**Full Stack Development with MERN**

**1. Introduction**

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

**Team Members:**

* K.Kailshwaran – Technical Lead
* A.Kalaivendhan – Full-stack Developer
* B.Saran – Full-stack Developer
* M.Saran – UI Designer

**2. Project Overview**

**Purpose**: SB Foods is an innovative platform designed to enhance your food ordering experience. This intuitive app allows users to browse, explore, and place orders effortlessly. With detailed dish descriptions, customer reviews, pricing information, and ongoing promotions, SB Foods ensures that users can make well-informed decisions. Simply choose your dishes, enter your delivery details, and enjoy instant order confirmation, making your food journey smooth and hassle-free.

**Features of SB Foods**:

* **User Registration and Login**: Secure access with JWT authentication.
* **Explore and Order**: Discover meals with in-depth descriptions, reviews, and promotions.
* **Personalized Dashboard**: Keep track of past orders and personal preferences.
* **Admin Panel**: Manage menu items, promotions, and customer information.
* **Real-Time Updates**: Receive instant order confirmations and track orders live.

**3. Architecture of SB Foods**

* **Frontend**: Built with React, providing a dynamic and easy-to-use interface. It includes components for user authentication, browsing the menu, and placing orders, with React Router ensuring smooth navigation.
* **Backend**: Powered by Node.js and Express.js, the backend manages API requests, ensures secure authentication, and interacts with MongoDB for data storage. Controllers manage data related to the menu, user orders, and other operations.
* **Database**: MongoDB is used to store user information, menu items, orders, and promotions. The data is structured in collections for efficient retrieval and updates.

**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 the backend and frontend directories and install dependencies:
   * For Backend: cd backend  
     npm install
   * For Frontend: cd ../frontend  
     npm install
3. Set up environment variables by creating a .env file in the backend directory with necessary details like database connection strings and JWT secret.

**5. Folder Structure**

**Client**:  
The frontend directory contains:

* src/components: Reusable React components.
* src/pages: Different pages like CourseList, Login, Register.
* src/redux: Manages state data.
* src/styles: CSS files for styling.

**Server**:  
The backend directory includes:

* index.js: Configures database connections, handles business logic, defines routes for 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**

* **POST /register**: Register a new user or restaurant.
* **POST /login**: User login.

**Admin**

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

**User Management**

* **GET /fetch-user-details/**

: Get user details by ID.

* **GET /fetch-users**: Get a list of all users.

**Restaurant Management**

* **GET /fetch-restaurants**: Get all restaurants.
* **GET /fetch-restaurant-details/**

: Get restaurant details by owner ID.

* **GET /fetch-restaurant/**

: Get restaurant details by restaurant ID.

**Food Items**

* **GET /fetch-items**: Get all food items.
* **GET /fetch-item-details/**

: Get details of a specific food item.

* **POST /add-new-product**: Add a new food product.
* **PUT /update-product/**

: Update an existing product by ID.

**Orders**

* **GET /fetch-orders**: Get all orders.
* **PUT /cancel-order**: Cancel an order.
* **PUT /update-order-status**: Update the status of an order.

**Cart**

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

**Example Response**:

json

Copy code

{

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

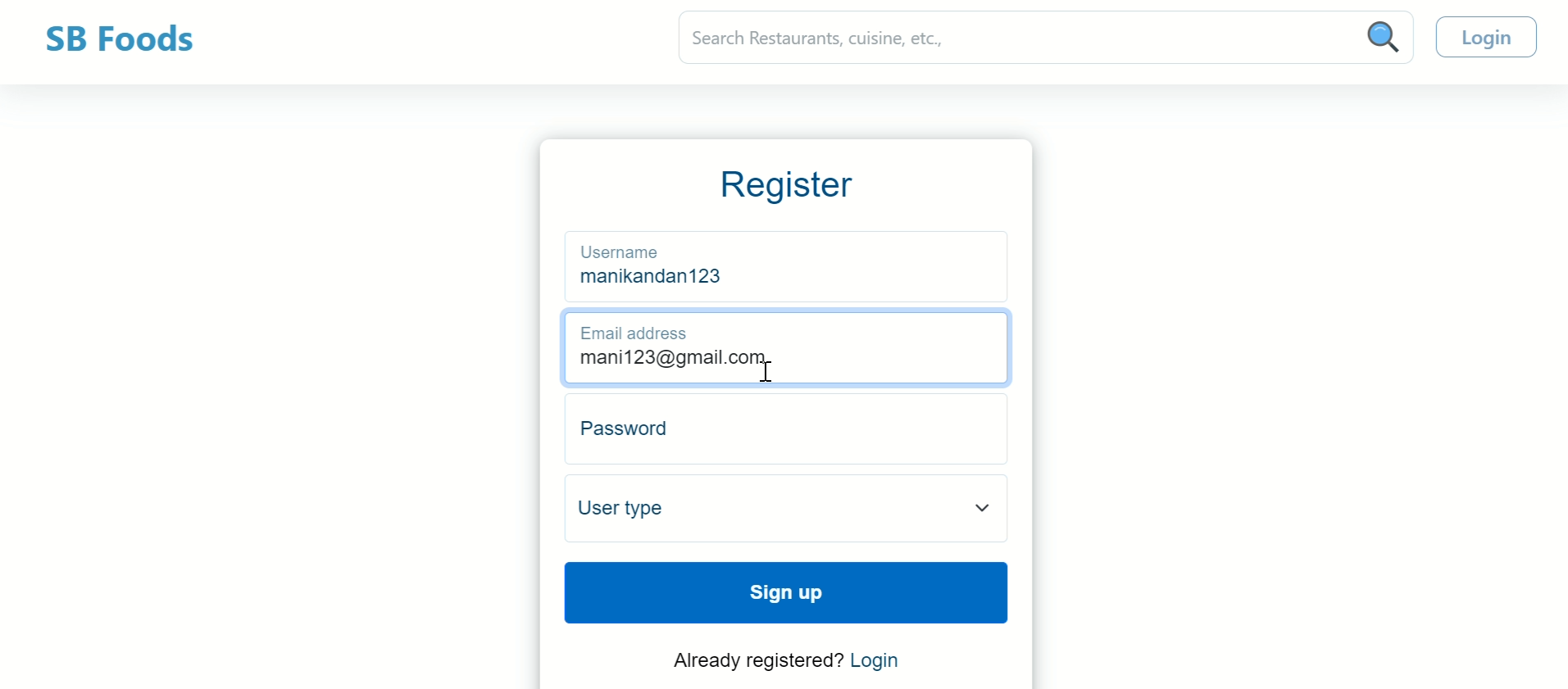
**9. User Interface**

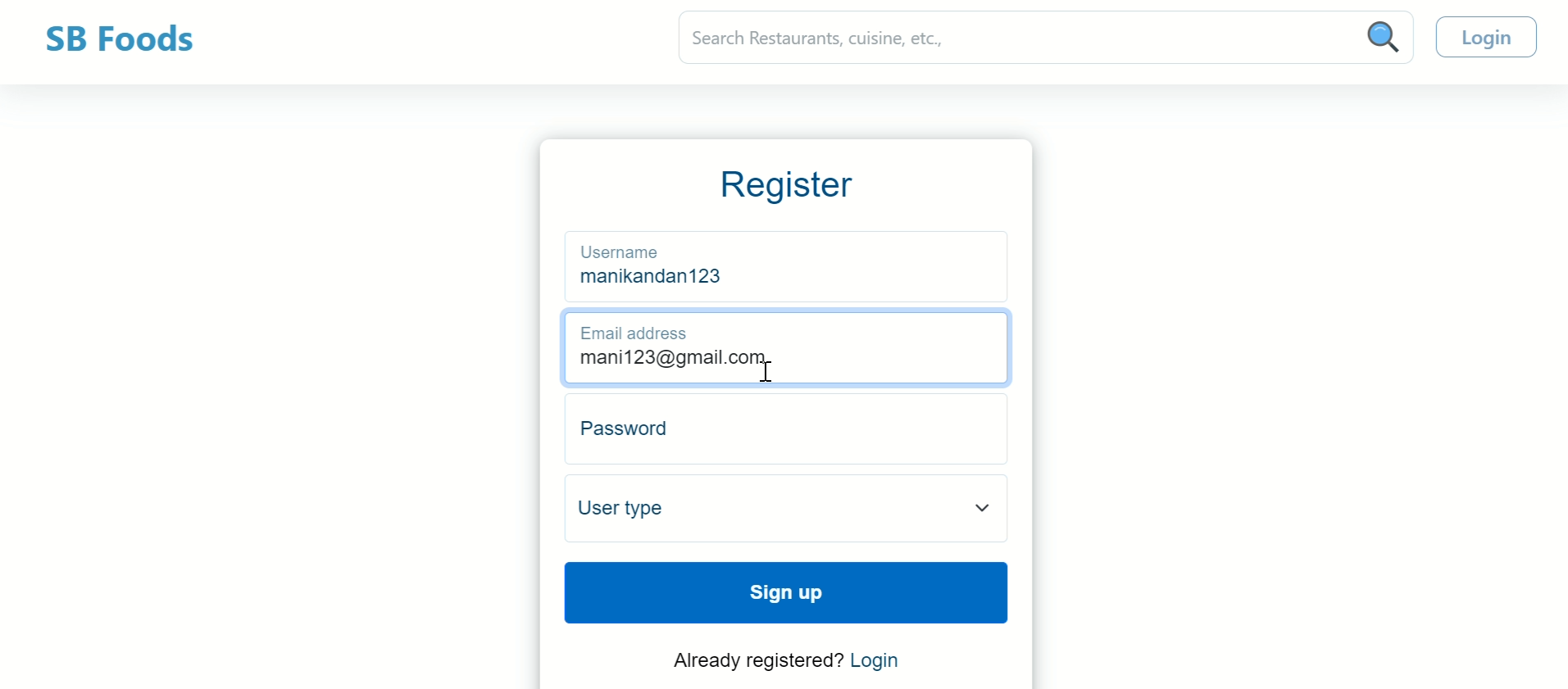
* **Login and Registration Form**: Secure user login and account management for ordering.
* **Menu Catalog Page**: Displays available dishes with descriptions, pricing, and promotions.
* **Dashboard Page**: Allows users to track past orders, favorite dishes, and current order status.

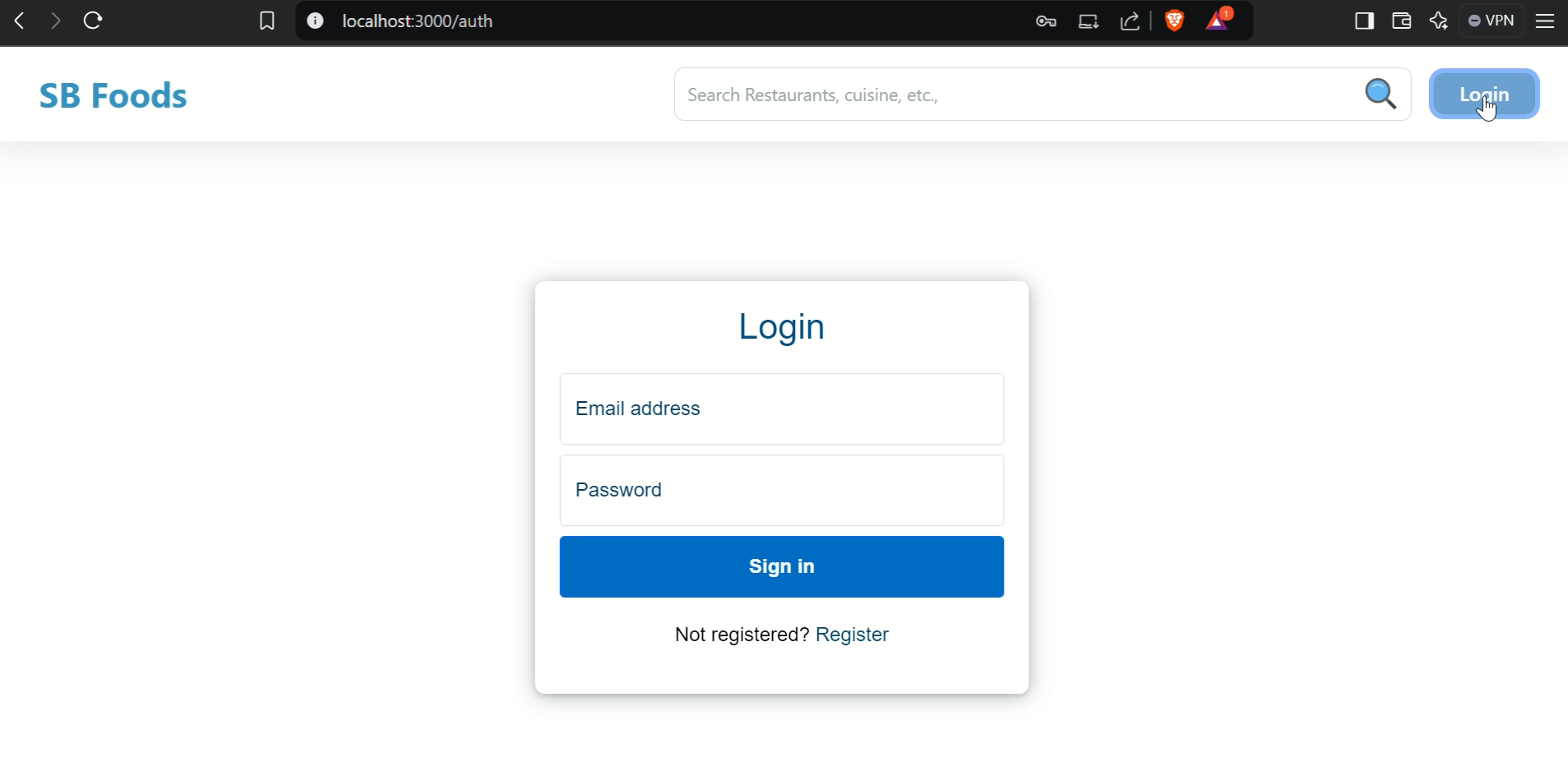
**10. Testing**

Testing is performed with tools like Jest and Postman for API validation. Unit tests are used for component functionality and API responses.

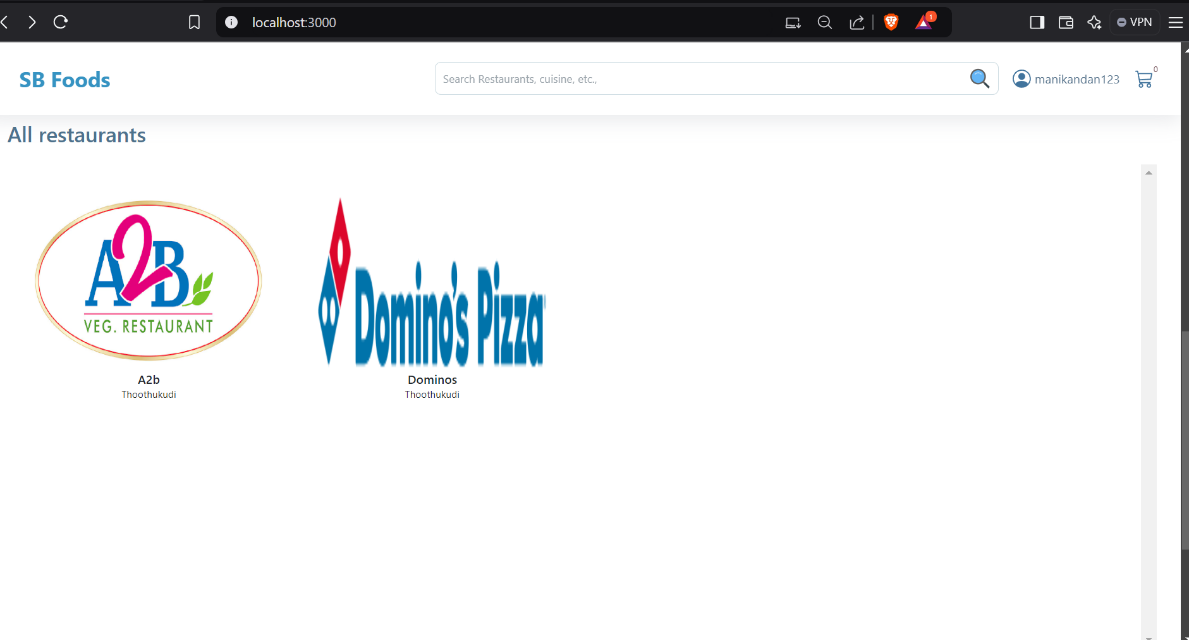
**11. Screenshots**

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

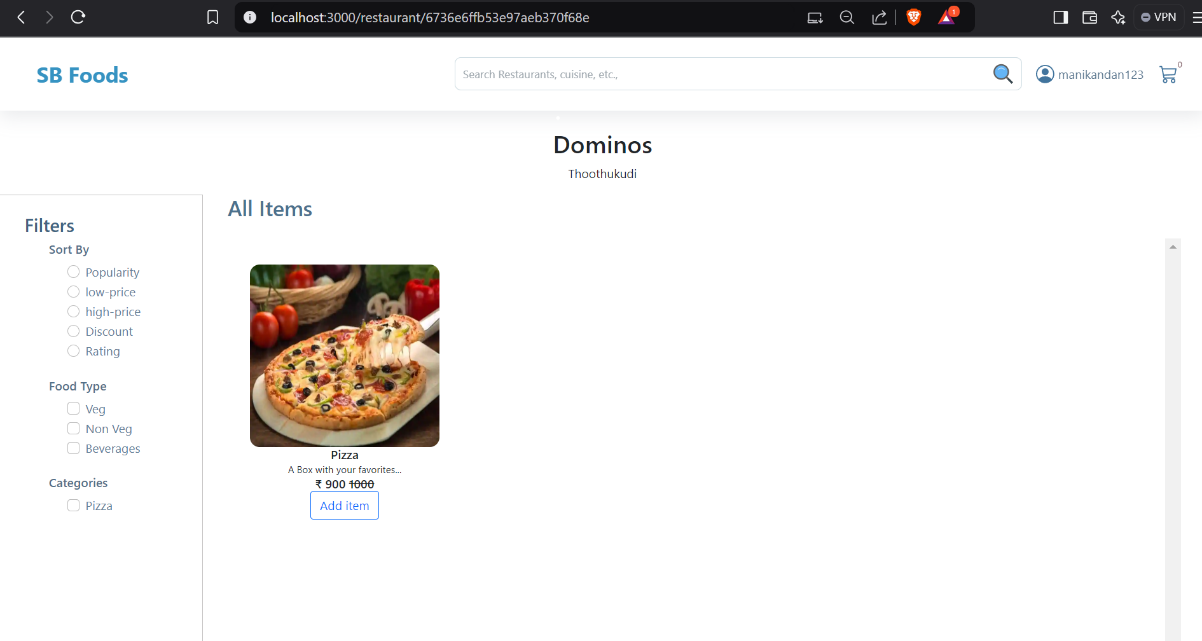




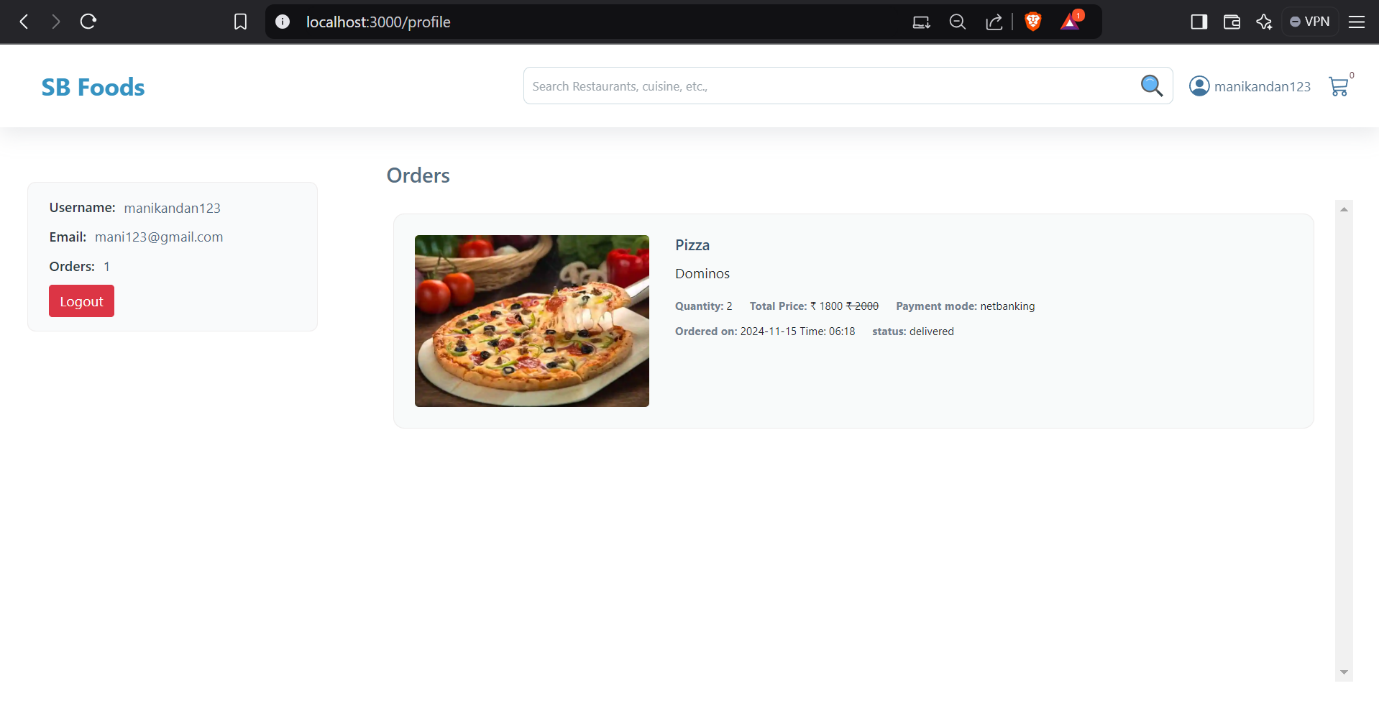
* ALL restaurant page displaying available restaurant



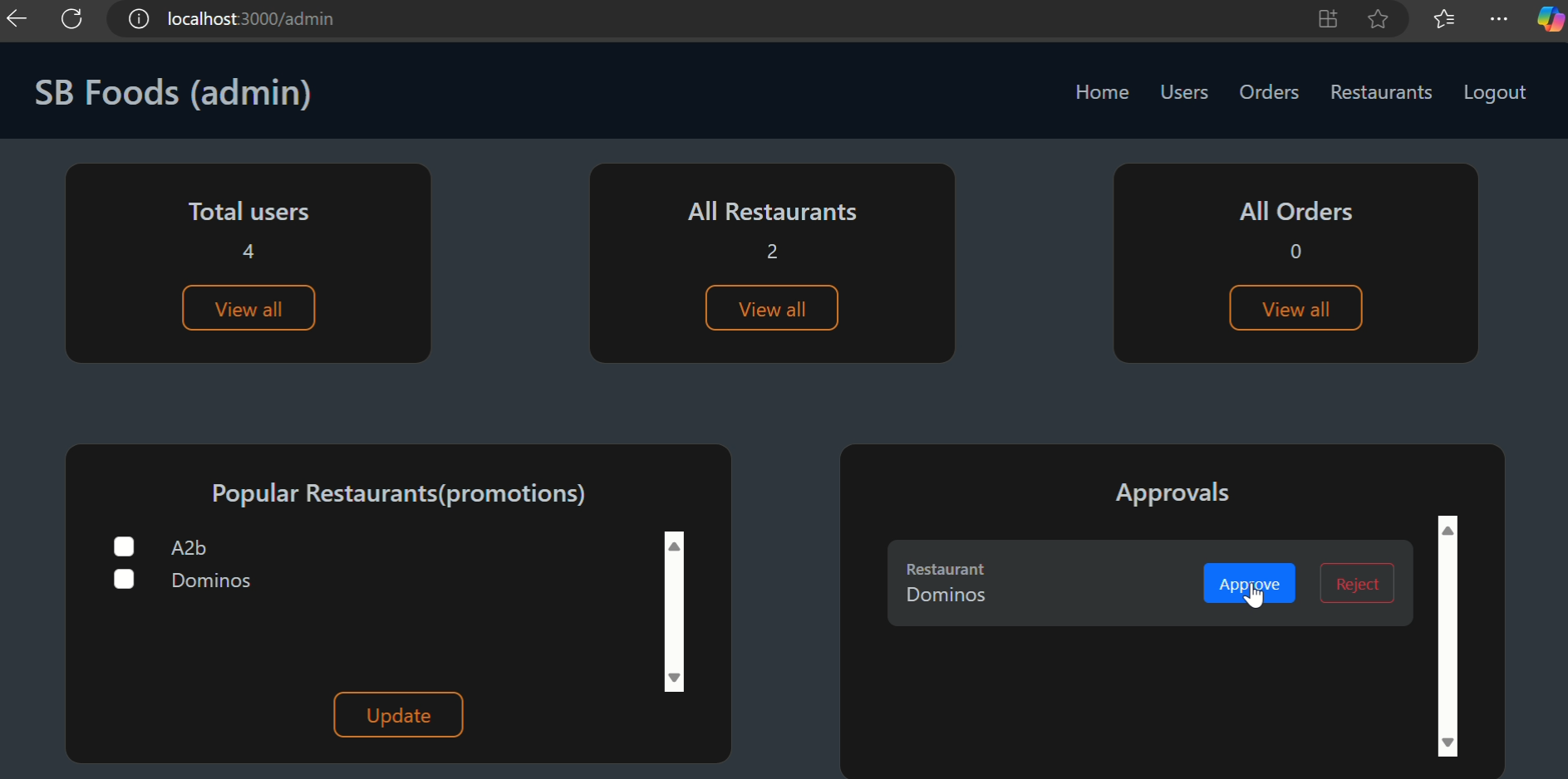
* Restaurant menu



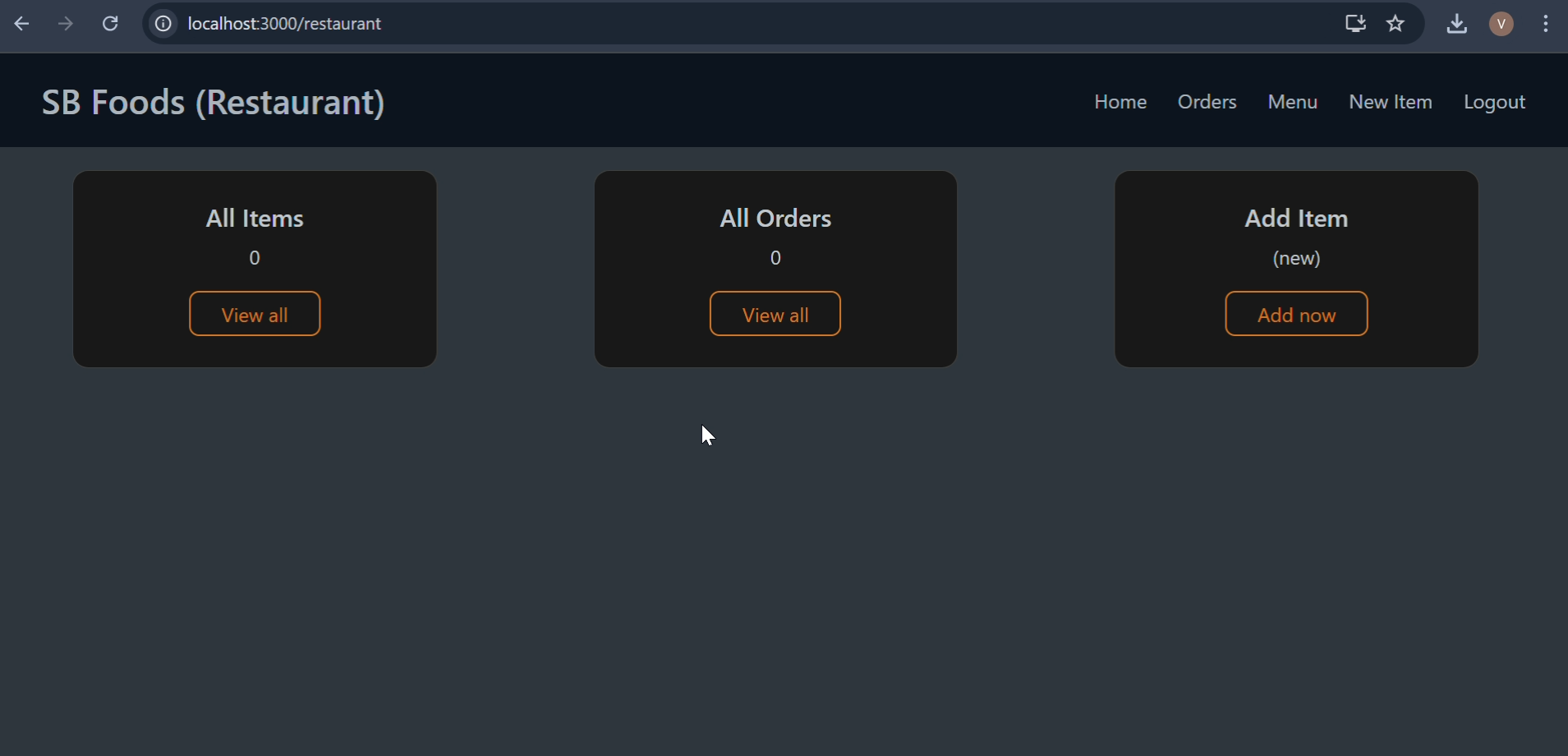
* User Profile



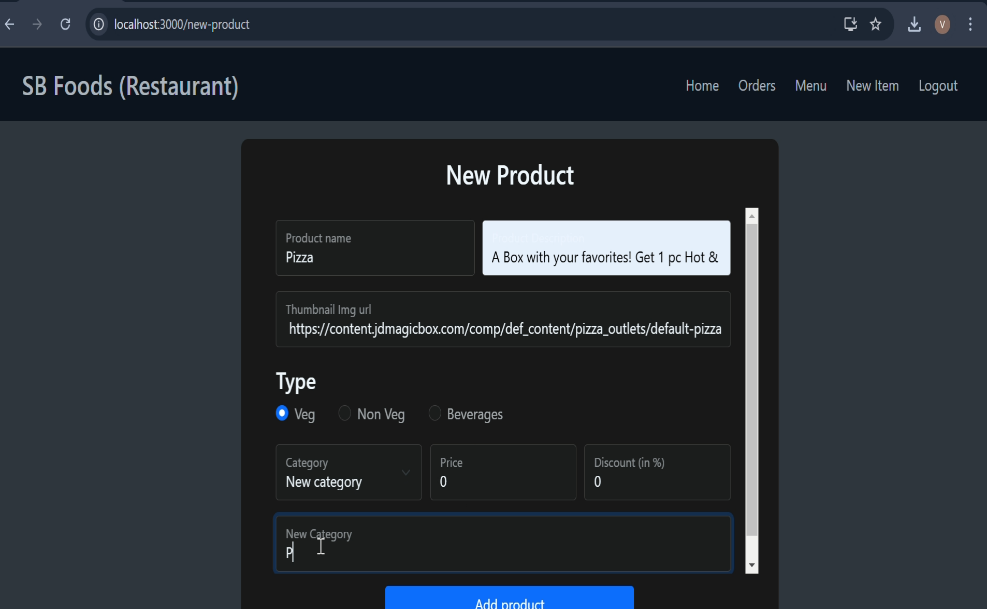
* Admin Dashboard



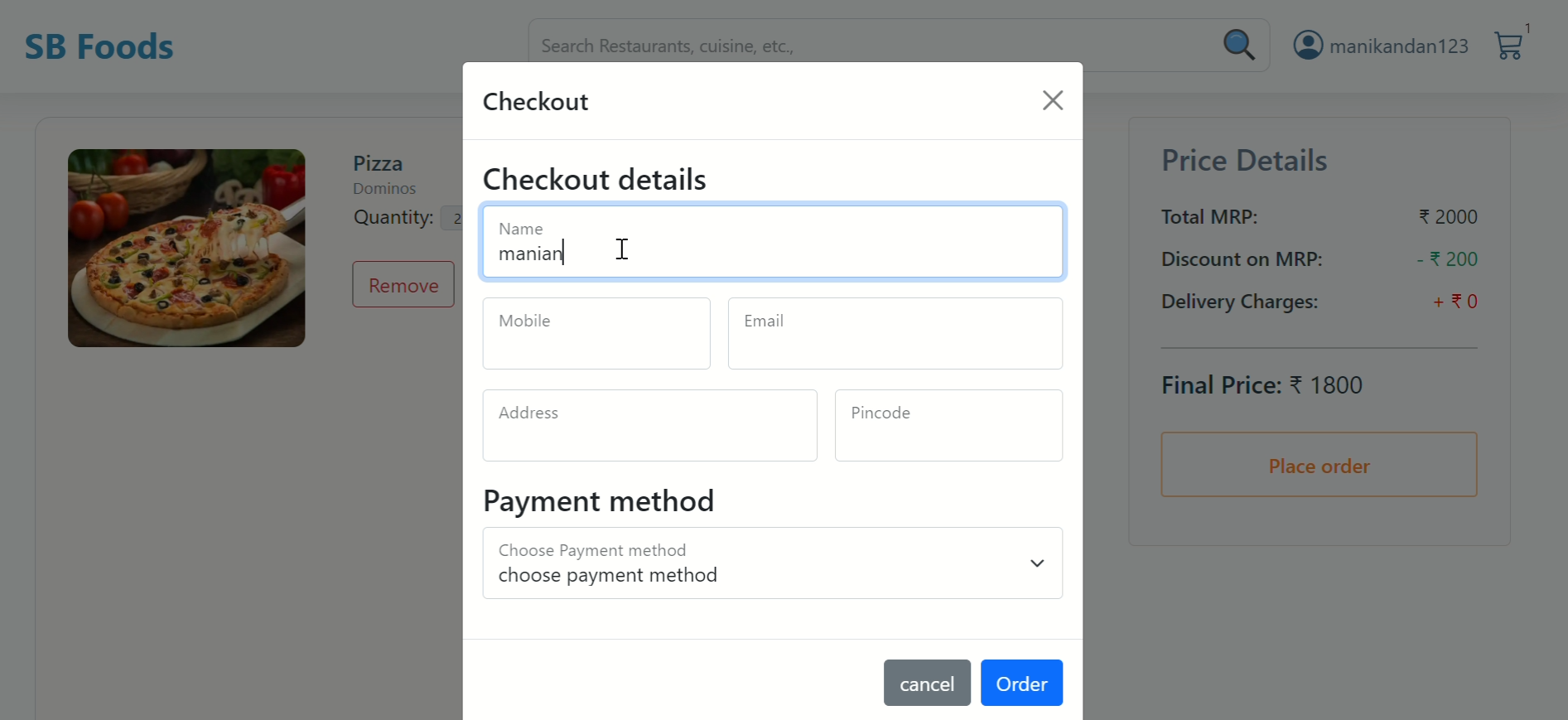
* **Restaurant DashBoard**



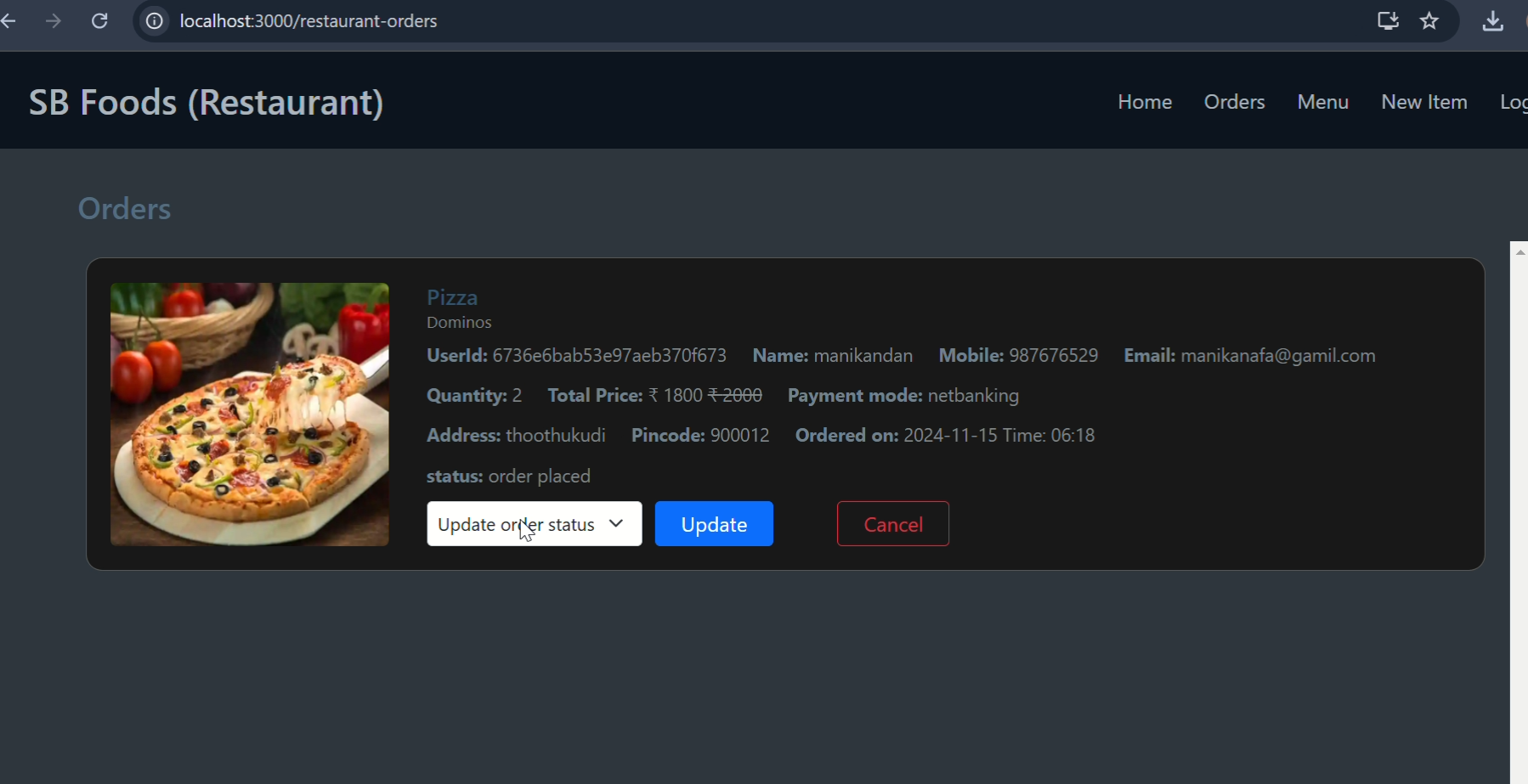
* **New Product**

****

* + - **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.