Project Report Format

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. IDEATION PHASE

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

3.2 Solution Requirement

3.3 Data Flow Diagram

3.4 Technology Stack

4. PROJECT DESIGN

4.1 Problem Solution Fit

4.2 Proposed Solution

4.3 Solution Architecture

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

7. RESULTS

7.1 Output Screenshots

8. ADVANTAGES & DISADVANTAGES

9. CONCLUSION

10. FUTURE SCOPE

11. APPENDIX

Source Code(if any)

Dataset Link

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview:

ShopSmart: Your Digital Grocery Store Experience is a full-stack web application developed using the MERN stack, which includes React.js for the frontend, Node.js and Express.js for the backend, and MongoDB with Mongoose for the database. The application is designed to provide a seamless and user-friendly online grocery shopping experience. It allows customers to browse products across different categories, view detailed product information, add items to a shopping cart, and securely complete the checkout process. The system also includes an administrative interface where admins can manage products, monitor orders, and oversee user accounts. The application follows a structured architecture with clear separation between frontend, backend, and database layers, ensuring maintainability, scalability, and security. Role-based access control is implemented to differentiate between user and admin functionalities, and authentication is handled securely using encrypted passwords and token-based authorization.

1.2 Purpose:

The purpose of the ShopSmart project is to develop a user-friendly digital grocery shopping platform that allows customers to browse products, manage their cart, and place orders securely. The system also provides administrative control for managing products, users, and orders efficiently. Additionally, the project aims to demonstrate practical implementation of full-stack development using the MERN stack with secure authentication and structured architecture.

Key objectives of the system include:

- To design and develop a responsive and intuitive grocery web application using React.js.
- To implement secure user authentication and role-based access control using JWT.
- To create RESTful APIs using Node.js and Express.js for handling business logic.
- To design and manage MongoDB database schemas using Mongoose.
- To implement complete product management, cart management, and order processing features.
- To provide an admin dashboard for managing products, users, and orders.
- To ensure data security, proper error handling, and clean project structure.

2.IDEATION PHASE

2.1 Problem Statement:

In today's fast-paced lifestyle, many customers find it difficult to visit physical grocery stores due to time constraints and inconvenience. Traditional shopping methods lack efficient digital management for product browsing, order tracking, and secure transactions. Small and medium grocery businesses also face challenges in managing inventory, customer data, and order processing effectively. Therefore, there is a need for a secure, user-friendly, and efficient digital grocery platform that simplifies online shopping while providing proper administrative control and system management.

Problem Statement -1:

I am	**a grocery customer**
I'm trying to	buy daily grocery items online quickly and conveniently
But	I cannot easily find all products in one place with clear stock availability
Because	many local stores don't provide a proper digital platform with real-time updates
Which makes me feel	frustrated and time-wasted

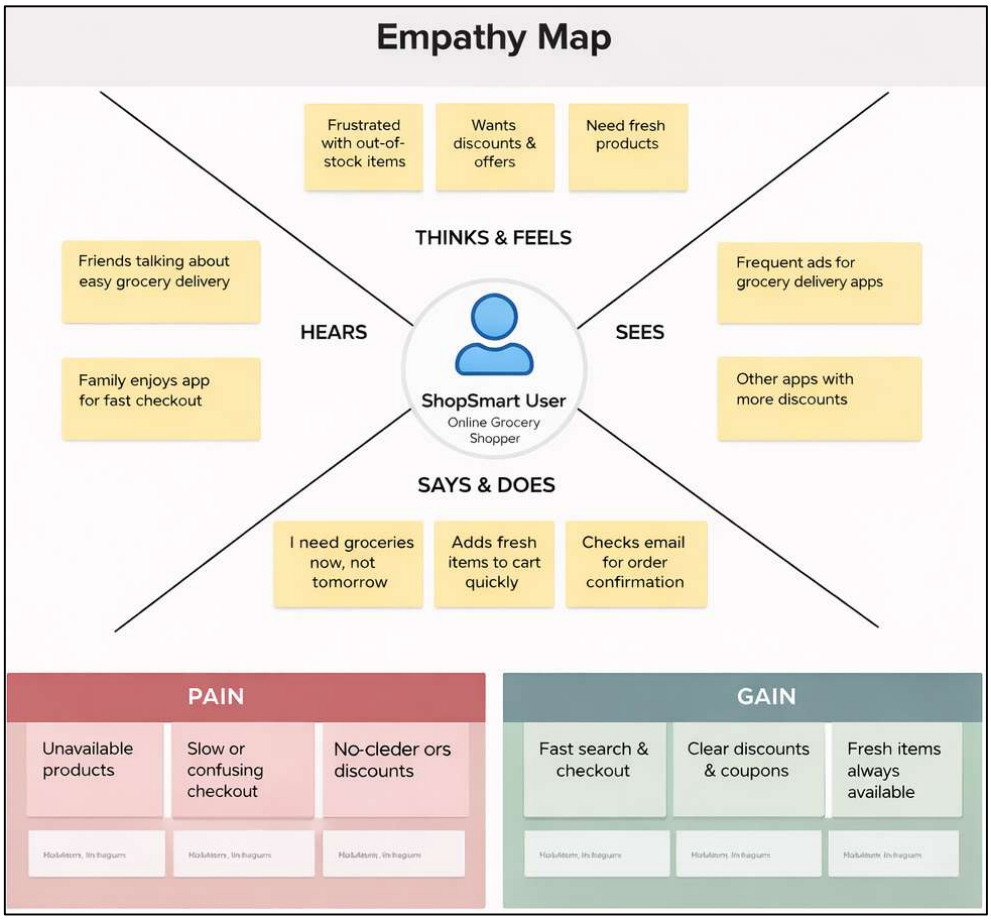
Problem Statement -2:

I am	**a store administrator**
I'm trying to	manage products, inventory, and customer orders efficiently
But	I struggle to track stock levels and customer bookings manually
Because	there is no centralized online management system
Which makes me feel	overwhelmed and stressed

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	A customer	Find quality grocery products at good prices	It is hard to compare products and availability	Many stores don't provide clear stock and price details	Confused and uncertain
PS-2	An admin	Monitor customer orders and sales	Tracking orders across different platforms is confusing	Data is not organized in one place	Stressed

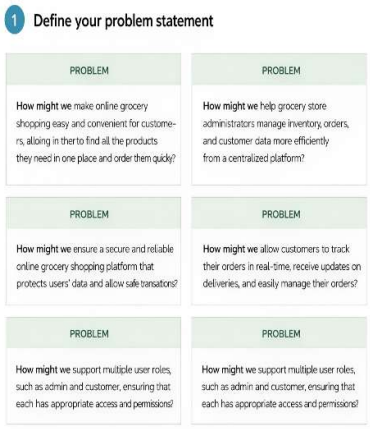
2.2 Empathy Map:

User: - Naveen Lukalapu (A working professional buying groceries online.)

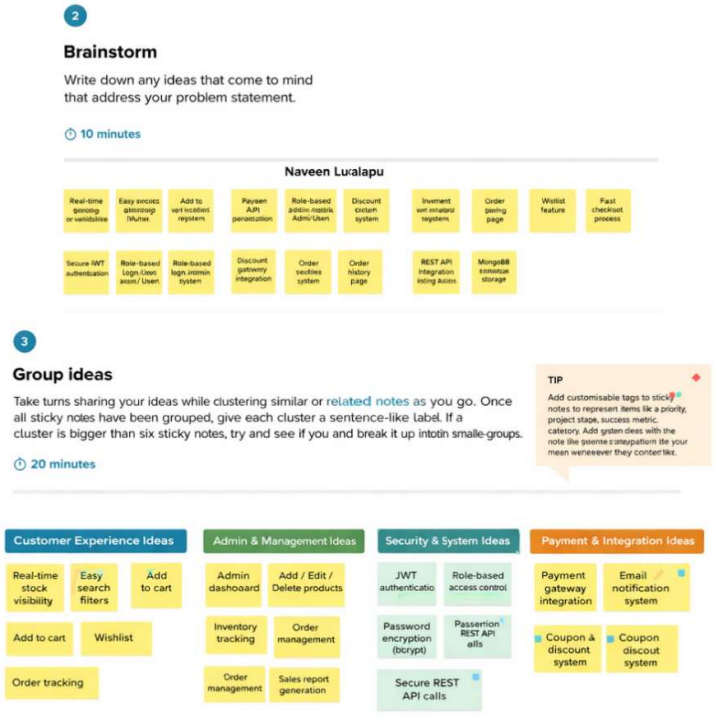


2.2 Brainstorm & Idea Prioritization:-

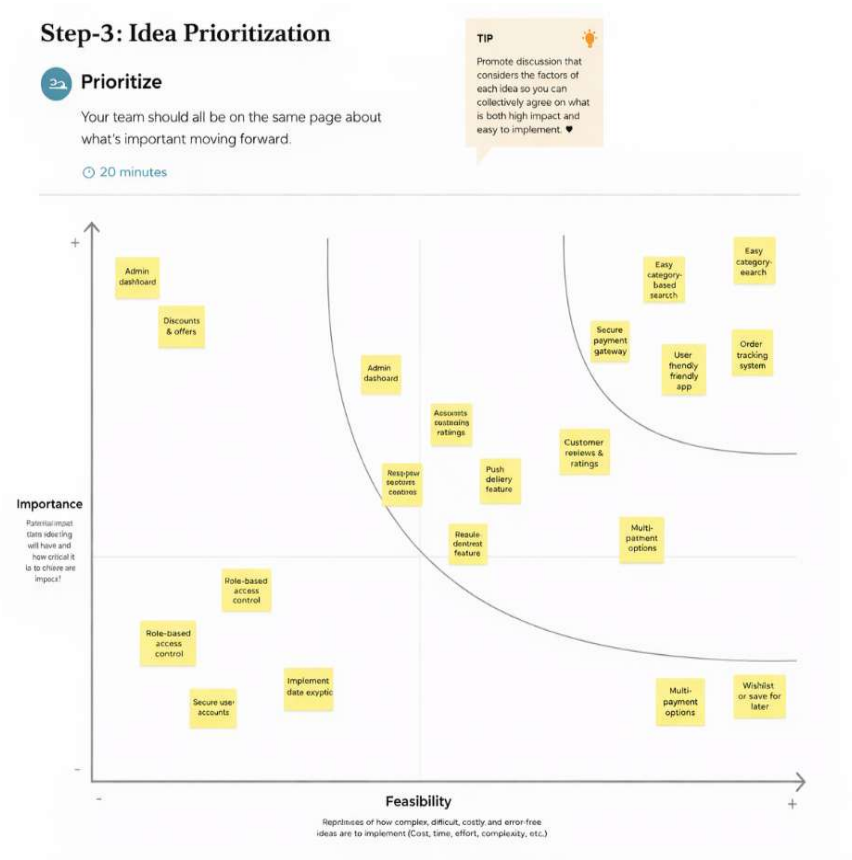
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

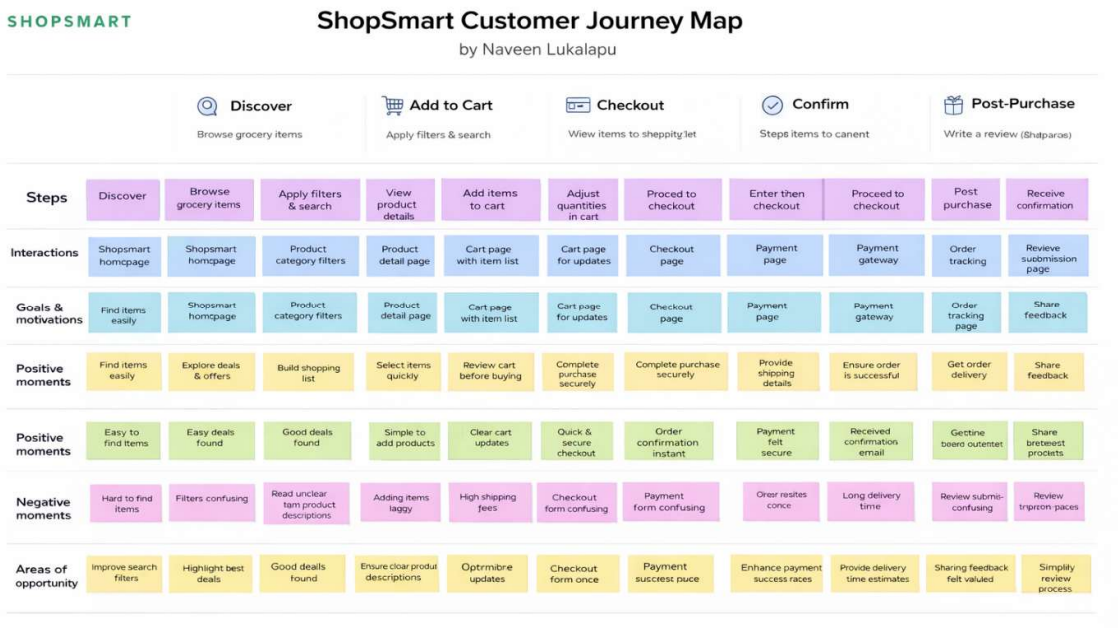


Step-3: Idea Prioritization



3. REQUIREMENT ANALYSIS

3.1 Customer Journey map: -



3.2 Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form (Name, Email, Password) Password Encryption (bcrypt) Validation of Email & Password
FR-2	User Login & Authentication	Login using Email & Password JWT Token Generation Role-based Login (Admin/User)

FR-3	Admin Login	Predefined Admin Credentials (Backend Only) Admin Authentication via JWT
FR-4	Product Management (Admin)	Add New Product Edit Product Details Delete Product Manage Stock Quantity
FR-5	Product Browsing (User)	View Product List Search Products Filter by Category View Product Details
FR-6	Cart Management	Add Item to Cart Update Quantity Remove Item View Cart Summary
FR-7	Order Management	Place Order Calculate Total Amount Reduce Product Stock After Order Cancel Order View Order History
FR-8	Payment Handling	Select Payment Method (COD / Online) Payment Status Update
FR-9	Review & Rating	Add Product Review View Reviews
FR-10	Admin Dashboard	View All Users View All Orders Generate Sales Report

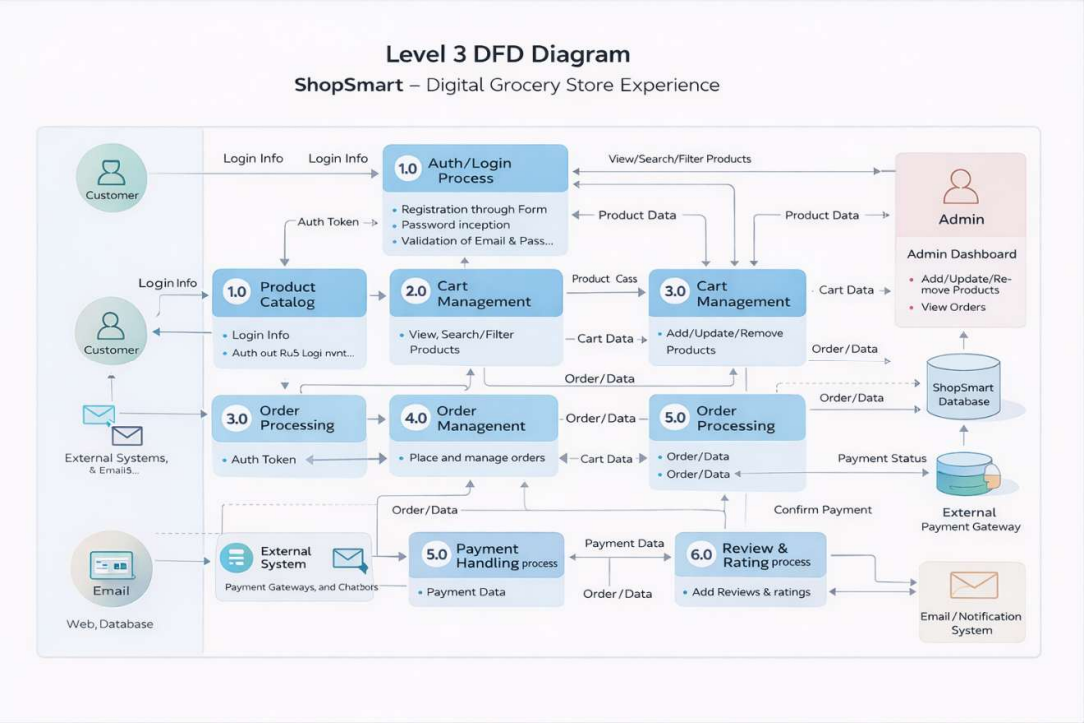
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application must have a clean, responsive UI for desktop and mobile devices.
NFR-2	**Security**	Passwords must be encrypted; JWT authentication must secure protected routes.
NFR-3	**Reliability**	System should correctly process orders without data loss.
NFR-4	**Performance**	API response time should be less than 2 seconds under normal load.

NFR-5	**Availability**	Application should be accessible 24/7 with minimal downtime.
NFR-6	**Scalability**	System should handle increasing number of users and products efficiently.

3.3 Data Flow Diagram: -



3.4 Technology Stack: -

Technical Architecture: -

The ShopSmart application follows a 3-Tier Client–Server Architecture:

- **Presentation Layer (Frontend):** React.js provides an interactive and responsive web interface where users can browse products, manage their cart, place orders, and track purchases, while admins can manage products and monitor orders.

- **Application Layer (Backend):** Node.js and Express.js handle REST API requests, user authentication, role-based access control, product management, cart operations, and order processing logic.

- **Data Layer (Database):** MongoDB, integrated using Mongoose, stores user accounts, product details, cart data, and order records in structured collections.

The frontend communicates with backend REST APIs using Axios, and JWT-based authentication secures protected routes for both users and admins. Passwords are encrypted using bcrypt to ensure secure credential storage.

Table-1: Components & Technologies:

S. No	Component	Description	Technology
1.	User Interface	Web application where users and admin interact (Product browsing, cart, dashboard)	React.js, HTML5, CSS3, JavaScript
2.	Application Logic-1	Authentication & Authorization (Login, Register, JWT validation)	Node.js, Express.js, JWT, bcrypt
3.	Application Logic-2	Product Catalog Management (CRUD operations for products)	Express.js, REST APIs, Node.js
4.	Application Logic-3	Cart & Order Processing Logic	Node.js , Express.js,
5.	Database	Stores Users, Products, Cart, Orders, Reviews	MongoDB, Mongoose
6.	Cloud Database	Database hosting in cloud	MongoDB Atlas
7.	File Storage	Product images storage	Cloudinary / Local File System
8.	External API-1	Payment Gateway Integration	Stripe API / Razorpay API
9.	External API-2	Email Notification Service	Nodemailer / SendGrid
10.	Machine Learning Model	Product Recommendation (future enhancement)	Recommendation Algorithm
11.	Infrastructure (Server / Cloud)	Application deployment	Local Server (Development), Render / Vercel / AWS (Production)

Table-2: Application Characteristics:

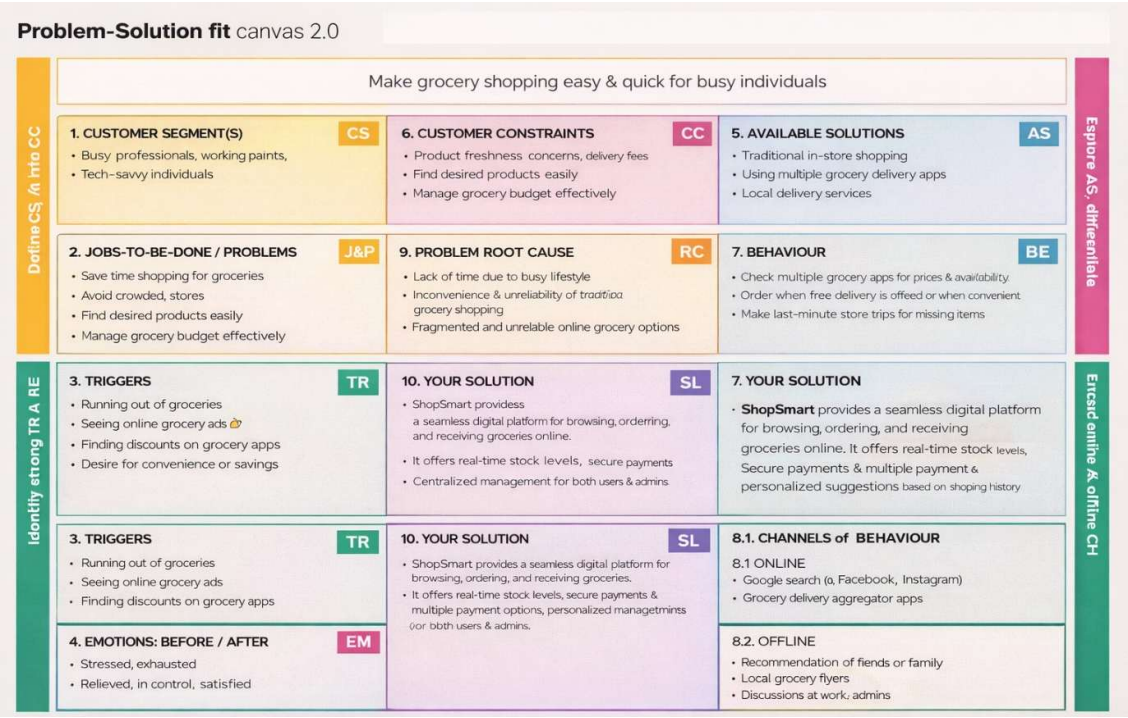
S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Frontend and Backend frameworks used	React.js, Node.js, Express.js

S. No	Characteristics	Description	Technology
2.	Security Implementations	Password hashing, JWT authentication, role-based access control	bcrypt, JWT, CORS, Helmet
3.	Scalable Architecture	3-Tier Architecture (Frontend – Backend – Database)	REST Architecture, MongoDB
4.	Availability	Hosted on cloud with uptime reliability	MongoDB Atlas, Cloud Hosting
5.	Performance	Fast API response and optimized queries	Express Middleware, Indexed MongoDB

4. PROJECT DESIGN

4.1 Problem Solution Fit : -

ShopSmart is designed to simplify grocery shopping for busy individuals by providing a seamless digital experience. It addresses the inefficiencies of traditional grocery shopping and fragmented online platforms by offering real-time product availability, secure payments, and centralized management for both users and admins.



4.2 Proposed Solution : -

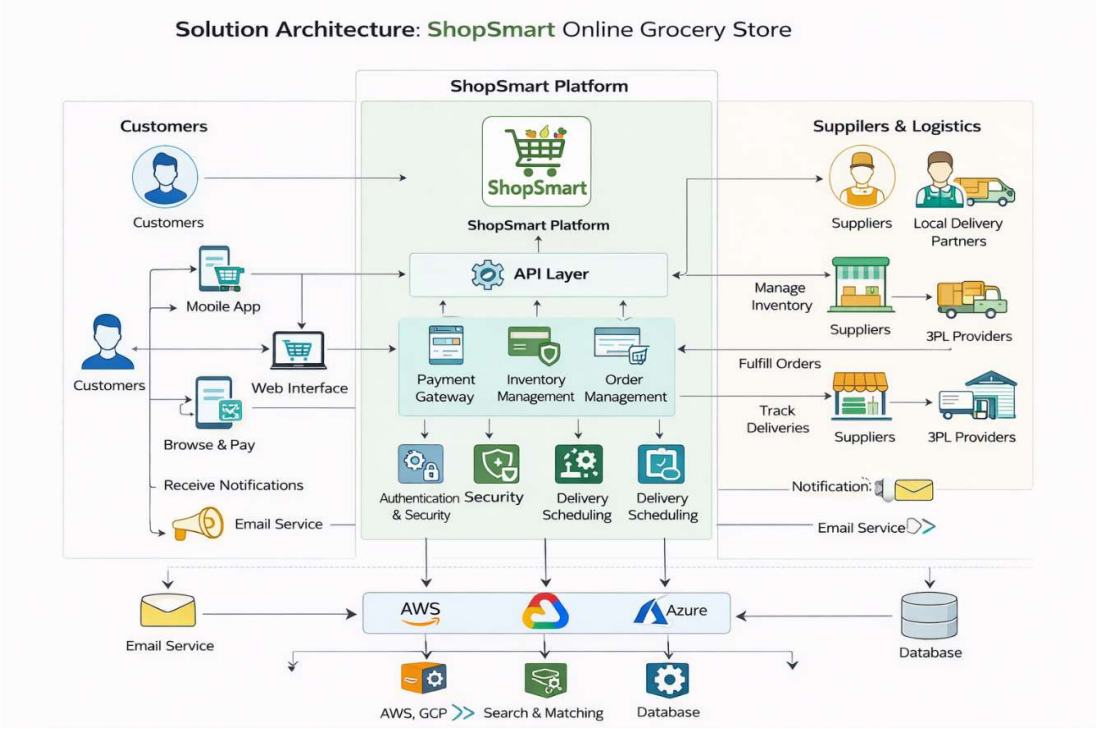
S.No	Parameter	Description
1	**Problem Statement (Problem to be solved)**	Busy individuals and working families face difficulty in purchasing groceries due to lack of time, long queues, limited product availability visibility, and fragmented online grocery services. There is no centralized, reliable, and user-friendly platform that ensures real-time stock updates, secure payments, and smooth order management.
2	**Idea / Solution Description**	ShopSmart is a full-stack digital grocery web application built using MERN stack (React, Node, Express, MongoDB). It allows users to browse products, add items to cart, place orders, and track deliveries seamlessly. It provides real-time stock visibility, secure authentication (JWT-based), role-based access (Admin & User), centralized product management, and smooth checkout process.
3	**Novelty / Uniqueness**	• Role-based access system (Admin controlled backend management) • Real-time inventory updates • Centralized product & order management • Secure JWT authentication • Clean UI with responsive design • Scalable backend architecture using REST APIs
4	**Social Impact / Customer Satisfaction**	• Saves time for working professionals and families • Reduces crowding in physical stores • Enables convenient shopping from home • Ensures secure transactions and data privacy • Provides personalized and smooth user experience
5	**Business Model (Revenue Model)**	• Commission on product sales • Delivery service charges • Featured product promotions for sellers • Subscription model for premium delivery benefits • Advertisement placements for grocery brands
6	**Scalability of the Solution**	• Built on scalable MERN stack architecture • MongoDB supports large-scale data handling

		• Can be deployed on cloud platforms (AWS, Azure, etc.) • Supports horizontal scaling and microservice expansion • Can be expanded to mobile application in future

4.3 Solution Architecture : -

Solution architecture defines the overall structure of the ShopSmart grocery web application and explains how different components interact with each other. It bridges the gap between business requirements and technical implementation by organizing the frontend, backend, database, and external services in a structured manner.

The architecture ensures secure communication between users and the system, manages product and order processing efficiently, and supports scalability and performance. It also defines how authentication, APIs, database operations, and third-party integrations (like payment gateways) work together to deliver a seamless grocery shopping experience.



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning : -

Product Backlog & Sprint Planning (ShopSmart)

Sprint	**Functional Requirement (Epic)**	**User Story No**	**User Story / Task**	**Story Points**	**Priority**
Sprint-1	Authentication	US-1	User can register with email & password	3	High
Sprint-1	Authentication	US-2	User can login securely	2	High
Sprint-1	Admin	US-3	Admin login with predefined credentials	2	High
Sprint-1	UI	US-4	Create homepage & navigation	3	Medium
Sprint-1	UI	US-5	Display product list	5	High
Sprint-1 Total				**15**	
Sprint-2	Functional Requirement	User Story No	User Story	**Story Points**	Priority
Sprint-2	Product	US-6	Admin add product	5	High
Sprint-2	Product	US-7	Admin edit/delete product	5	High
Sprint-2	Product	US-8	Product details page	4	Medium
Sprint-2	Search	US-9	Product search/filter	6	Medium

Sprint-2 Total				**20**	
Sprint-3	Functional Requirement	User Story No	User Story	**Story Points**	Priority
Sprint-3	Cart	US-10	Add to cart	5	High
Sprint-3	Cart	US-11	Update/remove cart items	5	High
Sprint-3	Order	US-12	Checkout process	6	High
Sprint-3	Order	US-13	Order placement	4	High
Sprint-3 Total				**20**	
Sprint-4	Functional Requirement	User Story No	User Story	**Story Points**	Priority
Sprint-4	Orders	US-14	Order history	5	Medium
Sprint-4	Orders	US-15	Admin view all orders	5	High
Sprint-4	Payment	US-16	Payment integration	6	Medium
Sprint-4	Email	US-17	Email confirmation	4	Medium
Sprint-4 Total				**20**	

Sprint Tracker (Velocity Table)

Sprint	**Total Story Points**	**Duration**	**Start**	**End (Planned)**	**Completed**	**Actual**
Sprint-1	20	7 days	Day 1	Day 7	20	Day 7
Sprint-2	20	7 days	Day 8	Day 14	18	Day 14
Sprint-3	20	7 days	Day 15	Day 21	16	Day 21
Sprint-4	20	7 days	Day 22	Day 28	14	Day 28

Velocity Calculation

Total completed story points = 20 + 18 + 16 + 14 = **68**

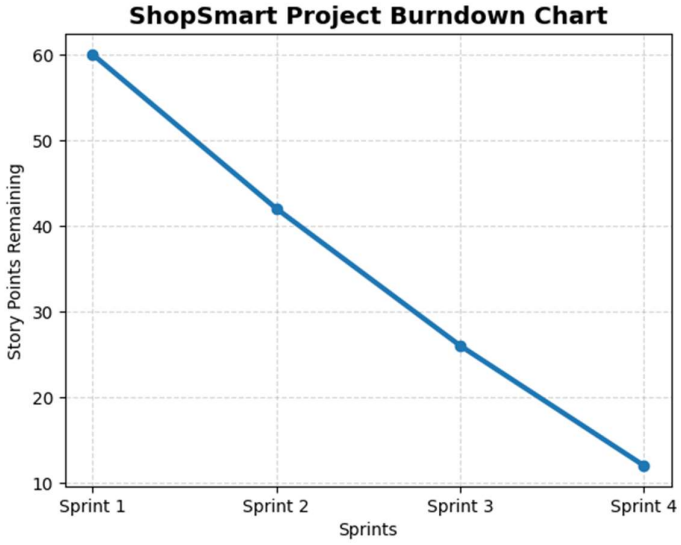
Number of sprints = 4

Average Velocity = 68 / 4 = 17 story points per sprint

If sprint duration = 7 days:

Velocity per day = 17 / 7 ≈ 2.4 points/day

Burndown Chart :



6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Test Scenarios & Results

Test Case ID	Scenario (What to test)	Test Steps (How to test)	Expected Result	Actual Result	Pass/Fail
FT-01	User Registration	Enter valid & invalid user details	Valid registration succeeds; errors for invalid input	Registration works correctly	Pass
FT-02	User Login	Enter correct & incorrect credentials	Login success for valid; error for invalid	Authentication works	Pass
FT-03	Admin Login	Login with predefined admin credentials	Admin dashboard opens	Admin access granted	Pass
FT-04	Product Listing	Load product page	Products display from DB	Products shown correctly	Pass
FT-05	Add Product (Admin)	Add new product from admin panel	Product stored in DB & visible	Product added	Pass
FT-06	Edit/Delete Product	Modify or remove product	DB updates & UI reflects	Update successful	Pass

FT-07	Add to Cart	Click add-to-cart button	Item added to user cart	Cart updated	Pass
FT-08	Update Cart	Change quantity/remove item	Cart recalculates total	Cart updates	Pass
FT-09	Checkout Process	Place order with cart items	Order stored & cart cleared	Order placed	Pass
FT-10	Order History	View past orders	User orders displayed	Orders shown	Pass
FT-11	Payment Integration	Simulate payment selection	Payment status stored	Payment recorded	Pass

Performance Testing

Test Case ID	**Scenario**	**Test Steps**	**Expected Result**	**Actual Result**	**Pass/Fail**
PT-01	Page Load Time	Load homepage/products	< 2 seconds	~1.5 sec	Pass
PT-02	API Response	Fetch products API	Fast response	Stable	Pass
PT-03	Concurrent Users	Multiple users add cart	No crash	Stable	Pass
PT-04	DB Query Speed	Search products	Quick retrieval	Fast	Pass
PT-05	Order Processing Load	Multiple orders placed	Orders saved correctly	Stable	Pass

7. RESULTS

7.1 Output Screenshots

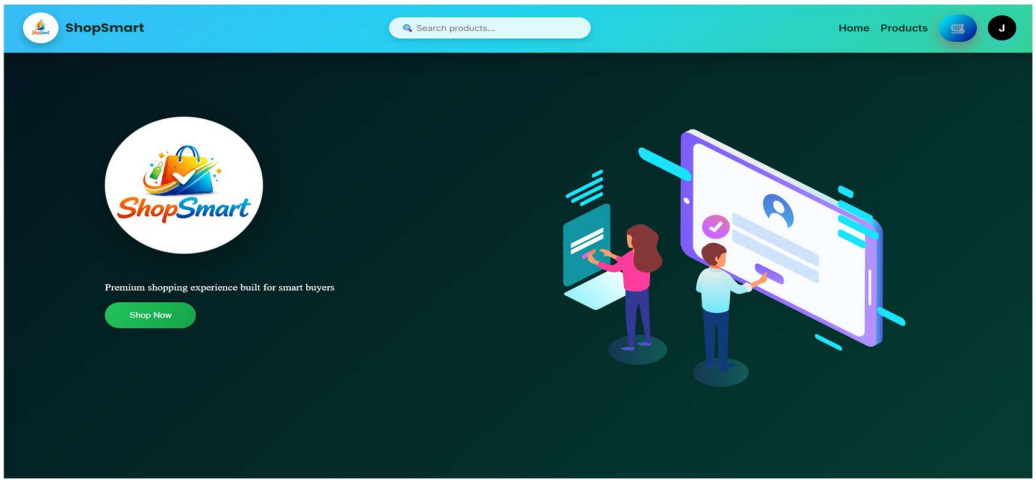


Fig : User Home Page

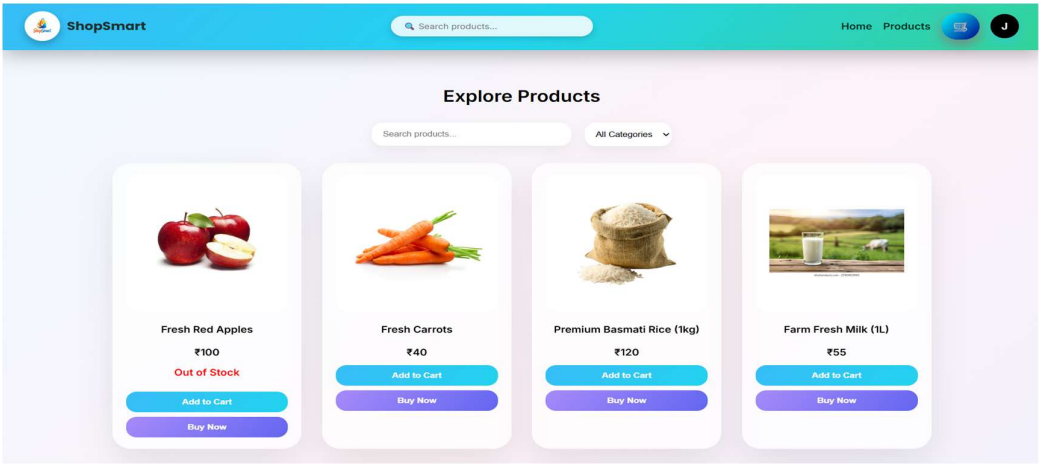


Fig : User Products Page

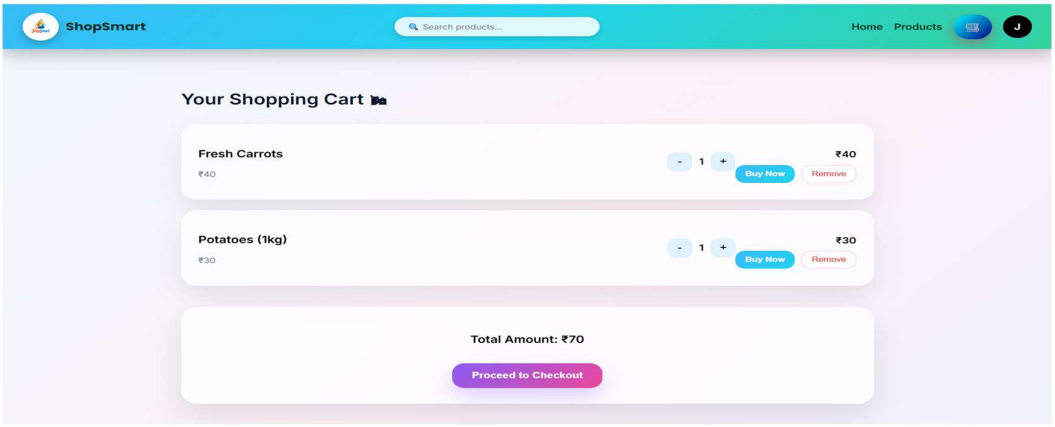


Fig : User Cart Page

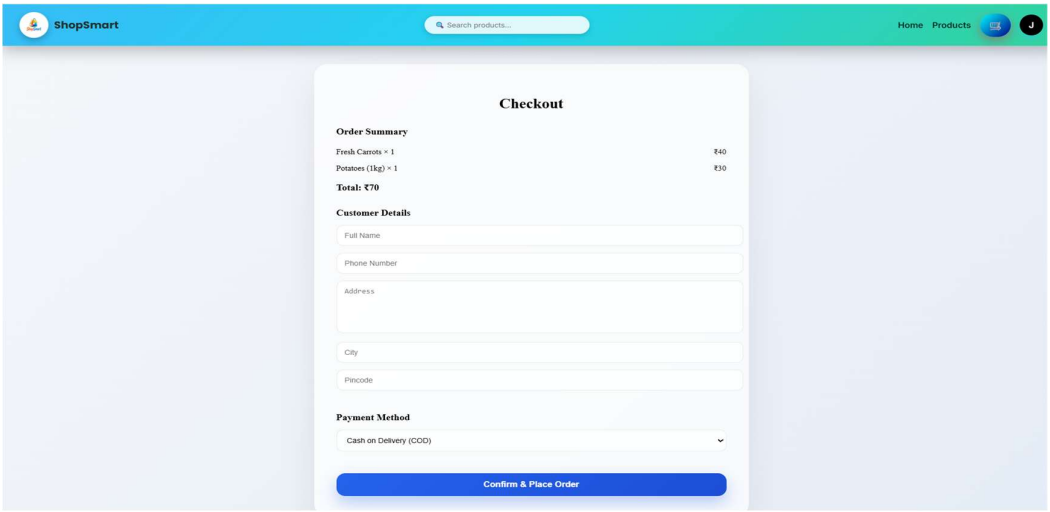


Fig : User Checkout Page

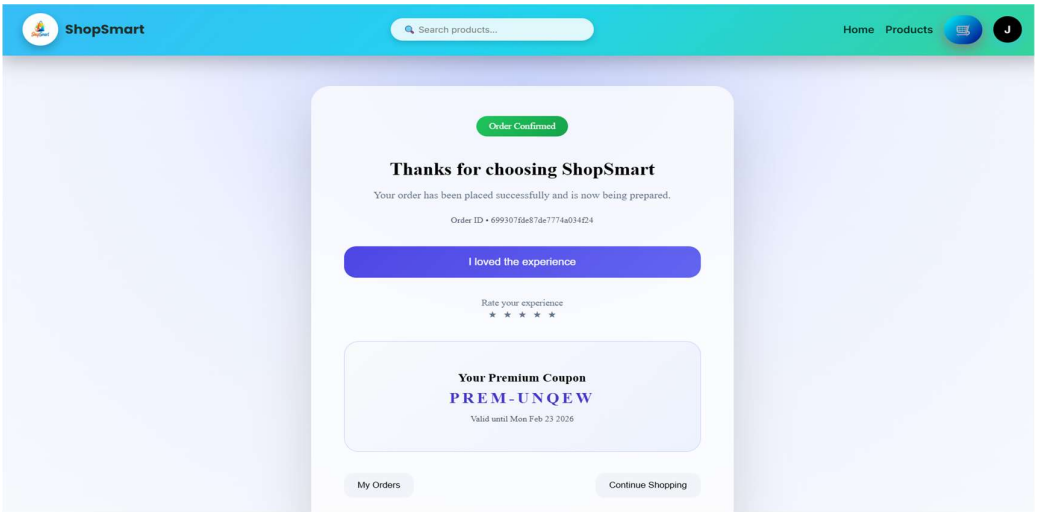


Fig : User Order Confirmation Page

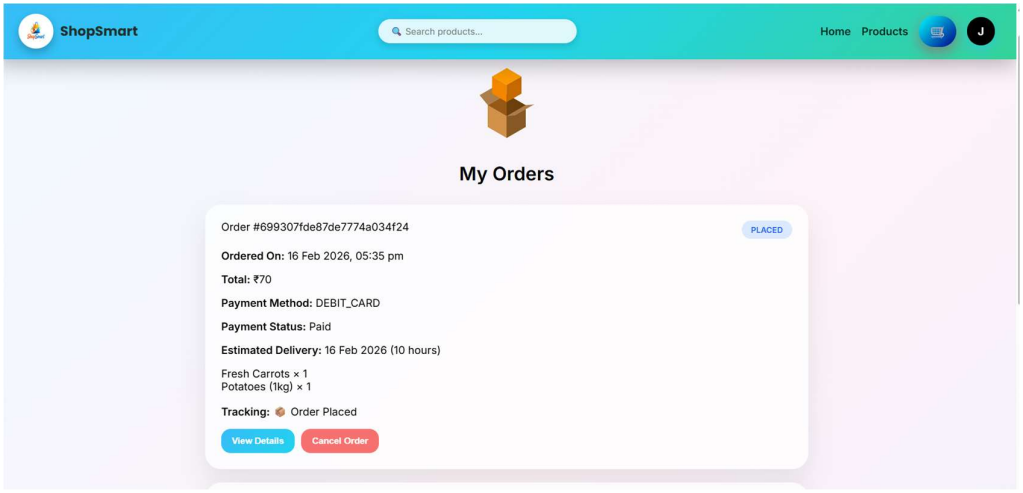


Fig : User Orders Page

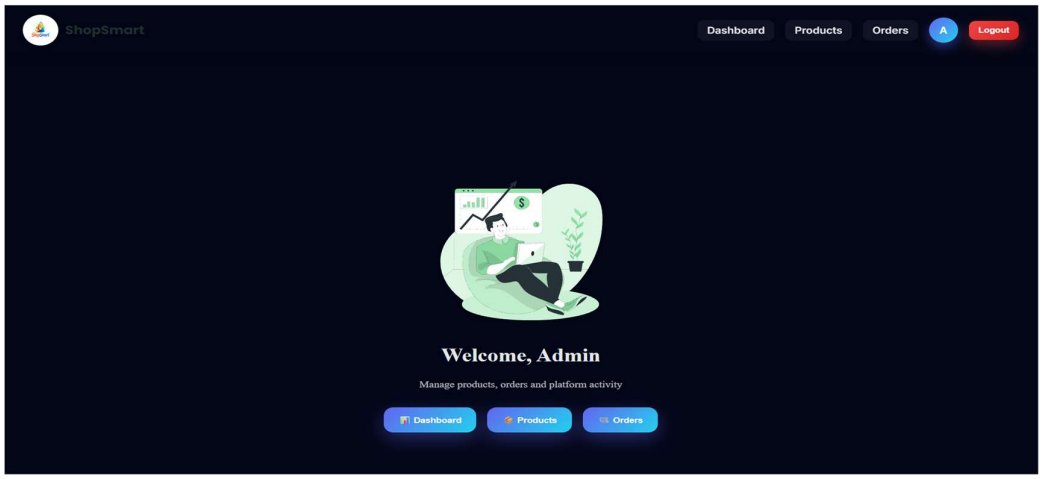


Fig : Admin Home Page

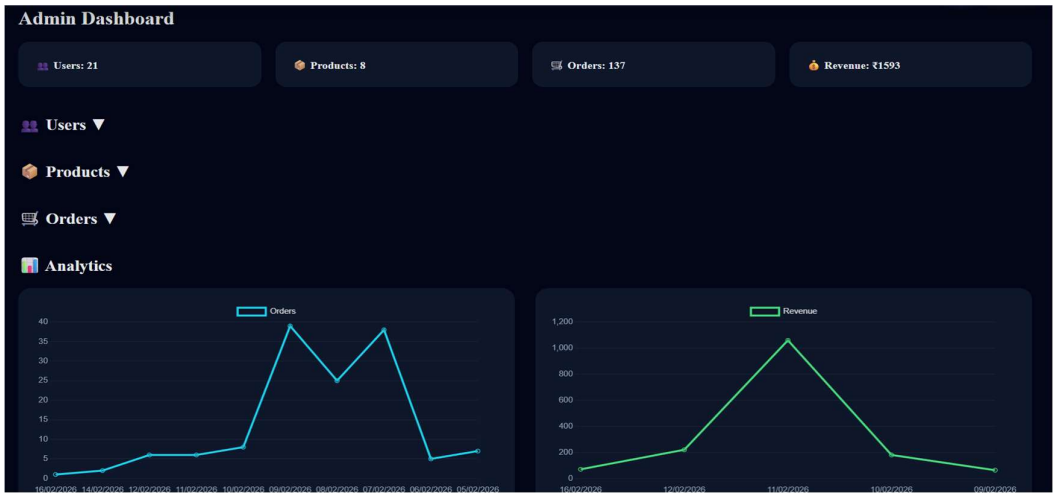


Fig : Admin Dashboard Page

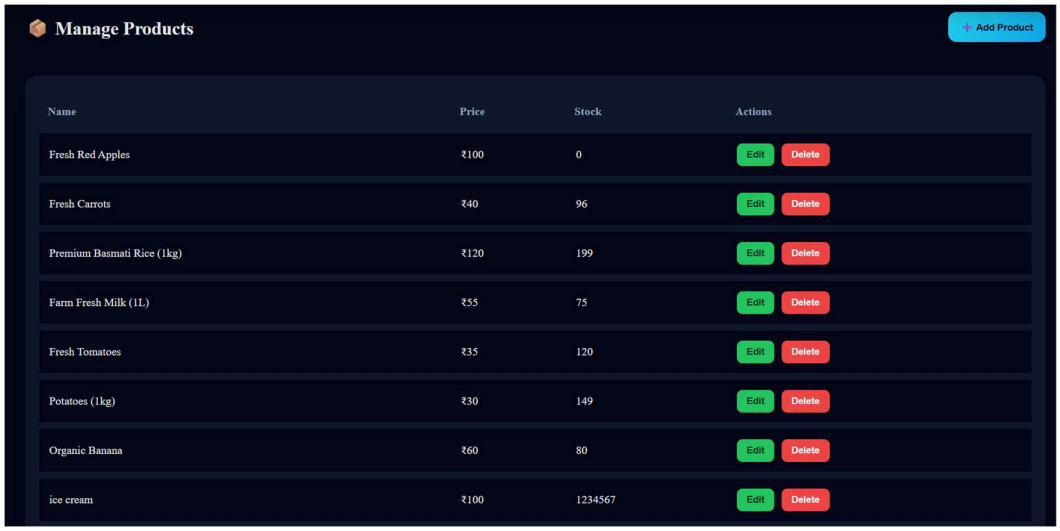


Fig : Admin Products Page

Fig : Admin Orders Page

8. ADVANTAGES & DISADVANTAGES

Advantages of the ShopSmart Project

1. **Full-Stack Implementation**
   The project demonstrates complete frontend and backend integration using the MERN stack, providing real-world development experience.
2. **Role-Based Access Control**
   Separate user and admin roles ensure secure access and proper authorization for sensitive operations like product and order management.
3. **Scalable Architecture**
   The modular folder structure (MVC pattern in backend) makes the project easy to maintain and extend.
4. **Secure Authentication**
   JWT-based authentication ensures stateless, secure communication between client and server.
5. **User-Friendly Shopping Flow**
   Features like product search, category filtering, cart management, and checkout provide a smooth user experience.
6. **Admin Management System**
   Admin dashboard enables efficient product CRUD operations and order status tracking.
7. **Academic and Practical Learning Value**
   Covers authentication, APIs, database design, and frontend routing, making it a strong academic project.

Disadvantages of the ShopSmart Project

1. **No Real Payment Gateway Integration**
   The system does not process real online payments, limiting production-level deployment.
2. **Basic UI/UX Design**
   The interface focuses more on functionality than advanced professional design.
3. **Token Stored in Local Storage**
   JWT stored in local storage may pose security risks in large-scale applications.
4. **Limited Scalability Optimization**
   No caching, load balancing, or performance optimization techniques are implemented.
5. **No Real-Time Features**
   Order updates and notifications are not real-time and require manual refresh.
6. **No Automated Testing Framework**
   The project mainly uses manual testing instead of automated unit and integration testing tools.
7. **Limited Deployment Configuration**
   Advanced DevOps practices like CI/CD pipelines and containerization are not implemented.

9. CONCLUSION

The ShopSmart project successfully demonstrates the design and implementation of a full-stack e-commerce web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The system was developed with the objective of providing a seamless online shopping experience while maintaining secure authentication and structured role-based access control. Throughout the development process, emphasis was placed on clean architecture, modular coding practices, and efficient database design to ensure maintainability and scalability.

The application enables users to register, log in securely, browse products, search and filter items, manage their shopping cart, and place orders efficiently. On the administrative side, the dashboard provides complete control over product management, order status updates, user management, and report generation. JWT-based authentication ensures secure communication between the frontend and backend, while middleware-based authorization restricts access to sensitive operations.

The project also demonstrates strong integration between client-side and server-side components through well-defined RESTful APIs. Proper error handling, validation mechanisms, and structured folder organization enhance the overall reliability and clarity of the system. Testing was performed across multiple modules to ensure functionality, security, and data integrity.

Although certain advanced features such as real-time notifications, payment gateway integration, performance optimization, and cloud deployment can be implemented in future versions, the current system effectively meets the fundamental requirements of an e-commerce platform. The project not only fulfills academic objectives but also provides practical exposure to real-world software development practices, including authentication strategies, database interactions, API design, and full-stack integration.

In conclusion, ShopSmart stands as a comprehensive and scalable e-commerce solution that reflects a strong understanding of modern web development technologies and application

architecture principles. With further enhancements, it has the potential to evolve into a fully production-ready commercial platform.

10. FUTURE SCOPE

The ShopSmart application has a strong foundational architecture and can be further enhanced with advanced features and improvements to make it production-ready and commercially scalable. The following are potential future developments:

1. Payment Gateway Integration

The system can be integrated with secure online payment gateways such as credit/debit card processing, UPI, and digital wallets. This will enable real-time payment transactions and make the platform suitable for real-world commercial deployment.

2. Advanced User Interface and Experience

Future improvements can focus on enhancing the UI/UX using modern design frameworks and responsive layouts. Features like animations, improved product galleries, better navigation menus, and dark/light themes can significantly improve user engagement.

3. Real-Time Notifications

Implementing real-time notifications using technologies like WebSockets can allow users to receive instant updates about order status changes, new product launches, discounts, or stock availability.

4. Inventory Management Enhancement

An automated inventory tracking system can be implemented to monitor stock levels and notify administrators when products are running low, reducing manual monitoring efforts.

5. Enhanced Security Measures

Security can be improved by:

- Storing tokens in HTTP-only cookies
- Implementing refresh tokens
- Adding multi-factor authentication (MFA)

- Strengthening input validation and rate limiting

6. Mobile Application Development

A dedicated mobile application using technologies like React Native or Flutter can expand the platform to Android and iOS users, increasing accessibility and market reach.

7. Cloud Deployment and Scalability

The project can be deployed using cloud platforms with:

- CI/CD pipelines
- Docker containerization
- Load balancing

This will ensure high availability and performance under heavy traffic.

11. APPENDIX

My Project Source code Files are available at :

https://drive.google.com/drive/folders/1IULsPfZQ3Jfpj-IWn1xDvNXlEhDfIRwe?usp=drive_link

My project Demo Video link is available at :

https://drive.google.com/file/d/1rk8Nj270eR39EGMRF183XiyVzjdErX5U/view?usp=drive_link

Github Resopitory Link :

https://github.com/Navi14-dev/Shopsmart-your-digital-grocery-store-experience