**OrderOnTheGo: Your On-Demand Food Ordering Solution**

Team ID: LTVIP2025TMID59421

Team Leader: Tanneru Ravisaiteja

|  |  |
| --- | --- |
| **MEMBER** | **RESPONSIBILITY** |
| 1. Tanneru Ravisaiteja | Frontend,Resource Gathering |
| 2.Seetha Ram Poola | Database,Documentation |
| 3. Seeda Lohith Venkata Sai Charan | Backend,Resource Gathering |
| 4. Shaik Amina | Frontend ,Testing |

**1.INTRODUCTION**

Project Title:

OrderOnTheGo: Your On-Demand Food Ordering Solution

# Project Overview

Purpose: The QuickBite project is a full-stack food ordering web application developed using MERN stack. It provides a seamless platform for users to browse, order food from various restaurants, and for restaurants to manage their products and orders.This full-stack web application allows users to browse a wide selection of dishes, explore restaurant menus, check reviews, and place orders with just a few clicks. Whether it’s a late-night craving or a planned dinner, QuickBite ensures a fast, reliable, and user-friendly process for both customers and restaurant partners.

Features:

1. Comprehensive Product Catalog
2. Order Details Page
3. Secure and Efficient Checkout Process
4. Order Confirmation and Details
5. Restaurant Dashboard for Order and Product Management

# Architecture

# Frontend:

The frontend is the user interface of the SB Foods application, where users interact with the platform through

a browser. It is built using React.js, a modern JavaScript library known for creating dynamic and responsive

single-page applications.

* React Components: The app is structured using reusable components (e.g., Navbar, FoodItem, Cart).
* State Management: Context API or Redux manages global states like user authentication and cart data.
* API Integration: Axios handles HTTP requests to communicate with the backend.

Backend: Developed using Node.js and Express.js for handling business logic, routing, and API endpoints.

* RESTful APIs: Modular and organized routes to manage users, products, orders, and authentication.
* Authentication: Uses JWT (JSON Web Tokens) to securely handle login sessions for users, restaurants, and admins.
* Middleware: Includes validation, logging, security headers, and error handling to ensure smooth operation.

Database : MongoDB is used to store information about users, restaurants, products, carts, and orders. The database is where all application data is stored and retrieved from. SB Foods uses MongoDB, a popular NoSQL database, with Mongoose as an Object Data Modeling (ODM) library.

* Collections: Similar to tables in relational databases. Collections include Users, Products, Orders, Carts, Restaurants, and Admin.
* Schemas: Each collection follows a defined structure (schema) with specific fields.

**4.Setup Instructions**

Prerequisites:

* + - Node.js and npm
    - MongoDB
    - Git
  + Installation:

1. Clone the repository:
2. Navigate into the directory: cd Food-Ordering-App-MERN
3. Install dependencies: npm install
4. Start the app: npm run dev or npm start

# 5.Folder Structure

* + Client: Contains the React frontend with all components, pages, and styles.
  + Server: Contains the Express backend with routes, models, controllers, and middleware.

QuickBite Foods-App/

│

├── client/

│ ├── public/

│ └── src/

│ ├── components/

│ ├── pages/

│ ├── context/

│ ├── services/

│ ├── App.js

│ └── index.js

│

├── server/

│ ├── models/

│ ├── routes/

│ ├── controllers/

│ ├── middleware/

│ ├── config/

│ ├── app.js

│ └── .env

│

├── README.md

├── .gitignore

└── package.json

# Running the Application

Frontend:

Navigate to client directory

cd client

start the development server:

npm start

The frontend application runs on : http://localhost:3000

Backend:

Navigate to server directory

cd server

start the backend server:

run `npm start`

# API Documentation

Include endpoints for user authentication, product listing, cart operations, order processing, etc.

Authentication APIs:

These endpoints handle user registration and login, supporting both regular users and restaurant owners.

* Register API (POST /auth/register):  
  Allows new users or restaurants to create an account by providing name, email, password, and role.
* Login API (POST /auth/login):
* Authenticates existing users and returns a JWT token for session management
* User APIs:

These APIs allow users to manage their profiles, cart, and orders.

* Get Profile (GET /users/me):  
  Fetches details of the currently logged-in user.
* Cart APIs:
  + POST /users/cart – Add items to the user's cart.
  + GET /users/cart – View all items in the cart.
  + DELETE /users/cart/:itemId – Remove an item from the cart.

Product (Food Item) APIs:

These APIs allow users to browse food items and restaurants to manage their menu.

* List Products (GET /products):  
  Retrieves all available food items on the platform.
* View Product (GET /products/:id):  
  Gets details of a specific food item.
* Add/Update/Delete Products (Restaurants only):
  + POST /products – Add new food items.
  + PUT /products/:id – Edit existing item.
  + DELETE /products/:id – Remove an item.

# Authentication

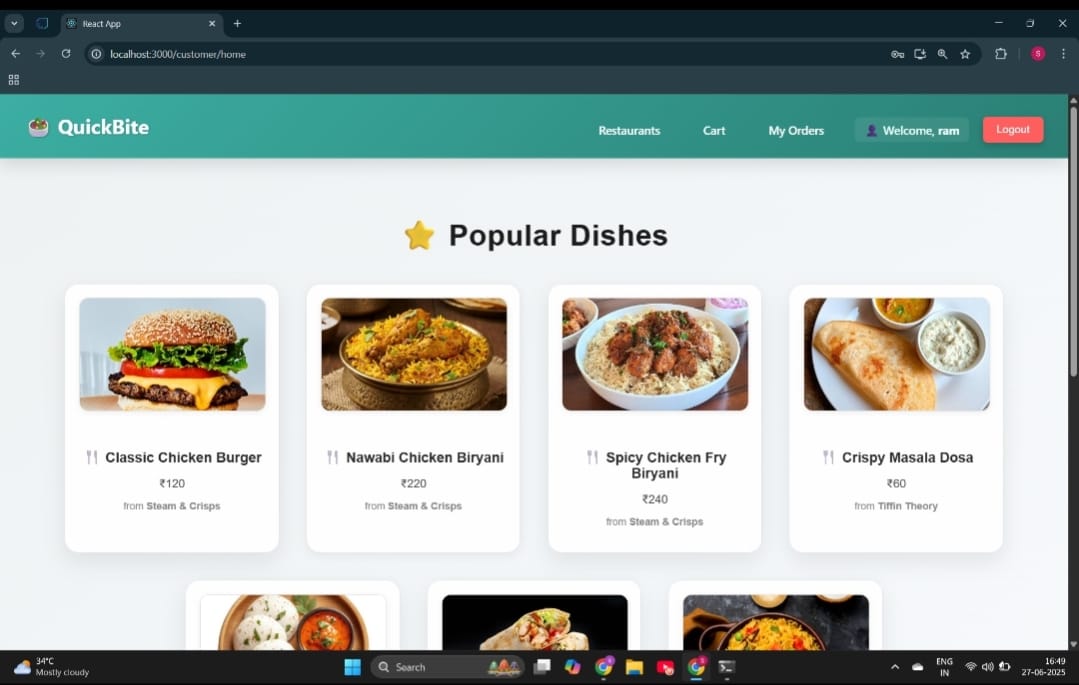
# Authentication ensures that only verified users (customers, restaurants, and admins) can access and perform actions within the application. It protects sensitive data, enforces role-based access, and maintains secure sessions for all users.

* Express.js – to build REST APIs and use middleware.
* Mongoose – to manage user data and roles in MongoDB.
* User: Browse food, manage cart, place orders, view history.
* Restaurant: Manage menu, handle orders.
* Admin: Approve restaurants, manage platform data.
* Endpoint: POST /auth/register
* User provides name, email, password, role.
* Password is hashed using bcrypt.
* Data is stored in the User collection.

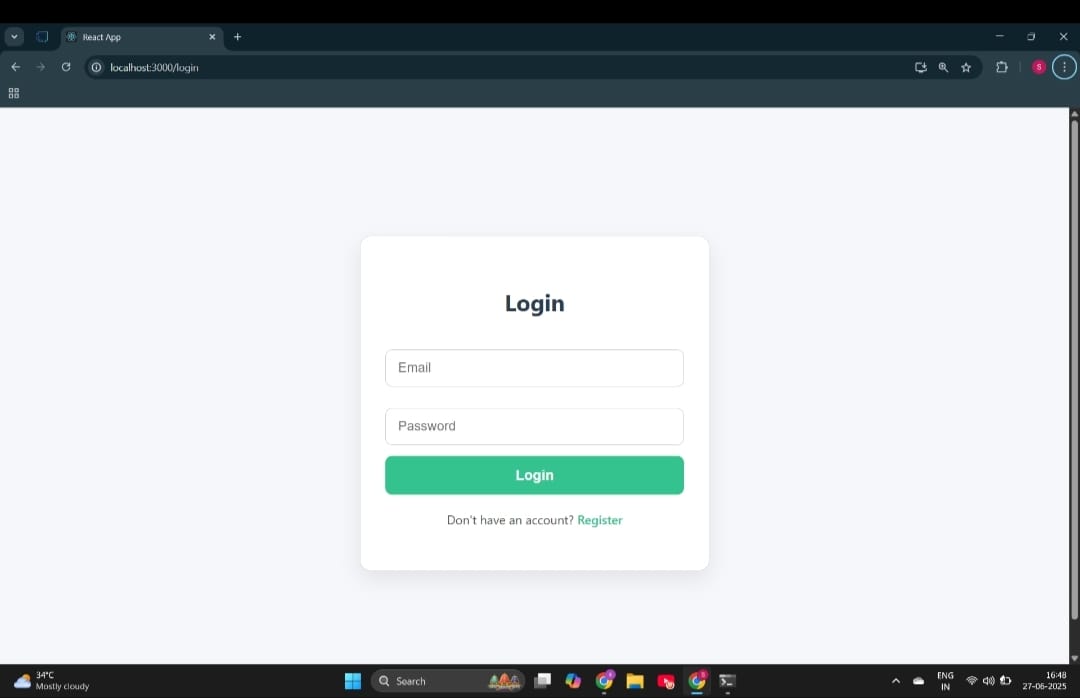
# Screenshots or Demo

* 1. Add screenshots or a video walkthrough link here to demonstrate the application in action.

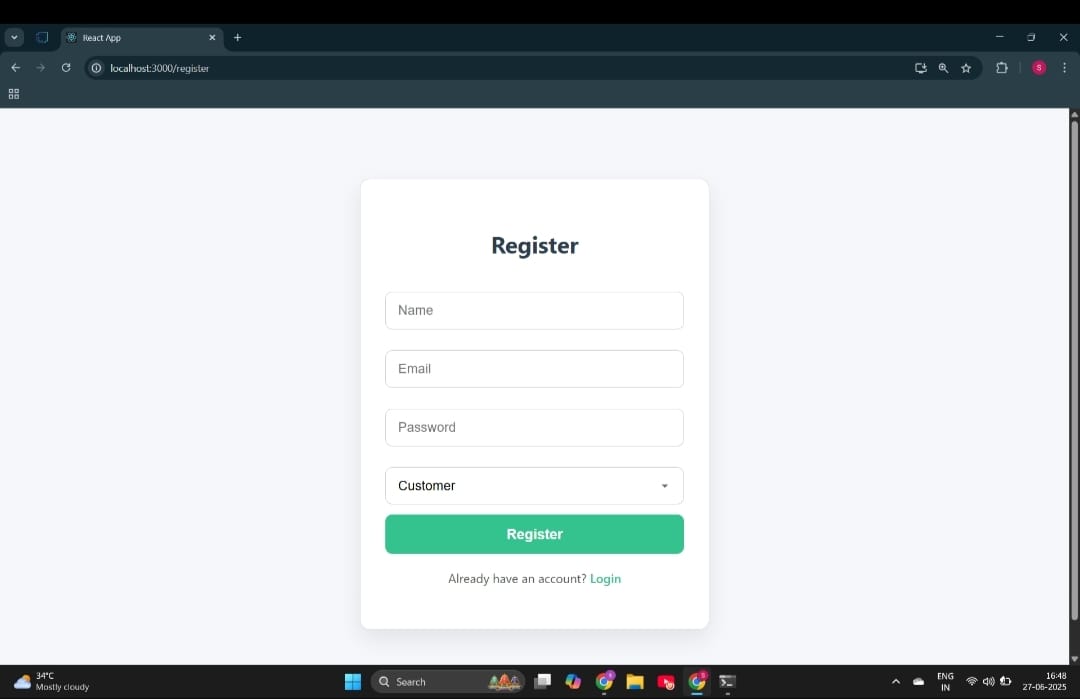
Home Page:



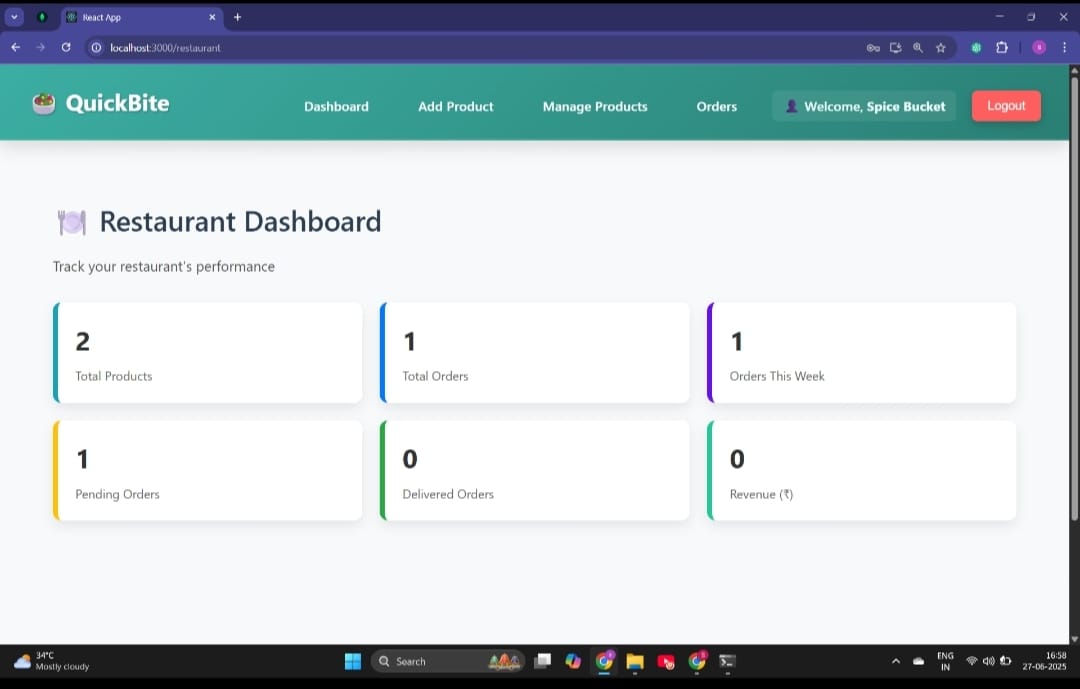
Login Page:



Register Page:



Restaurant Dashboard:



# Testing

Some minor UI responsiveness issues on older mobile browsers.

* Validate props, state changes, and UI rendering.
* CartPage updates on adding a product.
* OrderPage reacts to cart changes.
* Ensure API endpoints return correct responses.
* Validate data logic, authentication, and security.
* Catch edge-case failures before deployment.
* Test individual components like Navbar, FoodCard, LoginForm.
* Test controller functions independently:
  + - registerUser(), createOrder(), addToCart()
    - Mock database with dummy data.

1. **Known Issues**

* Order status updates (e.g., “Preparing”, “Delivered”) require manual page refresh.
* Some dashboard pages (especially Admin and Restaurant views) may not display well on smaller screens.
* Orders are marked as “Paid” without real transaction processing.
* Restaurants uploading food images cannot preview them before submission.

1. **Future Enhancements**

* Enable live tracking of delivery agents using Google Maps API or location sharing.
* Enhances user experience and transparency.
* Support for UPI, card payments, and wallets.
* Notify users and restaurants of order status changes, new orders, offers, etc.
* Use Firebase Cloud Messaging (FCM) or OneSignal.Improves performance on slow connections and mobile devices.