A PROJECT REPORT

ON

SHOPSMART:

Your Digital Grocery Store Experience

Team ID: LTVIP2025TMID59696

Team Members:

Gali Keerthi (Backend Developer)
Danaboina Nandini (Frontend developer)
Dadinaboyina Pavani Tejaswi (Backend Developer)
Dachepalli Nikhitha (Database Designer)

Under The Guidance of

GANESH M

INTRODUCTION:

ShopSmart is a modern and intelligent grocery shopping web application built to bring convenience to customers and efficiency to shop owners. It acts as a bridge between users who want to order groceries online and shopkeepers who manage product listings and customer orders digitally.

In today's busy life, people prefer online services to save time and effort. ShopSmart understands that need. With just a few clicks, users can browse products, add them to their cart, and place orders from the comfort of their home. Shop owners can easily handle product uploads, update prices, and manage orders using a smart and responsive dashboard.

Why Choose ShopSmart?

ShopSmart is built for real-world grocery needs — whether you're a busy customer or a shopkeeper managing many orders. It replaces manual processes with a smooth, digital system. Everything from product management to customer service is streamlined.

No more long queues, lost inventory, or complicated billing. With ShopSmart, everything happens online, quickly and securely.

Technology Stack

- **Frontend**: Built using **Angular**, which ensures a fast and responsive user interface for both desktop and mobile devices.
- **Backend**: Developed using **Node.js** and **Express.js**, providing powerful API support, fast processing, and smooth user authentication.
- **Database**: Uses **MongoDB**, a flexible and high-performance NoSQL database, combined with **Mongoose** for easy schema design.
- **Authentication**: Secured with **JWT (JSON Web Tokens)** and **bcrypt**, which encrypts passwords and manages user sessions safely.

Project Overview: ShopSmart

Purpose and Goals:-

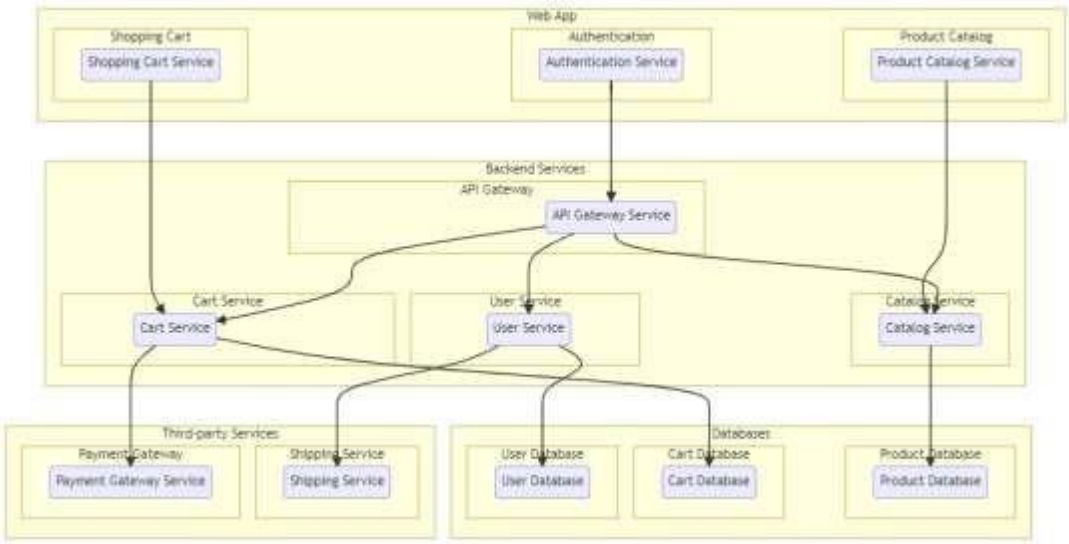
The purpose of ShopSmart is to revolutionize the grocery shopping experience by offering a seamless, user-friendly digital platform that benefits both customers and shop owners. The application is designed to simplify day-to-day shopping, reduce manual workload for vendors, and provide a modern solution to manage groceries online.

Feature Highlights: Key Functionalities

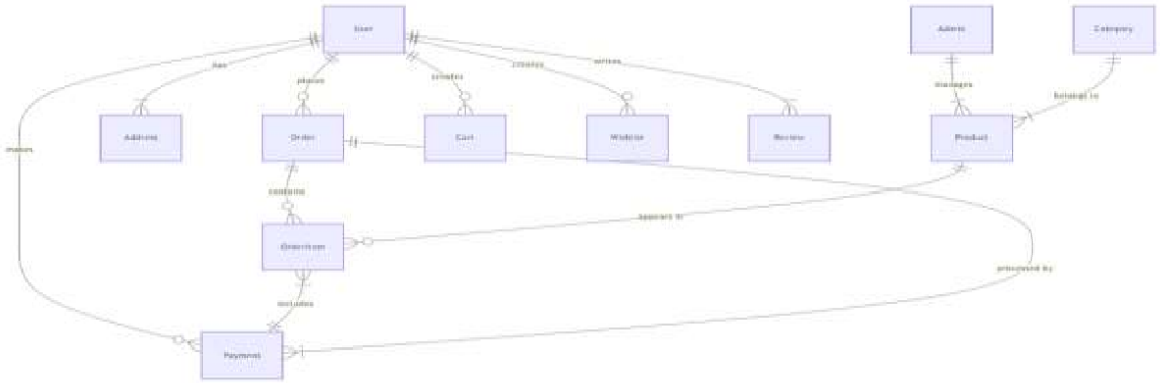
Feature	Description
User Authentication	Secure login and signup using JWT and bcrypt for encrypted user data.
Product Management	Shopkeepers can add, edit, and delete grocery products via easy-to-use forms.
Cart & Order System	Customers can add products to cart and place orders conveniently.
Admin Dashboard	Admins/shop owners manage inventory, orders, and view user activity.
Email Notifications	Confirmation emails sent for registrations and successful orders.
Responsive UI	Angular-powered frontend works smoothly on both mobile and desktop devices.
MongoDB Integration	Fast, flexible data handling with Mongoose schema modeling.
Real-Time Updates	Users see latest changes in stock, prices, and order status instantly.

ARCHITECTURE:-

Front end and Backend



Entity diagram:



Setup Instructions:-

Prerequisites

To develop the ShopSmart full-stack grocery web application using Angular, Node.js, and MongoDB, the following prerequisites are essential:

Node.js & npm

- Required for running server-side code and managing packages.
- Download Node.js

MongoDB

- Used as the backend database for storing users, products, and orders.
- Download MongoDB

Angular CLI

- Tool for generating and managing Angular projects.
- Install via terminal:

```
npm install -g @angular/cli
```

Mongoose

- ODM for MongoDB, used for schema and model handling.

```
npm install mongoose
```

Git & GitHub

- For version control and code collaboration.
- Download Git

Code Editor (IDE)

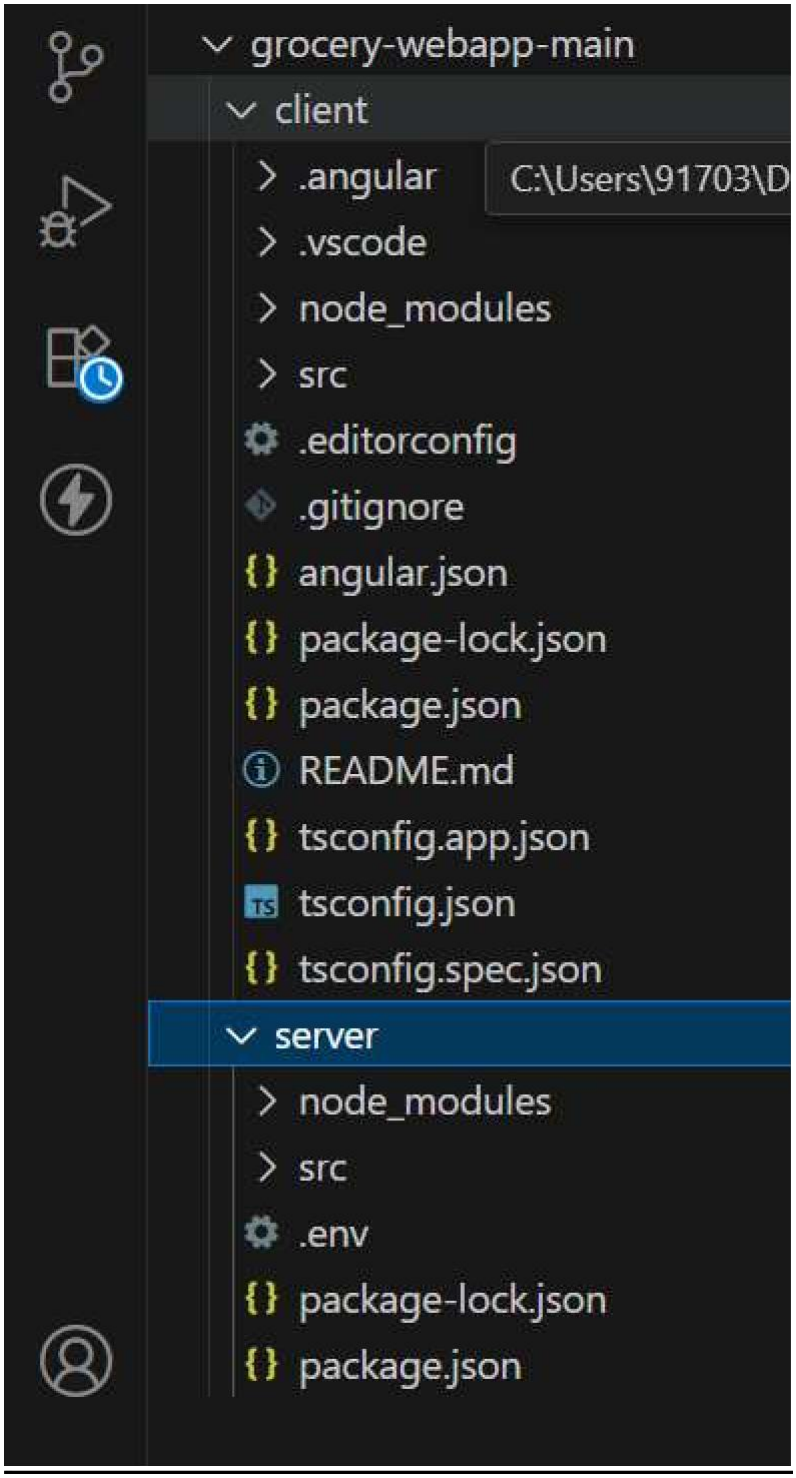
- Recommended: Visual Studio Code

INSTALLATIONS:-

1. Install Angular CLI globally:
   npm install -g @angular/cli
2. Install Nodemon globally (for auto-restart server):
   npm install -g nodemon
3. Create a new Angular project:
   ng new client
4. Go into the project folder:
   cd client
5. Install Angular project dependencies:
   npm install
6. Install Bootstrap for frontend styling:
   npm install bootstrap
7. Initialize a new Node.js backend project:
   npm init -y
8. Install backend core dependencies (Express, MongoDB, etc.):
   npm install express mongoose cors dotenv
9. Install authentication tools (bcrypt & JWT):
   npm install bcrypt jsonwebtoken
10. Install Nodemailer for email support:
    npm install nodemailer
11. Install Nodemon as a dev dependency:
    npm install --save-dev nodemon

FOLDER STRUCTURE:-

Client and server Architecture:-



Running the Application

To run the Grocery App locally, both frontend and backend servers must be started separately.

FRONT END:-

- cd client
- npm install
- npm start

This will start the Angular development server at:
http://localhost:4200

BACKEND:-

- cd server
- npm install
- npm run dev

This will run the backend API server at:
http://localhost:5000

Environment Configuration (.env)

Make sure to configure the .env file in the server directory:

- MONGO_URI=your_mongodb_connection_string
- JWT_SECRET=your_jwt_secret_key

API DOCUMENTATION:-

API Endpoint	Method	Description
/auth/register	POST	Registers a new user
/auth/login	POST	Logs in user and returns JWT token
/products	GET	Retrieves all available products
/products/add	POST	Adds a new product
/products/:id	DELETE	Deletes a product using its ID

API Endpoint	Method	Description
/orders/place	POST	Places an order with product list
/orders/user/:userId	GET	Gets all orders of a specific user

Authentication in ShopSmart

The **ShopSmart** application uses secure authentication and authorization techniques to protect user data and control access to different parts of the system.

1. Authentication Mechanism

- The app uses **JWT (JSON Web Tokens)** for authenticating users.
- When a user logs in, the backend generates a token and sends it to the frontend.
- The frontend stores the token securely (usually in localStorage).
- This token is then sent with every protected API request as a header to verify the user.

2. Authorization Control

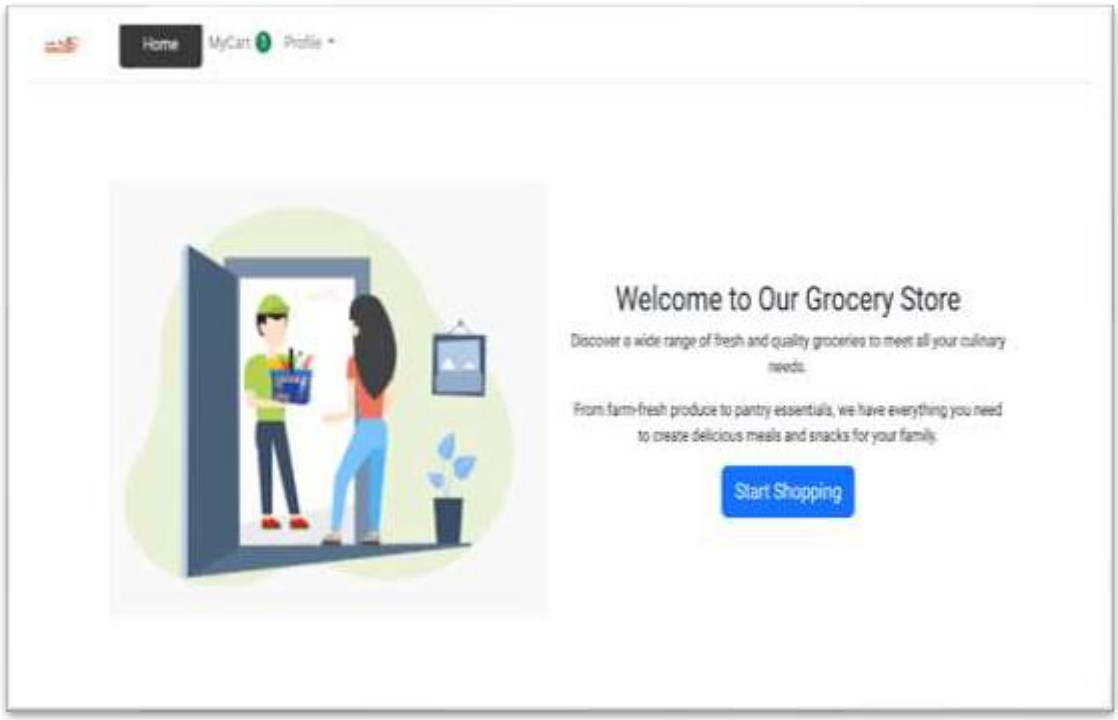
- Only logged-in users with a valid token can access protected features like placing orders or viewing account details.
- Shop owners or admins can access additional routes like product management or viewing all orders.
- The backend checks the role inside the JWT before allowing access.

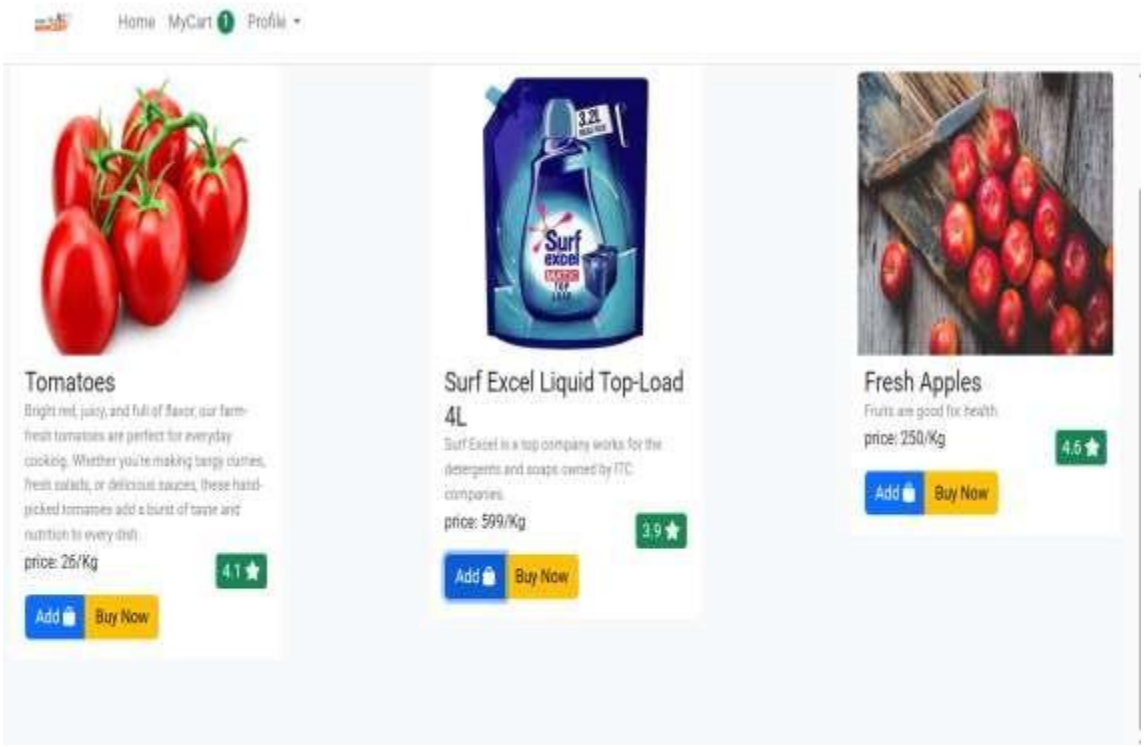
3. Password Security

- User passwords are hashed using **bcrypt** before being stored in the database.
- Even if the database is compromised, passwords remain unreadable and safe.

This approach ensures that only authorized users can access specific features, making the application secure for both customers and shop owners.

USER INTERFACE:-





Testing – ShopSmart Grocery App

The ShopSmart application was tested thoroughly to ensure functionality, security, and user experience across both frontend and backend components. A manual testing approach was mainly used during development.

Functional Testing

- Each feature was tested individually to verify it works as expected.
- Examples include:
  - User registration and login
  - Product creation, editing, and deletion
  - Cart operations and order placement
  - Admin dashboard visibility

Tools Used

- Postman was used to test backend API endpoints such as login, product CRUD, and order placement.
- Browser Developer Tools were used to inspect frontend elements and debug responsiveness, CSS issues, and console errors.

Frontend Testing

- All UI elements were checked for proper display on desktop and mobile views.
- Forms were tested with both valid and invalid inputs to ensure validation messages appear correctly.
-

Backend Testing

- APIs were tested with valid and invalid data.
- JWT-based authentication was verified for secure access.
- Role-based access control was checked to ensure restricted routes are protected.

Summary

- The app functions correctly under normal conditions.
- Manual testing covered all critical paths like login, cart, orders, and admin features.

DEMO LINK:-

Good [morning/afternoon], I'm going to present a demo of our project titled ShopSmart – Your Digital Grocery Store Experience.
This is a full-stack web application designed to simplify online grocery shopping for both customers and shopkeepers.
In this demo, I will walk you through the main features of the application, including:

- User registration and login
- Browsing available grocery products
- Adding items to the cart and placing an order
- Admin functionalities like managing products and viewing orders

The frontend of our app is built using Angular, and the backend is developed using Node.js and Express.js. We've used MongoDB as the database and implemented JWT-based authentication to ensure security.
Now, let me take you through the working of our application."

DEMO LINK:-

https://drive.google.com/file/d/14rl2jFOIX6F277vQZTB_EDSLyqmZATbY/view?usp=sharing

Known Issues – ShopSmart
Despite working well overall, the current version of ShopSmart has a few limitations:

1. **Limited Form Validation**
   Some forms lack proper client-side or server-side input validation messages.
2. **Static Admin Role**
   Admin access is hardcoded, and there is no dynamic role management system.
3. **No Payment Integration**
   Users can place orders, but online payment functionality is not yet available.
4. **No Email for Admin**
   While users receive confirmation emails, the admin does not get notified when an order is placed.
5. **Not Optimized for Offline Use**
   The app requires an active internet connection and is not a PWA (Progressive Web App) yet.

Future Enhancements – ShopSmart

We plan to add the following features in future versions to improve the app's performance, scalability, and user experience:

1. **Online Payment Integration**
   Add Razorpay or Stripe to allow secure online payments during checkout.
2. **Dynamic Role-Based Access Control**
   Allow dynamic assignment of roles (admin/user) with proper permissions.
3. **Order Tracking & Status Updates**
   Users will be able to track their orders and get status updates in real-time.
4. **Progressive Web App (PWA)**
   Convert the app into a PWA to enable offline support and installable web experience.
5. **Multi-Language Support**
   Add support for different languages to cater to a wider audience.
6. **Enhanced UI/UX**
   Improve responsiveness, animations, and overall interface polish.
7. **Admin Notifications via Email or SMS**
   Notify admins automatically when a new order is placed.