Full Stack Development with MERN Project

Documentation format

1. Introduction

- **Project Title:** OrderOnTheGo - SB Foods
- **Team Members:**

1.Team Leader : Bhavya Tatineni

Coordinator

Builds RESTful APIs using Node.js and Express.js, manages authentication and server logic.

2.Team member : Bande Raveendra

Works on the React-based UI, handles component design, page routing, and user interactions.

3. **Team member : Bandreddi Mahitha**

Designs and manages MongoDB schemas, handles CRUD operations and ensures data consistency.

4.Team member : Banala Lahari

Responsible for overall planning, coordination, GitHub management, and integration of frontend and backend.

2. Project Overview

Project Purpose: OrderOnTheGo – SB Foods

The OrderOnTheGo – SB Foods project is a full-stack web application developed to streamline and enhance the online food ordering experience. The primary objective is to deliver a modern, user-friendly platform that allows customers to conveniently browse, select, and order food items from various restaurants through a responsive web interface.

Key goals of the application include:

Enabling users to access and explore food menus at any time.

Allowing customers to add items to a cart and place orders efficiently.

Reducing the dependency on physical visits or phone calls to restaurants.

Providing a robust backend system for managing products, orders, and user data.

The project aims to emulate the essential functionalities of popular food delivery platforms such as Swiggy, Zomato, and Uber Eats, utilizing open-source technologies to ensure flexibility, scalability, and ease of deployment.

Features: For Users:

- **Sign Up / Log In** – Create an account and access your orders.
- **Browse Food Items** – View a list of available dishes with images, prices, and descriptions.
- **Add to Cart** – Add favorite food items to your cart.
- **Cart Storage** – Your cart items are saved even if you refresh the page.
- **Place Orders** – Enter your address and choose payment method to place an order.
- **Order Confirmation** – Get a message when your order is successfully placed.

For Admin (Future Scope):

- **Add or Update Products** – Admin can manage food items.
- **View Orders** – Admin can see orders placed by users.

3. Architecture

Frontend (React.js)

- Built using React with multiple pages (Home, Products, Cart, etc.)
- Uses React Router for navigation and Context API for managing the cart
- Axios is used for API calls to the backend
- Cart and user info are stored in localStorage

Backend (Node.js + Express.js)

- Handles API routes like register, login, get products, and place orders
- Uses Express middleware for JSON handling and CORS
- Connects to MongoDB using Mongoose

Database (MongoDB)

- Stores user, product, and order data
- Collections:
  - users: name, email, password, address
  - products: name, description, price, image
  - orders: userId, items, address, payment method

4. Setup Instructions

Prerequisites

- **Node.js & npm** – For running frontend and backend
- **MongoDB** – Local database (use Compass or terminal)
- **Git** – To clone the project
- **VS Code** – Recommended editor

Installation Steps

Clone the Project

```
git clone https://github.com/srikanthramagani/OrderGo.git
cd OrderGo
```

1. **Install & Run Backend**

```
cd server
npm install
node server.js
```

2. **Install & Run Frontend**
   Open a new terminal:

```
cd client
npm install
npm start
```

3. **Start MongoDB**
   - Use MongoDB Compass or run mongod in terminal.

Your app will run at:

- Frontend: http://localhost:3000
- Backend API: http://localhost:5000

5. Folder Structure

- **Client(React frontend):**

```
client/
├── public/              → Static assets
├── src/
│   ├── components/
│   │   └── pages/       → All page components (Home, Cart, Login, etc.)
│   ├── context/         → Cart context (global state)
│   ├── App.jsx          → Main component with routes
│   └── index.js         → Entry point of the app
```

- **Server(Node.js backend):**

```
server/
```

```
├── models/          → Mongoose schemas (User, Product, Order)
├── server.js        → Main Express server file
```

6. Running the Application

Frontend :

```
cd client
npm start
```

Runs the React app at: http://localhost:3000

Backend :

```
cd server
npm start   # Or use: node server.js
```

Runs the Node.js server at: http://localhost:5000

7. API Documentation

- **POST /api/register :** Registers a new user.
- **POST /api/login** : Logs in an existing user.
- **GET /api/products** : Retrieves a list of available food products**.**
- **POST /api/orders** : Places a new order.

8. Authentication

How Authentication Works:

- Users register by providing their name, email, password, and address using the endpoint:

  POST /api/register

- They log in with their email and password using:

  POST /api/login

Method Used:

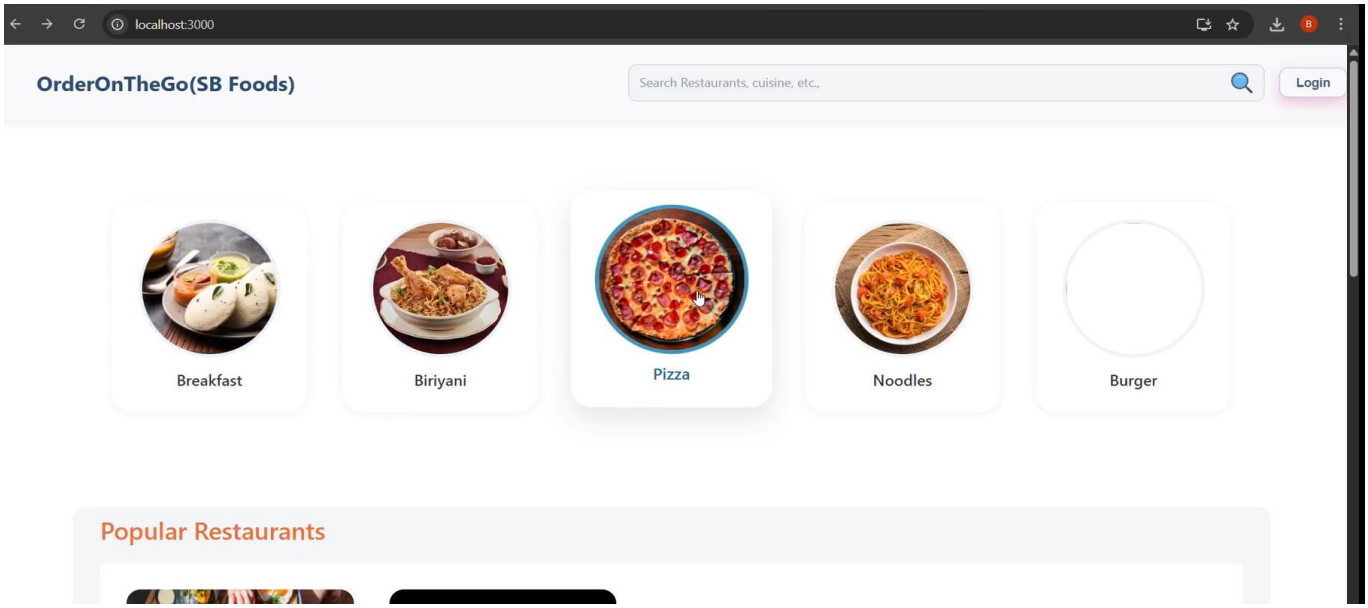
- The current setup uses **basic email and password matching**.
- There is **no token-based authentication** or sessions implemented at this stage.
- After login, the user's details can be stored on the frontend (e.g., in localStorage) to maintain the login state.

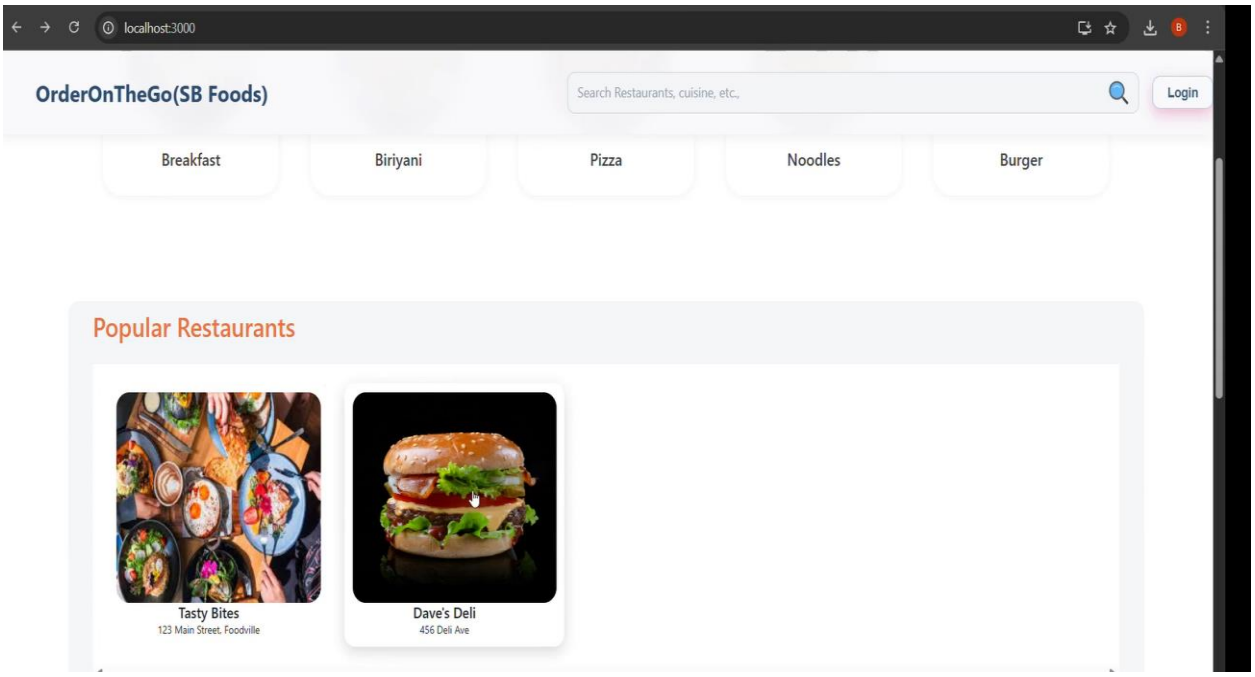
Recommendations for Improvement:

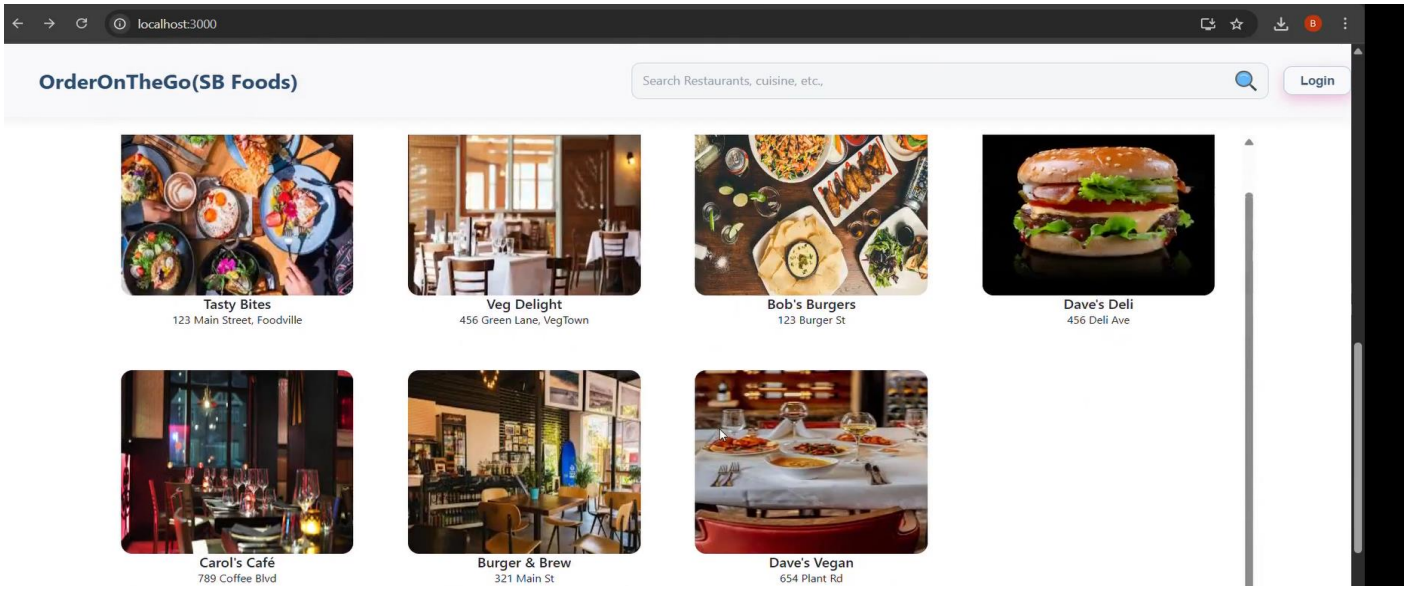
To enhance security in the future, it is recommended to:

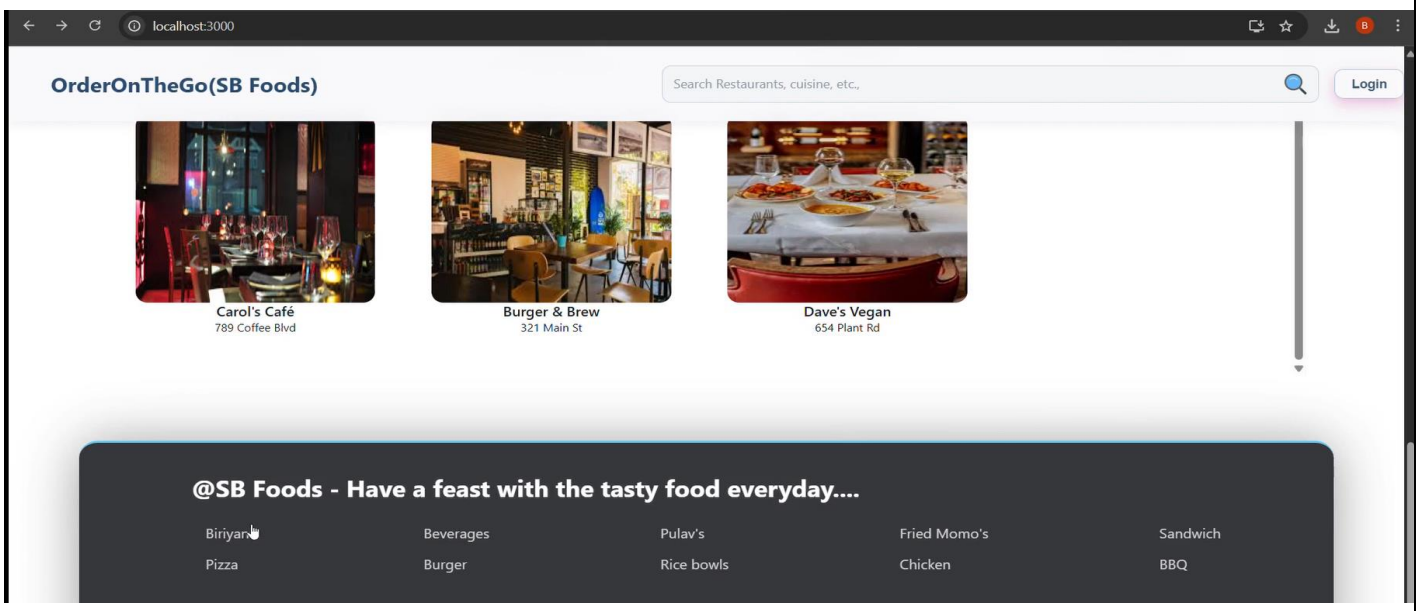
- Implement **JWT (JSON Web Token)** authentication.
- Use **middleware** to protect private API routes.
- Store tokens securely (e.g., in localStorage or HTTP-only cookies).

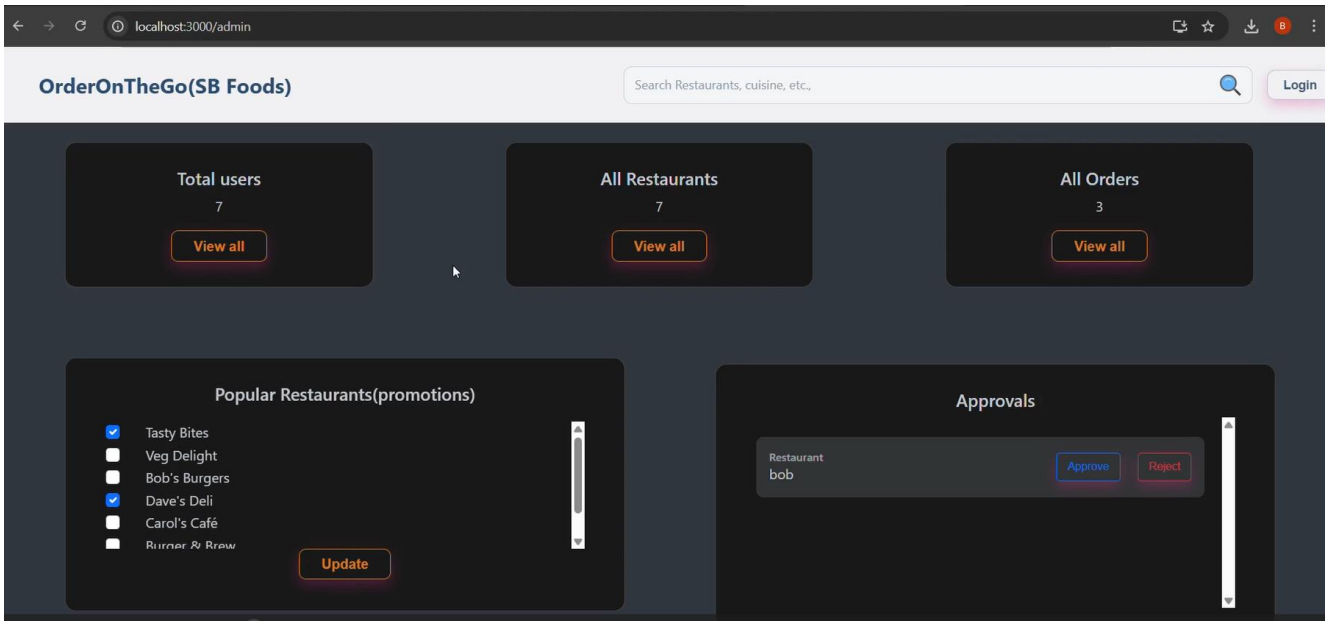
9. User Interface

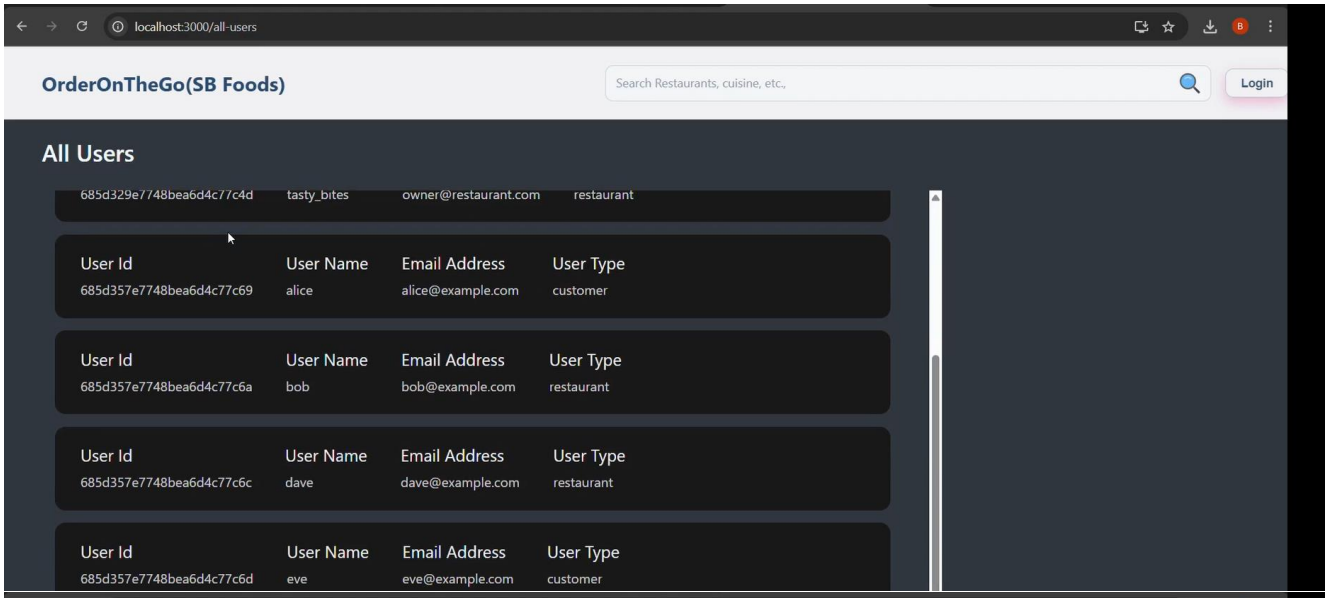


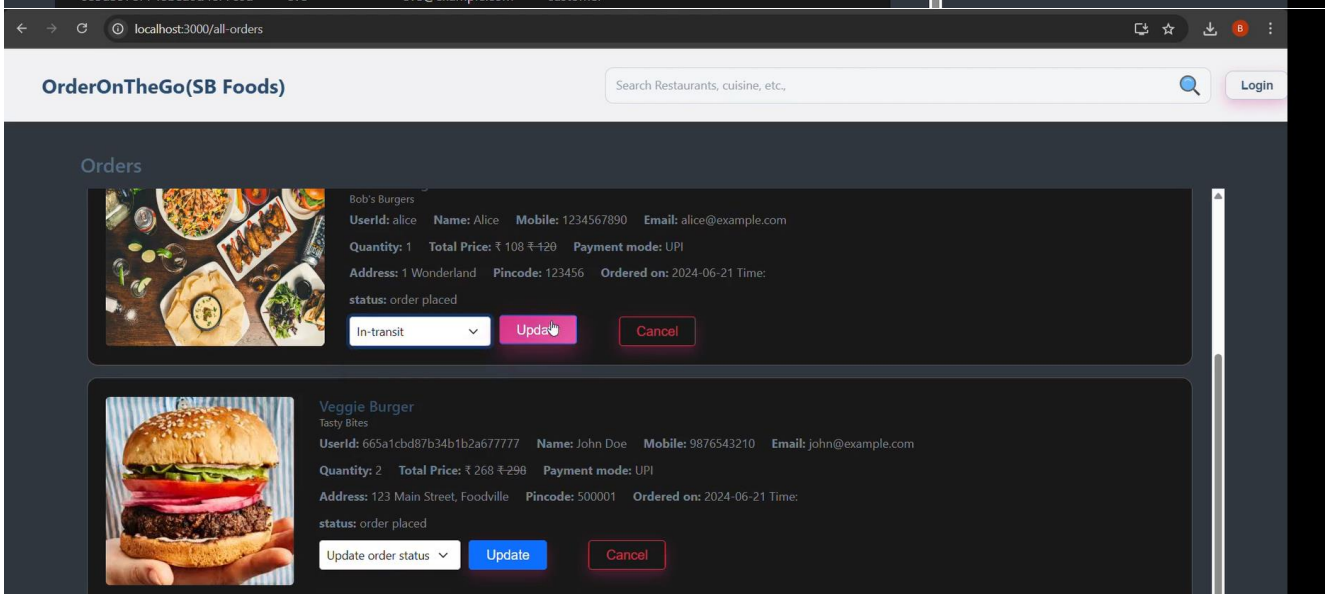


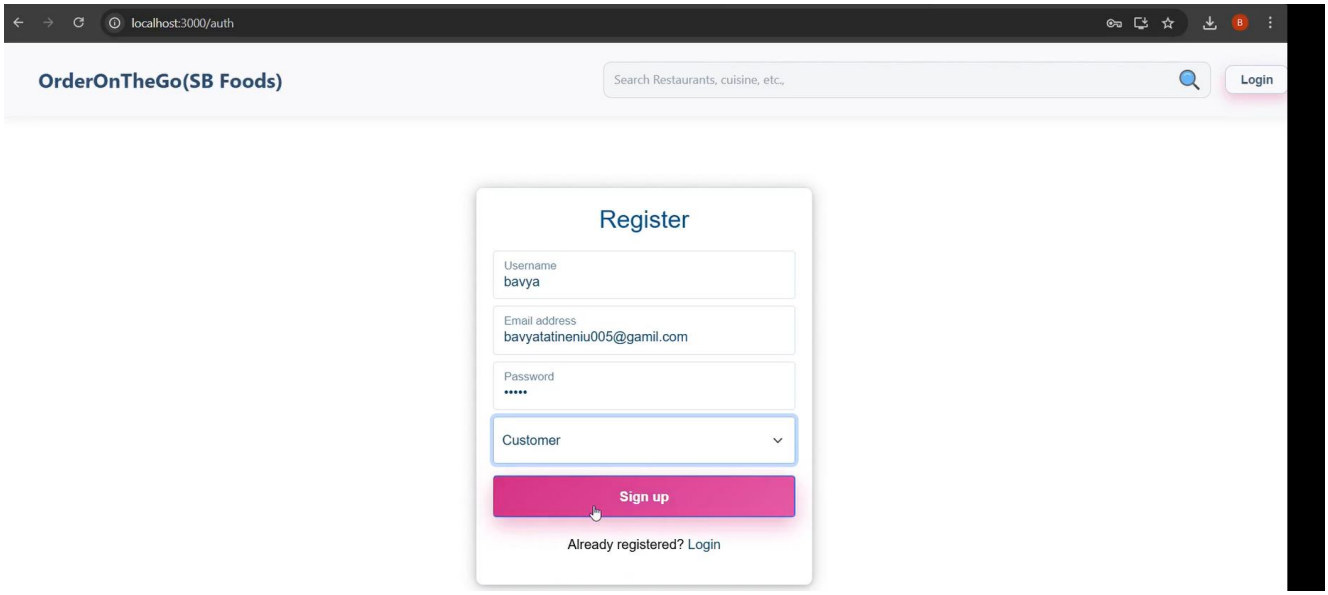

localhost:3000
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
Tasty Bites
123 Main Street, Foodville
Veg Delight
456 Green Lane, VegTown
Bob's Burgers
123 Burger St
Dave's Deli
456 Deli Ave
Carol's Café
789 Coffee Blvd
Burger & Brew
321 Main St
Dave's Vegan
654 Plant Rd

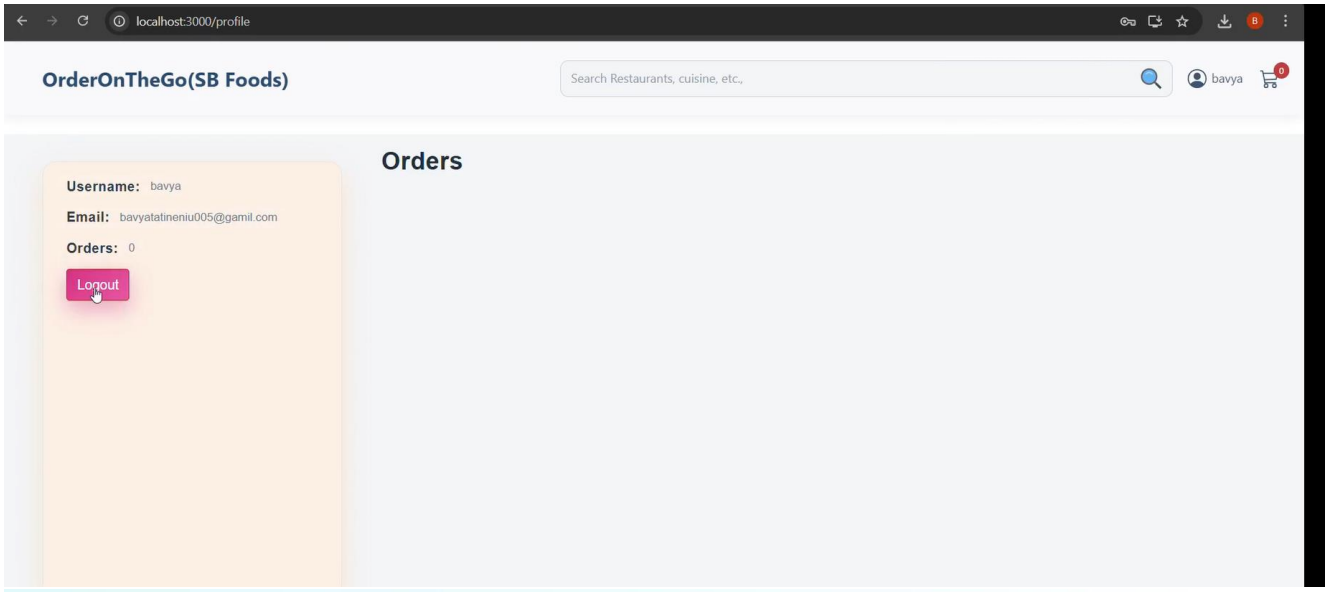

localhost:3000
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
Carol's Café
789 Coffee Blvd
Burger & Brew
321 Main St
Dave's Vegan
654 Plant Rd
@SB Foods - Have a feast with the tasty food everyday....
Biriyani
Beverages
Pulav's
Fried Momo's
Sandwich
Pizza
Burger
Rice bowls
Chicken
BBQ


localhost:3000/admin
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
Total users
7
View all
All Restaurants
7
View all
All Orders
3
View all
Popular Restaurants(promotions)
Tasty Bites
Veg Delight
Bob's Burgers
Dave's Deli
Carol's Café
Update
Approvals
Restaurant
bob
Approve
Reject


localhost:3000/all-users
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
All Users
685d329e7748bea6d4c77c4d tasty_bites owner@restaurant.com restaurant
User Id User Name Email Address User Type
685d357e7748bea6d4c77c69 alice alice@example.com customer
User Id User Name Email Address User Type
685d357e7748bea6d4c77c6a bob bob@example.com restaurant
User Id User Name Email Address User Type
685d357e7748bea6d4c77c6c dave dave@example.com restaurant
User Id User Name Email Address User Type
685d357e7748bea6d4c77c6d eve eve@example.com customer


localhost:3000/all-orders
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
Orders
Bob's Burgers
UserId: alice Name: Alice Mobile: 1234567890 Email: alice@example.com
Quantity: 1 Total Price: ₹ 108 ~~₹ 120~~ Payment mode: UPI
Address: 1 Wonderland Pincode: 123456 Ordered on: 2024-06-21 Time:
status: order placed
In-transit
Update
Cancel
Veggie Burger
Tasty Bites
UserId: 665a1cbd87b34b1b2a677777 Name: John Doe Mobile: 9876543210 Email: john@example.com
Quantity: 2 Total Price: ₹ 268 ~~₹ 298~~ Payment mode: UPI
Address: 123 Main Street, Foodville Pincode: 500001 Ordered on: 2024-06-21 Time:
status: order placed
Update order status
Update
Cancel


localhost:3000/auth
OrderOnTheGo(SB Foods)
Search Restaurants, cuisine, etc.,
Login
Register
Username
bavya
Email address
bavyatatineniu005@gamil.com
Password
•••••
Customer
Sign up
Already registered? Login



10. Testing

- **Manual testing** was done by using the app (register, login, cart, order flow).
- **Postman** was used to test backend APIs.
- **Browser DevTools** helped inspect React components and API requests.

11. Screenshots or Demo

https://drive.google.com/drive/folders/1OOS9KjxfeMRJcuTkFtbiu0xEAoSQyVV1?usp=sharing

12 . Known Issues

- Lack of Authentication Tokens: The login system does not implement JWT (JSON Web Tokens) or session-based authentication, resulting in less secure user sessions.

- No Order History Feature: Users are currently unable to view a history of their past orders after placing them.

- Cart Volatility: The shopping cart is stored in the browser's localStorage, which means it resets when the user logs out or clears browser data.

- Absence of Automated Testing: The application lacks automated testing

frameworks; all testing is performed manually.

- No Real-Time Order Updates: Changes made by the admin, such as order status updates, are not reflected in real-time on the user interface.

---

13. Future Enhancements

- Frontend Testing with Jest: Integrate the Jest testing framework to automate unit and component testing for frontend code.
- Backend API Testing with Supertest: Implement Supertest for comprehensive backend API testing.
- Payment Gateway Integration: Add secure payment processing using platforms such as Razorpay or Stripe.
- Role-Based Admin Access: Introduce role-based access control (RBAC) to manage admin permissions and enhance backend security.