Grocery Webapp – Detailed Project Report

1.Introduction:

Project Title: Grocery Webapp : Simplifying Grocery Shopping **Team**

Members:

- Muthu Kumar - Project Lead , Database Engineer
- Madhavan- Frontend Developer
- Abdul Razack A- Backend Developer
- Sethuraman - UI/UX Developer
- Sundaresan – Web Developer

2.Project Overview:

Purpose:

To create an intuitive web application that allows users to browse, search, and purchase groceries online, making grocery shopping efficient and user-friendly.

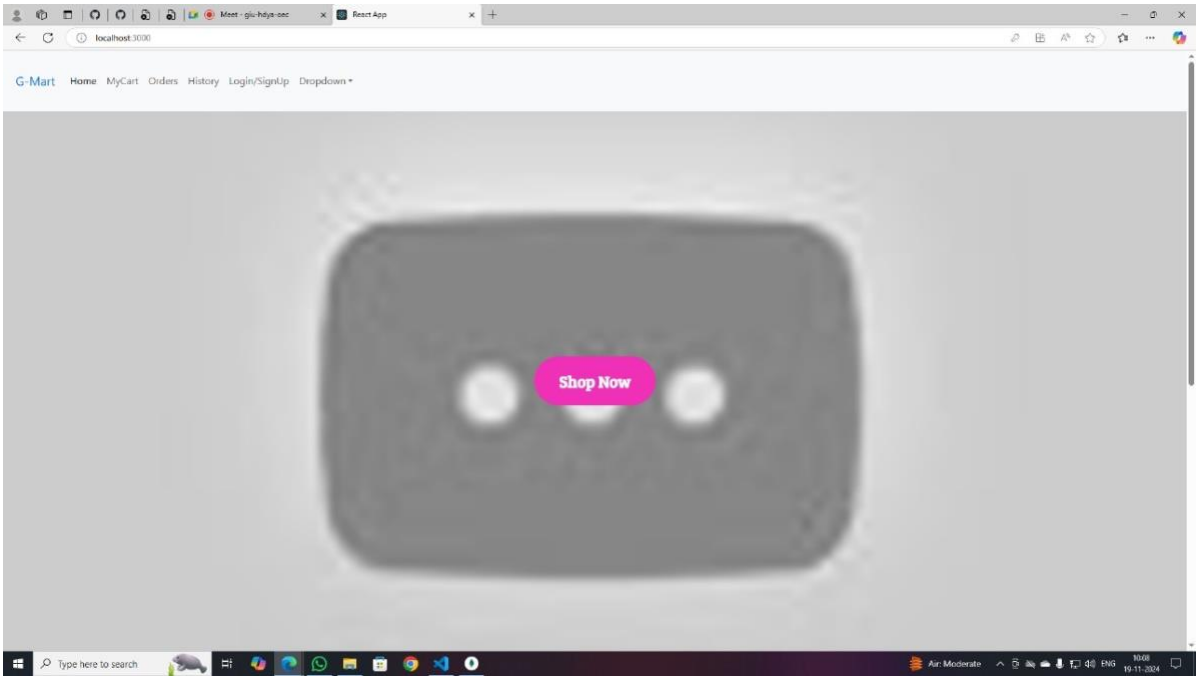
Features:

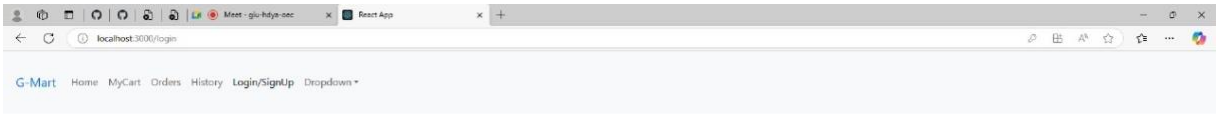
- User authentication and profile management.
- Real-time product search and filtering.
- Cart management with seamless checkout.
- Admin dashboard for managing inventory.

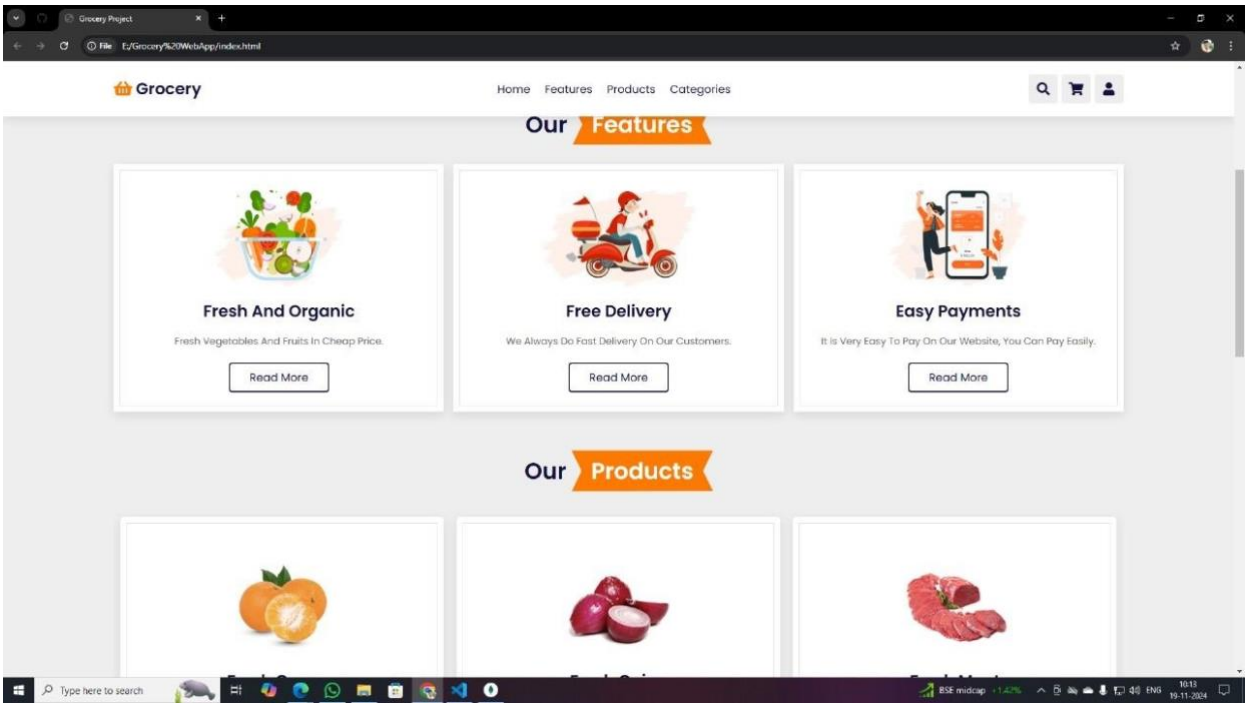
3. Architecture:

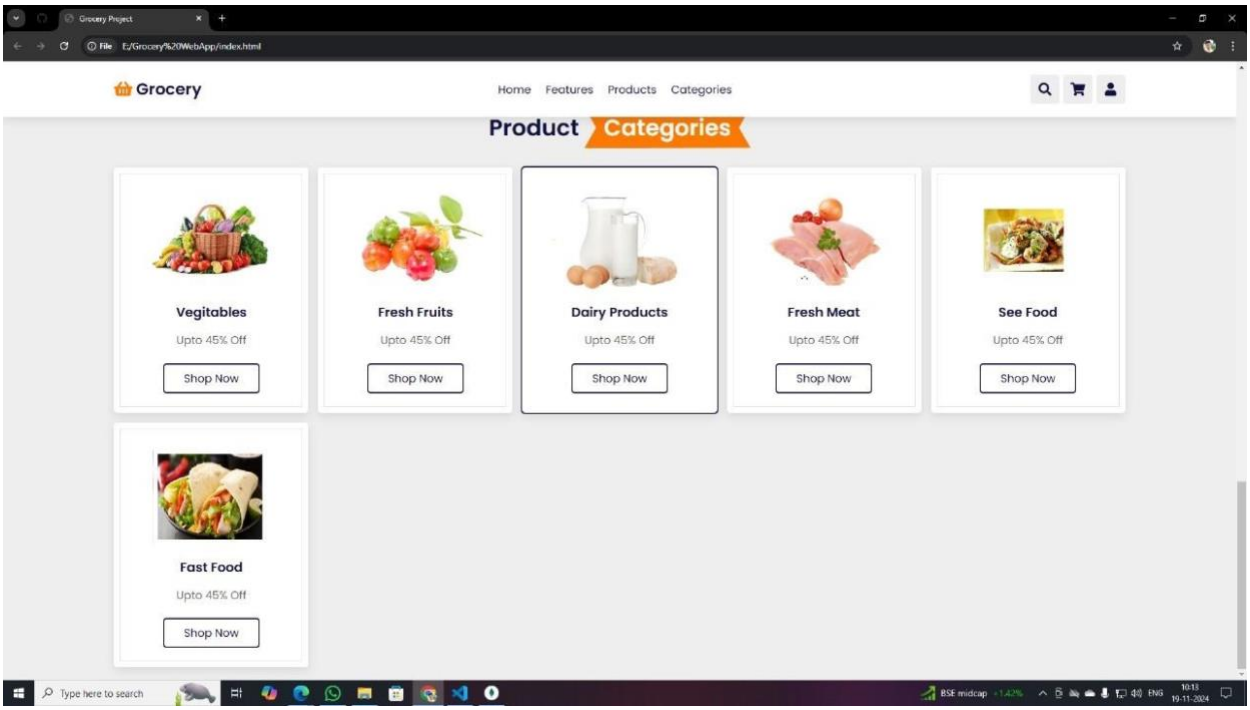
Frontend:

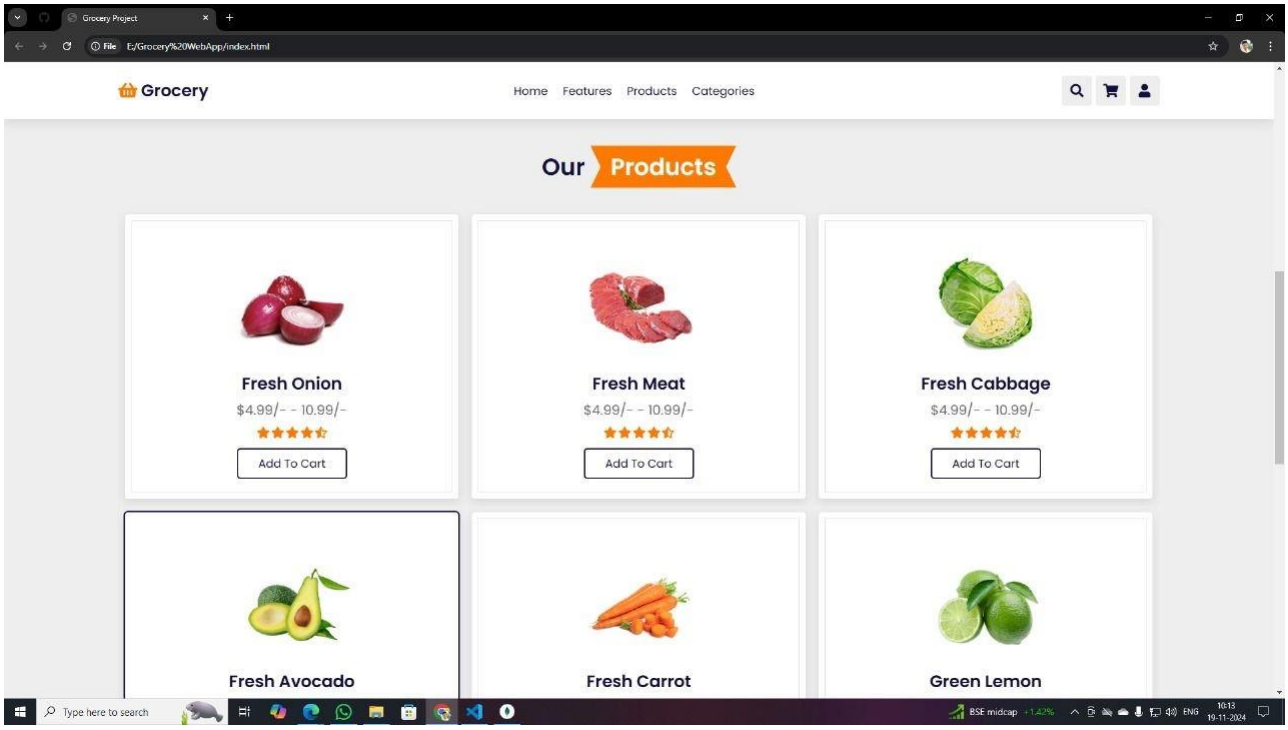
The frontend is built using React with component-based architecture, styled using TailwindCSS for modern and responsive design.






Grocery
Home Features Products Categories
Our Features
Fresh And Organic
Fresh Vegetables And Fruits In Cheap Price.
Read More
Free Delivery
We Always Do Fast Delivery On Our Customers.
Read More
Easy Payments
It Is Very Easy To Pay On Our Website, You Can Pay Easily.
Read More
Our Products

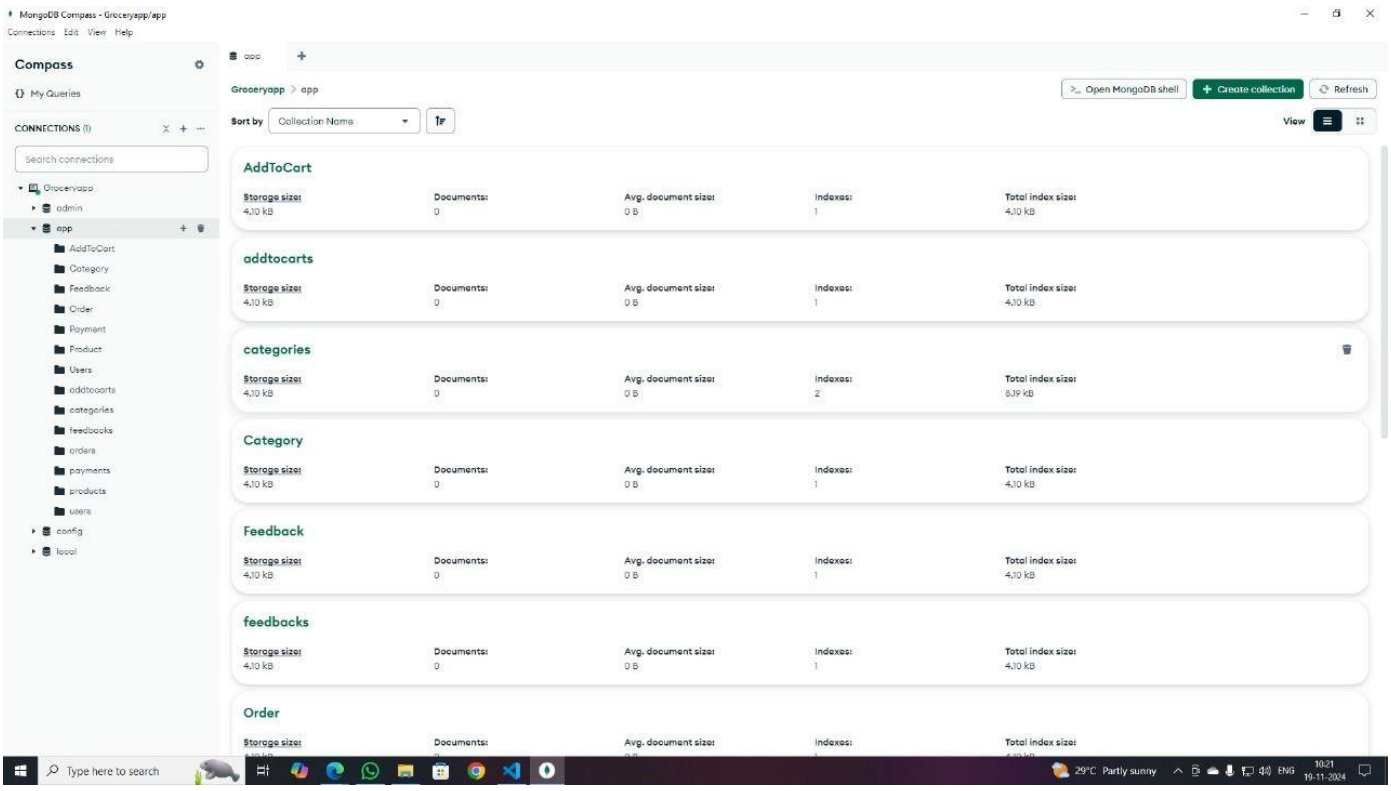

Grocery
Home Features Products Categories
Product Categories
Vegitables
Upto 45% Off
Shop Now
Fresh Fruits
Upto 45% Off
Shop Now
Dairy Products
Upto 45% Off
Shop Now
Fresh Meat
Upto 45% Off
Shop Now
See Food
Upto 45% Off
Shop Now
Fast Food
Upto 45% Off
Shop Now



Backend:

Node.js and Express.js power the backend, managing API endpoints and handling requests.

- Login
- Verification
- Web Page
- Server Connection
- Database Connection
- Close



Database:

MongoDB serves as the database, with collections for users, products, orders, and categories.

4. Setup Instructions

Prerequisites:

- Node.js (v16+)
- MongoDB (local or cloud instance)
- Git

5.Installation:

A. Clone the repository:

git clone

https://github.com/your-repo/groceryweb.git

B. Navigate to directories and install dependencies:

cd client npm install cd ../server npm install

C. Set up environment variables:

- Create .env file in the server directory.
- Add variables like DB_URI, JWT_SECRET, and PORT.

5. Folder Structure

Client:

- src/components/ - React components.
- src/pages/ - Page-level components.
- src/services/ - API interaction logic.

Server:

- routes/ - API routes.
- controllers/ - Business logic.
- models/ - Mongoose schemas.

6.Running the Application:

Frontend:

cd client

npm start

Backend: cd

server npm

start

7.API Documentation:

- GET /api/products: Fetch all products.
- POST /api/auth/login: User login. • POST /api/orders: Place an order.

Example:

Request:

POST /api/auth/login

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Response: json

```
{
  "token": "jwt-token",
  "user": { "name": "John Doe" }
}
```

8.Authentication:

- JWT-based authentication is implemented.
- Tokens are issued at login and verified for secured endpoints.
- Middleware ensures only authorized users access restricted resources.

9.User Interface:

The user interface includes:

- A home page showcasing featured products.

- A responsive product search and filter page.
- A secure checkout flow.

10.Testing:

Tools Used: Jest and Cypress.

- Unit tests for critical functions.
- End-to-end tests for checkout and user registration flows.

11.Screenshots or Demo:

Screenshots:

- Login Page

- Product Listing Page

- Checkout Flow

Demo:[Live Demo Link]

12.Known Issues:

- Minor UI glitches on older browsers.
- Payment gateway sandbox mode occasionally times out.

13.Future Enhancements:

- Implement AI-driven product recommendations.
- Add support for multiple languages.
- Introduce subscription-based discounts.