



# **Project Title:** Online Complaints

## **Team Members:**

**Team Leader : Paladugu Veera Raghavaiah**

**Team member : Kudeti Vyshnavi**

**Team member : Medikonduru Sri Sai Bhavana Keerthi**

**Team member : Vejendla Neeraj**



# Project Overview

## Purpose:

---

Overall, digital grocery stores aim to provide a seamless, efficient, and personalized shopping experience for customers, while also offering benefits such as increased convenience, time-saving, and cost-effectiveness.

---

## Features:

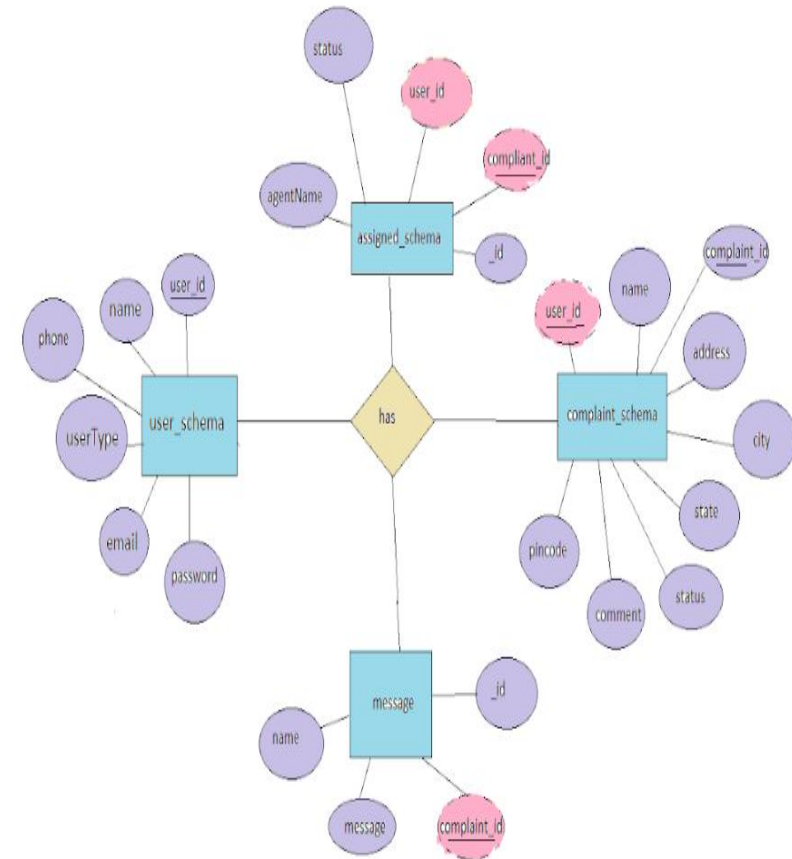
---

- User-Friendly Interface
- Personalized Shopping Experience
- Real-Time Inventory Management
- Flexible Delivery and Pickup Options
- Mobile-Friendly Design

# ER DIAGRAM

This is the er diagram of the project which shows the relationship between user and agent

It shows how user which have required fields can raise a complaint by fillings required fields. It illustrates how these entities relate to each other, helping us understand the underlying database structure and the flow of information within the app. He / She can also communicate with the agent with chat window which follows the message schema which uses userId and complaintId from other schemas



## PRE-REQUISITES

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js:

### Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

Download: <https://nodejs.org/en/download/>

Installation instructions: <https://nodejs.org/en/download/package-manager/>

### Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

Installation: Open your command prompt or terminal and run the following command:

```
npm install express
```

### MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

## React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: <https://reactjs.org/docs/create-a-new-react-app.html>

**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. To Connect the Database with Node JS go through the below provided link:

<https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/>

**Front-end Framework:** Utilize Reactjs to build the user-facing part of the application, including entering complaints, status of the complaints, and user interfaces for the admin dashboard.

For making better UI we have also used some libraries like material UI and bootstrap.

**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.

- Visual Studio Code: Download from <https://code.visualstudio.com/download>

To run the existing Video Conference App project downloaded from GitHub:

Follow below steps:

Clone the Repository:

Open your terminal or command prompt.

Navigate to the directory where you want to store the e-commerce app.

Execute the following command to clone the repository:

**git clone:** <https://github.com/awdhesh-student/complaint-registery.git>

Install Dependencies:

- Navigate into the cloned repository directory:

```
cd complaint-registery
```

- Install the required dependencies by running the following commands:

```
cd frontend
```

```
npm install
```

```
cd ../backend
```

```
npm install
```

Start the Development Server:

- The online complaint registration and management app will be accessible at <http://localhost:3000>

## PROJECT STRUCTURE:

### APPLICATION FLOW:

#### Online Complaint Registration and Management System

##### Customer/Ordinary User:

**Role:** Create and manage complaints, interact with agents, and manage profile information.

**Flow:**

##### Registration and Login:

Create an account by providing necessary information such as email and password.

Log in using the registered credentials.

##### Complaint Submission:

Fill out the complaint form with details of the issue, including description, contact information, and relevant attachments.

Submit the complaint for processing.

##### Status Tracking:

View the status of submitted complaints in the dashboard or status section.

Receive real-time updates on the progress of complaints.

##### Interaction with Agents:

Connect with assigned agents directly using the built-in messaging feature.

Discuss complaints further and provide additional information or clarification.

##### Profile Management:

Manage personal profile information, including details and addresses.

##### Agent:

**Role:** Manage complaints assigned by the admin, communicate with customers, and update complaint statuses.

**Flow:**

### **Registration and Login:**

- Create an account using email and password.
- Log in using the registered credentials.

### **Complaint Management:**

- Access the dashboard to view and manage complaints assigned by the admin.
- Communicate with customers regarding their complaints through the chat window.

### **Status Update:**

- Change the status of complaints based on resolution or progress.
- Provide updates to customers regarding the status of their complaints.

### **Customer Interaction:**

- Respond to inquiries, resolve issues, and address feedback from customers.

**Admin:Role:** Oversee the overall operation of the complaint registration platform, manage complaints, users, and agents, and enforce

### **Management and Monitoring:**

- Monitor and moderate all complaints submitted by users.
- Assign complaints to agents based on workload and expertise.

### **Complaint Assignment:**

- Assign complaints to the desired agents for resolution.
- Ensure timely and efficient handling of complaints.

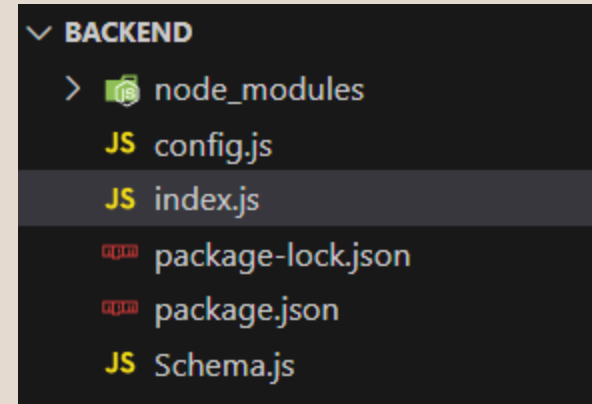
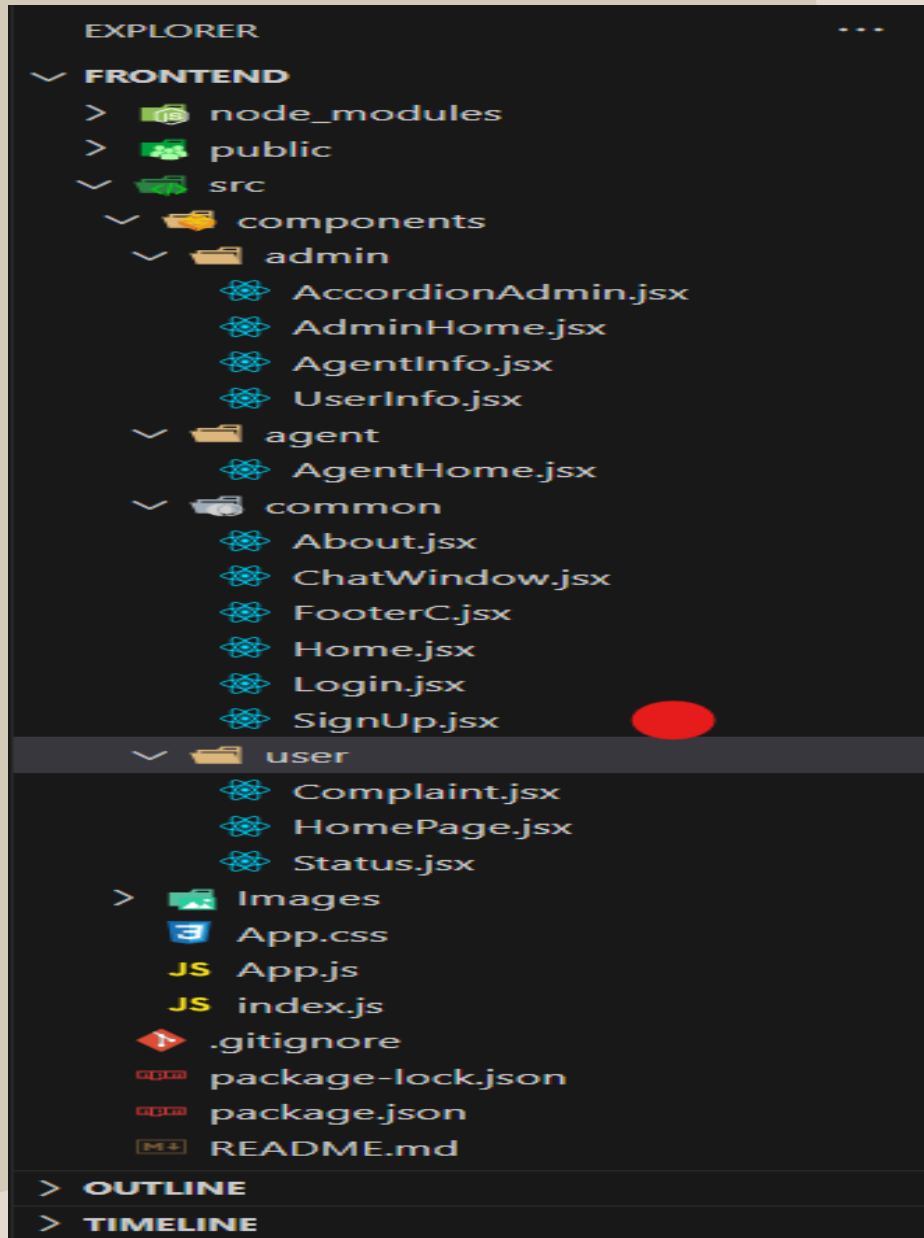
### **User and Agent Management:**

- Manage user and agent accounts, including registration, login, and profile information.
- Enforce platform policies, terms of service, and privacy regulations.

### **Continuous Improvement:**

- Implement measures to improve the platform's functionality, user experience, and security measures.
- Address any issues or concerns raised by users or agents for better service delivery.





The first image is of frontend part which is showing all the files and folders that have been used in UI development

The second image is of Backend part which is showing all the files and folders that have been used in backend development

# Project Setup and Configuration

## 1. Create project folders and files:

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

- Client folders.
- Server folders

2.

## Install required tools and software:

For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries includes

- Node.js.
- MongoDB.
- Bcrypt
- Body-parser

Also, for the frontend we use the libraries such as

- React Js.
- Material UI
- Bootstrap
- Axios

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.

```

{
  "name": "task1",
  "version": "0.1.0",
  "proxy": "http://localhost:8000",
  "private": true,
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@testing-library/jest-dom": "^5.16.5",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.4.0",
    "bootstrap": "^5.2.3",
    "mdb-react-ui-kit": "^6.1.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.7.4",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.11.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}

```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

```

1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "nodemon index.js"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "bcrypt": "^5.1.0",
14     "cors": "^2.8.5",
15     "express": "^4.18.2",
16     "express-session": "^1.17.3",
17     "mongoose": "^7.1.1",
18     "nodemon": "^2.0.22"
19   }
20 }

```

# Backend Development

## •Set Up Project Structure:

- Create a new directory for your project and set up a package.json file using npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

## •Set Up Project Structure:

- Create a new directory for your project and set up a package.json file using npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

## •Create Express.js Server:

- Set up an Express.js server to handle HTTP requests and serve API endpoints.
- Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

## Define API Routes:

- Create separate route files for different API functionalities such as authentication, stock actions, and transactions.
- 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 Bank, users, transactions, deposits and loans.
- 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:

- Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

•

## Handle new transactions:

- Allow users to make transactions to other users using the user's account id.
- Update the transactions and account balance dynamically in real-time.

•

## Admin Functionality:

- Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.

•

## 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.

# Database Development

## 1. User Schema:

- The user schema defines the structure of user data stored in the database. It includes fields such as name, email, password, phone, and userType.
- Each user must provide a name, email, password, phone number, and userType (e.g., customer, agent, admin).
- User data is stored in the "user\_Schema" collection in the MongoDB database.

## 2. Complaint Schema:

- The complaint schema specifies the format of complaint data registered by users.
- It contains fields like userId, name, address, city, state, pincode, comment, and status.
- Complaints are associated with users through the userId field, and each complaint must have a name, address, city, state, pincode, comment, and status.
- Complaint data is stored in the "complaint\_schema" collection in the MongoDB database.

## 3. Assigned Complaint Schema:

- The assigned complaint schema defines how complaints are assigned to agents for resolution.
- It includes fields such as agentId, complaintId, status, and agentName.
- Each assigned complaint is linked to a specific agent (identified by agentId) and complaint (identified by complaintId).

- The status field indicates the current status of the assigned complaint.
- Assigned complaint data is stored in the "assigned\_complaint" collection in the MongoDB database.

### **1.Chat Window Schema:**

- The chat window schema governs the structure of messages exchanged between users and agents regarding specific complaints.
- It comprises fields like name, message, and complaintId.
- Messages are associated with a complaint through the complaintId field, allowing for easy tracking and retrieval of chat history for each complaint.
- Message data is stored in the "message" collection in the MongoDB database

# Frontend Development

## 1.Setup React Application:

Bringing Customer Care Registry to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To start the development process for the frontend, follow the below steps.

- Install required libraries.
- Create the structure directories.

## 2.Design UI components:

Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface. Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of Customer Care Registry, like making specific complaints or managing your Product complaints.

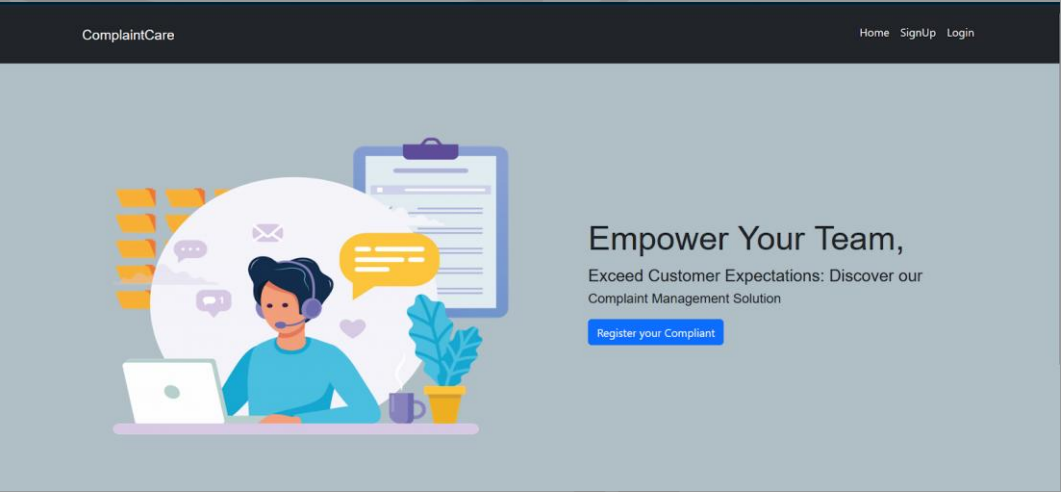
## 3.Implement frontend logic:

In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

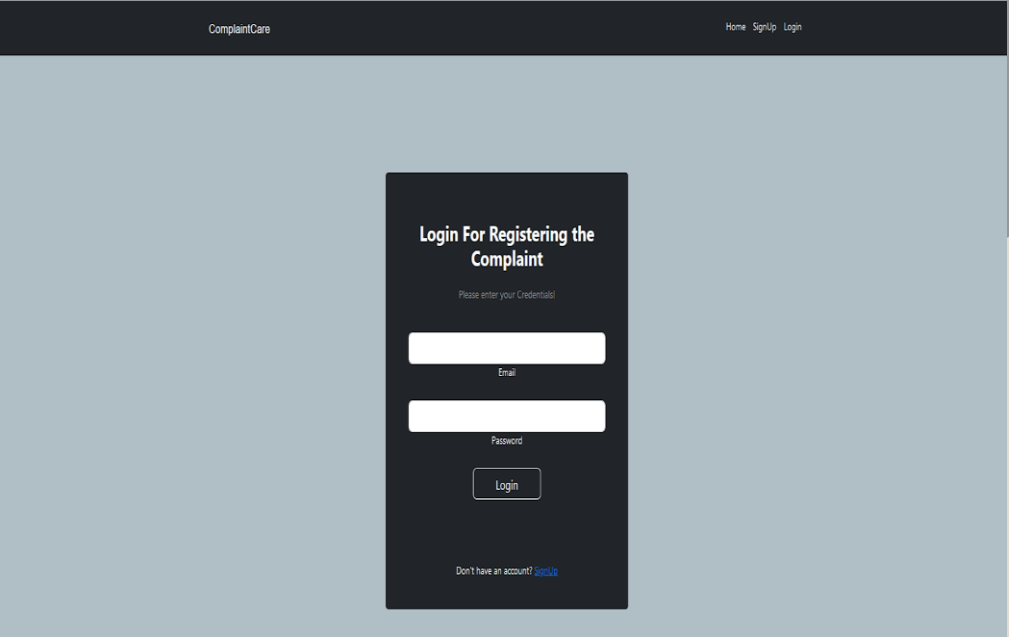
- Integration with API endpoints.
- Implement data binding.



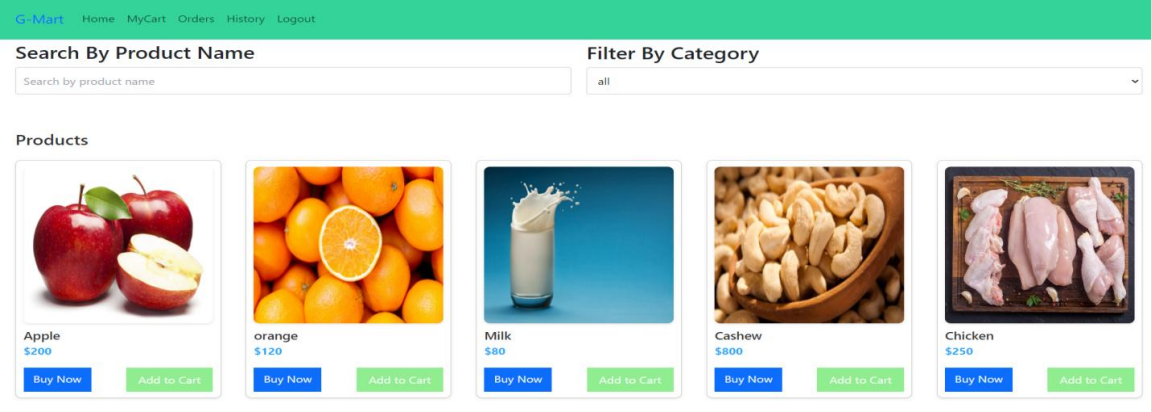
•Landing Page



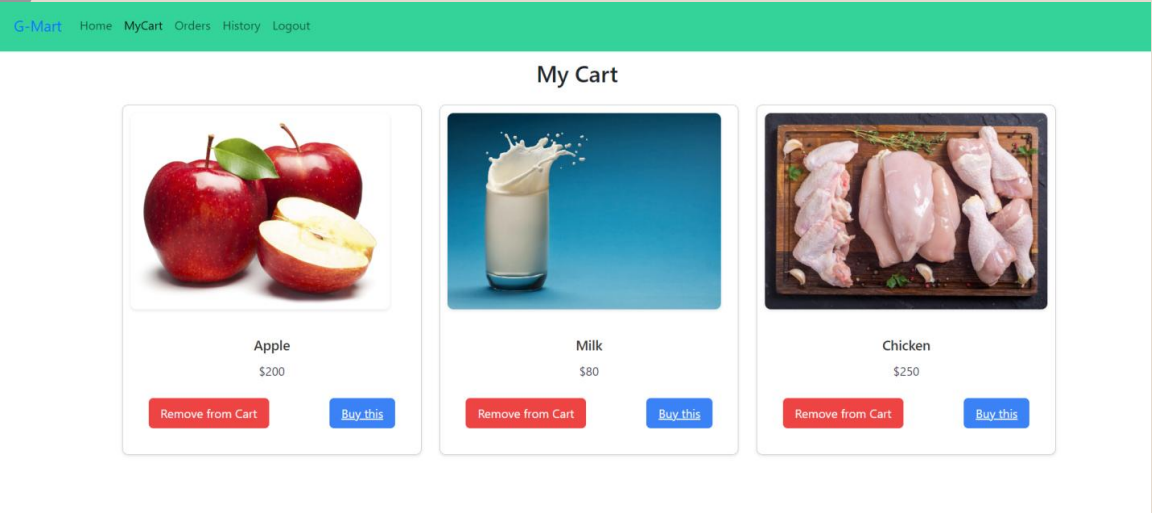
•Login Page



Items Page:



My Cart:-



# My History Page:

G-Mart

Home

MyCart

Orders

History

Logout

My Orders

Order ID: 6614b085c30b51d3c700f20a

Name: syed arshad

Phone: 9505221870

Date: 2024-04-09T03:05:41.563Z

Price: 400

Status: Pending

Payment Method: credit

# Place Order Page:-

G-Mart

Home

MyCart

Orders

History

Logout

My History

Order ID: 6614a8ddc30b51d3c700f1b4

Name: syed arshad

Phone: 9505221870

Date: 2024-04-09T02:36:47.498Z

Price: 400

Status: Delivered

Payment Method: debit

# Users Page:-

G-Mart

Home

MyCart

Orders

History

Logout

Order Details

First Name:

Enter your first name

Last Name:

Enter your last name

Phone:

Enter your phone number

Quantity:

Enter the quantity

Address:

Enter your address

Payment Method:

Cash on Delivery (COD)

Submit

# Add Product page:-

Grocery Web App

Dashboard

Users

Products

Add product

Orders

Logout

Add Product

Product Name

Enter product name

Rating

Enter product rating

Price

Enter product price

Image URL

Enter image URL

Category

Select Category

Count in Stock

Enter count in stock

Description

Enter product description

Add Product

# Add Product

Product Name

Enter product name

Rating

Enter product rating

Price

Enter product price

Image URL

Enter image URL

Category

Select Category

Count in Stock

Enter count in stock

Description

Enter product description

Add Product

Admin Orders Page:

# Orders

**Order ID:** 6614a8ddc30b51d3c700f1b4  
**Fullname:** syed arshad  
**Phone:** 9505221870  
**Product ID:** 660e84641c659d0b2b10e135  
**Quantity:** 2  
**Total price:** 400  
**Payment Method:** debit  
**Address:** hyderabad  
**Created At:** 2024-04-09T02:36:47.498Z  
**Status:** Delivered

Delivered

**Order ID:** 6614b085c30b51d3c700f20a  
**Fullname:** syed arshad  
**Phone:** 9505221870  
**Product ID:** 660e84641c659d0b2b10e135  
**Quantity:** 2  
**Total price:** 400



Thank You