Project Report Format

1.INTRODUCTION

1.1 Project Overview

SBFoods: Your Smart Online Food Delivery Platform 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, fast, and user-friendly online food ordering and delivery experience.

The platform allows customers to browse restaurants and food items across various categories, view detailed descriptions, add items to a cart, customize orders, and securely complete the checkout process. Users can also track their orders in real time and view order history.

The system includes an administrative interface where admins can manage restaurants, food items, orders, delivery status, and user accounts efficiently. The application follows a structured three-tier architecture with clear separation between the frontend, backend, and database layers, ensuring scalability, maintainability, and security.

Role-based access control is implemented to differentiate between customer and admin functionalities. Authentication is handled securely using encrypted passwords and token-based authorization (JWT), ensuring protected access to sensitive operations.

1.2 Purpose

The purpose of the SBFoods Delivery App project is to develop a user-friendly online food ordering platform that allows customers to explore menus, manage their cart, place orders, and track deliveries securely and efficiently.

The system also provides administrative control for managing food items, users, and orders effectively. Additionally, the project demonstrates the practical implementation of full-stack web development using the MERN stack with secure authentication, structured architecture, and real-world delivery workflow integration.

Key Objectives of the System:

• To design and develop a responsive and interactive food delivery 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 and order processing.
• To design and manage MongoDB database schemas using Mongoose for users, food items, carts, and orders.
• To implement complete food management, cart management, order placement, and order tracking features.
• To provide an admin dashboard for managing food items, users, orders, and delivery status.

2. IDEATION PHASE

2.1 Problem Statement

In today's fast-paced lifestyle, many people find it difficult to cook meals at home or visit restaurants due to busy schedules and time constraints. Traditional food ordering methods such as phone calls or visiting restaurants lack convenience, transparency, and real-time tracking. Small and medium restaurants also struggle to manage orders, delivery coordination, and customer data efficiently without a centralized digital system.

Therefore, there is a need for a secure, user-friendly, and efficient online food delivery platform that simplifies food ordering, enables real-time order tracking, and provides proper administrative control for managing menu items, customers, and delivery operations.

Problem Statement – 1

I am a food customer
I'm trying to order food online quickly and conveniently
But I cannot easily browse menus, compare items, or track my order status in real time
Because many local restaurants do not provide a proper digital platform with centralized ordering and live updates
Which makes me feel frustrated, confused, and uncertain about my order

---

Problem Statement – 2

I am a restaurant administrator
I'm trying to manage menu items, customer orders, and delivery status efficiently
But I struggle to track orders and coordinate deliveries manually
Because there is no centralized online management system
Which makes me feel overwhelmed and stressed

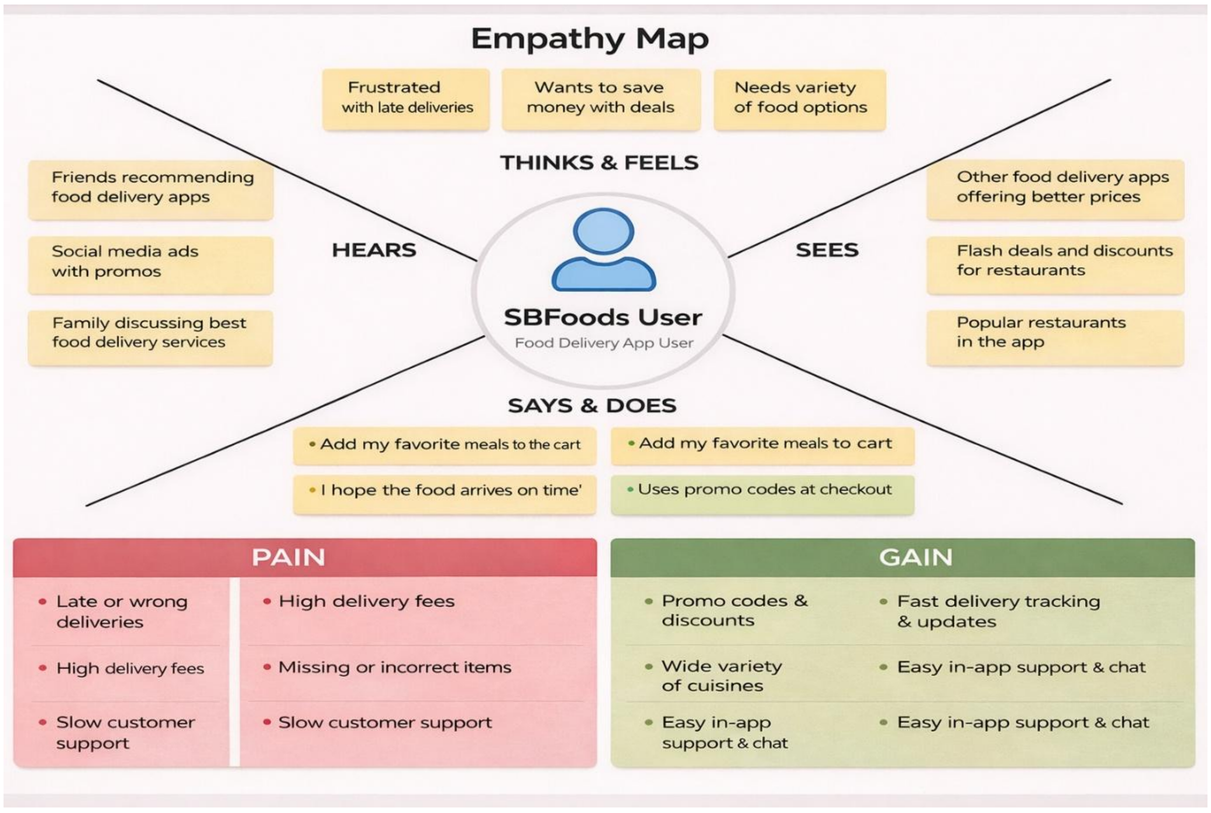
Structured Problem Statements Table Format

Problem Statement (PS)	I am	I'm trying to	But	Because	Which makes me feel
PS-1	**A customer**	**Find and order quality food quickly at**	**It is hard to compare menu items and**	**Many restaurants do not provide real-**	**Confused and frustrated**

Problem Statement (PS)	**I am**	**I'm trying to**	**But**	**Because**	**Which makes me feel**
		reasonable prices	**track delivery status**	**time digital ordering systems**	
PS-2	**An admin**	**Monitor customer orders and manage food delivery operations**	**Tracking orders and updating delivery status manually is difficult**	**Data is not organized in one centralized system**	**Stressed and overburdene**

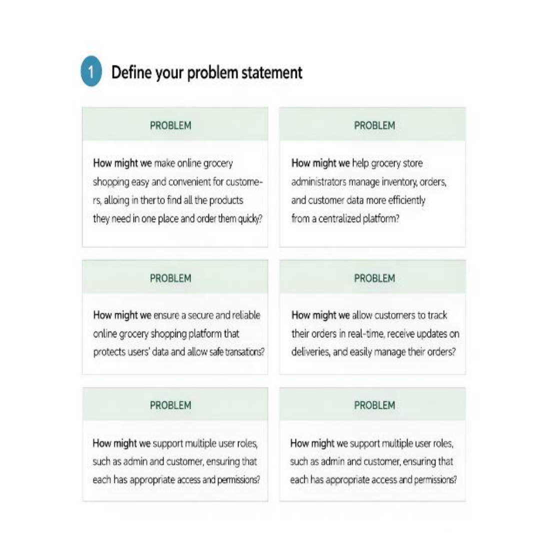
2.2 Empathy Map:

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


Empathy Map
Frustrated with late deliveries
Wants to save money with deals
Needs variety of food options
THINKS & FEELS
Friends recommending food delivery apps
Social media ads with promos
Family discussing best food delivery services
HEARS
SBFoods User
Food Delivery App User
SEES
Other food delivery apps offering better prices
Flash deals and discounts for restaurants
Popular restaurants in the app
SAYS & DOES
• Add my favorite meals to the cart
• I hope the food arrives on time'
• Add my favorite meals to cart
• Uses promo codes at checkout
PAIN
• Late or wrong deliveries
• High delivery fees
• Slow customer support
• High delivery fees
• Missing or incorrect items
• Slow customer support
GAIN
• Promo codes & discounts
• Wide variety of cuisines
• Easy in-app support & chat
• Fast delivery tracking & updates
• Easy in-app support & chat
• Easy in-app support & chat

2.2 Brainstorm & Idea Prioritization:-

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



1 Define your problem statement

PROBLEM

How might we make online grocery shopping easy and convenient for custome-rs, alloing in ther to find all the products they need in one place and order them quicky?

PROBLEM

How might we help grocery store administrators manage inventory, orders, and customer data more efficiently from a centralized platform?

PROBLEM

How might we ensure a secure and reliable online grocery shopping platform that protects users' data and allow safe tansations?

PROBLEM

How might we allow customers to track their orders in real-time, receive updates on deliveries, and easily manage their orders?

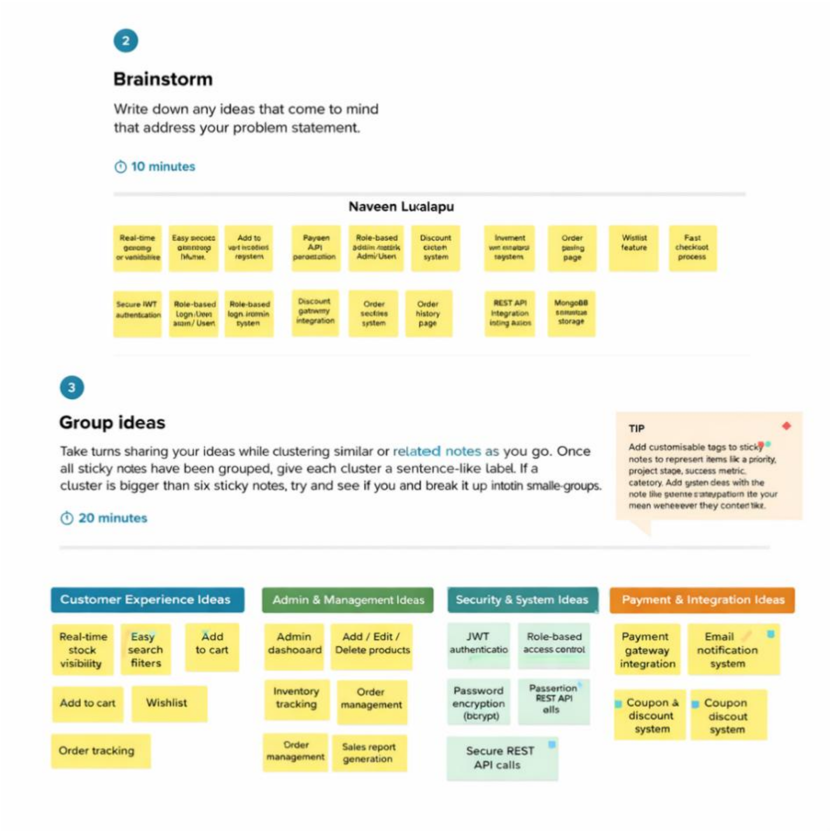
PROBLEM

How might we support multiple user roles, such as admin and customer, ensuring that each has appropriate access and permissions?

PROBLEM

How might we support multiple user roles, such as admin and customer, ensuring that each has appropriate access and permissions?

Step-2: Brainstorm, Idea Listing and Grouping



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Naveen Lukalapu
Real-time gerong or vanibiliise
Easy secoes oinnrong Mumer
Add to vart incolies reystem
Payeen API perontcition
Role-based addin matirk Admi/User
Discount cicteh system
Invement wm ennatoral taystem
Order pasing page
Wistlist feature
Fast checkoot process
Secure IWT autrentcation
Role-based Logn/Uem aisin/User
Role-based logn iromin tysten
Discount gatrwny integration
Order secties system
Order history page
REST API integration inting Aalos
MongoBB snmaise storage

3

Group ideas

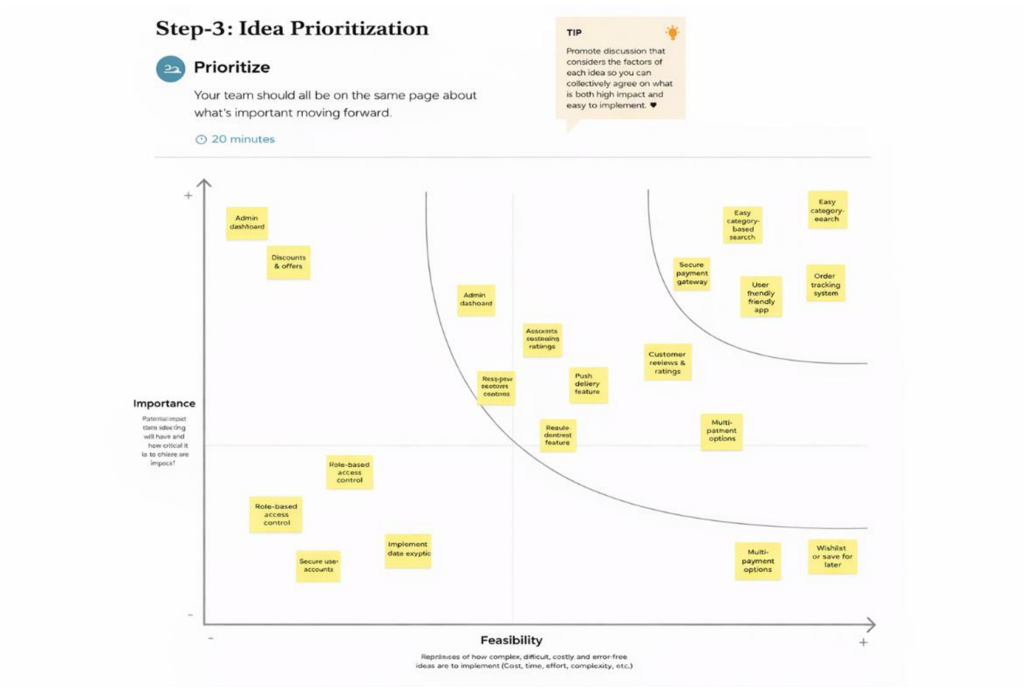
Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up intotin smalle-groups.

20 minutes

TIP

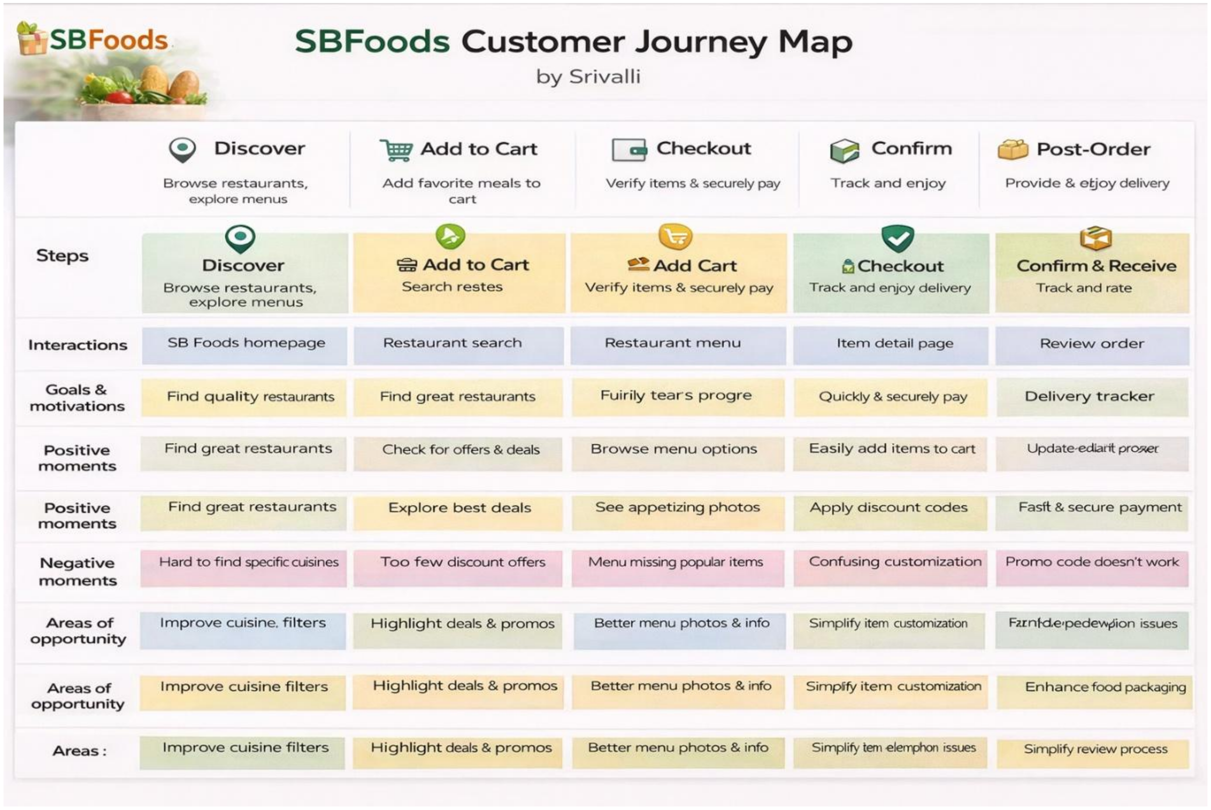
Add customisable tags to sticky notes to represent items lik a priority, project stape, success metric, catetory. Add gysten deas with the note like guente sateypatiorn ite your mean weheeever they conter like.

Step-3: Idea Prioritization


Step-3: Idea Prioritization
Prioritize
Your team should all be on the same page about what's important moving forward.
20 minutes
TIP
Promote discussion that considers the factors of each idea so you can collectively agree on what is both high impact and easy to implement.
Importance
Feasibility
Admin dashboard
Discounts & offers
Admin dashoard
Assounts costeaing ratings
Ress-pew seotorrs ceatrres
Push deliery feature
Reaule-dentrest feature
Customer reviews & ratings
Multi-patment options
Secure payment gateway
Easy category-based searcch
Easy category-eearch
User fhendly friendly app
Order tracking system
Role-based access control
Role-based access control
Implement date exyptic
Secure user accounts
Multi-payment options
Wishlist or save for later

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map: -



SBFoods

SBFoods Customer Journey Map

by Srivalli

	Discover Browse restaurants, explore menus	Add to Cart Add favorite meals to cart	Checkout Verify items & securely pay	Confirm Track and enjoy	Post-Order Provide & etjoy delivery
Steps	Discover Browse restaurants, explore menus	Add to Cart Search restes	Add Cart Verify items & securely pay	Checkout Track and enjoy delivery	Confirm & Receive Track and rate
Interactions	SB Foods homepage	Restaurant search	Restaurant menu	Item detail page	Review order
Goals & motivations	Find quality restaurants	Find great restaurants	Fuirily tear's progre	Quickly & securely pay	Delivery tracker
Positive moments	Find great restaurants	Check for offers & deals	Browse menu options	Easily add items to cart	Update-edlarit proxer
Positive moments	Find great restaurants	Explore best deals	See appetizing photos	Apply discount codes	Fasft & secure payment
Negative moments	Hard to find specific cuisines	Too few discount offers	Menu missing popular items	Confusing customization	Promo code doesn't work
Areas of opportunity	Improve cuisine. filters	Highlight deals & promos	Better menu photos & info	Simplify item customization	Fzrnfde-pedewpion issues
Areas of opportunity	Improve cuisine filters	Highlight deals & promos	Better menu photos & info	Simpify item customization	Enhance food packaging
Areas :	Improve cuisine filters	Highlight deals & promos	Better menu photos & info	Simplify tem elemphon issues	Simplify review process

3.2 Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

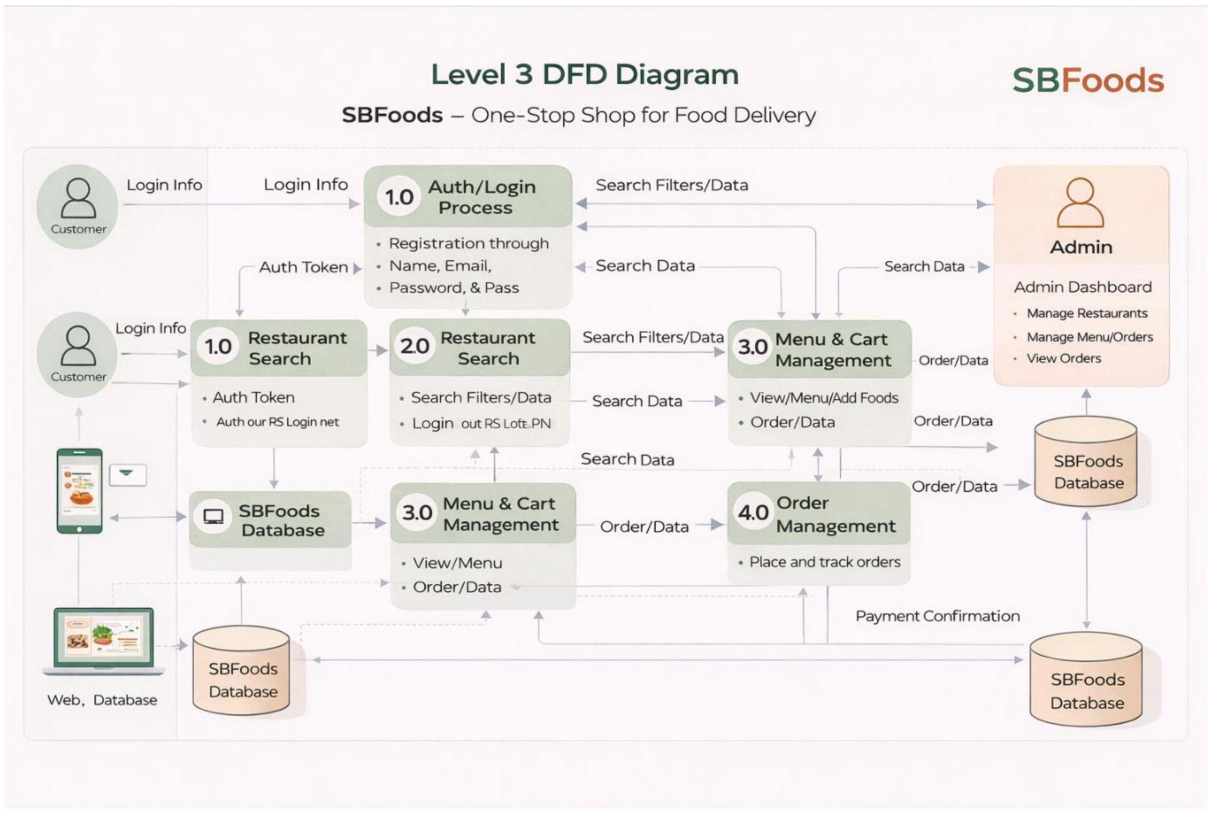
FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	User Registration	Registration through Form (Name, Email, Password) Password Encryption (bcrypt) Validation of Email & Password
FR-2	User Login & Authentication	Login using Email & Password JWT Token Generation Role-based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

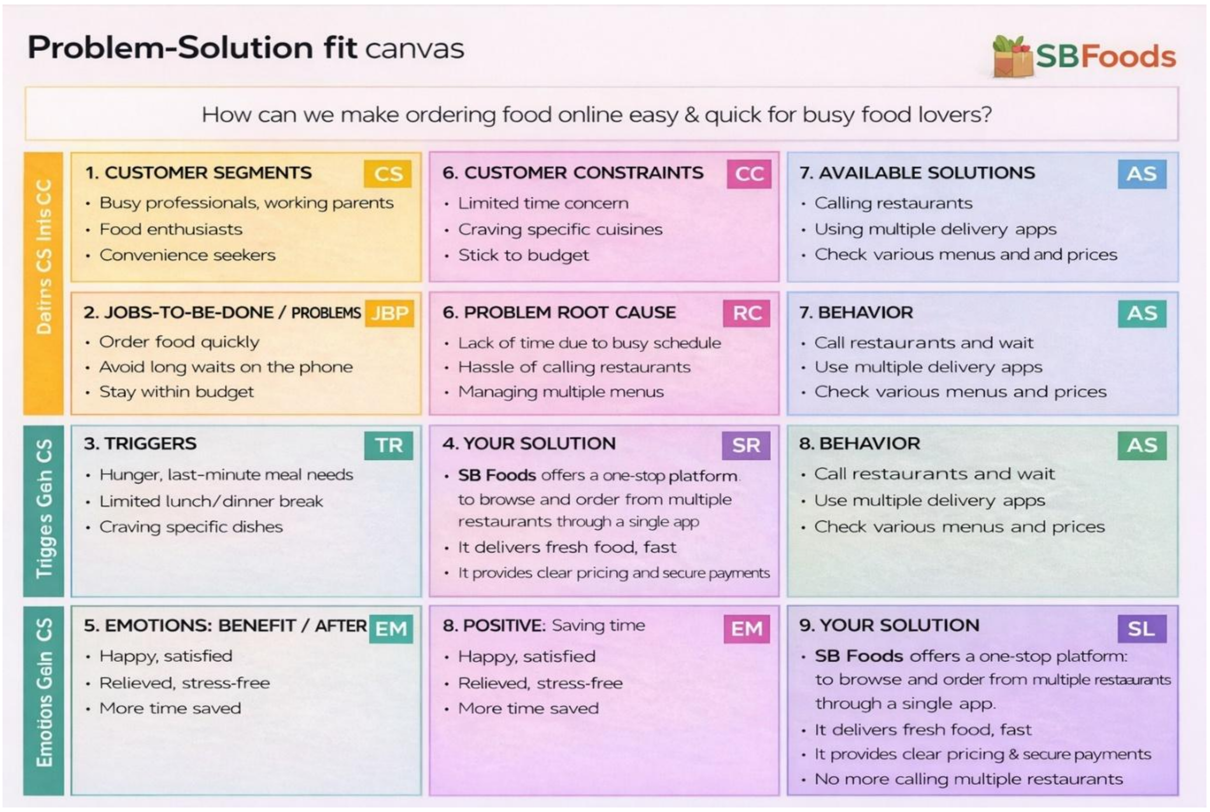
Table-2: Application Characteristics:

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



Problem-Solution fit canvas

SBFoods

How can we make ordering food online easy & quick for busy food lovers?

Datins CS Inis CC	**1. CUSTOMER SEGMENTS** CS • Busy professionals, working parents • Food enthusiasts • Convenience seekers	**6. CUSTOMER CONSTRAINTS** CC • Limited time concern • Craving specific cuisines • Stick to budget	**7. AVAILABLE SOLUTIONS** AS • Calling restaurants • Using multiple delivery apps • Check various menus and and prices
	2. JOBS-TO-BE-DONE / PROBLEMS JBP • Order food quickly • Avoid long waits on the phone • Stay within budget	**6. PROBLEM ROOT CAUSE** RC • Lack of time due to busy schedule • Hassle of calling restaurants • Managing multiple menus	**7. BEHAVIOR** AS • Call restaurants and wait • Use multiple delivery apps • Check various menus and prices
Trigges Geh CS	**3. TRIGGERS** TR • Hunger, last-minute meal needs • Limited lunch/dinner break • Craving specific dishes	**4. YOUR SOLUTION** SR • **SB Foods** offers a one-stop platform to browse and order from multiple restaurants through a single app • It delivers fresh food, fast • It provides clear pricing and secure payments	**8. BEHAVIOR** AS • Call restaurants and wait • Use multiple delivery apps • Check various menus and prices
Emotions Geln CS	**5. EMOTIONS: BENEFIT / AFTER** EM • Happy, satisfied • Relieved, stress-free • More time saved	**8. POSITIVE:** Saving time EM • Happy, satisfied • Relieved, stress-free • More time saved	**9. YOUR SOLUTION** SL • **SB Foods** offers a one-stop platform: to browse and order from multiple restaurants through a single app. • It delivers fresh food, fast • It provides clear pricing & secure payments • No more calling multiple restaurants

4.2 Proposed Solution : -

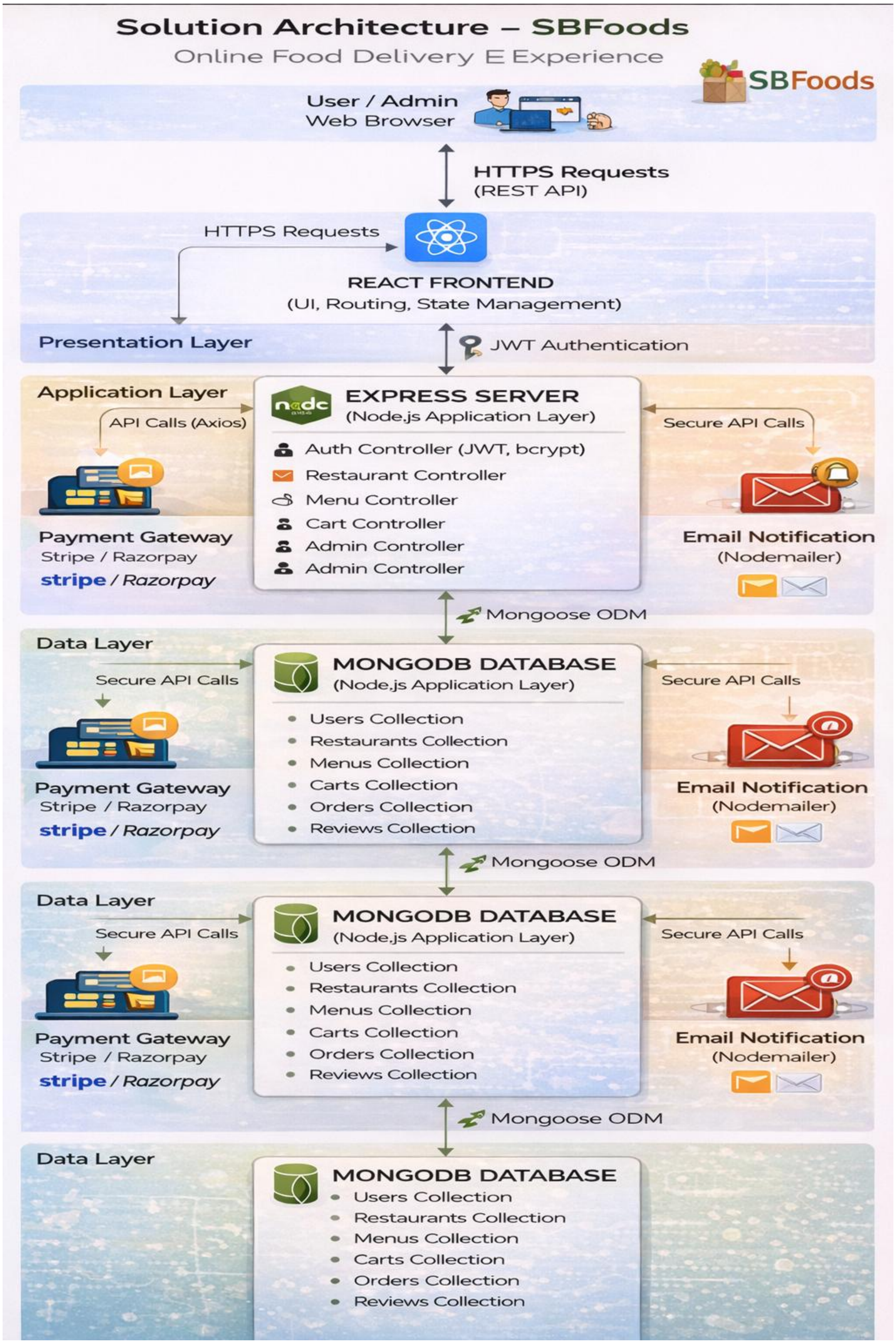
S.No	Parameter	Description
1	**Problem Statement (Problem to be solved)**	Busy individuals and working families face difficulty in purchasing groceries due to lack of time, long queues, limited product availability visibility, and fragmented online grocery services. There is no centralized, reliable, 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

.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


Solution Architecture – SBFoods
Online Food Delivery E Experience
SBFoods
User / Admin
Web Browser
HTTPS Requests
(REST API)
HTTPS Requests
REACT FRONTEND
(UI, Routing, State Management)
Presentation Layer
JWT Authentication
Application Layer
API Calls (Axios)
EXPRESS SERVER
(Node.js Application Layer)
Auth Controller (JWT, bcrypt)
Restaurant Controller
Menu Controller
Cart Controller
Admin Controller
Admin Controller
Secure API Calls
Payment Gateway
Stripe / Razorpay
stripe / Razorpay
Email Notification
(Nodemailer)
Mongoose ODM
Data Layer
Secure API Calls
MONGODB DATABASE
(Node.js Application Layer)
Users Collection
Restaurants Collection
Menus Collection
Carts Collection
Orders Collection
Reviews Collection
Secure API Calls
Payment Gateway
Stripe / Razorpay
stripe / Razorpay
Email Notification
(Nodemailer)
Mongoose ODM
Data Layer
Secure API Calls
MONGODB DATABASE
(Node.js Application Layer)
Users Collection
Restaurants Collection
Menus Collection
Carts Collection
Orders Collection
Reviews Collection
Secure API Calls
Payment Gateway
Stripe / Razorpay
stripe / Razorpay
Email Notification
(Nodemailer)
Mongoose ODM
Data Layer
MONGODB DATABASE
Users Collection
Restaurants Collection
Menus Collection
Carts Collection
Orders Collection
Reviews Collection

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**

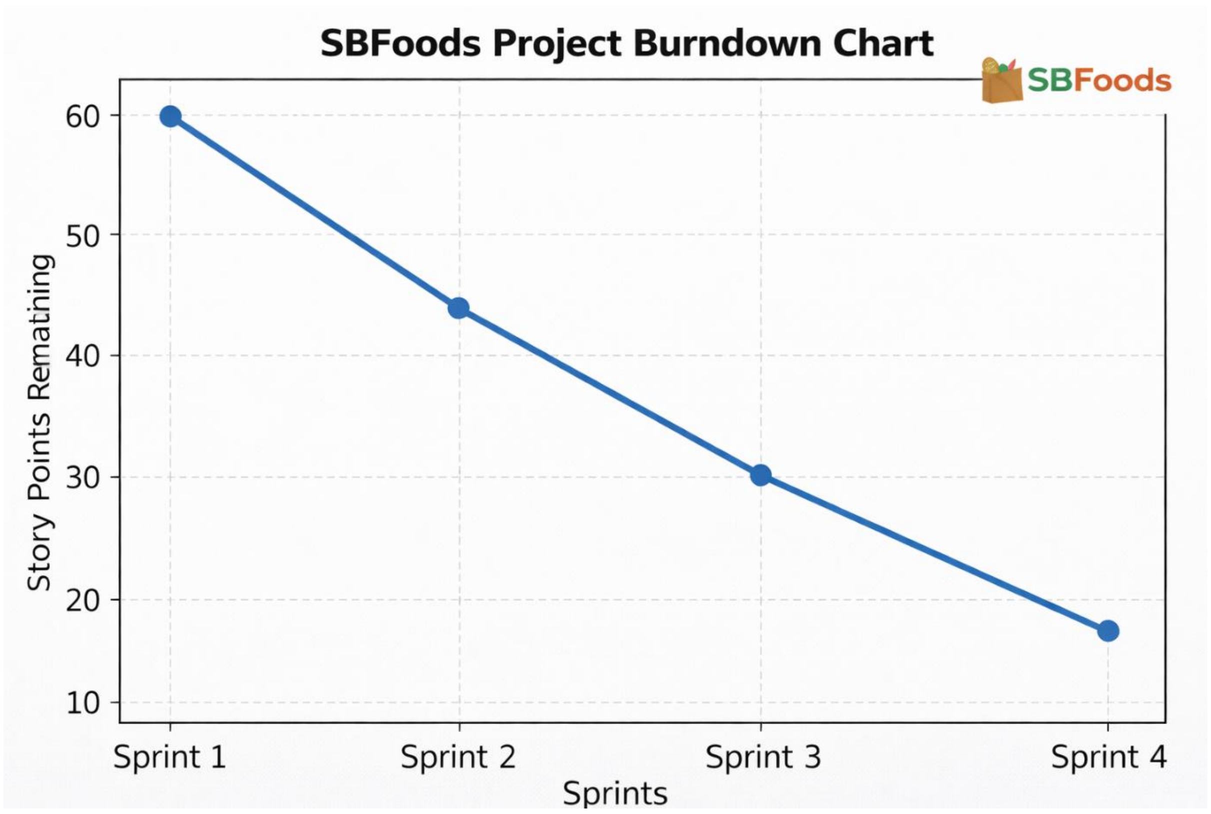
Number of sprints = 4

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

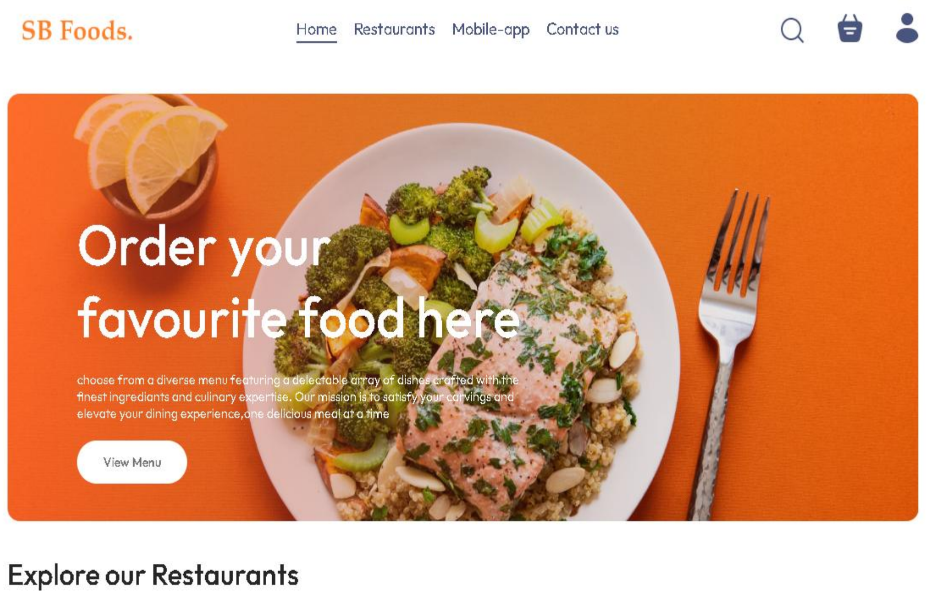
FT-03	Admin Login	Login with predefined admin credentials	Admin dashboard opens	Admin access granted	Pass
FT-04	Product Listing	Load product page	Products display from DB	Products shown correctly	Pass
FT-05	Add Product (Admin)	Add new product from admin panel	Product stored in DB & visible	Product added	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


SB Foods.

Home Restaurants Mobile-app Contact us

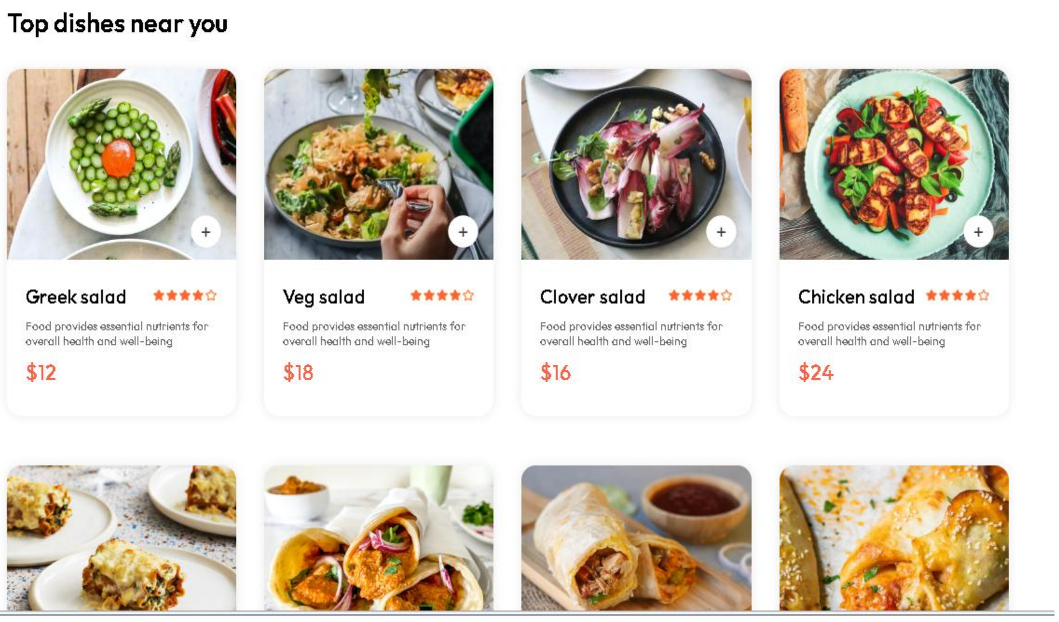
Order your favourite food here

choose from a diverse menu featuring a delectable array of dishes crafted with the finest ingrediants and culinary expertise. Our mission is to satisfy your carvings and elevate your dining experience,one delicious meal at a time

View Menu

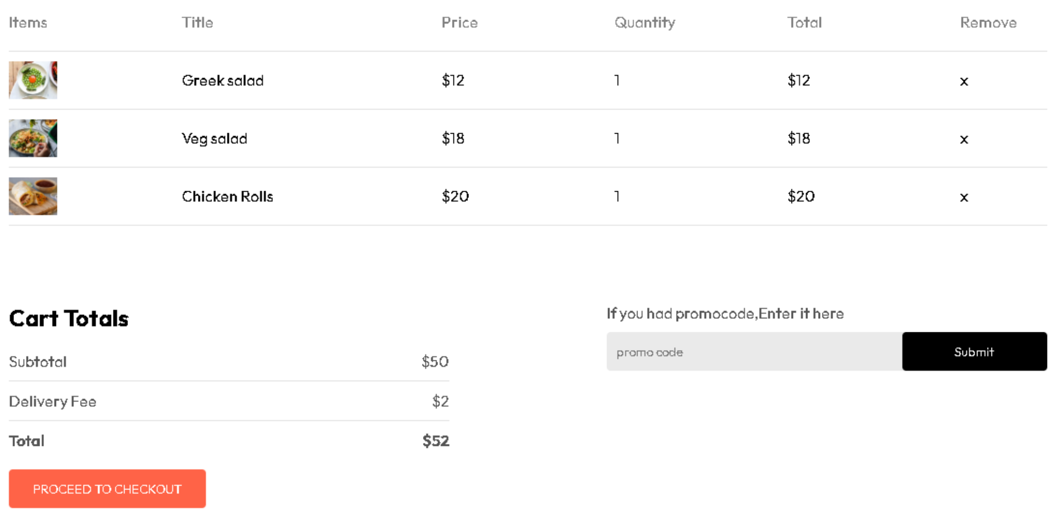
Explore our Restaurants

Fig : User Home Page


Top dishes near you

Greek salad ★★★★☆	Veg salad ★★★★☆	Clover salad ★★★★☆	Chicken salad ★★★★☆
Food provides essential nutrients for overall health and well-being	Food provides essential nutrients for overall health and well-being	Food provides essential nutrients for overall health and well-being	Food provides essential nutrients for overall health and well-being
$12	$18	$16	$24

Fig : User Products Page

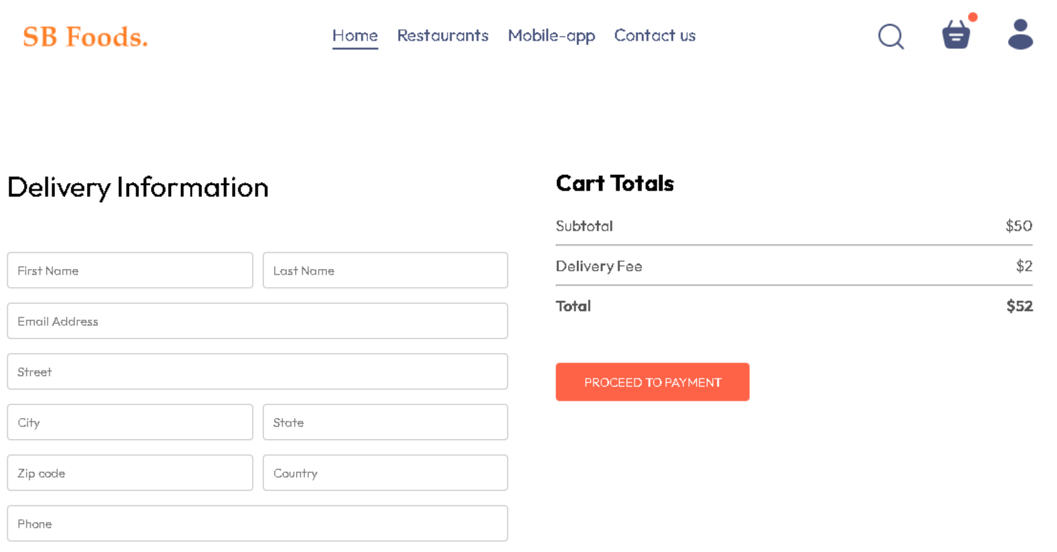


Items	Title	Price	Quantity	Total	Remove
	Greek salad	$12	1	$12	x
	Veg salad	$18	1	$18	x
	Chicken Rolls	$20	1	$20	x

Cart Totals

Subtotal	$50
Delivery Fee	$2
Total	**$52**

Fig :User Cart Page

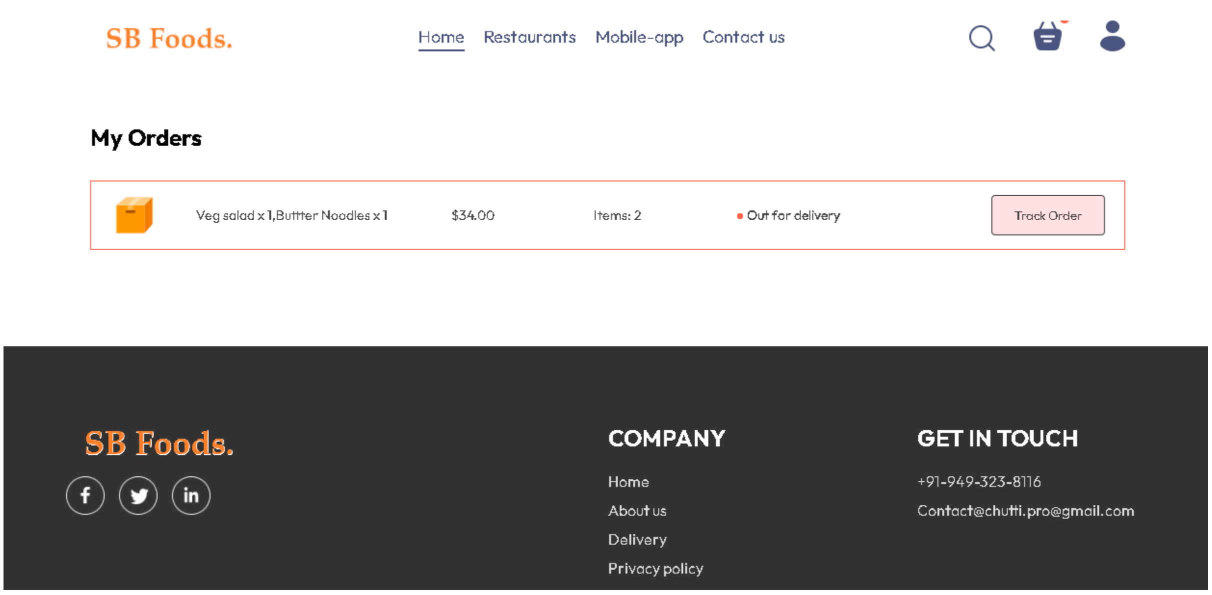


Delivery Information

Cart Totals

Subtotal	$50
Delivery Fee	$2
Total	**$52**

Fig : User Checkout Page



Veg salad x 1,Butter Noodles x 1 $34.00 Items: 2 ● Out for delivery Track Order

Fig : User Order Confirmation Page

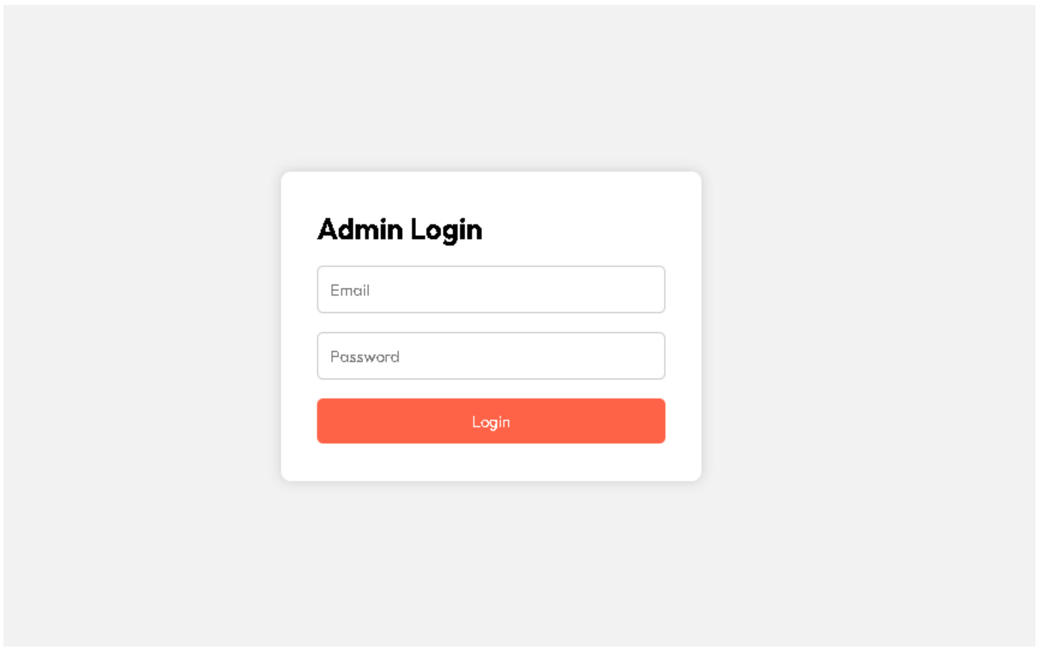
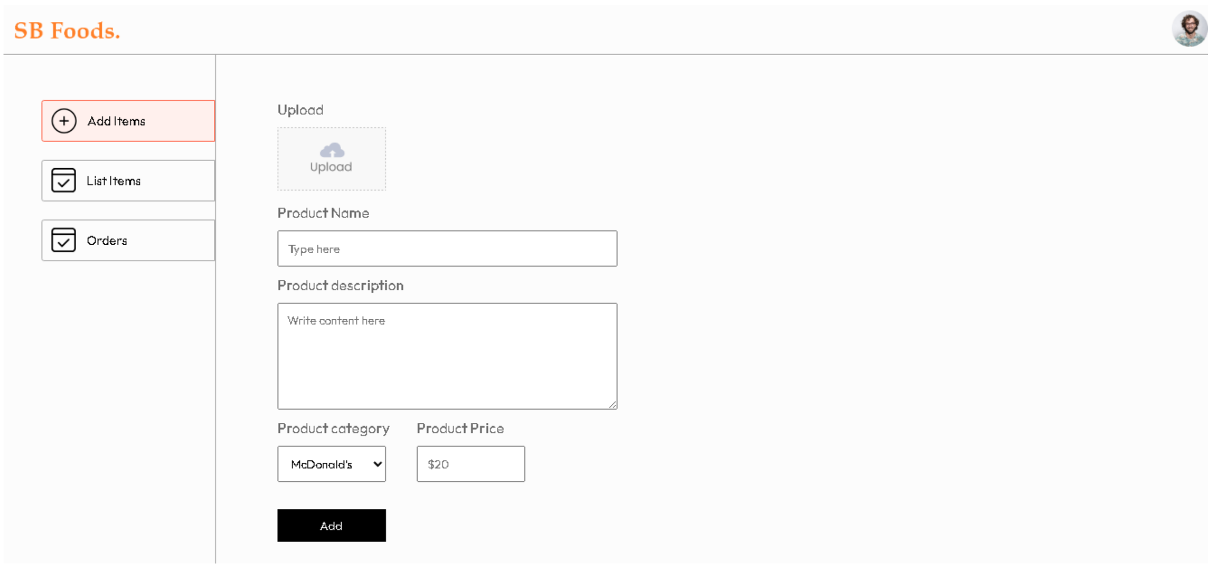


Fig : Admin Home Login Page


SB Foods.

Add Items

List Items

Orders

Upload

Upload

Product Name

Type here

Product description

Write content here

Product category

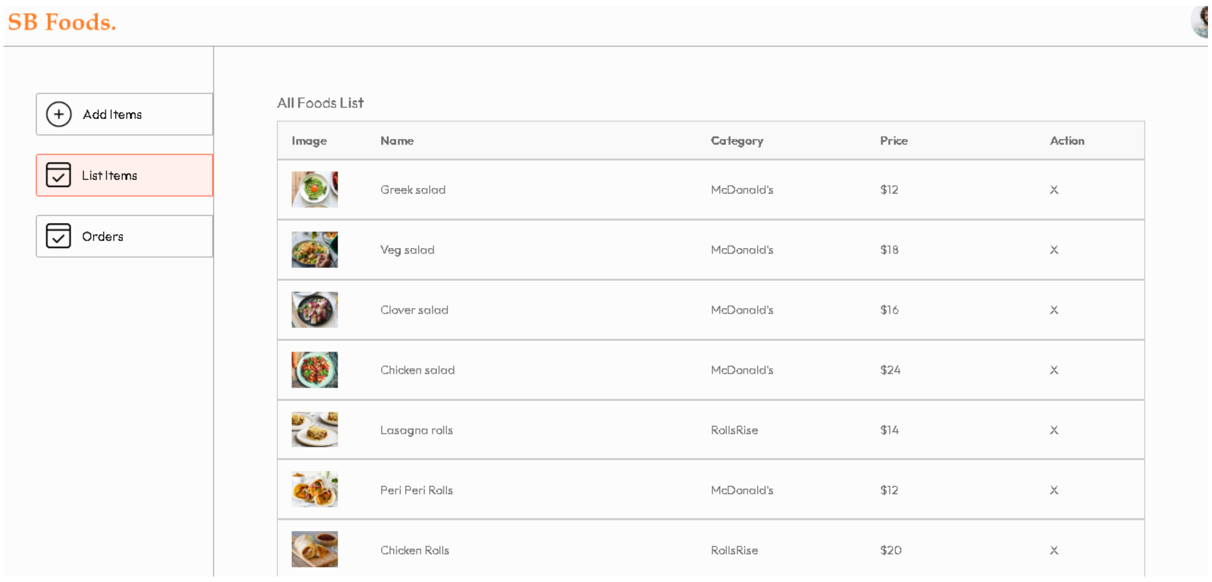
McDonald's

Product Price

$20

Add

Fig : Admin Product Adding Page


SB Foods.

Add Items

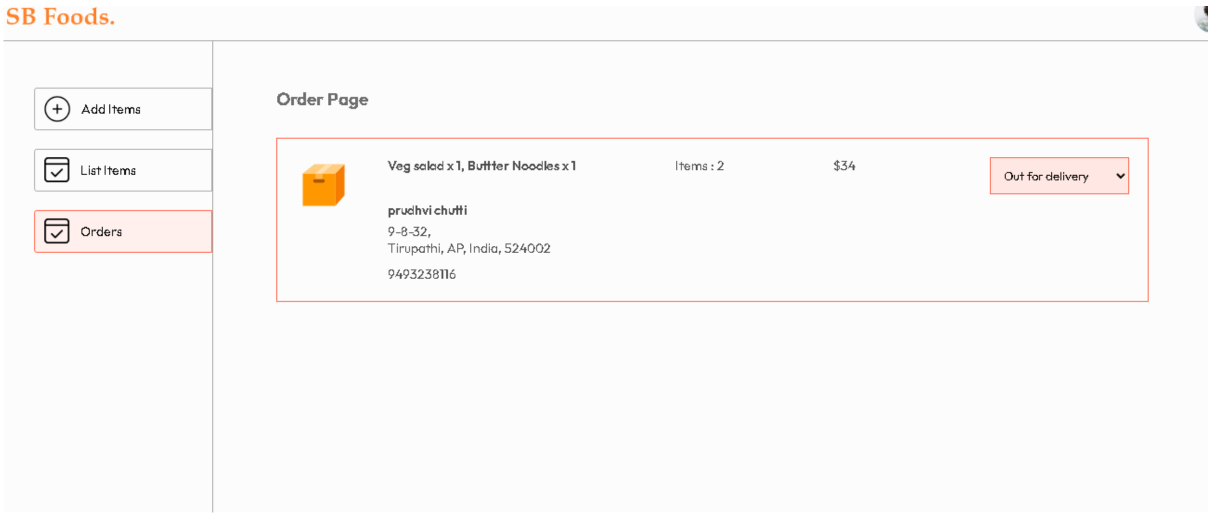
List Items

Orders

All Foods List

Image	Name	Category	Price	Action
	Greek salad	McDonald's	$12	X
	Veg salad	McDonald's	$18	X
	Clover salad	McDonald's	$16	X
	Chicken salad	McDonald's	$24	X
	Lasagna rolls	RollsRise	$14	X
	Peri Peri Rolls	McDonald's	$12	X
	Chicken Rolls	RollsRise	$20	X

Fig : Admin Products Page


SB Foods.

Order Page

Veg salad x 1, Butter Noodles x 1

Items : 2

$34

Out for delivery

prudhvi chutti

9-8-32,
Tirupathi, AP, India, 524002

9493238116

Fig : Admin Orders Page

8. ADVANTAGES & DISADVANTAGES

Advantages of the 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 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.

9. CONCLUSION

The **SBFoods Food Delivery App** project successfully demonstrates the design and implementation of a full-stack online food ordering and delivery system using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The system was developed with the objective of providing a smooth, efficient, and secure digital food ordering experience while maintaining structured architecture and role-based access control.

Throughout the development process, emphasis was placed on clean system architecture, modular coding practices, and efficient database schema design to ensure scalability, maintainability, and security. The separation of frontend, backend, and database layers enhances system clarity and future expandability.

The application enables users to register and log in securely, browse food categories, view detailed menu items, add items to the cart, place orders, and track delivery status efficiently. On the administrative side, the dashboard provides full control over food item management, order processing, delivery status updates, and user account management. JWT-based authentication ensures secure communication between client and server, while middleware-based authorization restricts access to protected resources.

The project demonstrates effective integration between frontend and backend through well-structured RESTful APIs. Proper validation, error handling, and organized folder structure contribute to the reliability and stability of the system. Testing was conducted across different modules to ensure smooth order processing, authentication security, and accurate data management.

Although advanced features such as real-time order tracking with live maps, push notifications, online payment gateway integration (Stripe/Razorpay), performance optimization, and cloud deployment can be implemented in future enhancements, the current version successfully fulfills the core requirements of a modern food delivery platform.

In conclusion, **SBFoods** stands as a scalable and practical food delivery solution that reflects a strong understanding of full-stack web development, secure authentication mechanisms, API design principles, and database management. With further improvements and feature enhancements, it has the potential to evolve into a fully production-ready commercial food delivery platform.

10. FUTURE SCOPE

The **SBFoods Food Delivery App** 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. Online Payment Gateway Integration

The system can be integrated with secure online payment gateways such as credit/debit cards, UPI, net banking, and digital wallets. This will enable real-time payment processing and allow users to complete transactions seamlessly. Integration with platforms like Stripe or Razorpay can make the application 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 such as animated food cards, improved menu filtering, personalized recommendations, dark/light themes, and smoother navigation can significantly improve user engagement and overall experience.

---

3. Real-Time Order Tracking and Notifications

Real-time order tracking can be implemented using WebSockets or similar technologies. Users can receive live updates about order confirmation, food preparation, dispatch, and delivery status. Push notifications and SMS/email alerts can further improve communication between customers and the platform.

---

4. Delivery Management System

An advanced delivery management module can be added to:

- Assign delivery partners automatically
- Track delivery locations using GPS
- Optimize delivery routes
- Monitor delivery performance

This will improve operational efficiency and reduce delivery time.

11. APPENDIX

My Project Source code Files are available at :

https://github.com/ch-prudhvi/Food-Delivery-App/tree/master

My project Demo Video link is available at :

https://drive.google.com/file/d/17b611Jx7IchesBcQkP8KSSN2S59-gzdj/view?usp=sharing

GitHub Repository Link :

https://github.com/padmajakaturi/Shopez-one-stop-shop-for-online-purchases