-quality groceries delivered right to your doorstep.

Scenario Based Case Study:

Meet priya, 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.

- **Priya's Solution:** The Grocery Web App.

Priya discovers the Grocery Web App, 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: Priya 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: Priya 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 Priya's purchase history and preferences to provide personalized recommendations, helping her discover new products and brands that align with her tastes.

Convenient Delivery Options: Priya 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.

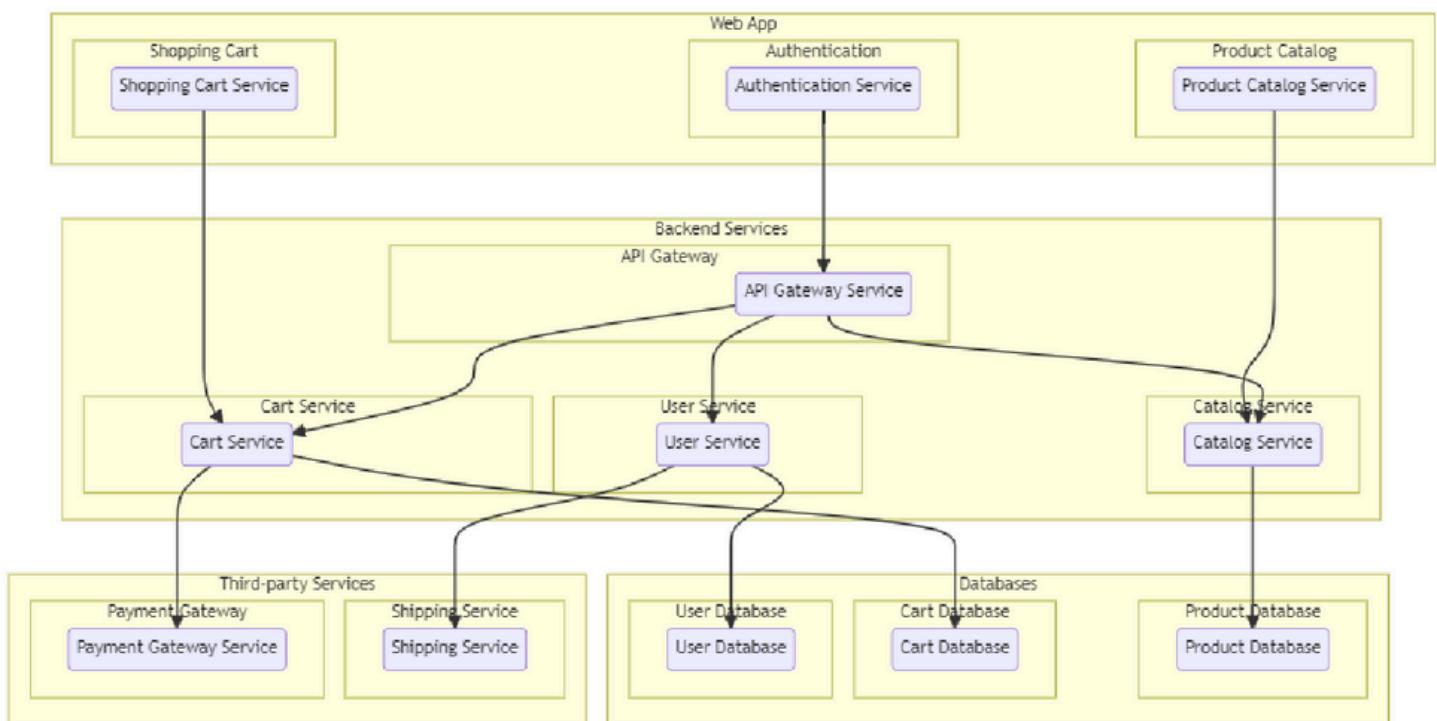
Secure Payment Gateway: The app integrates with a secure payment gateway, allowing Priya to pay for her groceries online using various payment methods, including credit/debit cards and digital wallets.

Order Tracking: Priya 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 Priya with any queries or concerns she may have.

Priya's Experience: Thanks to the Grocery Web App, Priya 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.

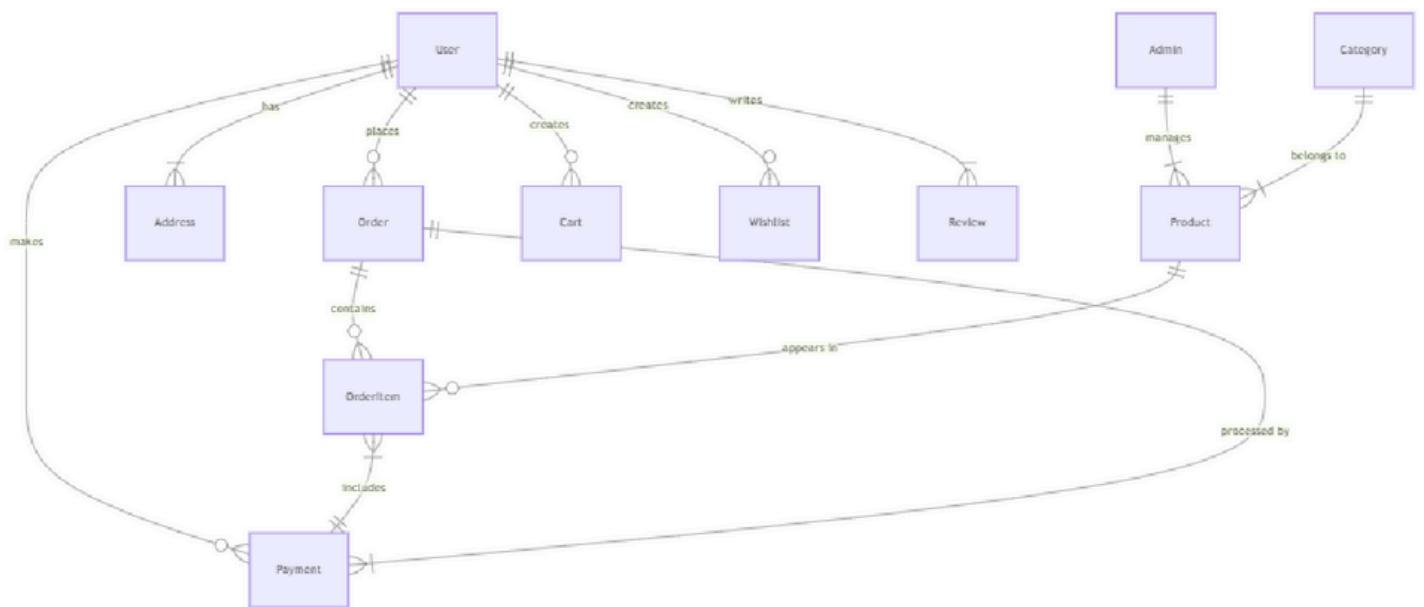
Technical Architecture:-



The technical architecture of a 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 a 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-Appiaction

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 advanced 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.

- **VisualStudioCode:** 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.

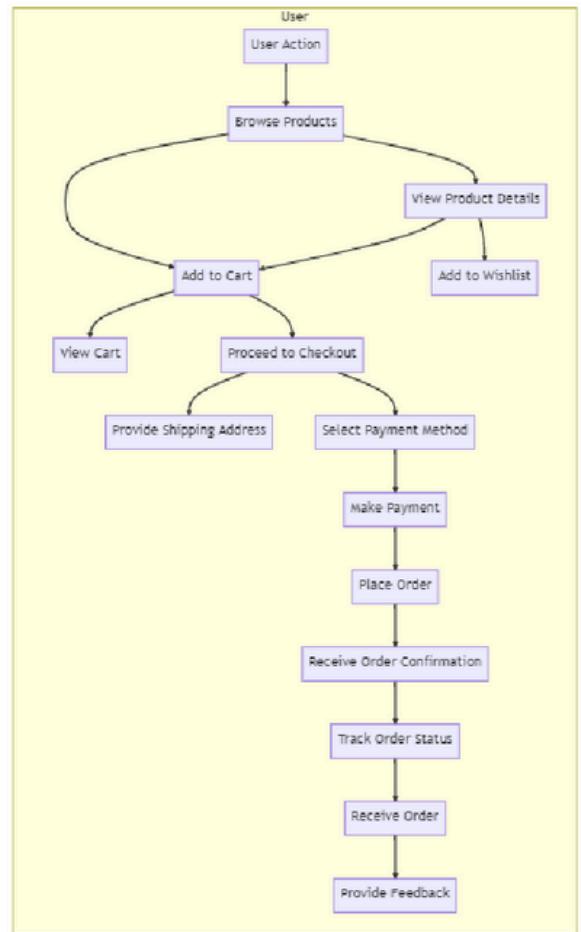
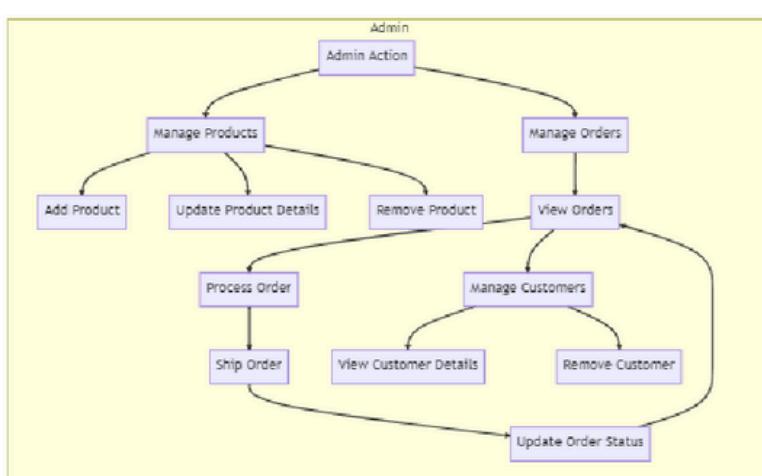
Admin:-

- 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:-

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

Demo link:-

https://drive.google.com/file/d/1HLowclqs2d8lxTprS2jqPmR4AOnUW8xD/view?usp=drive_link

Use the code in:-

https://drive.google.com/drive/folders/1RP-29p9mf-bbLAK5r7S4HSF_Itai0i1G?usp=drive_link

or follow the videos below for better understanding.

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.

Reference Link:-

https://drive.google.com/file/d/1Acv3Lx3PtJcOYkUjREWAzIoCi6w96Tl/view?usp=drive_link

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.

Reference Link:-

https://drive.google.com/file/d/18-gnCVAZdojBhl7QMO_yShH1JL5A6v32/view?usp=drive_link

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.

```
1 const mongoose = require('mongoose');
2
3 mongoose.connect(
4   'mongodb+srv://shopsmart:test123@cluster0.jsjtxes.mongodb.net/grocerydb?retryWrites=true&w=majority&appN
5   {
6     useNewUrlParser: true,
7     useUnifiedTopology: true
8   }
9 )
10 Tabnine | Edit | Test | Explain | Document
11 .then(() => console.log('✅ Connected to MongoDB Atlas'))
12 Tabnine | Edit | Test | Explain | Document
13 .catch((err) => console.error('✗ MongoDB connection error:', err));
14
```

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, required: true },
  lastname: { type: String, required: true },
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

// 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, max: 255 },
  rating: { type: Number, required: true },
  unit: { type: String, default: 'Kg' }, // Optional field for unit
  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 paymentSchema = new mongoose.Schema({
  user: { type: String, ref: 'User', required: true },
  name: { type: String, required: true },
  order: { type: String, ref: 'Order', required: true },
  amount: { type: Number, required: true },
  paymentMethod: { type: String, required: true },
  deliveryStatus: { type: String, required: true },
  status: { type: String, enum: ['Pending', 'Success', 'Failed'], default: 'Pending' },
  createdAt: { type: Date, default: Date.now }
});

const feedbackSchema = new mongoose.Schema({
  user: { type: String, required: true },
  message: { 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),
  Payment: mongoose.model('Payment', paymentSchema),
  Feedback: mongoose.model('Feedback', feedbackSchema)
};

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/1MFQ7NIQ02ynLioEMySPpEzdX-c0i6Wer/view?usp=drive_link

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:-

The screenshot shows the homepage of a digital grocery store named ShopSmart. At the top, there is a navigation bar with links for Home, MyCart, and Profile. The main header features a large green banner with the text "WELCOME TO SMARTSHOP" in white, flanked by a checkmark icon and a delivery truck icon. Below the banner, there are illustrations of fruit like an orange and an apple. The main title "Welcome to ShopSmart" is centered, followed by the tagline "Your Digital Grocery Store Experience". A prominent blue "Start Shopping" button is located below the tagline. In the footer, there are three columns: "About Us" (with placeholder text), "Quick Links" (listing Home, Products, Cart, and Contact), and "Contact Us" (with address, email, and phone number).

About Us
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed euismod nulla nec semper luctus.

Quick Links
Home
Products
Cart
Contact

Contact Us
1234 Main Street, City, Country
Email: info@example.com
Phone: +1 234 567890

Login Page:-



Home MyCart Profile ▾



Login

Email

Enter email

Password

Enter password

Sign In

[Don't have an account? click here](#)

Items Page:-



Home MyCart Profile ▾

Logout

Find Your Groceries

Search products...



Categories ▾



Available Products



Tomatoes

Fresh red tomatoes

₹30/Kg



Bananas

Ripe yellow bananas

₹40/Dozen



Apples

Kashmir apples

₹100/Dozen



Brown Bread

Healthy multi-grain brown bread

₹35/Pack

My Cart:-

The screenshot shows a user interface for a shopping cart. At the top, there is a navigation bar with icons for Home, MyCart (which is highlighted), Profile, and Logout. A small notification badge with the number '3' is visible above the MyCart button. The main title 'My Cart' is centered at the top, accompanied by a shopping cart icon.

The cart contains three items, each displayed in a separate card:

- Apples**
Price: ₹100
[Buy this product](#) X
- Croissant**
Price: ₹50
[Buy this product](#) X
- Rice**
Price: ₹60
[Buy this product](#) X

My Orders Page:-



Home MyCart Profile ▾

Logout

My Orders

Name: akshay

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Order Canceled

My History Page:-



Home MyCart Profile ▾

Logout

History

Product Name: Grapes

Quantity: 2

Total Price: 100

Payment Method: cash-on-delivery

Address: andhrapradesh

Created At: Jun 26, 2025

Status: Delivered

Product Name: Rice

Quantity: 2

Total Price: 120

Payment Method: paypal

Address: 6-221, andhrapradesh

Created At: Jun 26, 2025

Status: Canceled

Place Order Page:-



Order Details

Firstname

Lastname

Phone

Quantity

Address

Payment method:

[Confirm Order](#)

Admin Dashboard Page:



Profile ▾ [Logout](#)

Grocery



Dashboard

[Dashboard](#)

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

Users

Total users: 4

[View Users](#)

Orders

Total orders: 5

[View Orders](#)

Add Products

Click to add new products

[Add Products](#)

Add Category

Click to add new category

[Add Category](#)

Users Page:-



Profile ▾ [Logout](#)

Grocery



Users

[Users](#)

[Products](#)

[Add Products](#)

[Add Category](#)

[Orders](#)

[Feedback](#)

User Id: 68565e8b5b8a9f2270576ee1

First Name: Admin

Last Name: User

Username: admin123

Email: admin@gmail.com

User Id: 6858ff136cb82a281fedd7f0

First Name: Akshay

Last Name: Chatla

Username: akshay828

Email: akshaychatla99@gmail.com

User Id: 685a3817186404bc217e3388

First Name: test

Last Name: 123

Username: test123

Email: test@gmail.com

User Id: 685cf00a3fd383cef55adbfa

First Name: test

Last Name: 1

Username: test1

Email: test1@gmail.com

Admin's Add Products page:-

The screenshot shows the 'Add Products' section of the admin interface. On the left, there's a sidebar with a profile icon, 'Profile' and 'Logout' buttons at the top, and a navigation menu with items: Dashboard, Users, Products, Add Products (which is highlighted), Add Category, Orders, and Feedback. The main area has a title 'Add Products'. It contains several input fields: 'Productname' (placeholder 'Enter productname'), 'Price' (placeholder 'Enter price'), 'Image' (placeholder 'Enter image url'), 'Category' (placeholder 'Enter category'), 'Stock' (placeholder 'Enter stock'), 'Rating' (placeholder 'Enter rating'), 'Unit' (placeholder 'Unit' with 'Kg' selected), and a large 'Description' text area (placeholder 'Enter description'). At the bottom is a grey button labeled 'Add Product'.

Admin Orders Page:-

The screenshot shows the 'Orders' section of the admin interface. The sidebar is identical to the previous one. The main area displays a table of orders. The first order row contains: Order ID: 685ce38f3fd383cef55adb0b, Fullname: Akshay Chatla, Phone: 07075768029, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-406 sri lanka colony vuyyuru, andhra pradesh, Created At: Jun 26, 2025, Status: Delivered, and a green 'Delivered' button. The second order row contains: Order ID: 685ceee53fd383cef55adbcb, Fullname: Akshay 12345, Phone: 1234567891, Product ID: 685b7ba0a6d4b8aac8496df4, Quantity: 2, Total price: 70, Payment Method: cash-on-delivery, Address: 6-221, andhra pradesh, and a grey 'Pending' button.

The demo of the app is available at:-

[Demo video.mp4](#)