Project Report

1. INTRODUCTION

The introduction outlines the intent and scope of our Grocery WebApp, which is designed as a clone of Blinkit. It highlights the technologies used and the core idea behind building a fully functional grocery delivery platform with user, seller, and admin roles. The aim is to deliver an educational, scalable, and real-world applicable full-stack application**.**

1.1 Project Overview:

The Grocery WebApp is a full-stack grocery ordering and delivery platform designed as a functional clone of Blinkit. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the application provides a seamless real-time shopping experience for customers while allowing sellers to manage their inventory and admins to monitor all platform activities. The system is cloud-deployed and features secure user authentication, product listing, order tracking, and payment integration

1.2 Purpose

The project aims to replicate Blinkit's structure and functionality for academic learning. It serves as a learning base for full-stack development, integrating backend APIs, frontend interfaces, cloud hosting, payment gateways, and role-based user interactions.

2. IDEATION PHASE

2.1 Problem Statement

Consumers increasingly prefer convenient, fast, and contactless methods of purchasing everyday items. Traditional in-store shopping often lacks flexibility, real-time inventory updates, and efficiency. There is a growing need for a reliable, responsive web platform that simplifies grocery shopping, integrates secure payments, and allows real-time inventory access for users and sellers alike.

2.2 Empathy Map Canvas

Users: Working professionals, homemakers, students

Needs: Quick, contactless delivery; real-time product availability

Pain Points: Stock-outs, slow delivery, complex navigation

2.3 Brainstorming

Through team discussions, we prioritized essential features: smart product search, interactive cart, seller portal, and admin dashboard. After technology research, MERN was selected for fast development and efficient state management. We explored other options but found MERN offered better full-stack integration and modularity.

Grouped Features:

- User Module: Registration, login, cart, checkout, search, order tracking
- Seller Module: Product listing, inventory dashboard, order fulfillment
- Admin Module: User/seller control, analytics, content moderation
- External: Razorpay, MongoDB Atlas, JWT

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

- User visits platform and registers/logs in
- Browses or searches for grocery products
- Adds products to the cart
- Proceeds to checkout and enters address/payment
- Payment completes using Razorpay API
- Order is processed and tracked via seller/admin view

3.2 Solution Requirement

- Frontend: Responsive product catalog, filters, authentication views
- Backend: Order logic, route protection, payment gateway integration
- Seller Module: Dashboard for product and stock management
- Admin Panel: CRUD operations for users/sellers, content moderation
- Deployment: Netlify (UI), Render (API), MongoDB Atlas (DB)
- Security: HTTPS, JWT auth, server-side validation, protected routes

3.3 Data Flow Diagram

- Frontend sends requests via Axios
- Backend routes handle logic (Express.js)
- JWT middleware protects private routes
- MongoDB stores all persistent data (users, orders, products)
- Razorpay handles payment and status callbacks
- Seller/admin receive real-time updates on dashboard

3.4 Technology Stack

- **Frontend:** React.js, Axios, Tailwind CSS
- **Backend:** Node.js, Express.js, JWT, Bcrypt
- **Database:** MongoDB Atlas (cloud-hosted NoSQL)
- **APIs:** Razorpay for payments, Cloudinary for file storage
- **Hosting:** Netlify (frontend), Render (backend)

4. PROJECT DESIGN

4.1 Problem Solution Fit

Our solution addresses the identified problem by enabling users to quickly browse, select, and purchase groceries with real-time inventory and fast order processing. Sellers manage their stock with ease, and admins oversee all system interactions, closely mirroring Blinkit's structure.

4.2 Proposed Solution

The proposed solution features three roles:

- **User Portal:** Browse, filter, search, add to cart, and purchase products.
- **Seller Portal:** Add/edit product listings, monitor stock, track orders.
- **Admin Panel:** Access system data, review user activity, manage performance

4.3 Solution Architecture

React frontend fetches and posts data via REST APIs.

Express backend handles business logic, session tokens, and validations.

MongoDB stores user, product, and order data.

Razorpay processes payments using secure callbacks.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Sprint 1: Authentication, homepage, product browsing
- Sprint 2: Cart, seller dashboard, inventory updates
- Sprint 3: Admin panel, user management, order logic
- Sprint 4: Razorpay integration, chatbot, voice search, final testing

Sprint Velocity

- Total Duration: 22 March 2025 – 12 April 2025
- Total Story Points: 40
- Velocity: 10 points/sprint
- Average Velocity: 2.2 days per story point

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

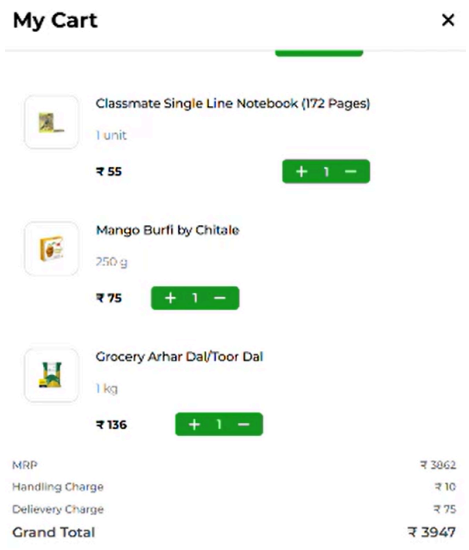
- Tested key modules using Lighthouse and Postman
- Evaluated UI rendering time, API latency, cart and checkout workflows
- Load-tested backend with 30 concurrent simulated users
- Validated Razorpay transaction callbacks and order logs

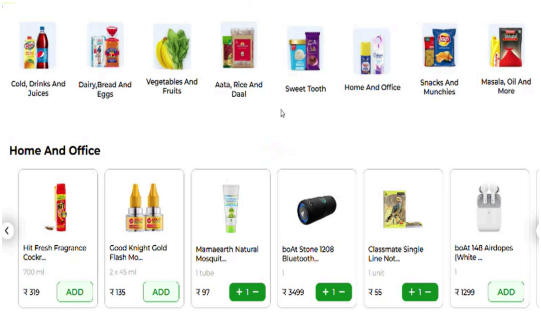
Performance monitoring

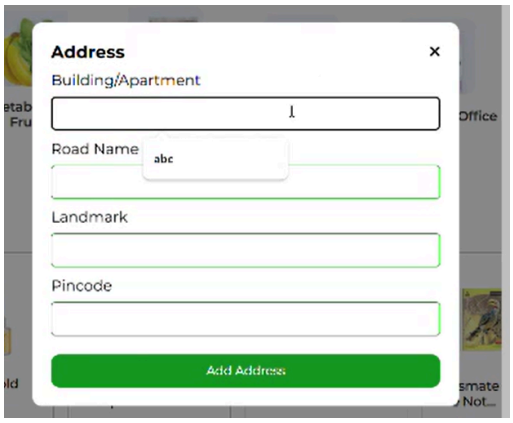
- Google DevTools used to measure initial page loads
- Backend monitored using logs and response time tracking
- Checked memory usage, CPU load during peak simulations

7. RESULTS

7.1 Output Screenshots







8. ADVANTAGES & DISADVANTAGES

Advantages:

- Consistent and responsive frontend design
- Optimized backend API response times

- Modular code structure (easy maintenance and scaling)
- Seamless integration of payment gateway

Disadvantages:

- Heavily reliant on consistent internet speed
- Limited to web-based interface

9. CONCLUSION

The Grocery WebApp delivers an effective online grocery shopping platform with all the essential features needed for real-time browsing, ordering, and checkout. The implementation was successful from both a technical and user experience standpoint. Every module underwent thorough testing, with special attention to speed and reliability.

Moreover, this project acted as a foundational step toward building more complex systems. It reinforced the team's understanding of client-server communication, middleware usage, security best practices, and real-time UI rendering, all of which are vital skills in today's full-stack development domain

10. FUTURE SCOPE

Plans for future development include native mobile apps, live delivery tracking, and smart recommendation systems. Additionally, we aim to improve customer engagement by incorporating push notifications, customer loyalty features, and multilingual support.

- Build a mobile version using React Native or Flutter
- Integrate delivery partner module with real-time map tracking
- Use AI/ML for personalized recommendations
- Integrate chatbots and speech-based search for ease of access
- Develop subscription and loyalty features

11. APPENDIX

GitHub https://github.com/NeelamKushwaha/Grocery_webapp_fullstackProject.git

Project Demo Link
https://drive.google.com/file/d/1UkKI_hIoIsxFNwyq0zidZ9EqUfoO9K1l/view?usp=sharing