**Full Stack Development with MERN**

**1. Introduction**

**Project Title:** **SB Foods - Food Ordering App**

**Team Members:**

* Manikandan A – Full-stack Developer
* Maharajan P – Full-stack Developer
* Srinath R – UI Designer
* Vasanth B – Technical Lead

**2. Project Overview**

**Purpose :** Introducing SB Foods, a revolutionary platform designed to transform your food ordering experience. This user-friendly app allows food enthusiasts to explore, discover, and order meals effortlessly. With detailed dish descriptions, customer reviews, pricing, and promotions at your fingertips, SB Foods ensures informed choices every time. Simply select your dishes, provide delivery details, and enjoy instant order confirmation – making your food journey seamless and efficient.

Features of SB Foods:

* **User Registration and Login:** Secure access with JWT authentication.
* **Explore and Order**: Discover meals with detailed descriptions, reviews, and promotions.
* **Personalized Dashboard**: Track past orders and preferences.
* **Admin Panel:** Manage menu items, offers, and customer data.
* **Real-Time Updates**: Instant order confirmations and live order tracking.

**3.Architecture of SB Foods**

* **Frontend:** Developed with React, the frontend delivers a dynamic and intuitive user interface. It includes components for user authentication, menu browsing, and order placement, using React Router for smooth navigation.
* **Backend:** Powered by Node.js and Express.js, the backend manages API requests, handles secure authentication, and interfaces with MongoDB for data storage. Controllers oversee menu data, user orders, and interactions.
* **Database**: MongoDB serves as the database, storing information about users, menu items, orders, and promotions. Collections are organized for efficient data retrieval and updates, ensuring seamless functionality.

**4. Setup Instructions**

**Prerequisites:**

* Node.js v14+
* MongoDB v4+
* (Optional) npm or yarn for package management

**Installation:**

1. Clone the repository: git clone <repository-url>
2. Navigate to both the backend and frontend directories and install dependencies:

For Backend: cd backend

npm install

For Frontend: cd ../frontend

npm install

1. Set up environment variables by creating a .env file in the backend directory with database connection strings, JWT secret, etc.

**5. Folder Structure**

**Client:**  
The frontend directory contains:

* src/components - Contains reusable React components.
* src/pages - Different pages (e.g., CourseList, Login, Register).
* src/redux – Handel the state data
* src/styles - CSS files for styling.

**Server:**  
The backend directory is organized as follows:

* index.js - Database connection configuration , Functions to handle business logic, Defines routes for various API endpoints and Defines routes for various API endpoints.
* schemas - MongoDB schemas and models.

**6. Running the Application**

**Frontend:**  
Start the frontend server: npm start

**Backend:**  
Start the backend server: npm start

**7. API Documentation**

**Authentication**

1. **POST /register** :Register a new user or restaurant.
2. **POST /login** : Login for users.

**Admin**

1. **POST /update-promote-list :** Update the promoted restaurant list.
2. **POST /approve-user** : Approve a restaurant.
3. **POST /reject-user** : Reject a restaurant.
4. **GET /fetch-categories** : Fetch all categories.
5. **GET /fetch-promoted-list** : Fetch the list of promoted restaurants.

**User Management**

1. **GET /fetch-user-details/:id** : Fetch user details by user ID.
2. **GET /fetch-users** : Fetch all users.

**Restaurant Management**

1. **GET /fetch-restaurants** : Fetch all restaurants.
2. **GET /fetch-restaurant-details/:id** : Fetch restaurant details by owner ID.
3. **GET /fetch-restaurant/:id** : Fetch restaurant details by restaurant ID.

**Food Items**

1. **GET /fetch-items** : Fetch all food items.
2. **GET /fetch-item-details/:id** : Fetch details of a specific food item.
3. **POST /add-new-product** : Add a new food product.
4. **PUT /update-product/:id** : Update an existing product by its ID.

**Orders**

1. **GET /fetch-orders** : Fetch all orders.
2. **PUT /cancel-order** : Cancel an order.
3. **PUT /update-order-status** : Update the status of an order.

**Cart**

1. **GET /fetch-cart** : Fetch all cart items.
2. **POST /add-to-cart** : Add an item to the cart.
3. **PUT /remove-item** : Remove an item from the cart.
4. **POST /place-cart-order** : Place an order from cart items.

**Example Response:**

{

"status": "success",

"data": {

"restaurants": [

{

"id": "restaurant-id-1",

"title": "Delicious Bites",

"address": "123 Food Street, Flavor Town",

"mainImg": "https://example.com/images/restaurant1.jpg",

"menu": [

{

"title": "Spaghetti Carbonara",

"description": "Classic Italian pasta dish with a creamy sauce.",

"itemImg": "https://example.com/images/carbonara.jpg",

"category": "Non-Veg",

"menuCategory": "Main Course",

"price": 12.99,

"discount": 2.0,

"rating": 4.5

},

{

"title": "Vegan Buddha Bowl",

"description": "A mix of fresh veggies, quinoa, and tahini sauce.",

"itemImg": "https://example.com/images/buddha-bowl.jpg",

"category": "Veg",

"menuCategory": "Salad",

"price": 10.49,

"discount": 0,

"rating": 4.8

}

]

}

]

}

}

**8. Authentication**

Authentication is handled using JWT tokens. After login, users receive a token stored in local storage. Protected routes require a valid token for access, which is verified using middleware.

**9. User Interface**

**Login and Registration Form**: Secure user access for ordering and account management.

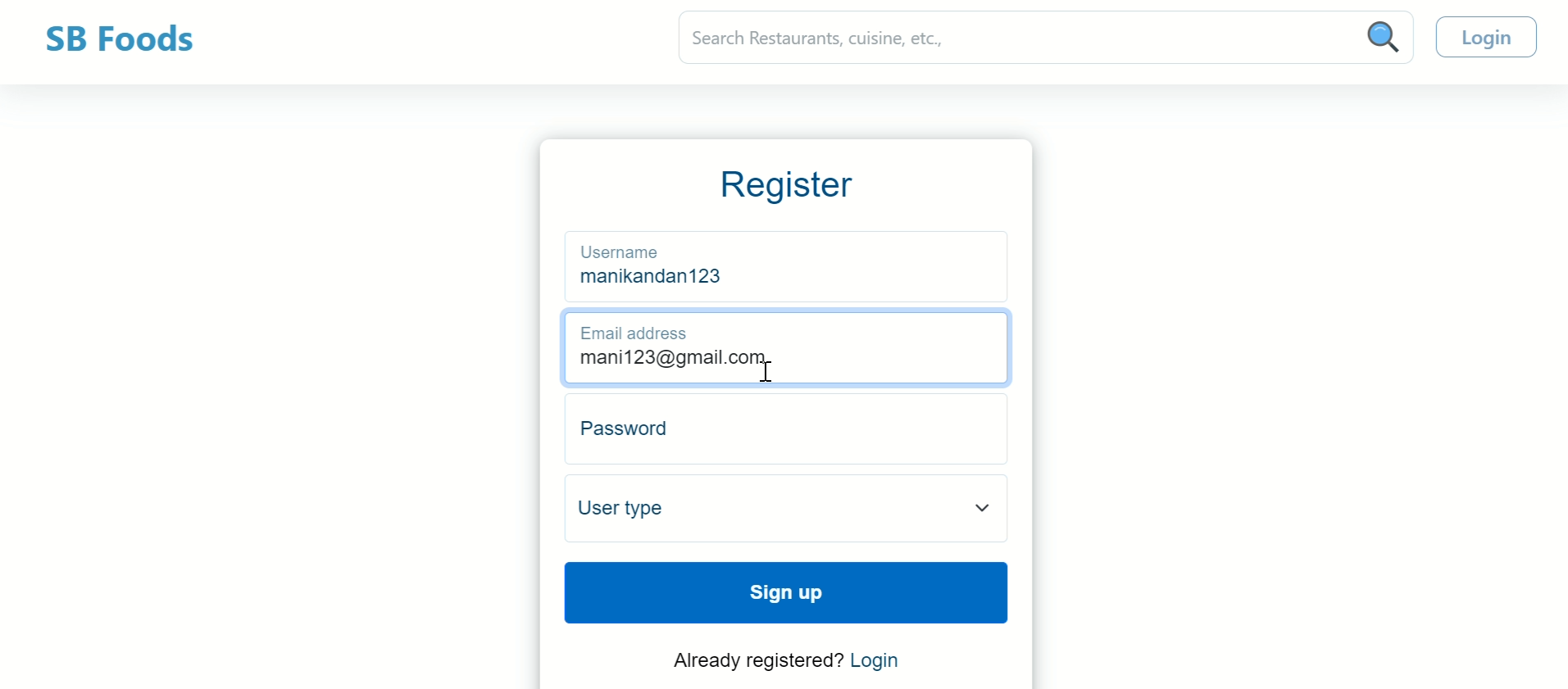
**Menu Catalog Page**: Displays available dishes with detailed descriptions, pricing, and promotions.

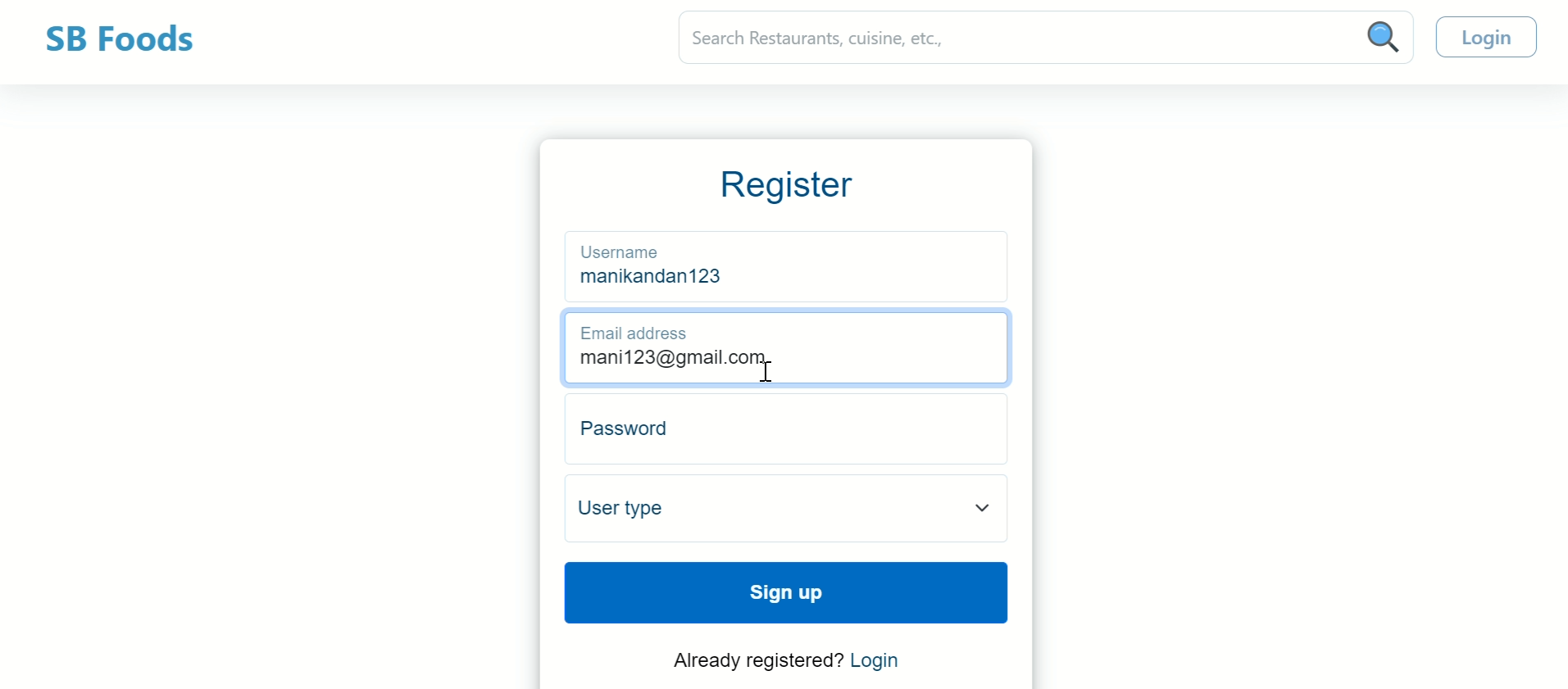
**Dashboard Page**: Allows users to track their past orders, favorite dishes, and current order status.

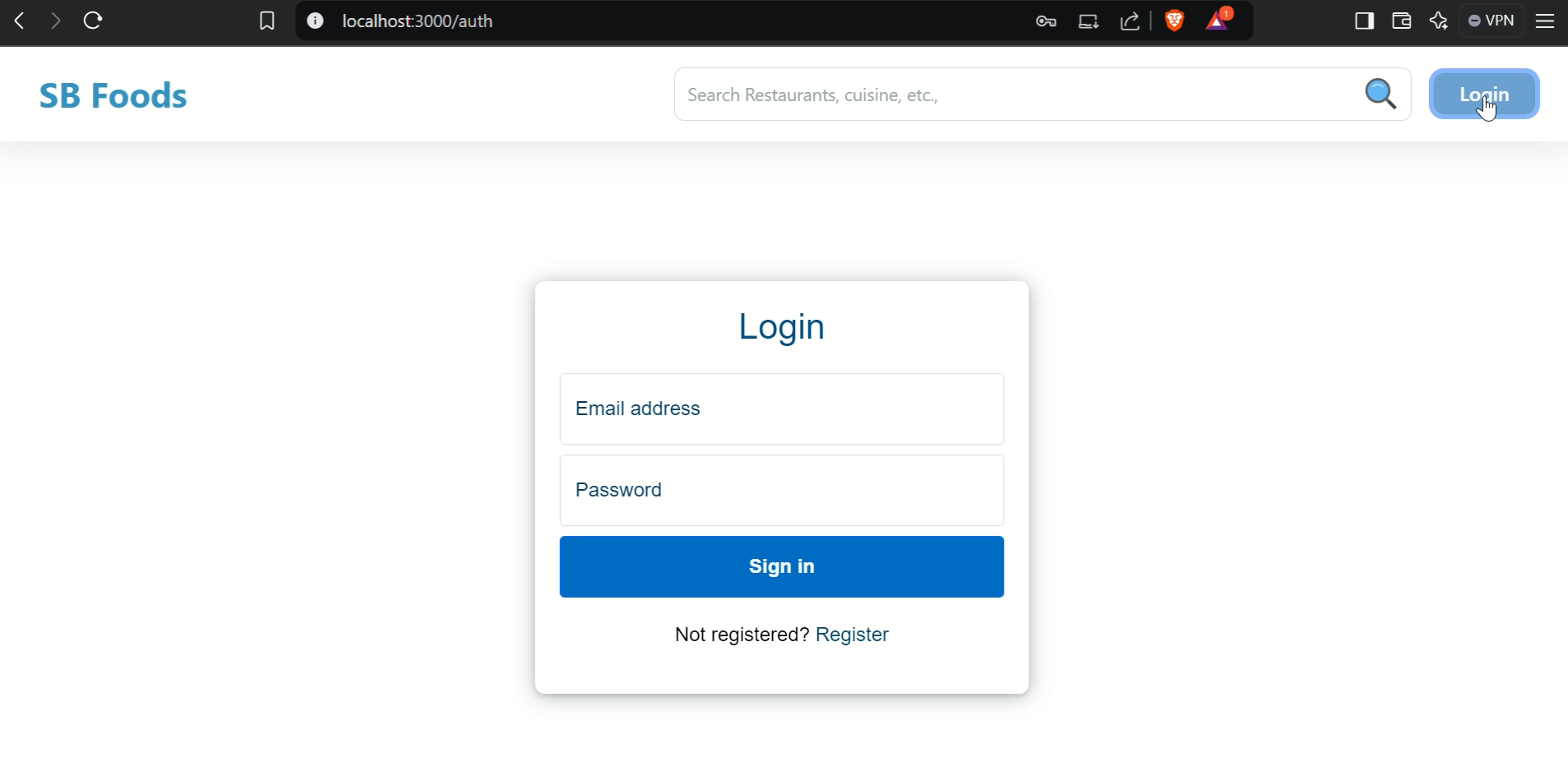
**10. Testing**

Testing is performed using tools like Jest and Postman for API testing. Unit tests cover component functionality and API response validation.

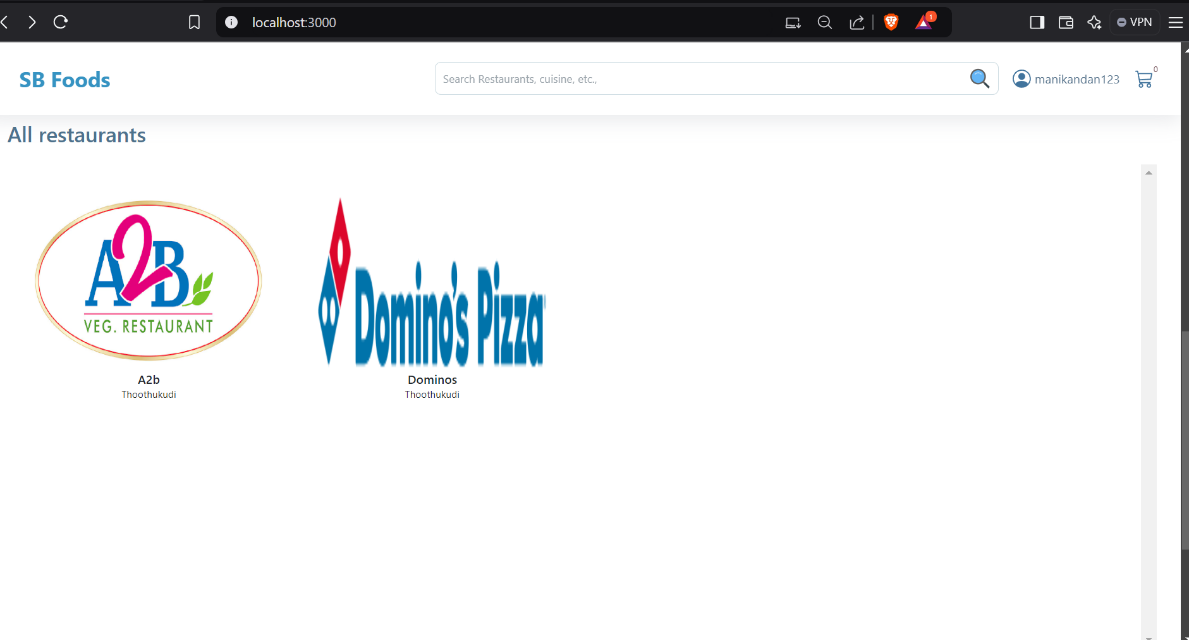
**11. Screenshots**

* Admin Dashboard
* ****
* A login and registration form.

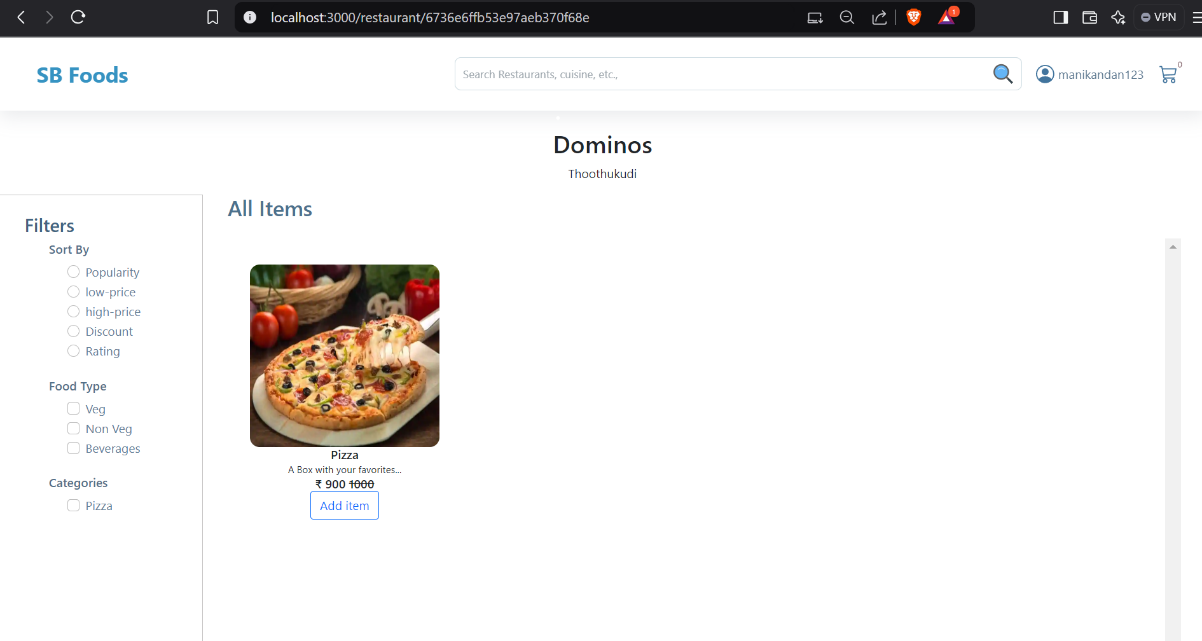




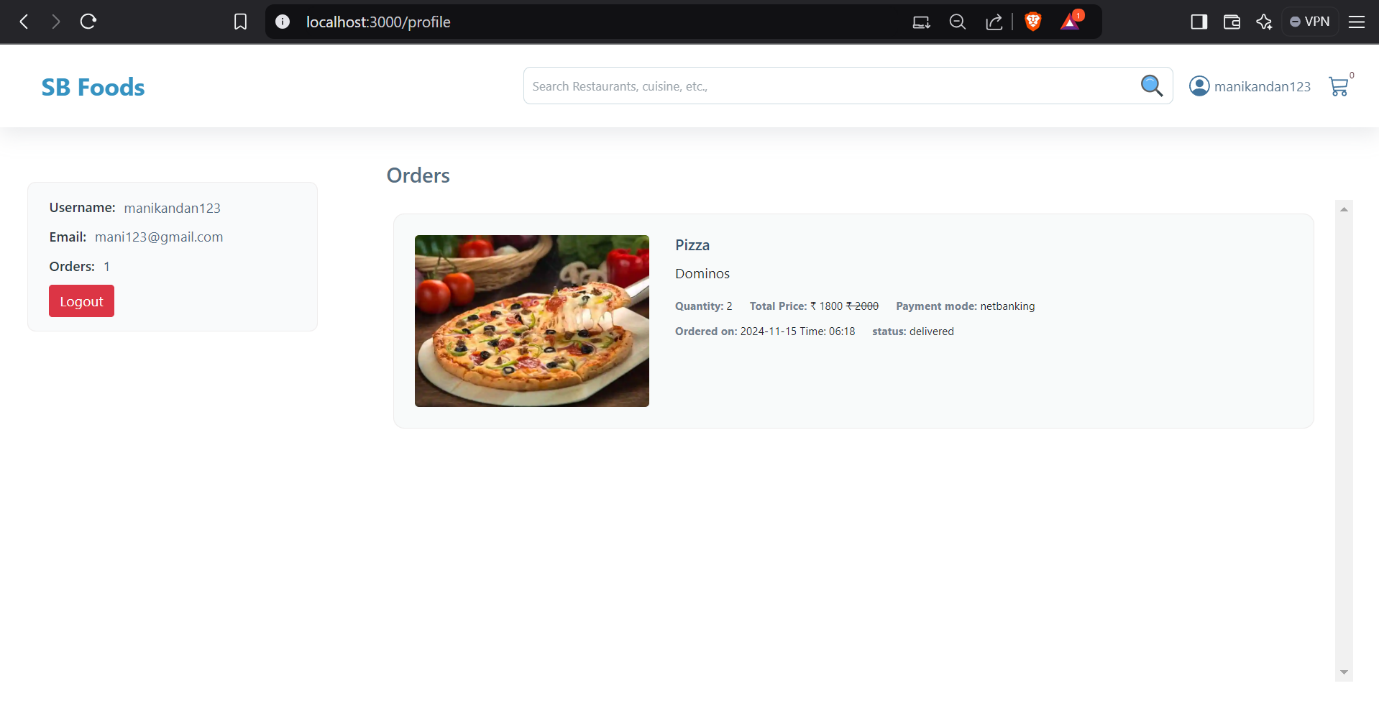
* ALL restaurant page displaying available restaurant



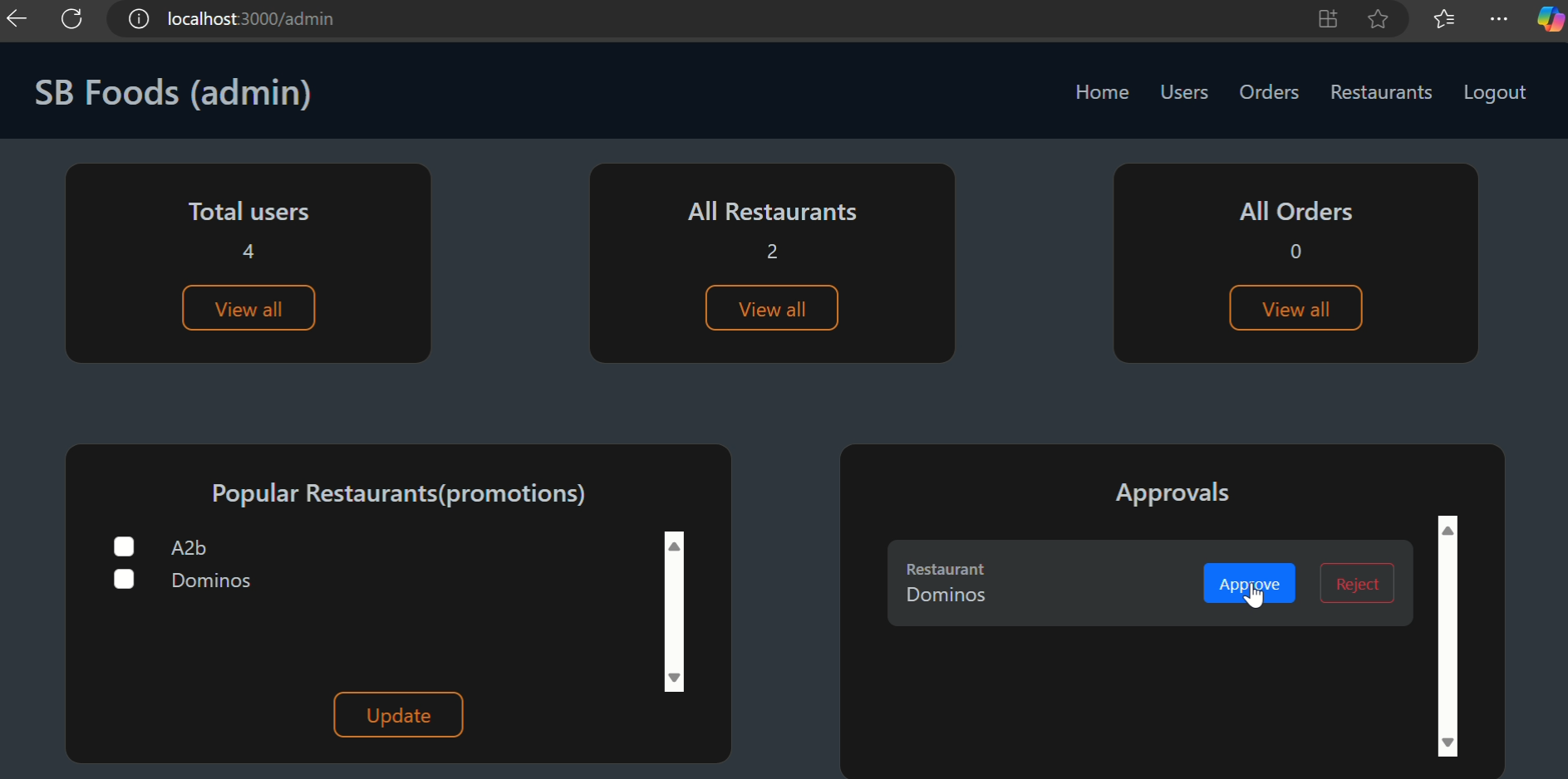
* Restaurant menu



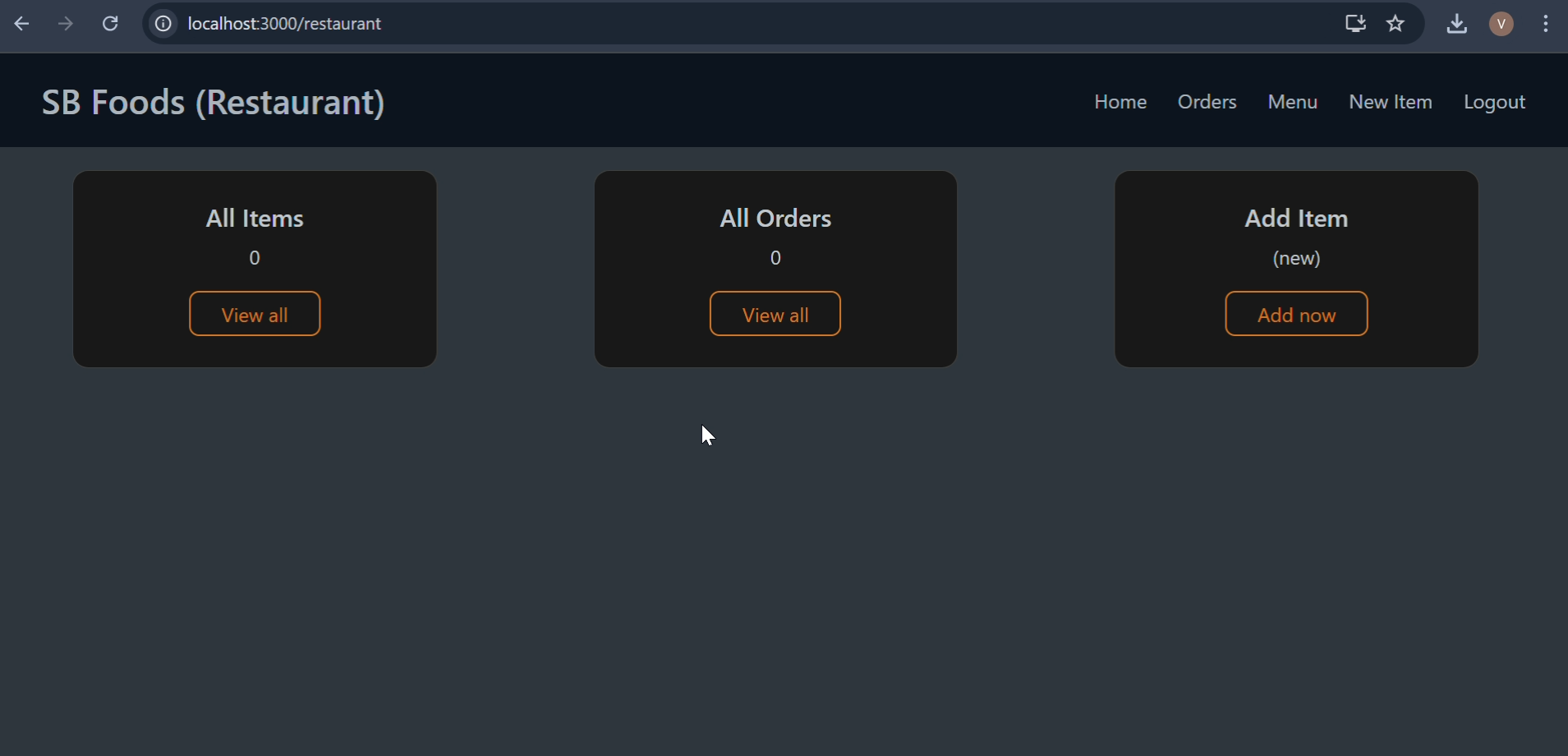
* User Profile



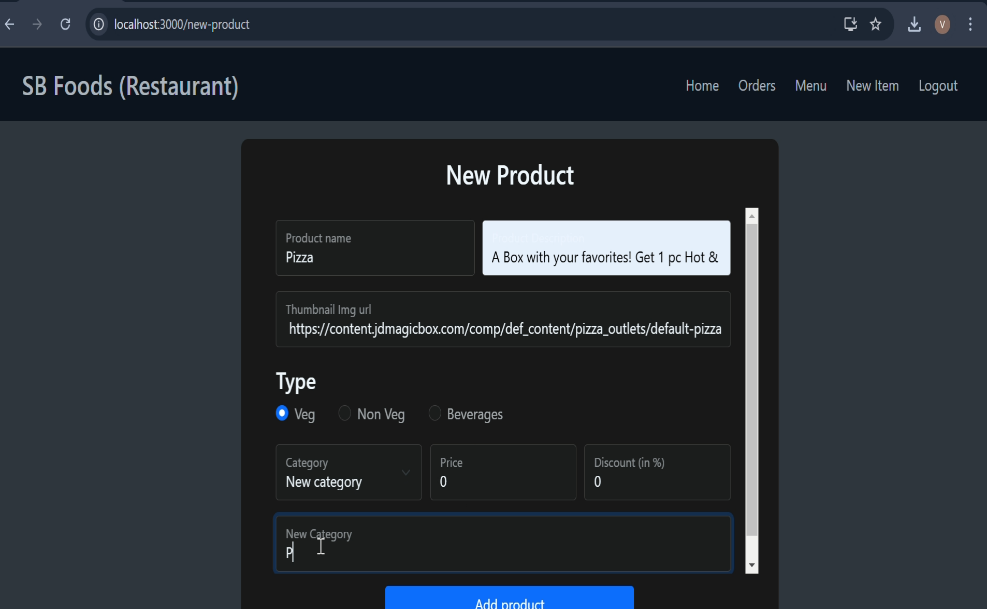
* Admin Dashboard



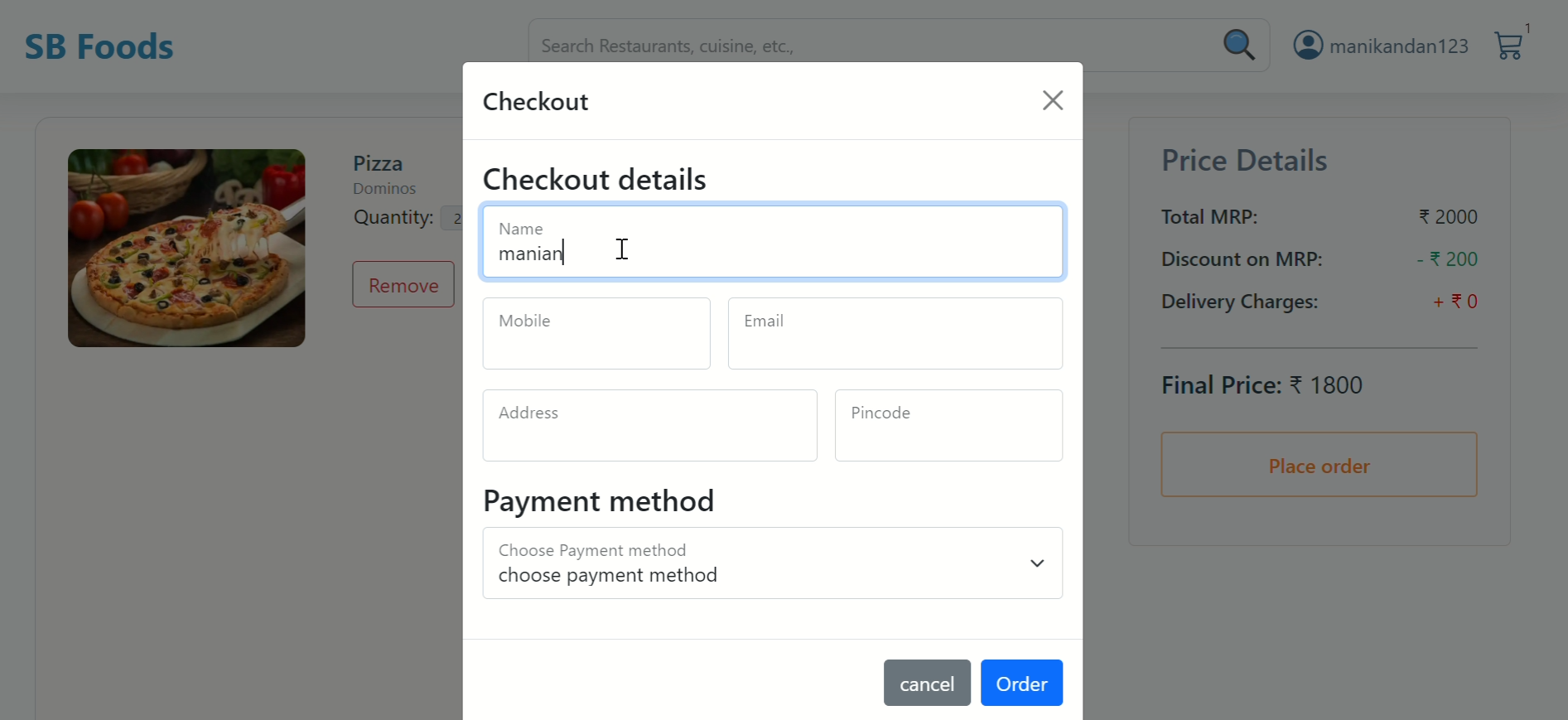
* **Restaurant DashBoard**



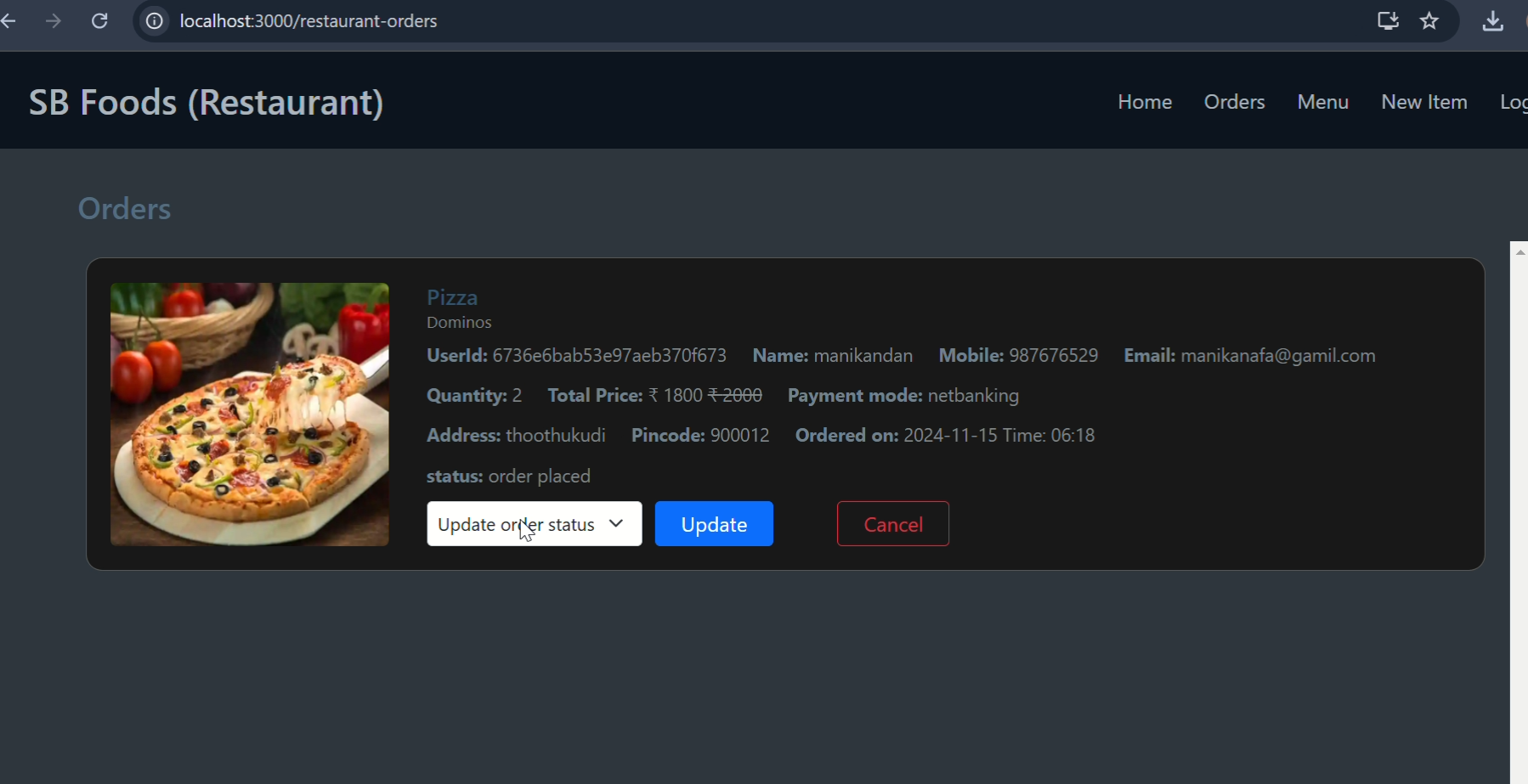
* **New Product**

****

* + - **Check out**

****

**All orders**

****

.

**12. Known Issues**

* Token expiration may log users out unexpectedly.
* multiple items load is delays in real-time

**13. Future Enhancements**

* **Introduce Special Meal Plans**: Offer curated meal plans based on user preferences and dietary needs.
* **Implement a Payment Gateway**: Enable secure payments for premium dishes and exclusive promotions.
* **Enhance Real-Time Communication**: Integrate live video streaming for interactive cooking classes and events.