**Full Stack Development with MERN**

**Project Documentation**

**1. Introduction**

# Project Title: OrderOnTheGo

**Team Members:**

**. Team ID: LTVIP2025TMID24694**

* **Pitta Vasanthi** : (Team Leader) Backend Development, Schemas & Controllers
* **Peddiboina Raju**: Backend Development, Rotes & API Integration
* **Pedasingu.Sai Sushma Sri**: Frontend Development, UI Design & Implementation
* **Pavurayila Bhanu Sai Teja**: Frontend Development, Authentication, Cart & Order Logic

**2. Project Overview**

# Purpose

*OrderOnTheGo* is a food ordering platform built with the **MERN stack**. The goal is to connect users with local restaurants through an easy-to-use, secure platform that allows browsing, ordering, and managing orders. Restaurant owners can manage their menu, and admins can promote restaurants and oversee the platform's operation.

# Features

* User authentication (sign up, login, JWT token)
* Browse restaurants, products, and categories
* Cart management (add/remove items)
* Order placement and tracking
* Restaurant management (add, update, delete products)
* Admin dashboard for restaurant promotions

**3. Architecture**

# Frontend

The frontend is built using **React.js**. It includes:

* **React Router** for dynamic page routing.
* **Redux** (optional) for state management.
* Custom components for user interface design, including reusable components like the header, footer, cart, and product cards.

# Backend

The backend uses **Node.js** and **Express.js** to handle:

* API routes for user authentication, cart management, and order processing.
* Role-based access control (user, restaurant, admin).
* JWT token authentication for secure communication.

# Database

The database uses **MongoDB** for storing:

* **Users** (authentication data)
* **Restaurants** (details, products, orders)
* **Orders** (order status, payment tracking)

**4. Setup Instructions**

# Prerequisites

* **Node.js**: Version 14.x or higher
* **MongoDB**: Installed locally or using MongoDB Atlas
* **npm**: Node package manager

# Installation

1. **Clone the repository**: git clone https://github.com/your-repo/OrderOnTheGo.git
2. **Install Dependencies**:
   * For **Frontend** (React):
   * cd client o npm install o For **Backend** (Node.js, Express):
   * cd server o npm install
3. **Set up environment variables**:
   * Create .env files in both client and server directories:
     + **Frontend**: Set the API base URL (REACT\_APP\_API\_URL)
     + **Backend**: Set MongoDB URI (MONGO\_URI), JWT secret (JWT\_SECRET)
4. **Start the Application**:
   * For **Frontend**:
   * cd client
   * npm start o For **Backend**:
   * cd server o npm start

1. **Folder Structure**

# Client

• client/: React.js frontend folder.

o s/

* components/: Reusable UI components (e.g., Header, Footer)
* pages/: React components for different routes (e.g., Home, Cart, Orders)
* redux/: Optional state management files (if Redux is used)

# Server

• server/: Node.js backend folder.

* controllers/: Logic to handle API requests. o models/: MongoDB schema models.
* routes/: API routes (e.g., user, restaurant, order). o middleware/: Custom middleware (e.g., authentication, error handling).
* config/: Environment variables and database connection.

# 6. Running the Application

To start the application locally, follow these commands:

* **Frontend**:
* cd client
* npm start
* **Backend**:
* cd server
* npm start

**7. API Documentation**

# User Endpoints

* **POST /api /register**: Register a new user
* **POST /api /login**: Log in and receive JWT token
* **GET /api /profile**: Get user details (Protected route)

**Restaurant Endpoints**

* **POST /api/restaurants/login**: Restaurant login
* **GET /api/restaurants/:id/products**: Get products for a specific restaurant
* **POST /api/restaurants/:id/products**: Add a product

**Order Endpoints**

* **POST /api/orders**: Create a new order
* **GET /api/orders/:id**: Get order details

# Admin Endpoints

• **POST /api/admin/promote/:id**: Promote restaurant to homepage

# 8. Authentication

Authentication in *OrderOnTheGo* is handled using **bcrypt** for secure password hashing and **React Context API** for client-side authentication state management.

* **Password Handling**:
  + User passwords are hashed using bcrypt before being stored in the database.
  + During login, the hashed password is compared with the entered password securely.
* **Session Management**:
  + Once a user logs in, their information (like user ID, role, etc.) is stored in React’s **Auth Context**.
  + This context is used across the app to control access to protected routes and display appropriate UI components (e.g., user vs. restaurant header).
* **Role-Based Access**:
  + React Context tracks user roles (user, restaurant, admin) to ensure each role sees the correct dashboard and has access to relevant features.

# 9. User Interface

Screenshots showcasing key UI elements:

* **Homepage**: Displays restaurant categories and popular restaurants.
* **Cart Page**: Lists added items with the option to update and checkout.
* **Restaurant Dashboard**: Allows restaurant owners to manage their products.

**10. Testing**

# Testing Strategy

* **Unit Tests**: Tests for backend API routes, models, and controllers.
* **Integration Tests**: Full-stack testing (frontend to backend communication).
* **Manual Testing**: Functional testing of user flows such as login, cart updates, and order placement.

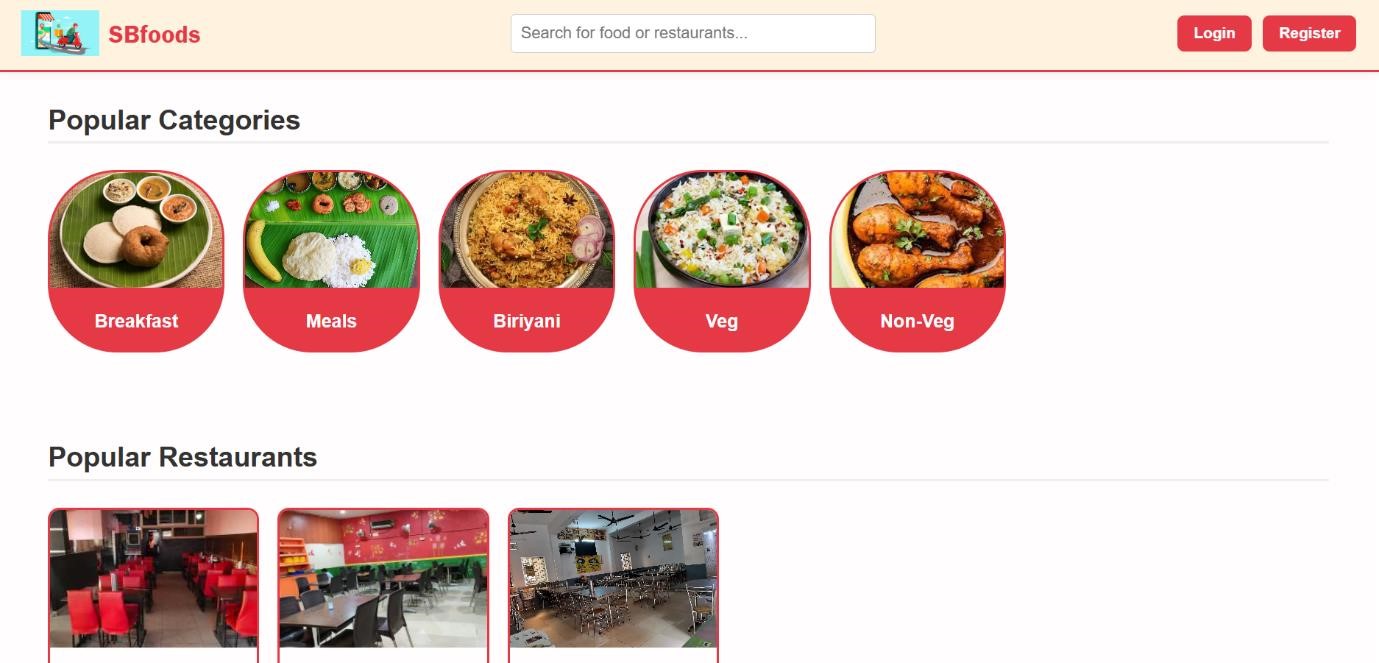
# Tools Used

* **Jest** for unit testing
* **Supertest** for API testing

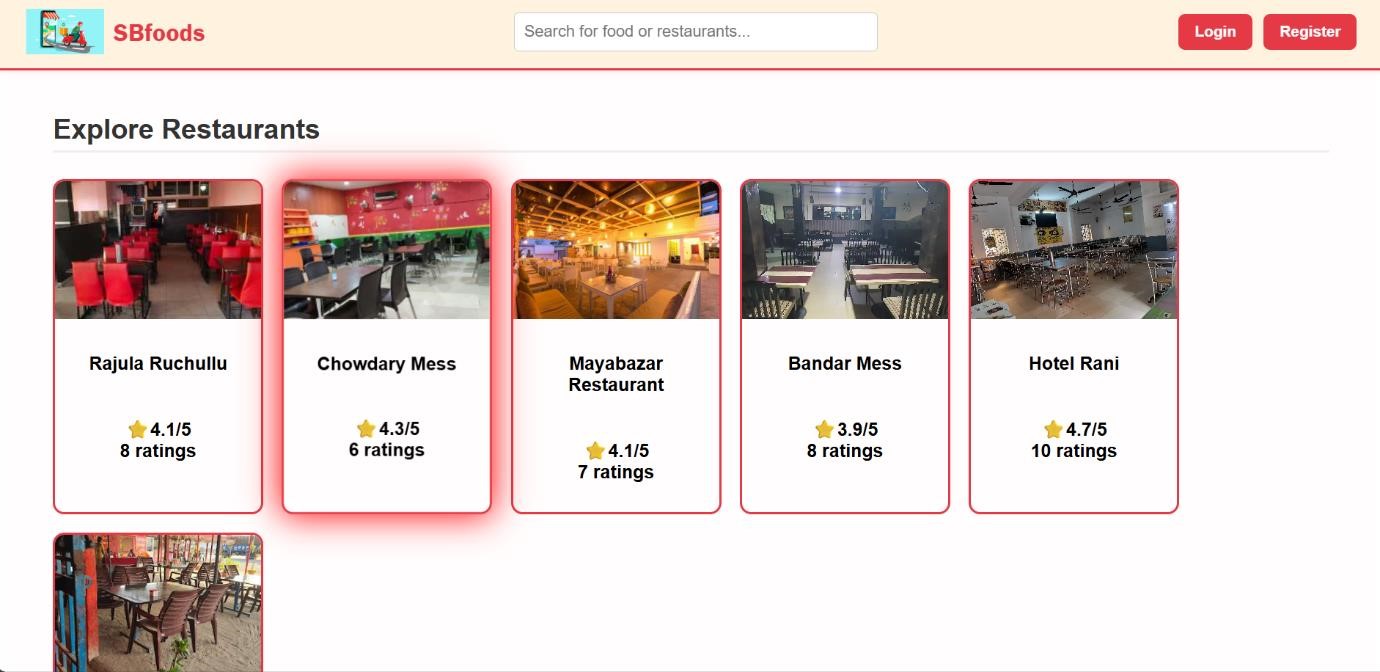
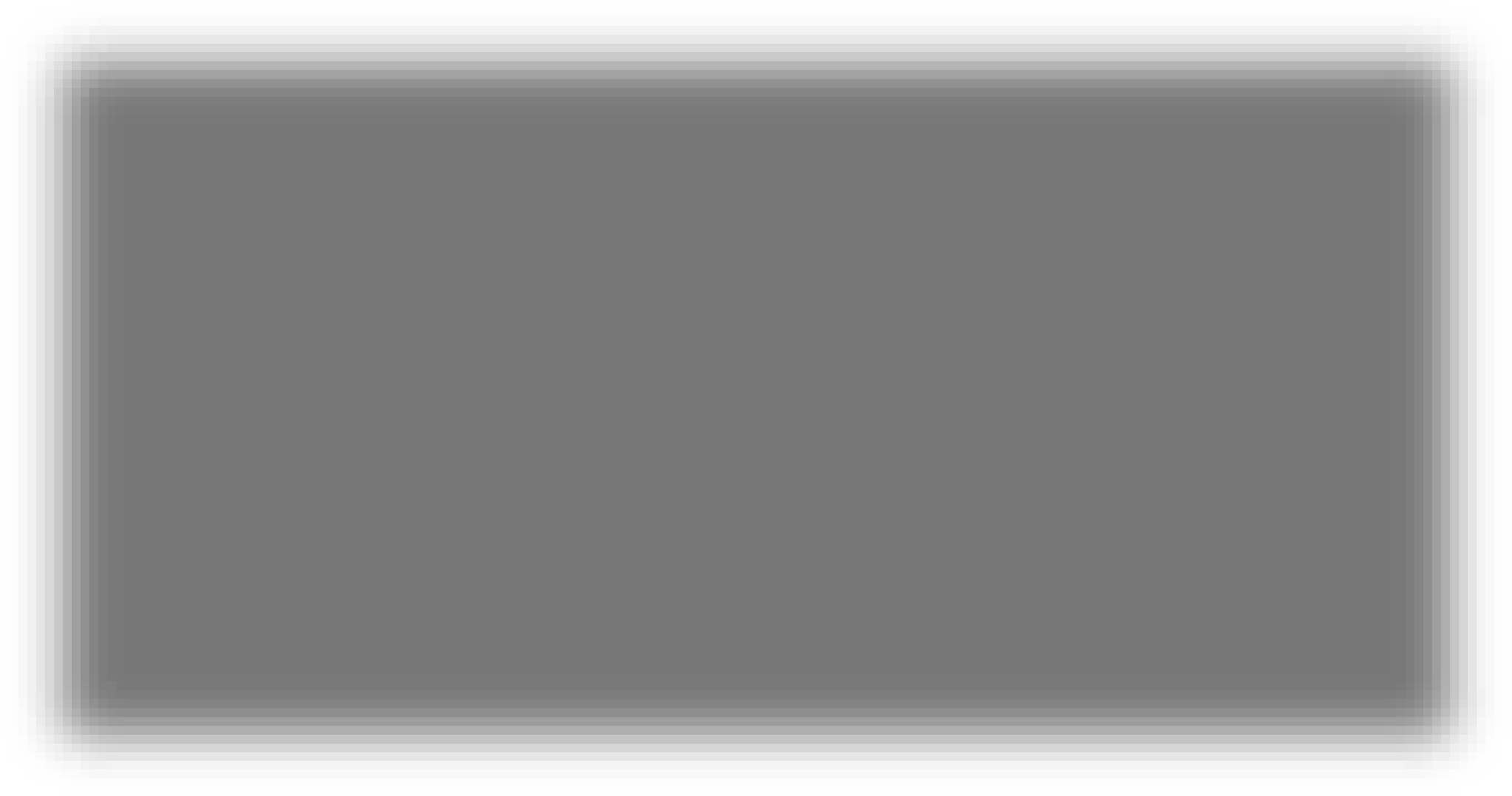
# 11. Screenshots

• **Screenshots**:

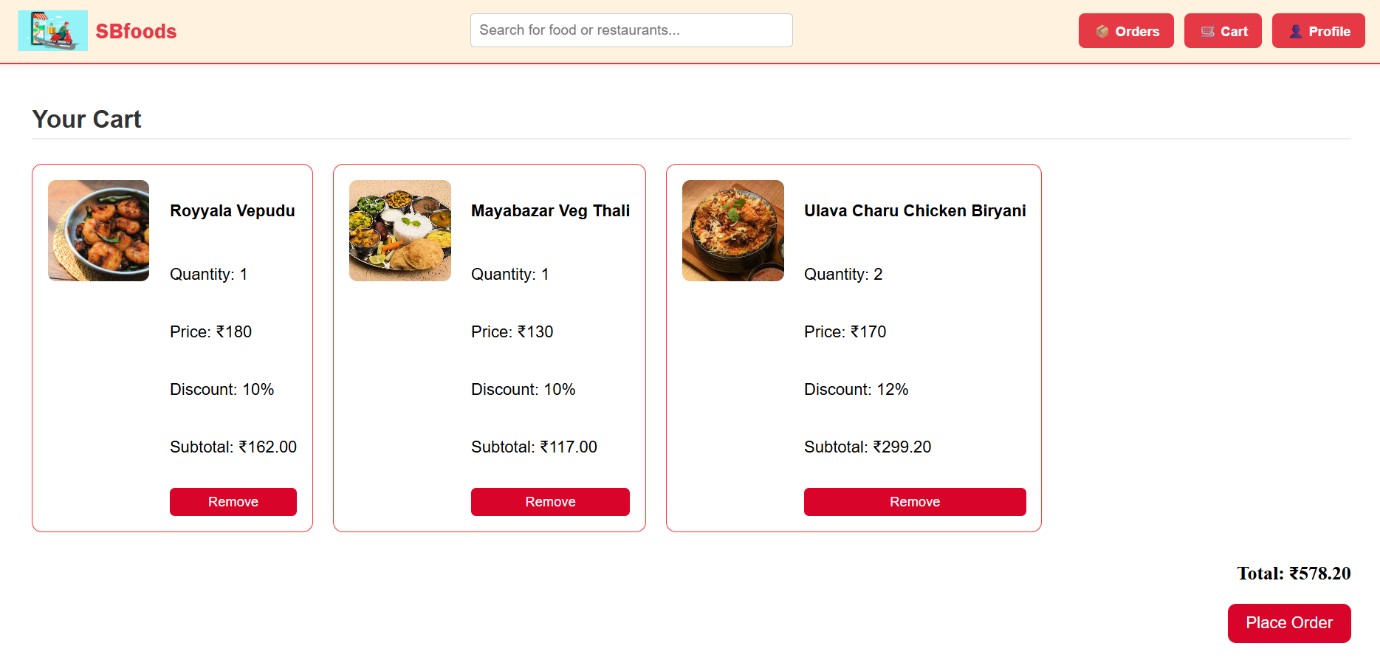
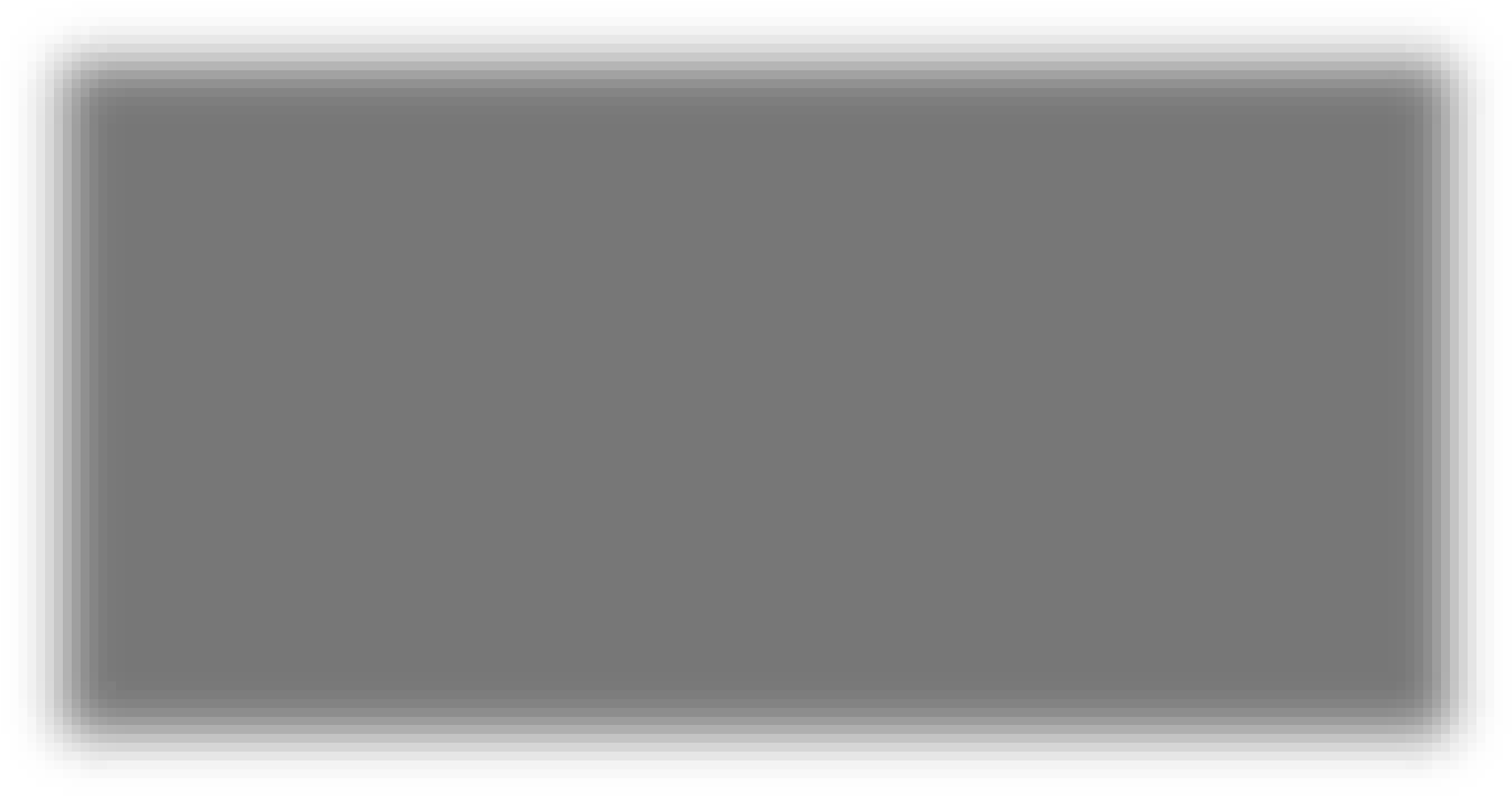
*Homepage*:



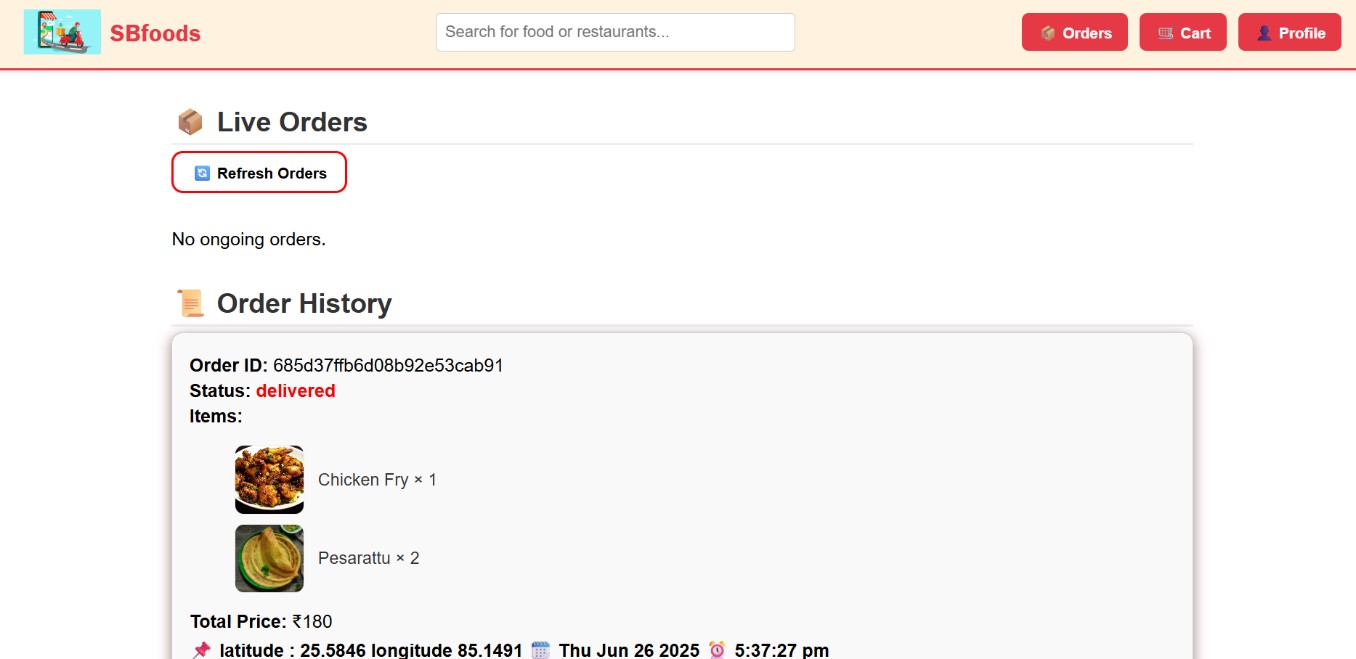
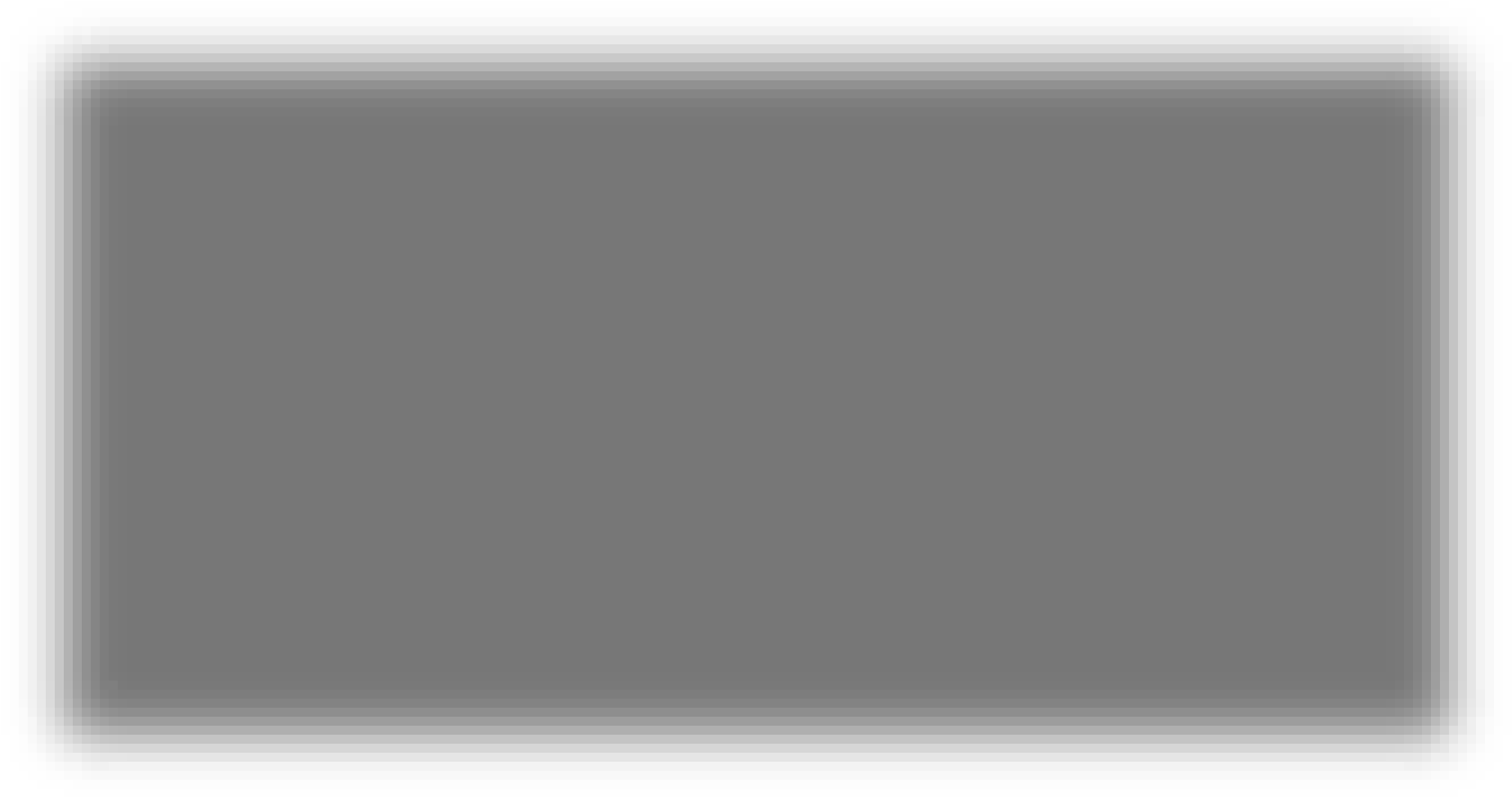
*Restaurants:*



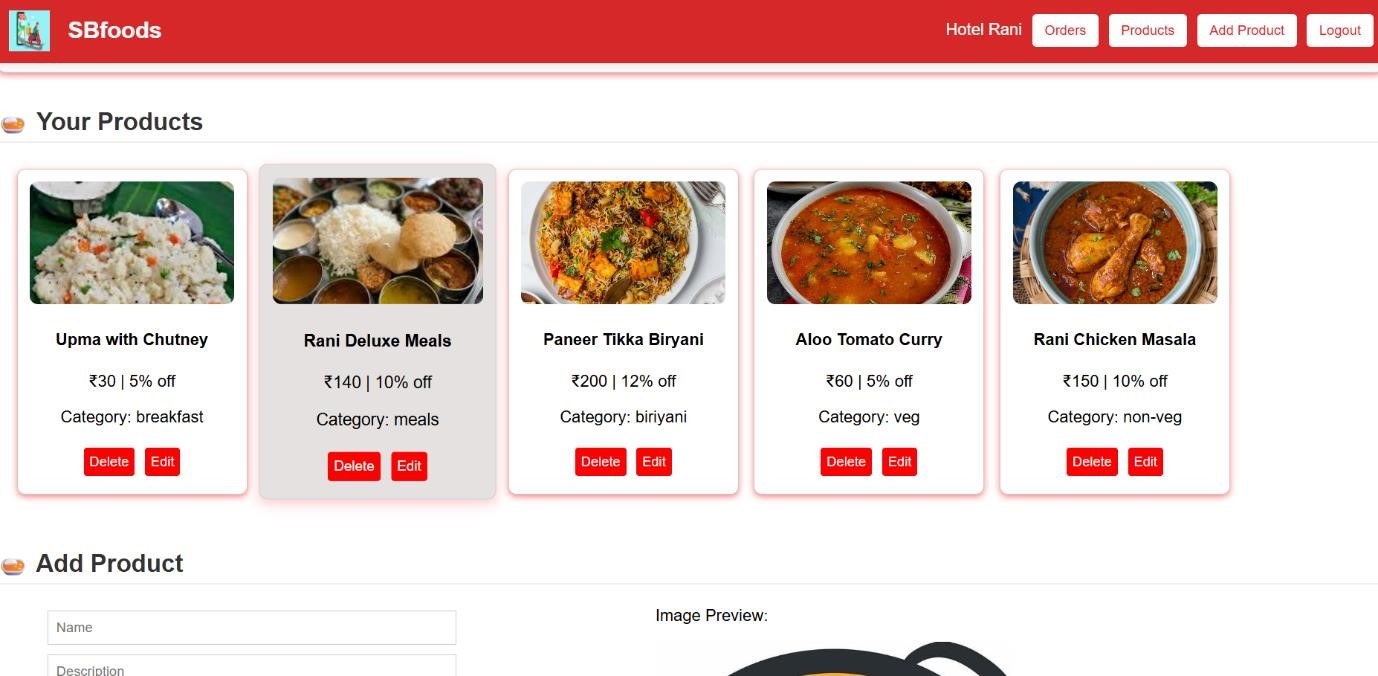
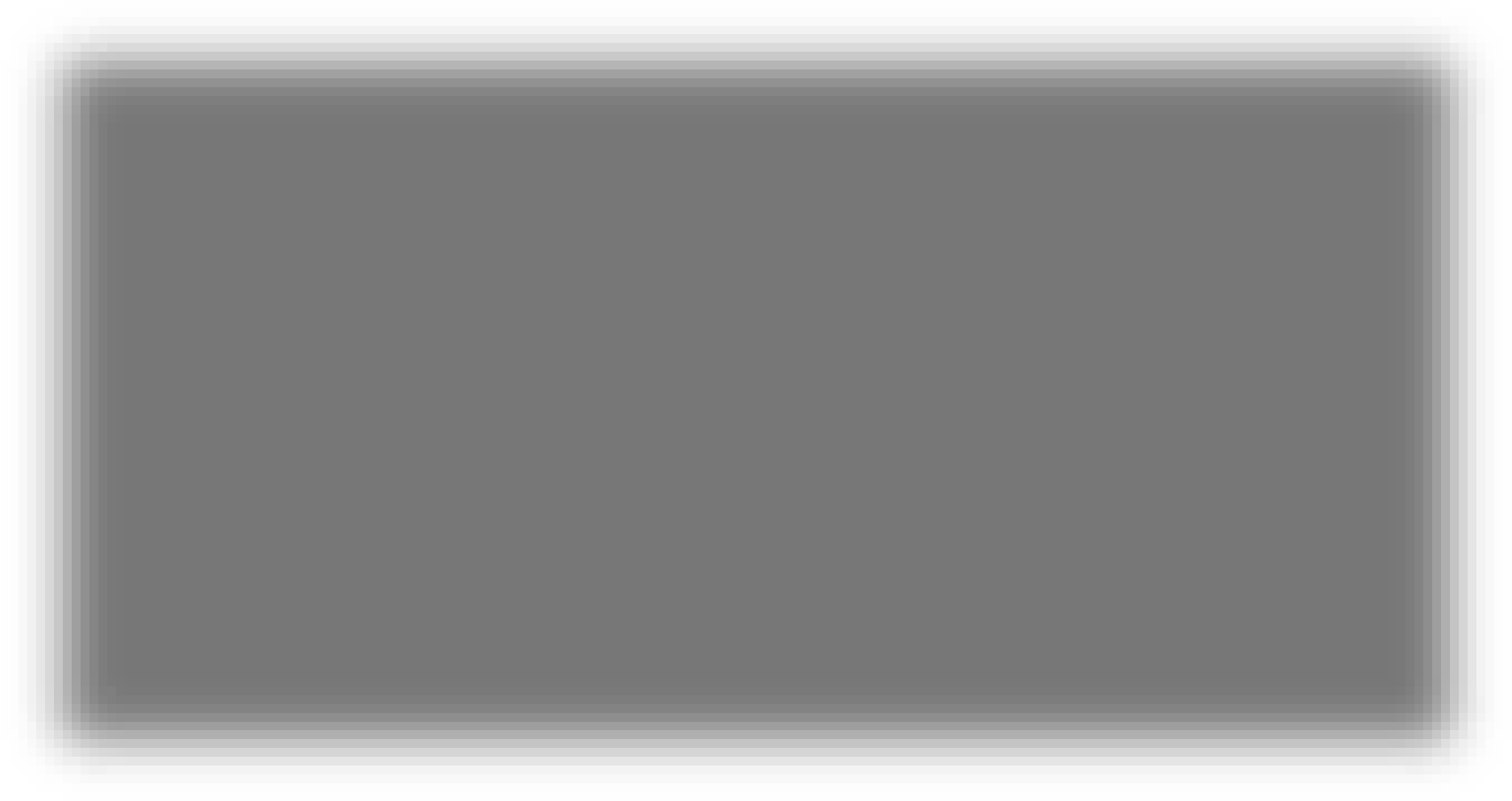
*Cart*:



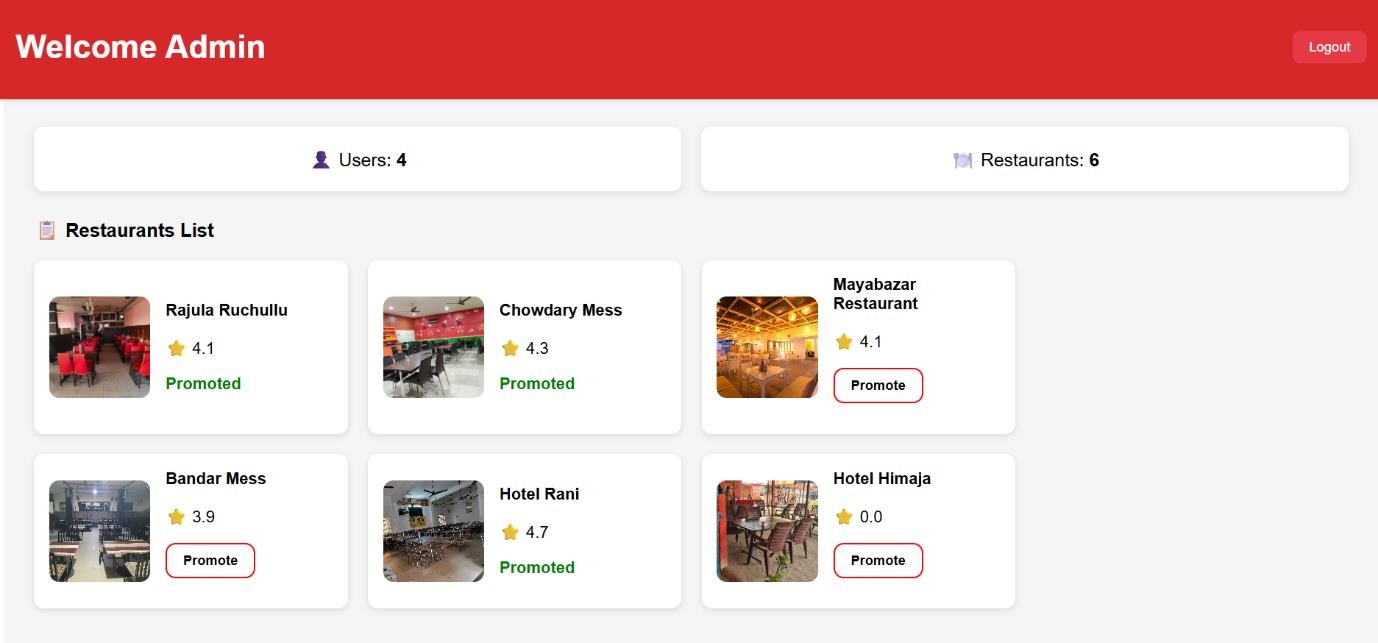
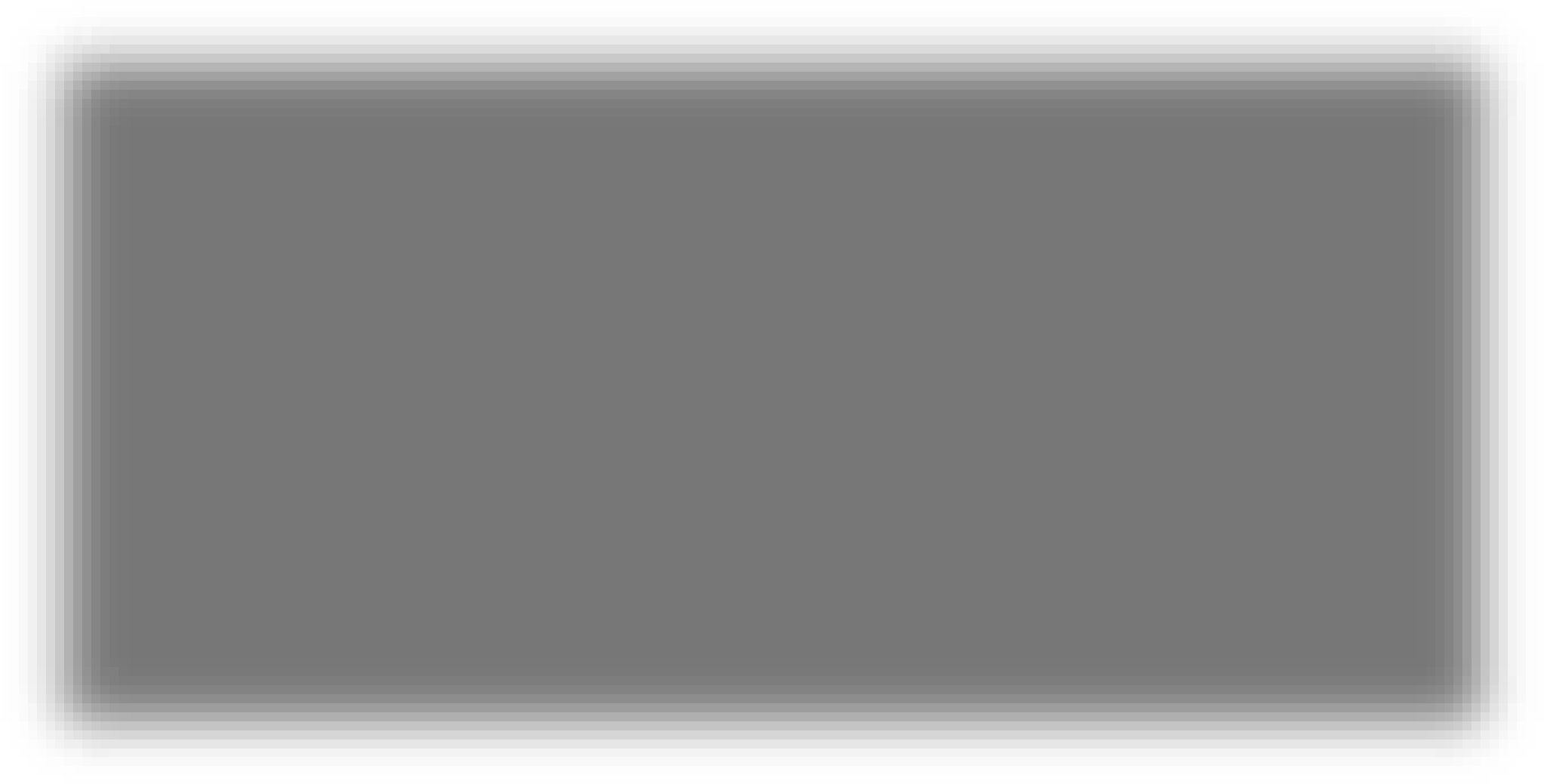
*Orders:*



*Restaurant Dashboard:*



*Admin Dashboard*:



# 12. Known Issues

* **Payment Integration**: Not implemented yet, placeholder functionality.
* **Mobile App**: Currently no mobile app version, only web-based.

# 13. Future Enhancements

* **Payment Gateway**: Integrate with a real payment provider (e.g., Stripe, Razorpay).
* **Mobile App**: Develop a mobile version using **React Native**.
* **Notifications**: Implement order status notifications for users and restaurants.
* **Analytics Dashboard**: Add real-time analytics for restaurants and admins.