



JEPPIAAR

ENGINEERING COLLEGE

JEPPIAAR NAGAR, RAJIVGANDHI SALAI

CHENNAI – 600119.

DEPARTMENT OF INFORMATION TECHNOLOGY

IV YEAR B.TECH – VII SEM

ACADEMIC YEAR 2024 - 25 (ODD SEM)

NM1042 - MERN Stack Powered by MongoDB

Grocery WebApp Project Report

(Naan Mudhalvan Project)

TEAM MEMBERS

Rakshitha Sree G	- 310821205070
Nivedha S	- 310821205056
Sushiha Eroni Vas A	- 310821205099
Sheeba Jebakani J	- 310821205086



JEPPIAAR

ENGINEERING COLLEGE

JEPPIAAR NAGAR, RAJIVGANDHI SALAI, CHENNAI – 600119.

DEPARTMENT OF INFORMATION TECHNOLOGY

This is a Bonafide Record Work of _____

Register No. _____ submitted for the Anna University Practical

Examination held on _____ in **NM1042 - MERN Stack Powered by**

MongoDB during the year _____.

Signature of the Faculty-In-Charge

Signature of the HOD

Date: _____

Examiners Internal: _____

External: _____

COLLEGE VISION & MISSION

Vision

To build Jeppiaar Engineering College as an institution of academic excellence in technological and management education to become a world class university.

Mission

- To excel in teaching and learning, research and innovation by promoting the principles of scientific analysis and creative thinking.
- To participate in the production, development and dissemination of knowledge and interact with national and international communities.
- To equip students with values, ethics and life skills needed to enrich their lives and enable them to contribute for the progress of society.
- To prepare students for higher studies and lifelong learning, enrich them with the practical skills necessary to excel as future professionals and entrepreneurs for the benefit of Nation's economy.

Program Outcomes

PO1	Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

To produce engineers with excellent knowledge in the field of Information Technology through scientific and practical education to succeed in an increasingly complex world.

Mission

- To demonstrate technical and operational excellence through creative and critical thinking for the effective use of emerging technologies.
- To involve in a constructive, team-oriented environment and transfer knowledge to enable global interaction.
- To enrich students with professional integrity and ethical standards that will make them deal social challenges successfully in their life.
- To devise students for higher studies and perpetual learning, upgrade them as competent engineers and entrepreneurs for country's development.

Program Educational Objectives (PEOs)

PEO 1	To support students with substantial knowledge for developing and resolving mathematical, scientific and engineering problems
PEO 2	To provide students with adequate training and opportunities to work as a collaborator with informative and administrative qualities
PEO 3	To shape students with principled values to follow the code of ethics in social and professional life
PEO 4	To motivate students for extensive learning to prepare them for graduate studies, R&D and competitive exams
PEO 5	To cater the students with industrial exposure in an endeavor to succeed in the emerging cutting-edge technologies

Program Specific Outcomes

PSO1	Students are able to analyze, design, implement and test any software with the programming and testing skills they have acquired.
PSO2	Students are able to design algorithms, data management to meet desired needs, for real time problems through analytical, logical and problem solving skills.
PSO3	Students are able to provide security solutions for network components and data storage & management which will enable them to work in the industry ethically.

Course Outcomes (COs)

C407.1	Configure various virtualization tools such as Virtual Box, VMware workstation
C407.2	Design and deploy a web application in a PaaS environment
C407.3	Learn how to simulate a cloud environment to implement new schedulers.
C407.4	Install and use a generic cloud environment that can be used as a private cloud.
C407.5	Install and use Hadoop

Table of Contents

S.No	Topic
1	Introduction
2	Project Overview
3	Architecture
4	Setup Instructions
5	Folder Structure
6	Running the Application
7	API Documentation
8	Authentication
9	User Interface
10	Testing
11	Screenshots or Demo
12	Known Issues
13	Future Enhancements

1. Introduction

Project Title: Grocery WebApp (MERN Stack)

Team Members and Roles:

- **Rakshitha Sree G [310821205070]:** *Team Lead and Backend Developer* - Responsible for API development, database integration, and deployment.
- **Sheeba Jebakani J [310821205086]** *Frontend Developer* - Designed the user interface, implemented React components, and ensured responsive design.
- **Nivedha S [310821205056]:** *Database Manager* - Designed and managed MongoDB schemas, ensured data consistency, and optimized queries.
- **Sushiha Eroni Vas A [310821205099]:** *Quality Assurance and Documentation* - Conducted application testing, bug fixes, and prepared the documentation.

2. Project Overview

Purpose: The Grocery eCommerce Platform is a full-stack web application designed to provide consumers with a convenient way to purchase grocery items online. It aims to offer a user-friendly interface for customers, secure payment processing, and comprehensive backend management tools for administrators. The platform will enable businesses to expand their market reach beyond geographical boundaries.

Key Objectives:

1. Develop a user-friendly platform for customers to browse and purchase grocery items.
2. Implement secure payment processing to ensure safe transactions.
3. Build a robust backend for administrators to manage inventory, orders, and customer data.
4. Ensure the platform is scalable and can handle high traffic volumes.

Features:

- **User Authentication:** Secure registration, login, and password recovery features for users.
- **Product Catalog:** Displays grocery items with detailed descriptions, prices, and images.
- **Shopping Cart:** Enables users to add, view, and modify products before checkout.
- **Order Management:** Allows users to track and manage their orders.

- Admin Dashboard: Provides administrators with tools to manage inventory, view orders, and analyze customer data.
- Secure Payment Gateway: Integration with payment providers for secure transaction processing.

3. Architecture

3.1. Application Overview:

This is a MERN (MongoDB, Express.js, React, Node.js) application with two primary modules: Admin and Customer. It features a separate Admin Login Portal for administrative tasks and a Customer Module for shopping activities.

3.2. Modules:

A. Admin Module:

The Admin Module is designed for administrators to manage the platform. Key features include:

- Product Management: Add, update, or delete products in the catalog.
- User Management: View and manage registered customers.
- Order Management: Track customer orders and view order statuses.
- Analytics Dashboard: Monitor metrics such as user count, order count, and product availability.

B. Customer Module:

The Customer Module is tailored for a smooth shopping experience. Key features include:

- Product Catalog: Browse products with images, prices, and descriptions.
- Cart Management: Add products to the cart, update quantities, or remove items.
- Order Placement: Checkout and place orders with ease.
- Payment Options: Secure payment gateways for online payments.

3.3. System Components:

A. Frontend:

- Built using React for a responsive and interactive user interface.
- Separate views for Admin and Customer.
- Interacts with the backend using Axios or Fetch API for RESTful API calls.

B. Backend:

- Built with Node.js and Express.js for API handling and business logic.
- Key functionalities:
 - Authentication and Authorization: Implemented with JWT (JSON Web Tokens) for secure role-based access control.
 - Business Logic: Separate endpoints for Admin and Customer operations.
 - Email Notifications: For account verification, order confirmations, and updates.
 - Payment Gateway Integration: Handles secure payment processing.

C. Database:

- Uses MongoDB for efficient storage and retrieval of data.
- Key collections:
 - Users: Stores user details with roles (Admin or Customer).
 - Products: Manages product inventory.
 - Orders: Tracks orders with user and product references.
 - Cart: Temporarily stores items selected by customers before checkout.

D. Security:

- Secure authentication with JWT and password hashing using bcrypt.
- Role-based access control ensures Admin and Customer permissions.
- Input validation and database sanitization prevent common security threats.

3.4. High-Level Flow:

1. Admin Workflow:

- Admin logs in via the Admin Portal.
- Manages product inventory (add, update, or delete products).

- Tracks orders and reviews customer details using the dashboard.

2. Customer Workflow:

- Customer registers and logs into the platform.
- Browses products, adds items to the cart, and proceeds to checkout.
- Completes payment using integrated payment gateways and places an order.
- Receives email confirmation and tracks the order status.

3. Payment Flow:

- Payment data is securely processed through APIs integrated with payment gateways (e.g., Razorpay, Stripe).
- Successful transactions generate order confirmations stored in the database.

4. Setup Instructions

Prerequisites:

Ensure the following are installed on your system before proceeding:

- **Node.js:** Version 16 or later (Install from [Node.js](#))
- **MongoDB:** Local installation or cloud-based service like MongoDB Atlas (Install from [MongoDB Community Edition](#))
- **Git:** For version control and cloning the repository (Install from [Git](#))

Installation Steps

1. Clone the Repository

To get started, clone the project repository to your local machine using Git:

```
git clone <https://github.com/rakshithasree/naan-mudhalvan-grocery-webapp>
```

2. Navigate to the Project Directory

After cloning the repository, navigate into the project folder:

```
cd naan-mudhalvan-grocery-webapp
```

You should now see two directories: client (frontend) and server (backend).

3. Install Dependencies

Install the required Node.js packages for both the **client** and **server**.

1. For the Frontend (Client):

- Navigate to the client folder:

```
cd client
```

- Install the dependencies:

```
npm install
```

- This will install all the necessary packages specified in package.json.

2. For the Backend (Server):

- Navigate to the server folder:

```
cd ../server
```

- Install the dependencies:

```
npm install
```

4. Set Environment Variables

Both the **client** and **server** require configuration through .env files for environment variables.

1. **Backend (Server) .env File:** Create a file named .env in the server directory and add the following:

```
MONGO_URI=mongodb://localhost:27017/
```

```
JWT_SECRET=secret_key
```

```
PORT=27017
```

- **MONGO_URI:** URL to connect to your MongoDB instance (update if you're using MongoDB Atlas).
- **JWT_SECRET:** A secret key for JSON Web Token authentication.
- **PORT:** Port number on which the backend server will run (default is 5000).

2. **Frontend (Client) .env File:** Create a file named .env in the client directory and add:

```
REACT_APP_API_BASE_URL=http://localhost: 27017/
```

- **REACT_APP_API_BASE_URL:** The base URL for the backend API (ensure the backend server is running on this URL).

5. Run the Application

Start both the backend and frontend services.

1. Start the Backend (Server):

- Navigate to the server folder:

```
cd server
```

- Run the server:

```
node index.js
```

- For development mode with automatic restarts:

```
npx nodemon index.js
```

- The backend will run on [http://localhost: 27017](http://localhost:27017).

2. Start the Frontend (Client):

- Navigate to the client folder:

```
cd ../client
```

- Run the React development server:

```
Npm start
```

- The frontend will run on <http://localhost:3000>.

6. Test the Application

- Open your browser and navigate to <http://localhost:3000> to view the application.
- The frontend should connect to the backend API running on [http://localhost: 27017](http://localhost:27017).

1. Database Setup:

- Ensure MongoDB is running locally or replace MONGO_URI in the .env file with your MongoDB Atlas connection string.

2. API Integration:

- Confirm that API calls in the frontend (client) point to the correct backend URL (<http://localhost:5000> or your hosted API URL).

3. Development Tools:

- Use nodemon for backend development to auto-restart the server on file changes.

5. Folder Structure

Root Directory

```
naan-mudhalvan-grocery-webapp/
|
├── client/
|
├── server/
|
├── .gitignore
|
├── README.md                # Project documentation
|
├── package.json
|
└── package-lock.json
```

Client Directory (Frontend)

```
client/
|
├── node_modules/
|
├── public/
|   ├── index.html
|   └── favicon.ico
|
├── src/
|   ├── components/
|   │   └── (e.g., Navbar, Footer, etc.)
|   └──
```

```

|   └─ context/
|   |
|   └─ images/
|   |
|   └─ pages/
|   |
|   └─ styles/
|       └─ index.css
|       |
|       └─ App.js
|       └─ App.css
|       └─ index.js
|       └─ reportWebVitals.js
|       └─ setupTests.js
|       └─ logo.svg
|
└─ package.json
└─ package-lock.json

```

Server Directory (Backend)

```
server/
|
├─ node_modules/
|
├─ index.js           # Main server file
|
├─ Schema.js
|
├─ .env              # Environment variables for
backend
|
```

```
|— package.json
└— package-lock.json
```

6. Running the Application

1. Start the Frontend:

```
cd client
npm start
```

2. Start the Backend:

```
cd server
npm start
```

7. API Documentation

Base URL: `http://localhost: 27017`

Authentication

Most APIs require a **JWT Token** for authorization. Include the token in the headers:

`Authorization: Bearer <jwt-token>`

Endpoints

1. User Management

1.1 Register a New User

POST `/api/users/register`

Description: Creates a new user account.

- **Request Body:**

```
{
  "name": "John Doe",
  "email": "john.doe@example.com",
  "password": "securepassword"
}
```

- **Response:**

```
{
  "message": "User registered successfully!"
}
```

1.2 Login User

POST /api/users/login

Description: Authenticates a user and returns a JWT token.

- **Request Body:**

```
{
  "email": "john.doe@example.com",
  "password": "securepassword"
}
```

- **Response:**

```
{
  "token": "jwt-token-string",
  "user": {
    "id": "userId123",
    "name": "John Doe",
    "email": "john.doe@example.com"
  }
}
```

2. Grocery Management

2.1 Get All Groceries

```
GET /api/groceries
```

Description: Fetches a list of all available groceries.

- **Response:**

$$\left[\begin{array}{c} \vdots \\ \vdots \end{array} \right]$$

```
    "id": "groceryId1",
    "name": "Apples",
    "price": 100,
    "quantity": 50,
    "category": "Fruits"
  },
  {
    "id": "groceryId2",
    "name": "Milk",
    "price": 50,
    "quantity": 30,
    "category": "Dairy"
  }
]
```

2.2 Add a New Grocery

POST /api/groceries

Description: Adds a new grocery item. (Admin only)

- **Request Body:**

```
{
  "name": "Bananas",
  "price": 60,
  "quantity": 100,
  "category": "Fruits"
}
```

- **Response:**

```
{
  "message": "Grocery item added successfully!"
}
```

2.3 Update a Grocery Item

PUT /api/groceries/:id

Description: Updates an existing grocery item. (Admin only)

- **Request Body:**

```
{  
  "name": "Bananas",  
  "price": 70,  
  "quantity": 120,  
  "category": "Fruits"  
}
```

- **Response:**

```
{  
  "message": "Grocery item updated successfully!"  
}
```

2.4 Delete a Grocery Item

DELETE /api/groceries/:id

Description: Deletes a grocery item. (Admin only)

- **Response:**

```
{  
  "message": "Grocery item deleted successfully!"  
}
```

3. Order Management

3.1 Place an Order

POST /api/orders

Description: Places a new order for groceries.

- **Request Body:**

```
{  
  "userId": "userId123",  
  "items": [  
    {
```

```
    "groceryId": "groceryId1",
    "quantity": 3
  },
  {
    "groceryId": "groceryId2",
    "quantity": 2
  }
]
```

- **Response:**

```
{
  "message": "Order placed successfully!",
  "orderId": "orderId123"
}
```

3.2 Get User Orders

GET /api/orders/:userId

Description: Fetches all orders for a specific user.

- **Response:**

```
[
  {
    "orderId": "orderId123",
    "userId": "userId123",
    "items": [
      {
        "groceryId": "groceryId1",
        "quantity": 3,
        "name": "Apples",
        "price": 100
      },
      {
```

```
        "groceryId": "groceryId2",
        "quantity": 2,
        "name": "Milk",
        "price": 50
    }
],
"totalAmount": 400,
"orderDate": "2024-11-25T14:30:00.000Z"
}
```

4. Admin Features

4.1 View All Users

GET /api/admin/users

Description: Fetches a list of all registered users. (Admin only)

- **Response:**

```
[
  {
    "id": "userId123",
    "name": "John Doe",
    "email": "john.doe@example.com",
    "isAdmin": false
  },
  {
    "id": "userId124",
    "name": "Admin User",
    "email": "admin@example.com",
    "isAdmin": true
  }
]
```

Error Responses

For all endpoints, errors are returned in the following format:

- **Response:**

```
{  
  "error": "Invalid credentials"  
}
```

8. Authentication

- **JWT Tokens:** Used for user session management.
- **bcrypt.js:** Ensures secure password storage with hashing.
- **Middleware:** Protects routes requiring user authentication.

9. User Interface

1. Landing Page

- **Purpose:** First impression of the platform; introduces users to the features and services.
- **Key Features:**
 - Hero banner with a carousel of promotional offers or seasonal deals.
 - Quick navigation to product categories (e.g., Vegetables, Fruits, Beverages).
 - Search bar for finding specific products quickly.
 - Buttons for Login/Signup and Cart Access.
 - Footer with links to About Us, Contact, Help Center, and Social Media Icons.

2. Registration and Login Pages

- **Purpose:** Enable secure user account creation and login.
- **Key Features:**
 - Registration Page:
 - Fields for name, email, phone number, password, and confirm password.

- Checkbox for agreeing to terms and conditions.
- "Register" button.
- Login Page:
 - Email and password fields.
 - "Forgot Password?" link for password recovery.
 - "Login" button.
- Multi-Factor Authentication (MFA):
 - After login, a prompt to enter a One-Time Password (OTP) sent to the user's registered email/phone.
 - Countdown timer for OTP expiration and "Resend OTP" button.

3. Homepage (After Login)

- **Purpose:** Provide a personalized experience based on user preferences.
- **Key Features:**
 - Product Categories:
 - Horizontal scroll or grid layout for categories like *Fresh Produce*, *Dairy*, *Snacks*, *Beverages*, *Bakery*.
 - Recommendations:
 - Section displaying products based on the user's past purchases or preferences.
 - Top Deals:
 - Highlighted discounts and offers on products.
 - Search and Filters:
 - Search bar with advanced filters (e.g., price range, brand, popularity, rating).
 - Quick Links:
 - "View Order History," "Track Orders," and "Wishlist."

4. Product Listing Page

- **Purpose:** Display a curated list of products based on the selected category or search query.
- **Key Features:**

- Grid or list layout for displaying products.
- Product cards showing:
 - Image
 - Name
 - Price
 - Ratings (stars and count)
 - "Add to Cart" button
- Filters and sorting options:
 - Price (low to high, high to low)
 - Ratings
 - Availability
- Pagination or infinite scroll for product navigation.

5. Product Detail Page

- **Purpose:** Provide detailed information about a selected product.
- **Key Features:**
 - Product image with zoom functionality.
 - Detailed description, including ingredients, expiry date, weight, and brand.
 - Price and availability (e.g., "In Stock").
 - Quantity selector and "Add to Cart" or "Buy Now" button.
 - Customer reviews and ratings section.
 - Related products carousel.

6. Shopping Cart Page

- **Purpose:** Allow users to view selected items and proceed to checkout.
- **Key Features:**
 - List of selected products with:
 - Product name, image, price, and quantity.
 - Option to increase/decrease quantity or remove items.

- Display of total price, discounts, and delivery charges.
- Button to proceed to checkout.

7. Checkout Page

- **Purpose:** Facilitate seamless order placement.
- **Key Features:**
 - Address selection or input.
 - Delivery options (e.g., same day, next day).
 - Payment methods:
 - Card, UPI, Net Banking, Wallets, and Cash on Delivery (COD).
 - Order summary with item details and total cost.
 - "Place Order" button.

8. Order Confirmation Page

- **Purpose:** Confirm the successful placement of an order.
- **Key Features:**
 - Display of order ID, delivery estimate, and payment summary.
 - Button to view order details or continue shopping.

9. Profile Page

- **Purpose:** Manage user account and preferences.
- **Key Features:**
 - Personal information (editable fields for name, email, phone number).
 - Saved addresses.
 - Order history with details and option to reorder.
 - Wishlist of saved products.
 - Logout button.

10. Admin Dashboard (For Admin Users Only)

- **Purpose:** Allow admins to manage the platform effectively.
- **Key Features:**

- Overview of sales, user registrations, and popular products.
- Manage:
 - Products (add, edit, delete).
 - Orders (update delivery status).
 - Users (view, block, delete).
- Reports:
 - Revenue trends
 - Stock levels
- Notifications for low stock or user issues.

Design Elements

1. Color Scheme:

- Fresh and vibrant colors like green, white, and yellow to emphasize freshness and groceries.
- Accent colors for buttons and highlights.

2. Typography:

- Clean and legible fonts like *Roboto* or *Open Sans*.

3. Icons:

- Use intuitive icons for cart, search, categories, etc.

4. Responsive Design:

- Optimized for both desktop and mobile devices.

Navigation

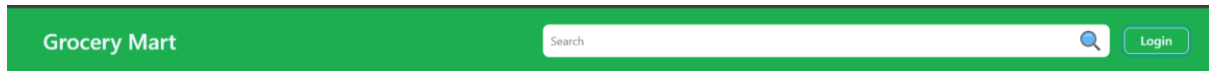
1. Header:

- Logo
- Search bar
- Cart icon with item count
- Profile icon with dropdown options (e.g., Login, Profile, Orders).

2. Footer:

- Links to privacy policy, terms of service, contact info, and social media.

Sample Screenshots:



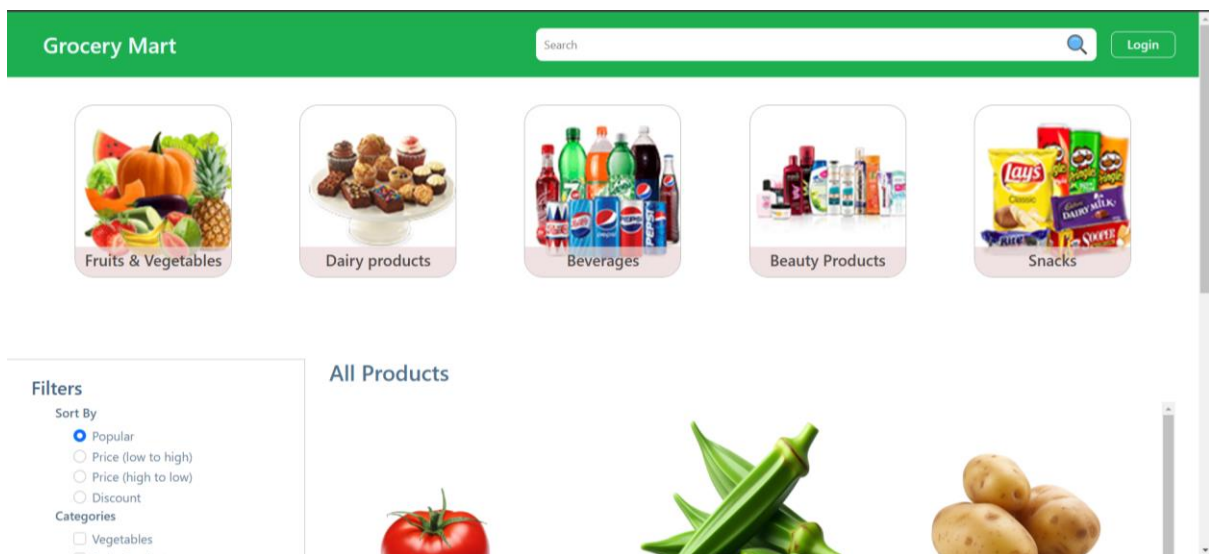
Login

Email address
rakshitha

Password

Sign in

Not registered? [Register](#)




Grocery Mart

Search

Login

← back



Ladies Finger

Vegetable

Choose size

Quantity 1

Price: ₹ 50 ~~₹ 52~~ (3% off)

Rating: 3.4/5

Free delivery in 5 days

Buy now

Add to cart

Grocery Mart

Search Electronics, Fashion, mobiles, etc.,

Naveen

0


Username: Naveen

Email: naveen@gmail.com

Orders: 2

Logout

Orders



Milk


Dairy Products

Size: Quantity: 2 Price: ₹ 38 Payment method: card

Address: coimbatore Pincode: 642206 Ordered on: 2024-11-26

Order status: order placed

Cancel



Paneer

Dairy Products

Size: Quantity: 1 Price: ₹ 49 Payment method: card

Address: coimbatore Pincode: 642206 Ordered on: 2024-11-26

Order status: order placed

Cancel

Grocery Mart

Search

Login

Checkout

Checkout details

Name

Mobile

Email

Address

Pincode

Payment method

Choose Payment method

choose payment method

cancel

Order

Price Details

Total MRP: ₹ 0

Discount on MRP: - ₹ 0

Delivery Charges: + ₹ 0

Final Price: ₹ 0

Place order

10. Testing

Tools:

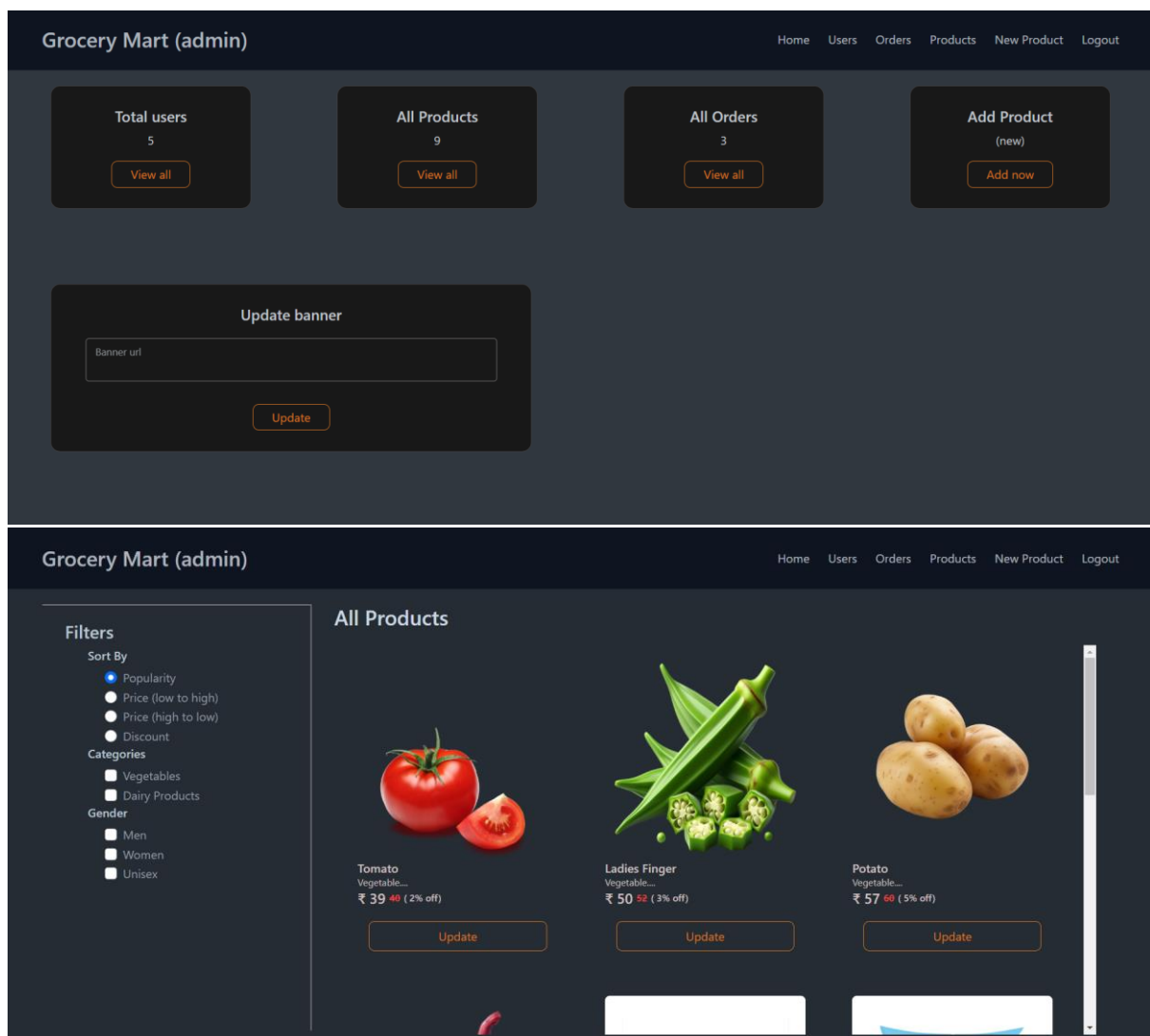
- **Postman:** For API testing.
- **Jest:** To test backend functionality.

Testing Process:

1. Unit tests for APIs.
2. Integration testing between the backend and database.
3. Manual UI testing for responsiveness and performance.

11. Screenshots or Demo


Screenshots:



Grocery Mart (admin)

[Home](#)[Users](#)[Orders](#)[Products](#)[New Product](#)[Logout](#)

Orders



Ladies Finger

Vegetable

Size: Quantity: 5 Price: ₹ 250 Payment method: cod

Userid: 6745807e98b6401705be8843 Name: Nivedha Email: nivedha@gmail.com Mobile: 0808079568


Ordered on: 2024-11-26 Address: chengalpattu Pincode: 607040

Order status: order placed

Update order status

Update

Cancel



Milk

Dairy Products

Size: Quantity: 2 Price: ₹ 38 Payment method: card

Userid: 6745740898b6401705be87e7 Name: Naveen Email: naveen@gmail.com Mobile: 9999900000

Ordered on: 2024-11-26 Address: coimnatore Pincode: 642206

Grocery Mart (admin)

[Home](#)[Users](#)[Orders](#)[Products](#)[New Product](#)[Logout](#)

All Users

User Id	User Name	Email Address	Orders
6745740898b6401705be87e7	Naveen	naveen@gmail.com	2

User Id	User Name	Email Address	Orders
6745807e98b6401705be8843	Nivedha	nivedha@gmail.com	1

Grocery Mart (admin)

[Home](#)[Users](#)[Orders](#)[Products](#)[New Product](#)[Logout](#)

New Product

Product name

Product Description

Thumbnail img url

Add on img1 url

Add on img2 url

Add on img3 url

Available Size

☐ S ☐ M ☐ L ☐ XL

Gender

☐ Men ☐ Women ☐ Unisex

Add product

Demo Link:

Link: https://drive.google.com/drive/folders/1CiKX-ZAeRmRh_RmiE67Dh0iQJg7cc0NA?usp=drive_link

12. Known Issues

1. Functional Issues

- Search bar lacks error tolerance; irrelevant results for typos.
- Cart items may not sync across devices.
- Wishlist removal or cart addition may behave inconsistently.

2. Performance Issues

- Slow product image loading and order history for large datasets.
- Notifications for order updates are delayed.
- Backend response time can lag during high traffic.

3. UI/UX Issues

- Mobile responsiveness issues on smaller screens.
- Pagination/infinite scroll fails intermittently.
- Generic error messages lack helpful details.

4. Security Issues

- Weak password policy without strict requirements.
- Gaps in email verification allow bypassing the process.
- OTP delays during multi-factor authentication.
- Sessions not always terminated after inactivity.

5. Integration Issues

- Occasional payment gateway failures.
- Inconsistent geolocation-based delivery results.
- Supplier APIs return outdated inventory data.

6. Backend Issues

- Scalability challenges for high concurrent users.
- Redundant data in the database increases load.
- Limited logging details hinder debugging.

7. Other Issues

- Stock synchronization delays for "out of stock" products.
- Coupon codes may cause incorrect discount calculations.

13. Future Enhancements

1. Functional Enhancements

- **Advanced Search:** Implement fuzzy search and autocomplete to handle typos and improve query results.
- **Smart Cart:** Enable cross-device cart synchronization and persistent cart items across sessions.
- **Wishlist Optimization:** Ensure seamless wishlist updates with real-time feedback and improved UI.

2. Performance Improvements

- **Optimized Image Loading:** Use lazy loading and image compression for faster page rendering.
- **Efficient Database Queries:** Optimize database structure and implement indexing to reduce query times.
- **Real-Time Notifications:** Enhance websocket handling for instant updates on order status and promotions.

3. UI/UX Upgrades

- **Responsive Design:** Make the platform fully responsive across all screen sizes and devices.
- **Enhanced Error Handling:** Provide detailed and user-friendly error messages for better guidance.
- **Improved Navigation:** Introduce breadcrumbs and intuitive menu structures for easier navigation.

4. Security Enhancements

- **Stronger Password Policy:** Require passwords to include a mix of characters, numbers, and special symbols.
- **Robust MFA:** Improve OTP delivery reliability and add alternative MFA options like app-based authentication.
- **Session Management:** Implement stricter session timeout policies with user notification.

5. Integration Enhancements

- **Reliable Payment Gateways:** Add redundancy by integrating multiple payment gateways with fallback mechanisms.
- **Geolocation Accuracy:** Use enhanced geolocation APIs for precise delivery availability.
- **Supplier API Validation:** Validate and cache supplier inventory data to reduce dependency on real-time API calls.

6. Backend Enhancements

- **Scalability:** Introduce horizontal scaling and load balancing to handle high traffic efficiently.
- **Data Redundancy Elimination:** Normalize the database to reduce duplication and improve performance.
- **Advanced Logging:** Implement a detailed logging system with error categorization for easier debugging.

7. Additional Features

- **Personalized Recommendations:** Use AI to suggest products based on user behavior and preferences.
- **Delivery Tracking:** Add real-time delivery tracking for better user experience.
- **Subscription Plans:** Offer subscription-based options for recurring orders like milk, bread, or vegetables.
- **Gamification:** Introduce reward points for purchases to encourage user retention and loyalty.