FULL STACK DEVELOPMENT WITH MERN (NM1042)
ON FOOD ORDERING APPLICATION

A NAAN MUDHALVAN PROJECT REPORT

Submitted By:

ANTONY JEROLD J (412721104005)

INDRAKANDAN N (412721104015)

KIRUBANANDHAN S (412721104021)

SARVESH R (412721104042)

WILFRED AROCKIA RAJ A (412721104059)

TAGORE ENGINEERING COLLEGE, RATHINAMANGALAM – 600 127

ANNA UNIVERSITY::CHENNAI

Abstract

SB Foods is an advanced Food Ordering WebApp built to transform the food ordering experience for customers and restaurants alike. Developed using the MERN stack (MongoDB, Express, React, Node.js), it combines cutting-edge technology with a focus on usability to create a seamless platform for menu exploration, order placement, and operational management.

The platform empowers customers with an intuitive interface for browsing menus, filtering options, and placing orders via a dynamic cart integrated with a secure Razorpay payment gateway. Enhanced features like order tracking, personalized profiles, and purchase history streamline the user experience. Restaurants gain access to a robust admin dashboard for efficient menu management, order processing, and customer engagement.

SB Foods integrates Cloudinary for optimized media storage and retrieval, while security is bolstered by JWT authentication and password hashing, ensuring robust data protection. The platform's scalable architecture makes it an ideal solution for both small businesses and large-scale food service providers.

With plans for future innovation, including AI-powered recommendations, subscription-based models, real-time delivery tracking, and a dedicated mobile app, SB Foods is positioned as a versatile and forward-thinking solution for modern food service needs.

ONLINE FOOD ORDERING APPLICATION

1. Introduction

Project Title: SB Foods - Food Ordering WebApp
Team Members:

Name	Register Number	Program ID
Sarvesh R	412721104042	37BA0272979BC2380FD4CCB08D41488A
Kirubanandhan S	412721104021	B52B710A4913DFF8BC64D708BA9BD226
Wilfred Arockia Raj A	412721104059	533063809077E47B502442F74703D3B5
Antony Jerold J	412721104005	2146CFBFE826A9E9DF54DB95CC7925B0
Indrakandan N	412721104015	782E4DFB206C27E982241AB65D2E00D8

2. Project Overview

Purpose:
SB Foods is a comprehensive Food Ordering WebApp designed to enhance the digital dining experience. It offers customers the ability to explore menus, place orders, and manage profiles seamlessly. Restaurants benefit from an admin dashboard to manage menus, orders, and operations.

Key Features:

1. **User Roles and Access Control**:
   - **Customers**: Browse restaurants, place orders, track order status, and manage profiles.
   - **Restaurants**: Manage menus, process orders, and gain insights into customer activity.
   - **Admin**: Oversee users, restaurants, and platform operations.
2. **Secure Payment Integration**:
   - Razorpay integration for secure and reliable transactions.
3. **Media Storage**:
   - Efficiently store and retrieve images and media assets using Cloudinary.
4. **Order Management**:
   - Customers can track orders and access purchase history.
   - Restaurants can process and update order statuses dynamically.

SB Foods simplifies operations for both customers and restaurants, ensuring a secure, user-friendly, and scalable platform.

3. Architecture

Frontend:

The frontend is built with React to provide a responsive and dynamic interface. Key elements include:

- **Framework**: React with custom CSS and Bootstrap for styling.
- **Routing**: JWT-protected routes secure sensitive pages like profile and admin dashboards.
- **State Management**: React Context API for global state sharing.

Backend:

The backend is powered by Node.js and Express, providing efficient server-side logic. Key components include:

- **Framework**: Express.js for handling REST APIs.
- **Authentication**: JWT tokens and password hashing ensure secure access.
- **Media Management**: Cloudinary handles image and media uploads.
- **Payment Gateway**: Razorpay integration for secure transactions.

Database:

MongoDB Atlas provides flexible and scalable data storage. Key schemas include:

- **User Schema**: Details for customers and admins, including authentication credentials.
- **Restaurant Schema**: Data about restaurant profiles, menus, and operational settings.
- **Order Schema**: Tracks order details, payment information, and delivery status.

4. Setu Instructions

Prerequisites:

- Install Node.js (v14+).
- Install MongoDB (local or MongoDB Atlas).
- Have npm or yarn installed.
- Configure environment variables for:
  - MongoDB connection (MONGO_URI).
  - Cloudinary API credentials (CLOUDINARY_CLOUD_NAME, CLOUDINARY_API_KEY, CLOUDINARY_API_SECRET).
  - Razorpay API keys.
  - JWT secret (JWT_SECRET).

Installation:

1. Clone the repository:
   git clone "https://github.com/example-repo/SB-Foods.git"

2. Navigate to project folders:
   - cd frontend
   - cd backend
3. Install dependencies:
   - npm install
4. Configure environment variables in .env files.

5. Running the Application

Frontend:

Start the React development server:

cd frontend

npm run dev

Backend:

Run the backend server:

cd backend

node index.js

Here's a proposed folder structure for the SB Foods WebApp, ensuring maintainability and scalability:

6. Folder Structure

Frontend

client/

```
├── src/
│   ├── assets/            # Static assets (images, icons, etc.)
│   ├── components/        # Reusable UI components
│   │   ├── Customer/      # Components specific to customer workflows
│   │   ├── Restaurant/    # Components specific to restaurant workflows
│   │   ├── Admin/         # Components specific to admin workflows
│   │   └── Shared/        # Common components (header, footer, modals, etc.)
│   ├── pages/             # Page-specific components
│   │   ├── Home/          # Landing page and restaurant listing
│   │   ├── Cart/          # Cart management page
```

```
│   │   ├── Profile/        # Customer profile and order history
│   │   ├── AdminDashboard/  # Admin dashboard pages
│   │   └── RestaurantDashboard/ # Restaurant dashboard pages
│   ├── services/           # API calls and service functions
│   ├── context/            # Context API for state management
│   ├── utils/              # Utility functions (e.g., formatters, validators)
│   ├── App.js              # Main application component
│   ├── index.js            # Entry point of the React app
│   └── styles/             # CSS and styling files
└── public/
    ├── index.html          # Main HTML file
    └── assets/             # Static assets (shared with the app)
```

Backend

```
server/
├── config/                 # Configuration files
│   ├── db.js               # MongoDB connection setup
│   ├── cloudinary.js       # Cloudinary configuration
│   └── razorpay.js         # Razorpay API setup
├── controllers/            # Business logic for routes
│   ├── authController.js    # Handles authentication logic
│   ├── restaurantController.js # Handles restaurant management logic
│   ├── orderController.js   # Handles order processing logic
│   └── adminController.js   # Handles admin-specific tasks
├── middleware/             # Custom middleware
│   ├── authMiddleware.js    # JWT token validation
│   └── errorHandler.js      # Centralized error handling
├── models/                 # Mongoose models
│   ├── User.js             # User schema
│   ├── Restaurant.js       # Restaurant schema
│   ├── MenuItem.js         # Menu item schema
│   └── Order.js            # Order schema
```

```
├── routes/                # Route definitions
│   ├── authRoutes.js      # Authentication routes
│   ├── restaurantRoutes.js  # Restaurant management routes
│   ├── orderRoutes.js     # Order-related routes
│   └── adminRoutes.js     # Admin routes
├── utils/                 # Utility functions
│   ├── jwtUtils.js        # JWT generation and verification
│   ├── passwordUtils.js   # Password hashing and validation
│   └── validators.js      # Input validation utilities
├── .env                   # Environment variables
├── server.js              # Entry point of the backend server
└── package.json           # Backend dependencies and scripts
```

7. API Documentation

Endpoints:

- /api/auth: Handles user authentication.
- /api/restaurants: Manages restaurant profiles and menus.
- /api/orders: Processes customer orders and tracks status.

Example:
POST /api/auth/login

- **Request Body**: {email, password}
- **Response**: {token, user}

8. Authentication in SB Foods

Authentication in SB Foods is designed to ensure secure access to the platform while managing different user roles (Customers, Restaurants, Admins). Below are the detailed steps and implementation for authentication.

Authentication Workflow

1. **Registration**
   - A new user (Customer, Restaurant Owner, or Admin) registers by providing details such as name, email, password, and role.
   - Passwords are securely hashed using bcrypt before storage in the database.
2. **Login**

- Users log in using their email and password.
- Upon successful authentication, a **JWT (JSON Web Token)** is generated and sent to the client.
- The token contains user information (e.g., ID, role) and is used for subsequent requests.

3. **Token Validation**

- For every protected route, the client must include the token in the **Authorization** header (Bearer <token>).
- The server validates the token using a secret key and extracts the user's information.

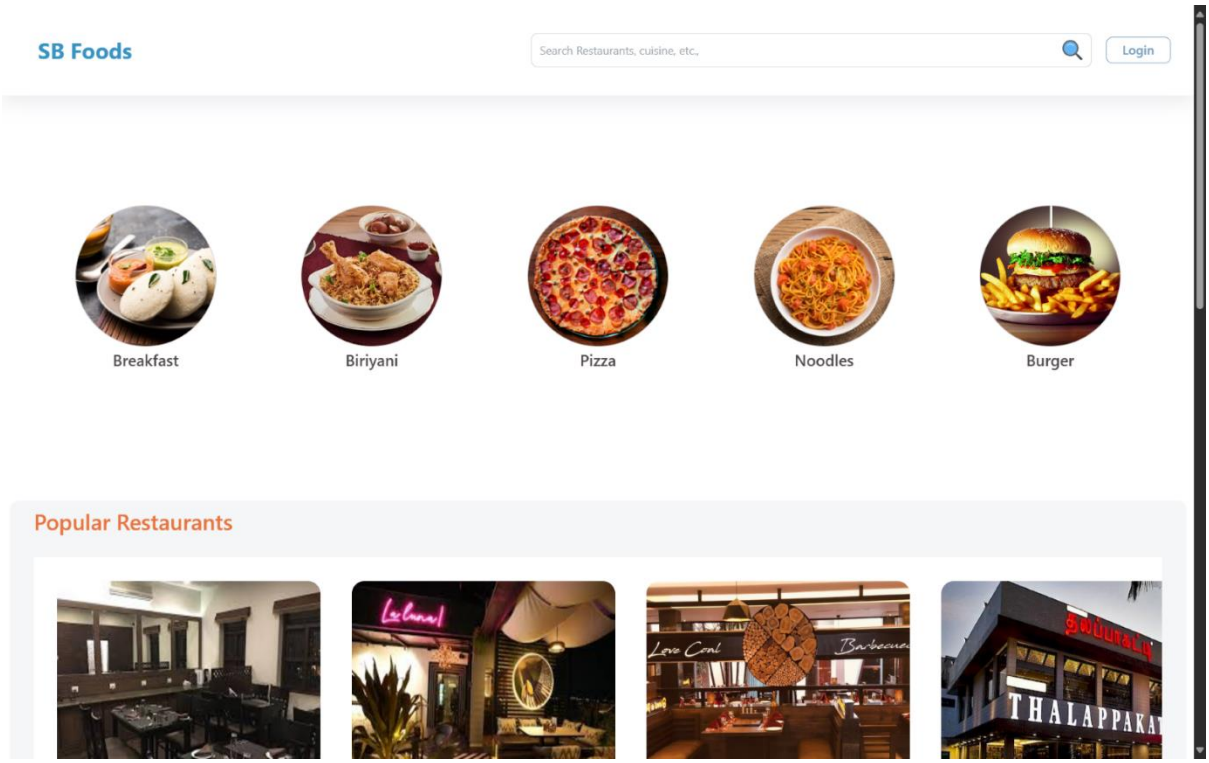
4. **Role-Based Access Control**

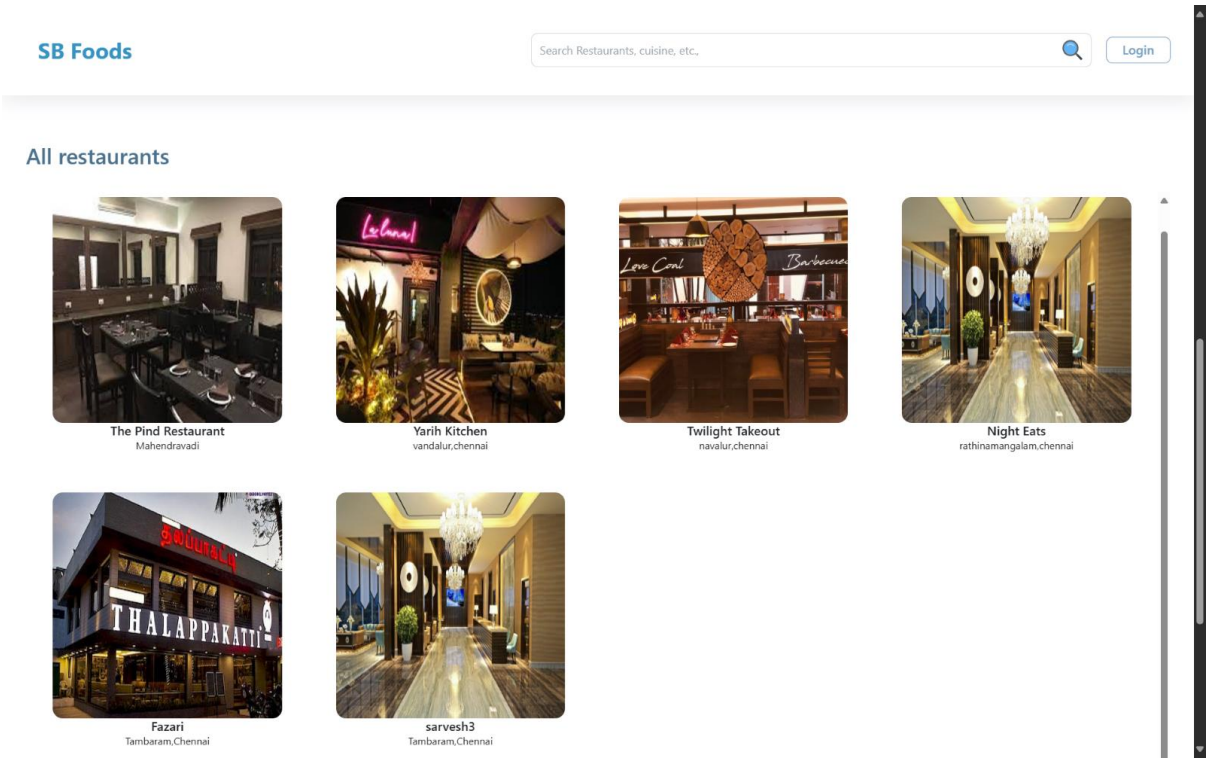
- Users are granted specific permissions based on their roles:

  - **Customers**: Access profile, cart, and order-related functionalities.
  - **Restaurants**: Manage menus, orders, and operational insights.
  - **Admins**: Manage users, restaurants, and platform-wide settings.

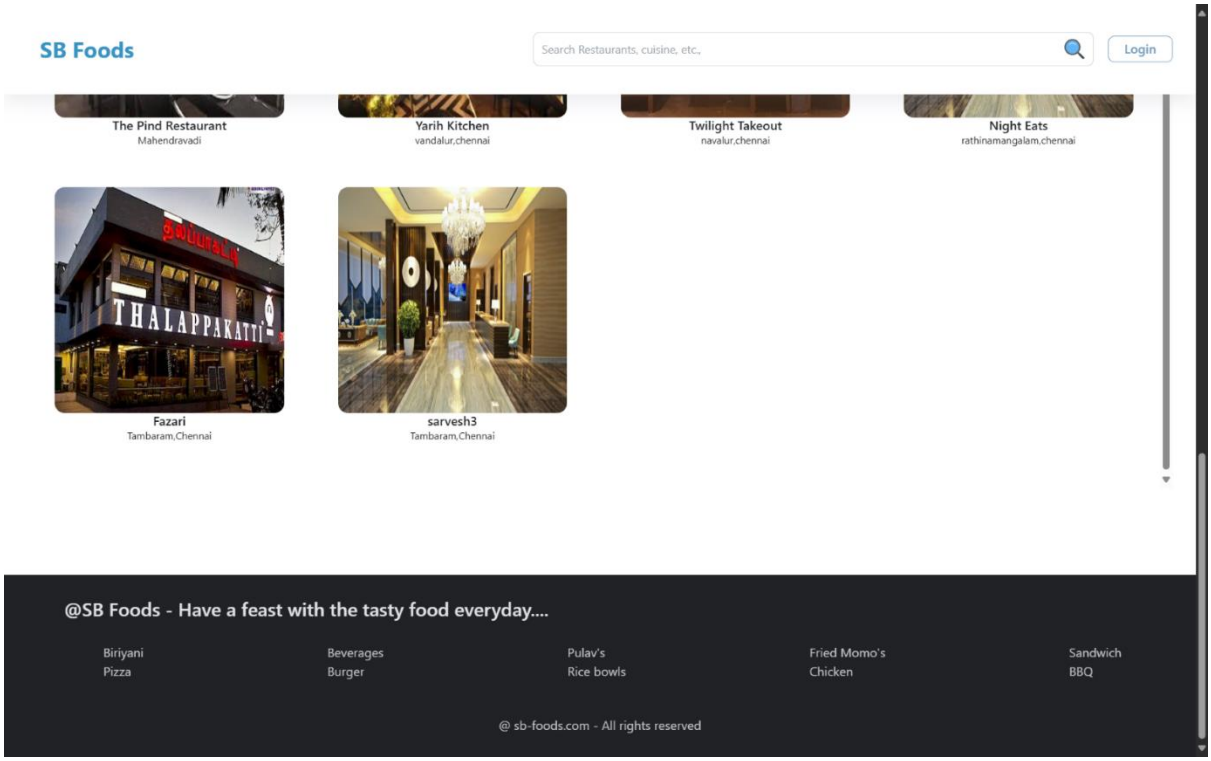
9. User Interface Screenshots

Customer Pages:

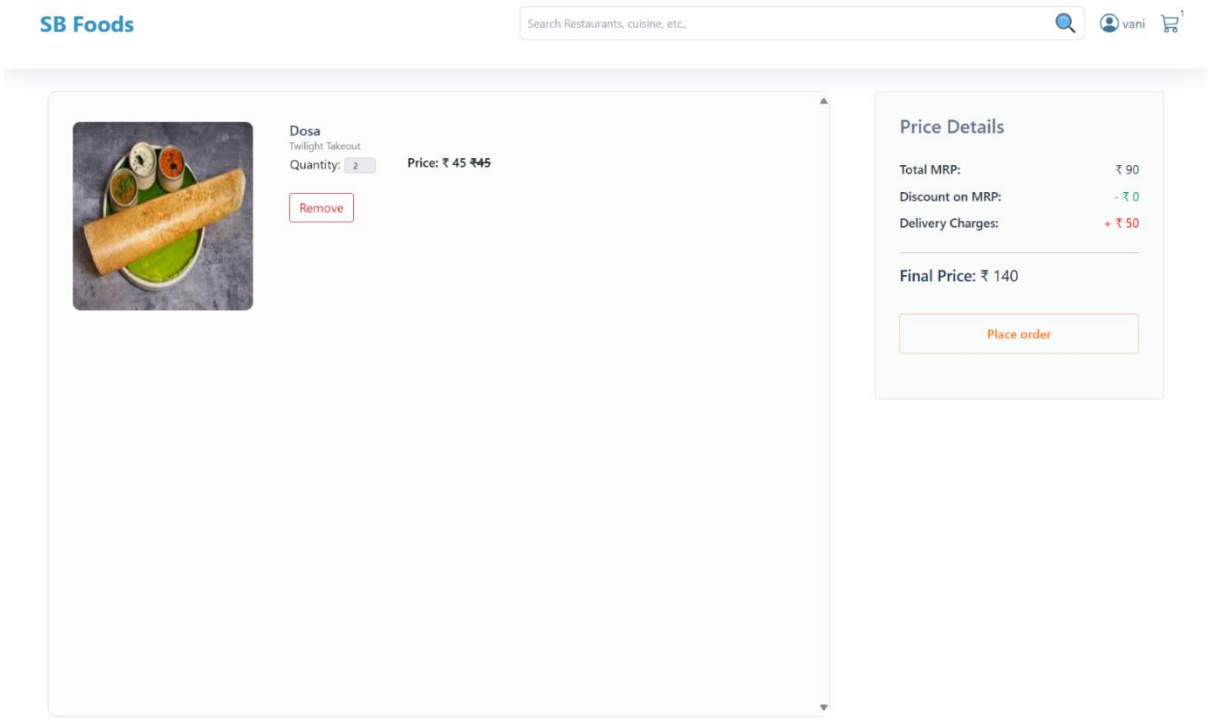




Landing Page

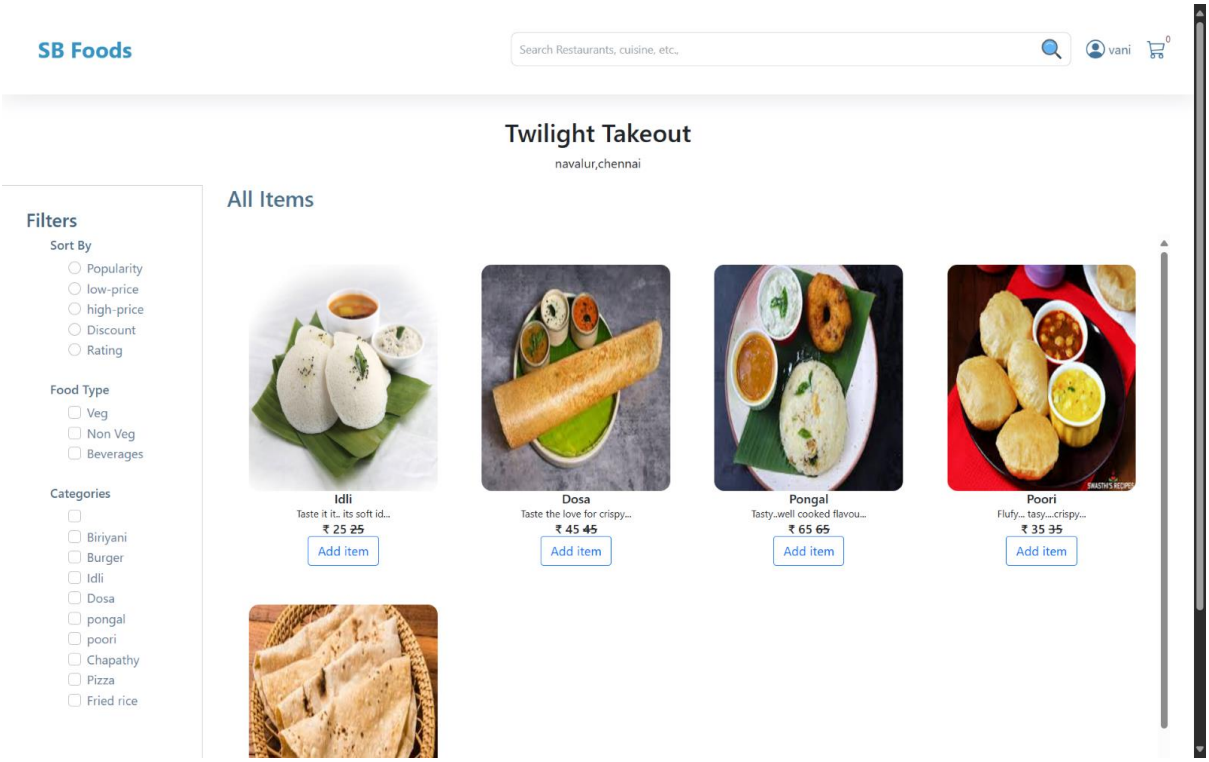


Restaurant and Menu View

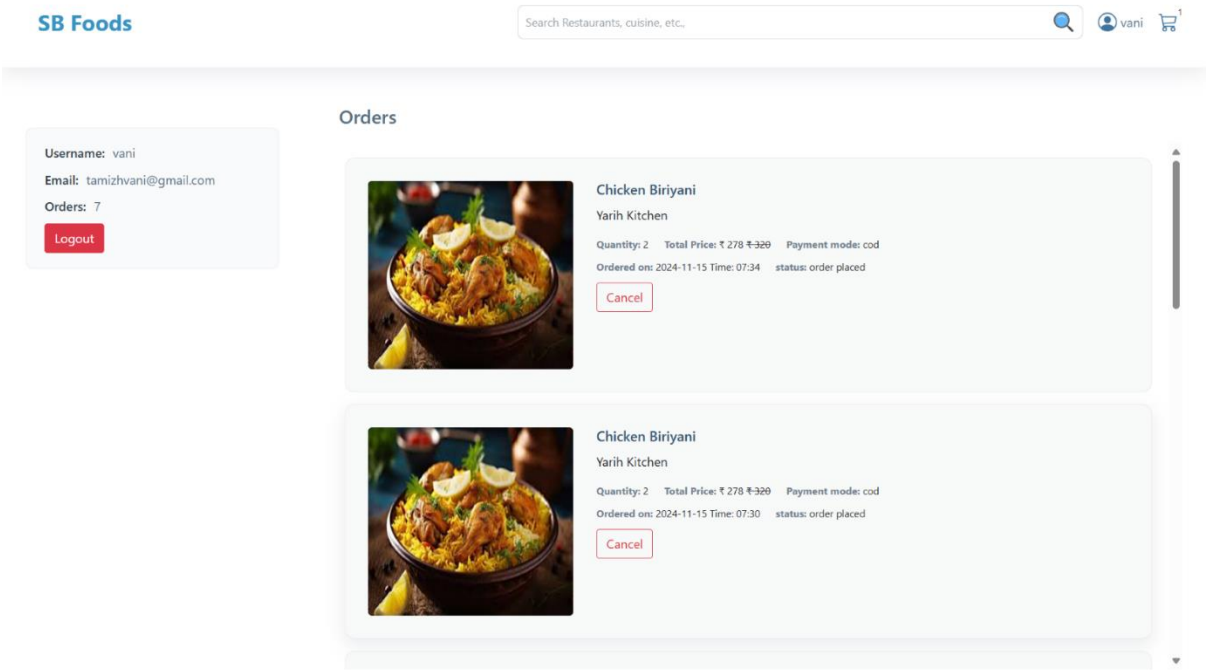


Dynamic Cart and Checkout

Restaurant Pages:

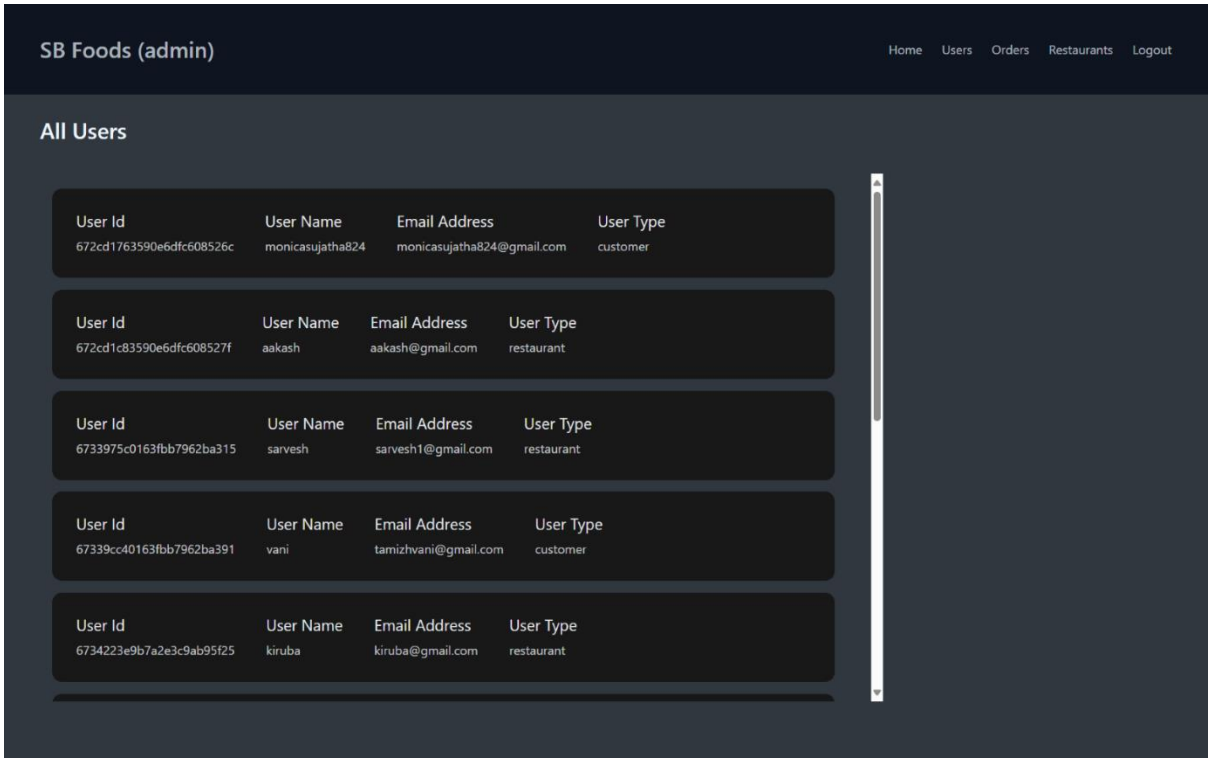


Menu Management

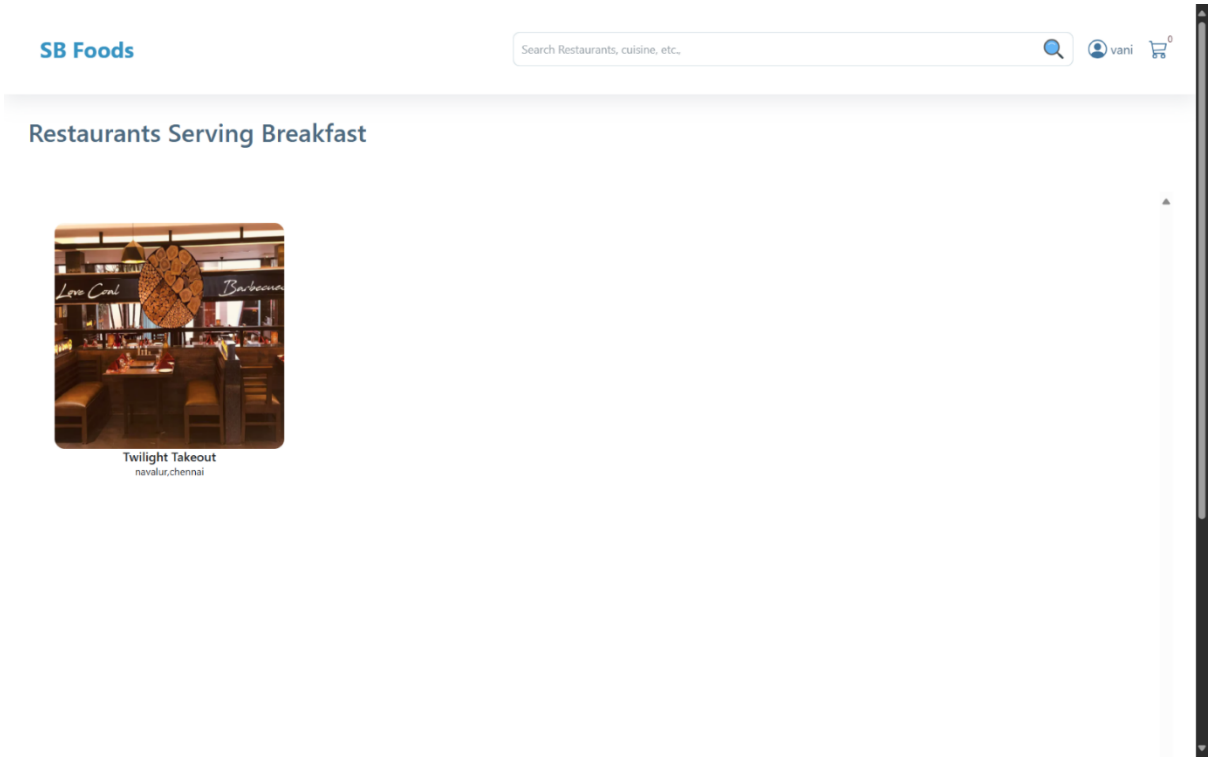


Order Processing Dashboard

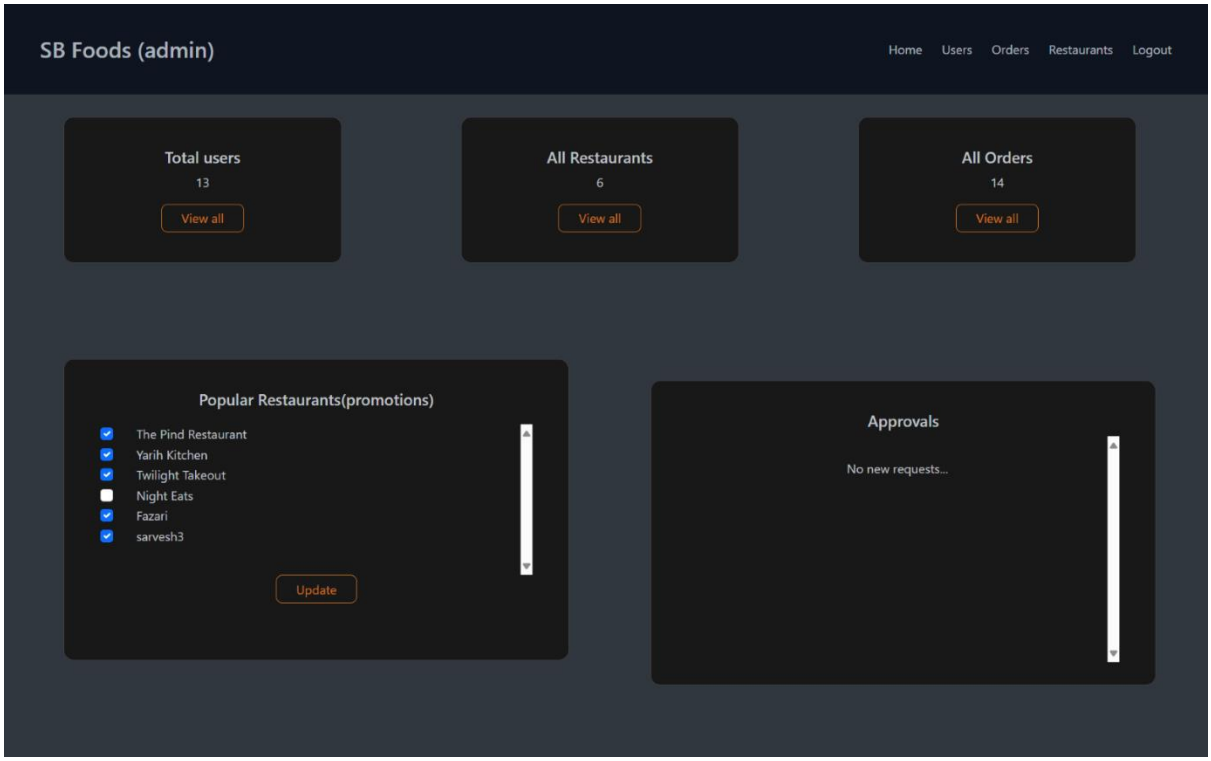
Admin Pages:



User Management



Restaurant Oversight



System Monitoring

10. Testing of SB Foods Application

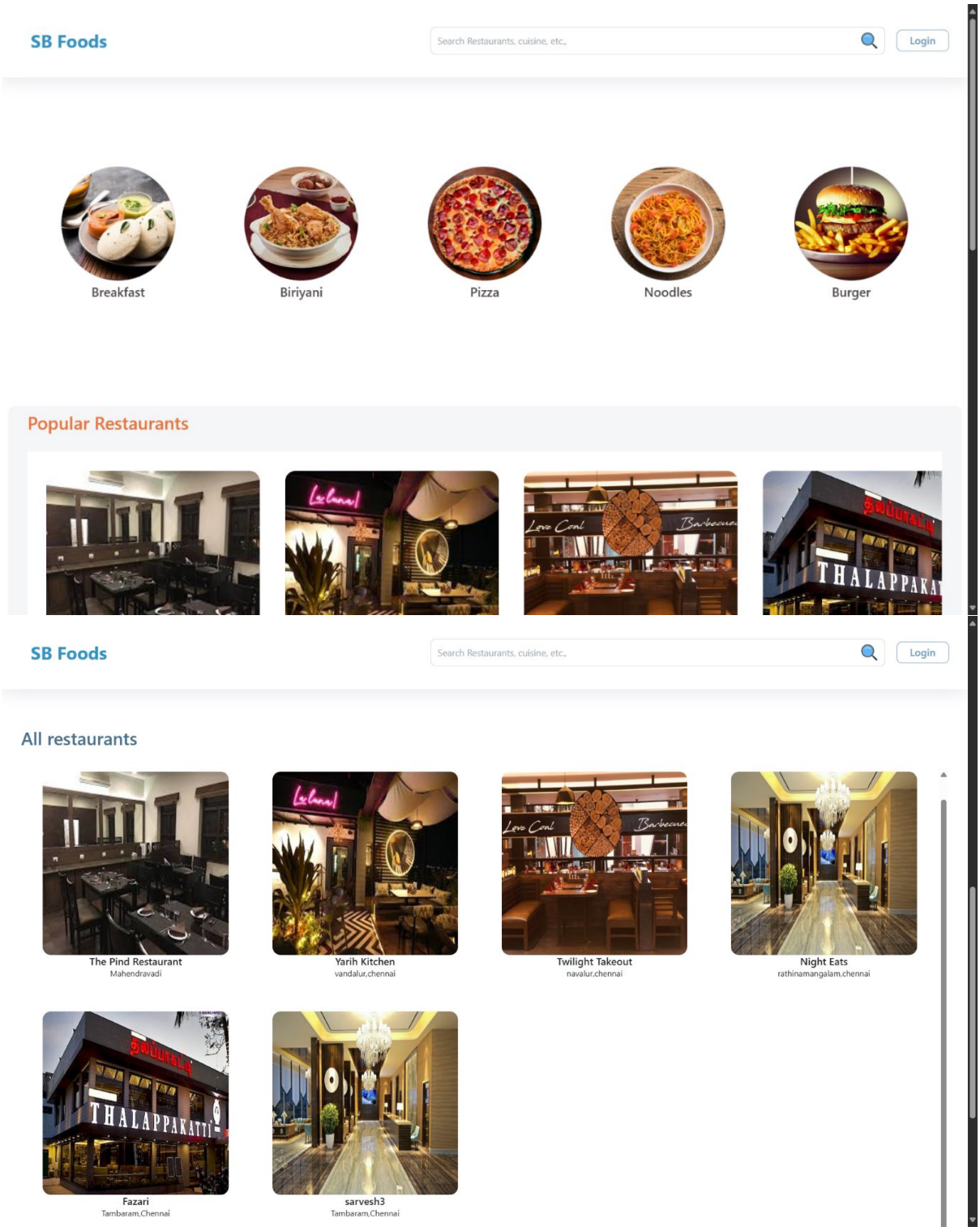
Testing ensures the functionality, reliability, and security of the SB Foods application. The testing process includes **Unit Testing**, **Integration Testing**, **End-to-End Testing**, and **Performance Testing** to verify all aspects of the platform work as intended.

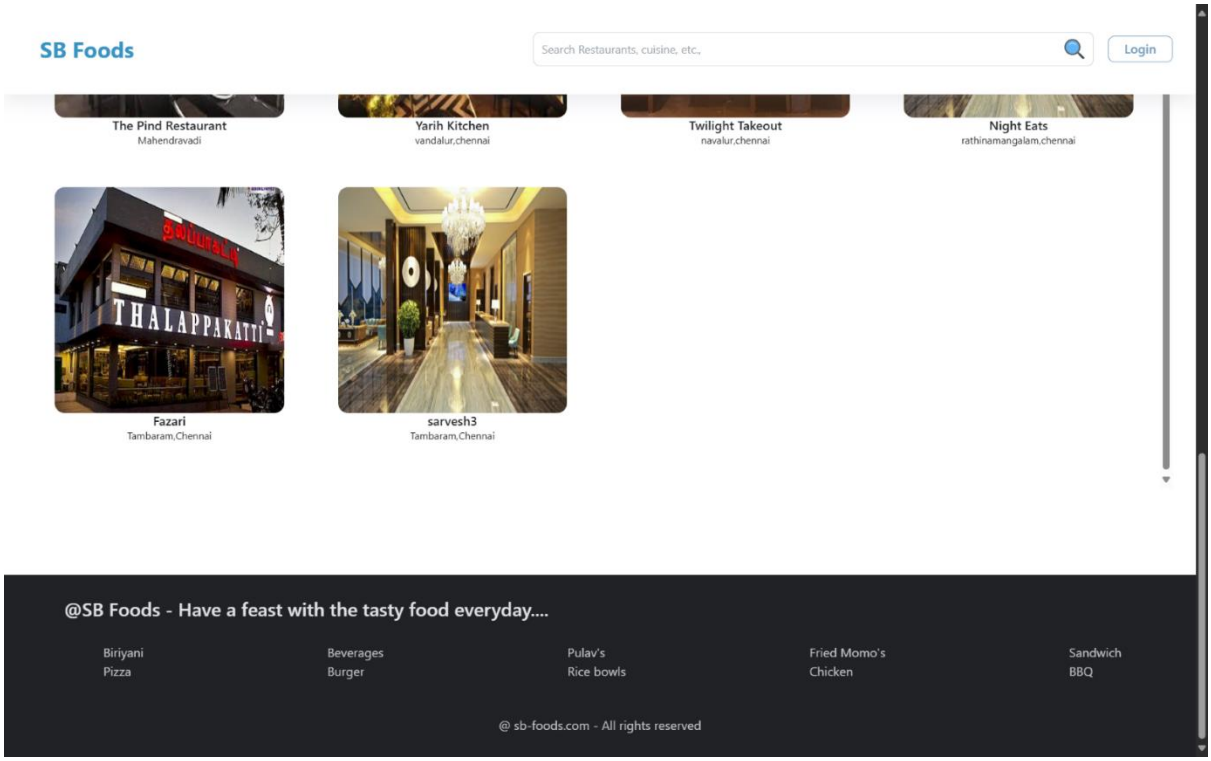
1. Types of Testing

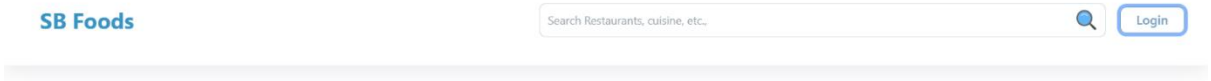
Unit Testing

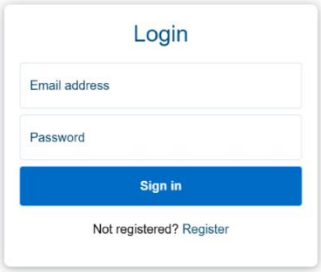
- **Focus**: Tests individual components or functions in isolation.
- **Tools**:
  - **Jest**: For testing backend logic (e.g., API routes, controllers, middleware).
  - **React Testing Library**: For testing React components.
- **Examples**:
  - Verifying that the cart functionality updates prices dynamically.
  - Testing JWT token generation and verification.

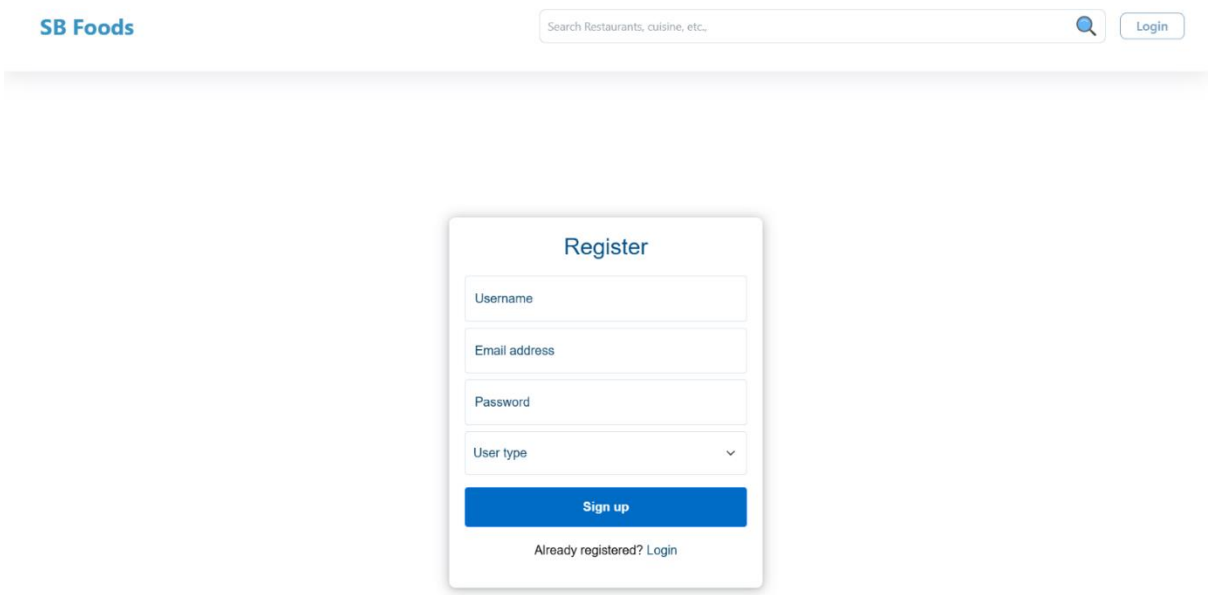
11. Screenshots and Demo

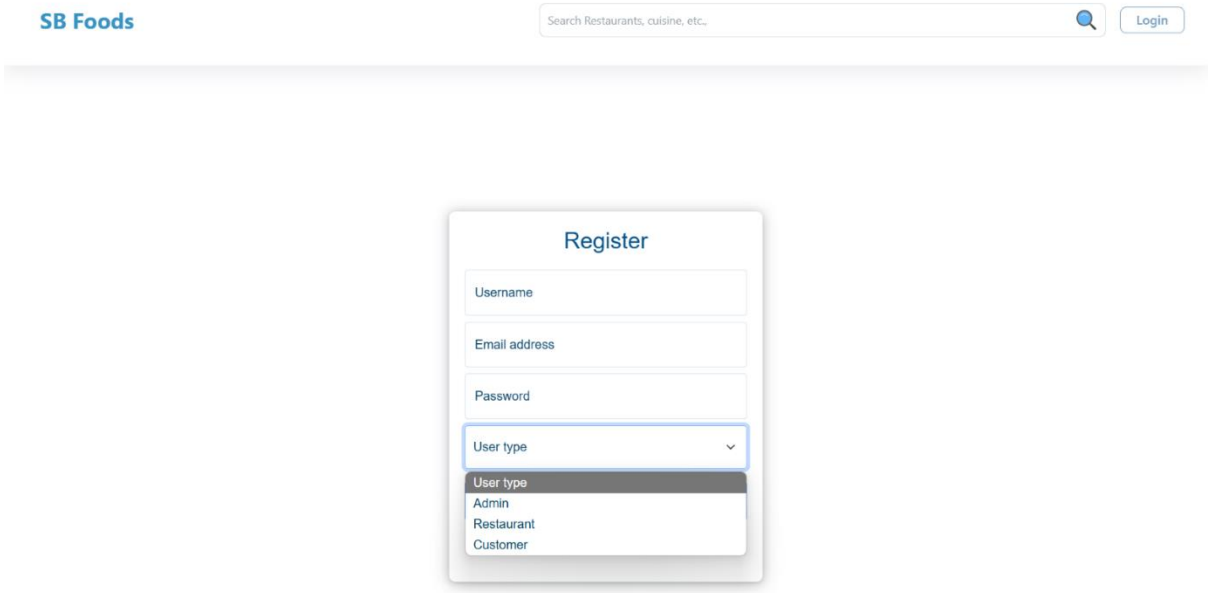


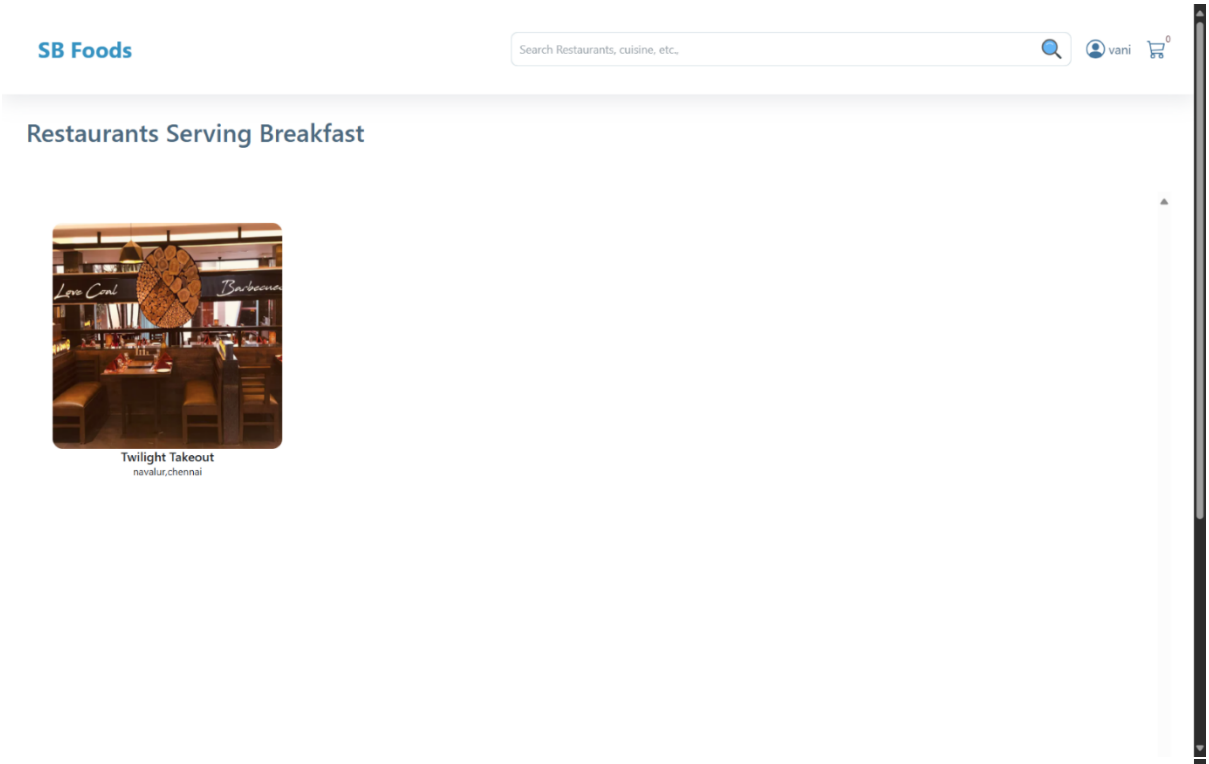

SB Foods
Search Restaurants, cuisine, etc.,
Login
The Pind Restaurant
Mahendravadi
Yarih Kitchen
vandalur,chennai
Twilight Takeout
navalur,chennai
Night Eats
rathinamangalam,chennai
THALAPPAKATTI
Fazari
Tambaram,Chennai
sarvesh3
Tambaram,Chennai
@SB Foods - Have a feast with the tasty food everyday....
Biriyani
Pizza
Beverages
Burger
Pulav's
Rice bowls
Fried Momo's
Chicken
Sandwich
BBQ
@ sb-foods.com - All rights reserved

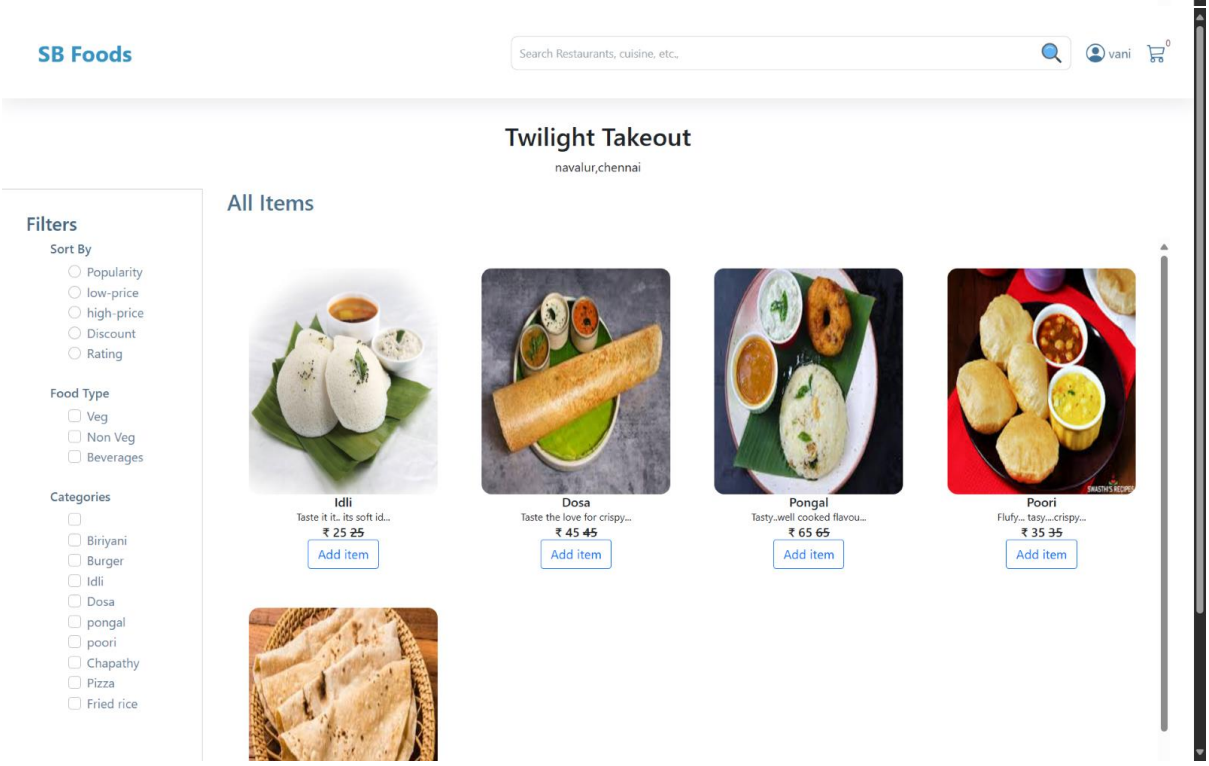

SB Foods
Search Restaurants, cuisine, etc.,
Login


Login
Email address
Password
Sign in
Not registered? Register


SB Foods
Search Restaurants, cuisine, etc.,
Login
Register
Username
Email address
Password
User type
Sign up
Already registered? Login


SB Foods
Search Restaurants, cuisine, etc.,
Login
Register
Username
Email address
Password
User type
User type
Admin
Restaurant
Customer





12. Future Enhancements

1. **AI-Driven Recommendations**: Personalized suggestions for dishes and restaurants.
2. **Subscription Services**: Regular delivery of meal plans.
3. **Real-Time Notifications**: Order updates and promotional offers.
4. **Delivery Integration**: APIs for real-time tracking.
5. **Mobile App**: A dedicated app for improved accessibility.