MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE

Kodambakkam, Chennai-600024

(An Autonomous Institution)

NM1042 – MERN STACK POWERED BY MONGODB

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

PROJECT TITLE: ONLINE FOOD ORDERING APPLICATION

FACULTY MENTOR: Mrs. MUTHULAKSHMI

Project submitted by,

NAAN MUDHALVAN ID	NAME	REGISTER NUMBER
DF72B677139D56D8E034 CEACB2CFC92D	*KRITHIKA PRIYA PR*	*311521243027*
3F7114219A2277BB0B549 51740EC9C9C	*SANDHYA ANAND*	*311521243043*
C29279220EBB9D5A5A04 154E988D8969	*JOTHIKA B*	*311521243019*
A678B3C4BFD45D488364 491B43F75DE4	*DEEPIKA E*	*311521243009*

TABLE OF CONTENTS

S.NO	TITLE
1	INTRODUCTION
2	PROJECT OVERVIEW
3	ARCHITECTURE
4	SETUP INSTRUCTONS
5	FOLDERS STRUCTURE
6	RUNNING THE APPLICATION
7	API DOCUMENTATION
8	AUTHENTICATION
9	TESTING
10	OUTPUTS SCREENSHOTS
11	KNOWN ISSUES
12	FUTURE ENHANCEMENTS
13	CONCLUSION

INTRODUCTION

Project Title: Online Food Ordering Application

Team Members & Role:

1.Krithika Priya.PR - Frontend Developer
2.Sandhya Anand - Backend Developer
3.Jothika.B - Database Developer
4.Deepika.E - DevOps Engineer

Objective:

The Online Food Ordering Application aims to provide an end-to-end solution for customers to browse menus, place food orders, and track them online. It also includes an admin panel where restaurant owners or administrators can manage menus, orders, and customers effectively. The project uses the MERN stack, leveraging MongoDB for the database, Express.js and Node.js for the backend, and React.js for the frontend.

Core focus:

This project was developed with scalability, user-friendly design, and real-world applicability in mind. It prioritizes secure user authentication, smooth navigation, and an intuitive admin panel for effortless order and menu management.

PROJECT OVERVIEW

Purpose:

The food ordering app addresses the need for seamless online food ordering and restaurant management. Customers can easily browse restaurant menus, customize their orders, and track delivery. Simultaneously, restaurant owners can streamline their operations with robust order and menu management functionalities.

Goals:

- Simplify the process of ordering food online.
- Provide restaurants with tools for efficient operation management.
- user data protection.

Key Features:

- **User Authentication**: Sign up, login, and profile management with roles for customers, restaurant owners, and delivery drivers.
- **Restaurant Listings**: Searchable restaurant profiles with menu items, filters, and ratings.
- **Order Management**: Add items to cart, customize orders, and process payments securely with multiple methods (credit card, PayPal, COD).
- **Real-Time Tracking**: Track order status and delivery progress on a live map.
- **Reviews & Ratings**: Rate restaurants and food items, leave feedback, and interact with restaurant owners.
- **Admin Panel**: Manage restaurant profiles, orders, and user accounts with analytics and reporting.
- **Delivery Driver Management**: Drivers can accept orders, update statuses, and deliver items.
- **Push Notifications**: Order updates, delivery status, and promotional alerts.

ARCHITECTURE

Frontend Architecture (React.js):

1. **Component-based architecture**: Utilizes **React.js** to build reusable and modular components for easy maintenance and scalability.
2. **State Management**: Manages complex states using **Redux** or **Context API** for smooth data flow and interaction between components (e.g., cart updates, user login status).
3. **Responsive UI**: Uses **CSS frameworks** like **Tailwind CSS** or **Bootstrap** to create a responsive, user-friendly design across devices.
4. **API Communication**: The frontend interacts with the backend through **RESTful APIs** to fetch and display data in real time.

Backend Architecture (Node.js and Express.js):

1. **Node.js & Express.js**: Developed with **Node.js** for high performance and **Express.js** for routing and middleware management.

2. **RESTful API**: Backend structured around **RESTful APIs** for user authentication, order management, restaurant listings, and payment processing.

3. **Middleware**: Uses middleware for **request validation**, **authentication** (JWT or OAuth), and **error handling** to ensure security and smooth user experience.

4. **Third-party Integrations**: Integrates services like **Stripe** for payment processing and push notification services for real-time updates.

Database Architecture (MongoDB):

1. **MongoDB (NoSQL)**: Uses **MongoDB** as a NoSQL database to store both structured and semi-structured data.

2. **Key Collections**:

   - **Users**: Stores user data (name, email, hashed passwords, role, and order history).

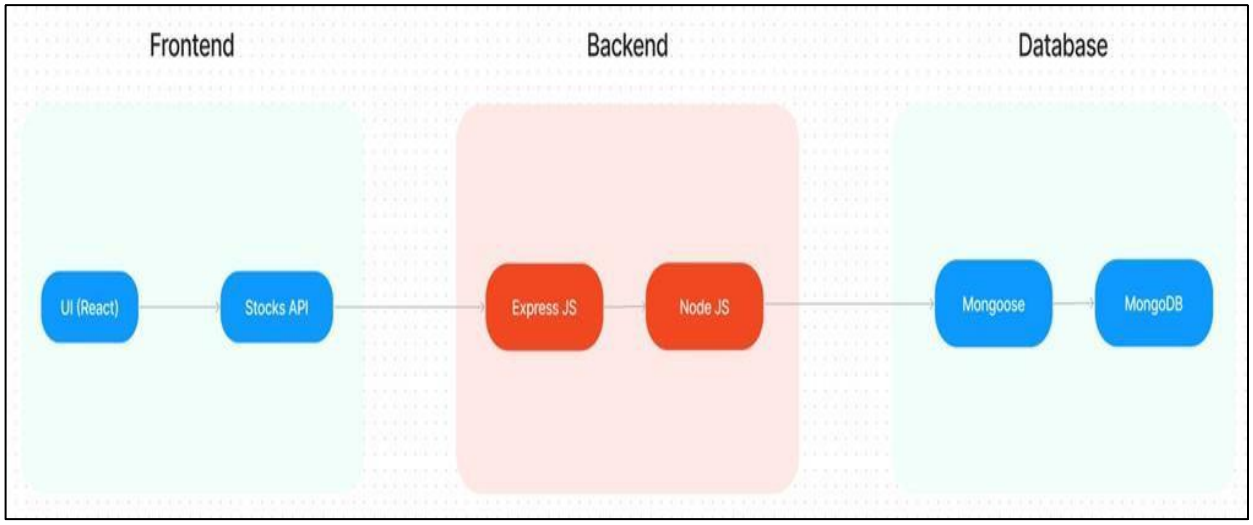
   - **Foods**: Stores menu details (name, category, price, availability, and customization options).

   - **Orders**: Stores order details (order ID, user ID, items, total price, delivery address, and order status).

3. **Mongoose ORM**: Uses **Mongoose** for schema definition, data validation, and efficient querying/updating of collections.

4. **Flexible Schema**: MongoDB's flexible schema design allows easy handling of varying data types like user preferences and menu variations.

Technical Architecture:



SETUP INSTRUCTIONS

Prerequisites:

- Node.js (v14.x or higher)
- MongoDB (local installation or Atlas cloud)
- npm or yarn (package managers)

Installation Steps:

1. Clone the repository:

```
git clone https://github.com/username/food-ordering-app.git
cd food-ordering-app
```

2. Install dependencies for the frontend:

```
cd client
npm install
```

3. Install dependencies for the backend:

```
cd ../server
npm install
```

4. Configure environment variables in a .env file inside the server directory:

```
MONGO_URI=your-mongodb-uri
JWT_SECRET=your-jwt-secret
PORT=5000
```

5. Start the development servers:

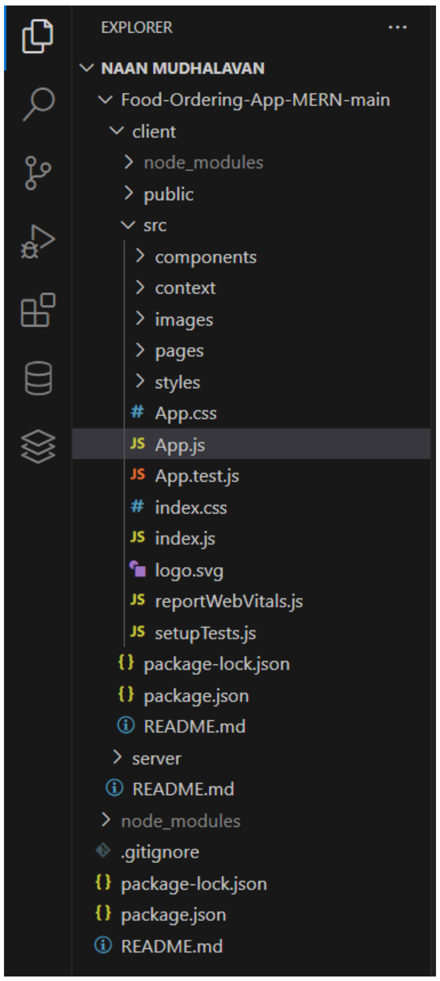
Frontend:

```
cd client
npm start
```

Backend:

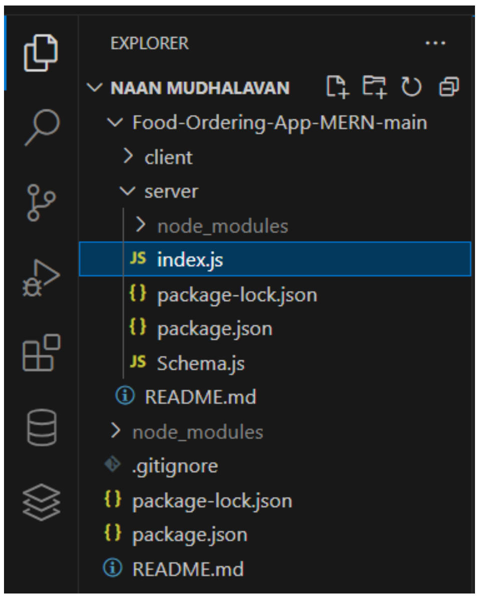
```
cd server
npm start
```

FOLDER STRUCTURE



```
EXPLORER
NAAN MUDHALAVAN
  Food-Ordering-App-MERN-main
    client
      node_modules
      public
      src
        components
        context
        images
        pages
        styles
        App.css
        App.js
        App.test.js
        index.css
        index.js
        logo.svg
        reportWebVitals.js
        setupTests.js
      package-lock.json
      package.json
      README.md
    server
    README.md
  node_modules
  .gitignore
  package-lock.json
  package.json
  README.md
```

Client Folder Structure



```
EXPLORER
NAAN MUDHALAVAN
  Food-Ordering-App-MERN-main
    client
    server
      node_modules
      index.js
      package-lock.json
      package.json
      Schema.js
    README.md
  node_modules
  .gitignore
  package-lock.json
  package.json
  README.md
```

Server Folder Structure

RUNNING THE APPLICATION

Commands to Start Locally:

1. Frontend:

- ➢ Navigate to the client directory and run:
- ➢ Cd client
- ➢ npm start
- ➢ This will launch the React app on http://localhost:3000.

2. Backend:

- ➢ Navigate to the server directory and run:
- ➢ Cd server
- ➢ npm start
- ➢ This will start the Node.js server on http://localhost:5000.

API DOCUMENTATION

Base URL

Development: http://localhost:5000/api
Production: https://your-production-url.com/api

AUTHENTICATION

1. Authentication

Authentication is the process of verifying the identity of a user.

- **User Registration**: When a new user registers, they provide basic information (like name, email, and password). The password is securely stored after being hashed (not stored in plain text).

- **User Login**: Upon login, the system checks if the provided email and password match the stored records. If they do, the system generates a **JSON Web Token (JWT)** for that user. This token contains crucial information such as the user's ID and role, and is used to verify their identity in subsequent requests.
- **Token Storage**: After a successful login, the generated JWT is sent to the client, where it is stored (typically in local storage or cookies). This token is then included in the headers of subsequent API requests to authenticate the user.
- **Token Expiration**: JWTs typically have an expiration time (e.g., 1 hour). When the token expires, the user will need to log in again to get a new token.

2. Authorization

Authorization determines what actions or data a user can access after they are authenticated.

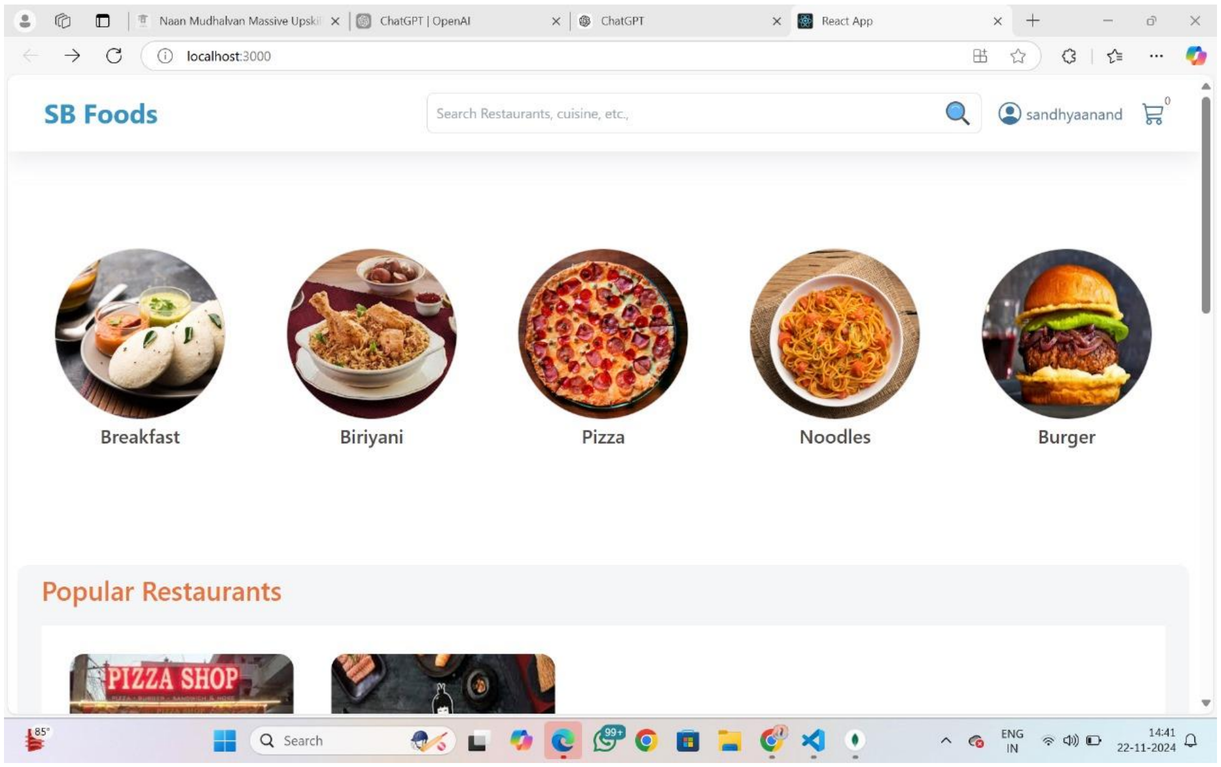
- **Role-Based Access Control (RBAC)**: The app assigns users specific roles (such as **customer**, **restaurant owner**, or **admin**). Each role has different levels of access:
    - **Customers** can browse menus, place orders, and view their order history.
    - **Restaurant owners** can manage their restaurant's menu and orders but cannot access customer data.
    - **Admins** have full control, managing all users, restaurants, and orders.
- **Permission Validation**: When a user attempts to access a specific resource or perform an action (like viewing another user's order or updating a restaurant menu), the system checks the user's role (stored in the JWT) to ensure they have the necessary permissions.

3. Token Validation

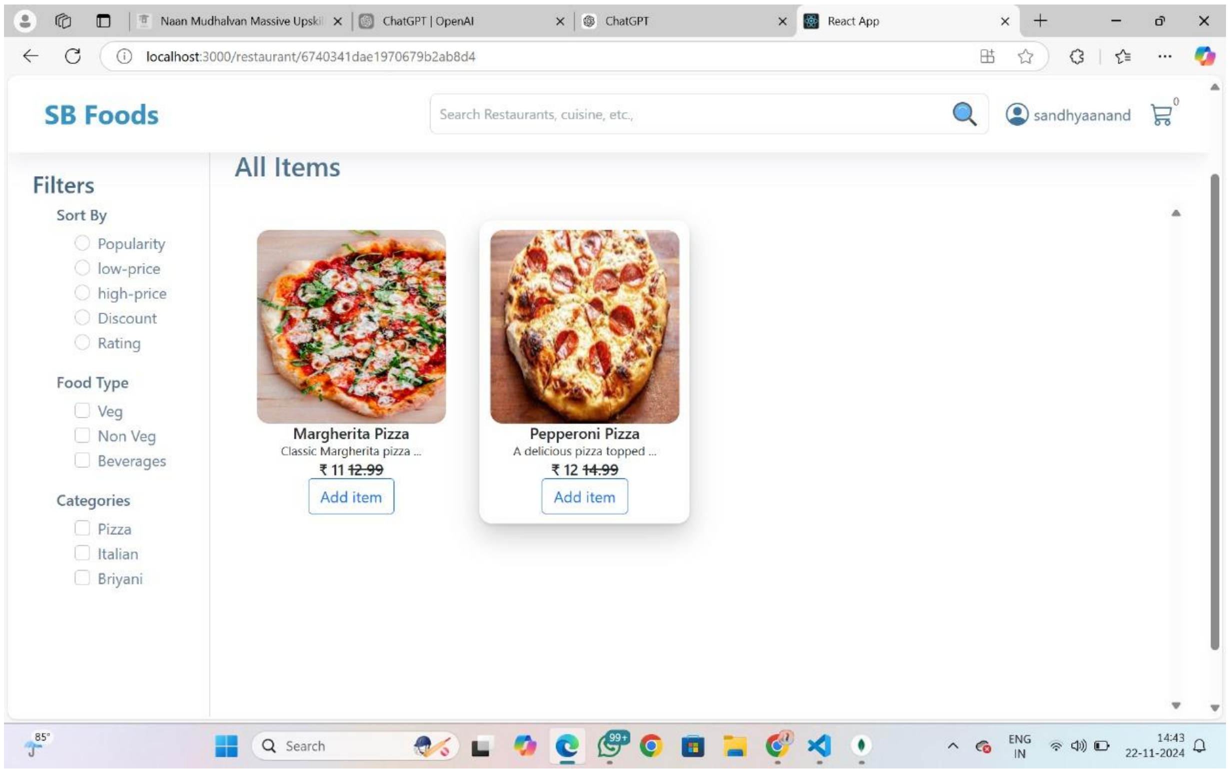
- **Middleware**: Every request to a protected resource must include the JWT. A middleware function on the backend checks if the token is present and valid before allowing the user to proceed with the request.
- If the token is invalid or expired, the request is rejected, and the user is prompted to log in again.

USER INTERFACE

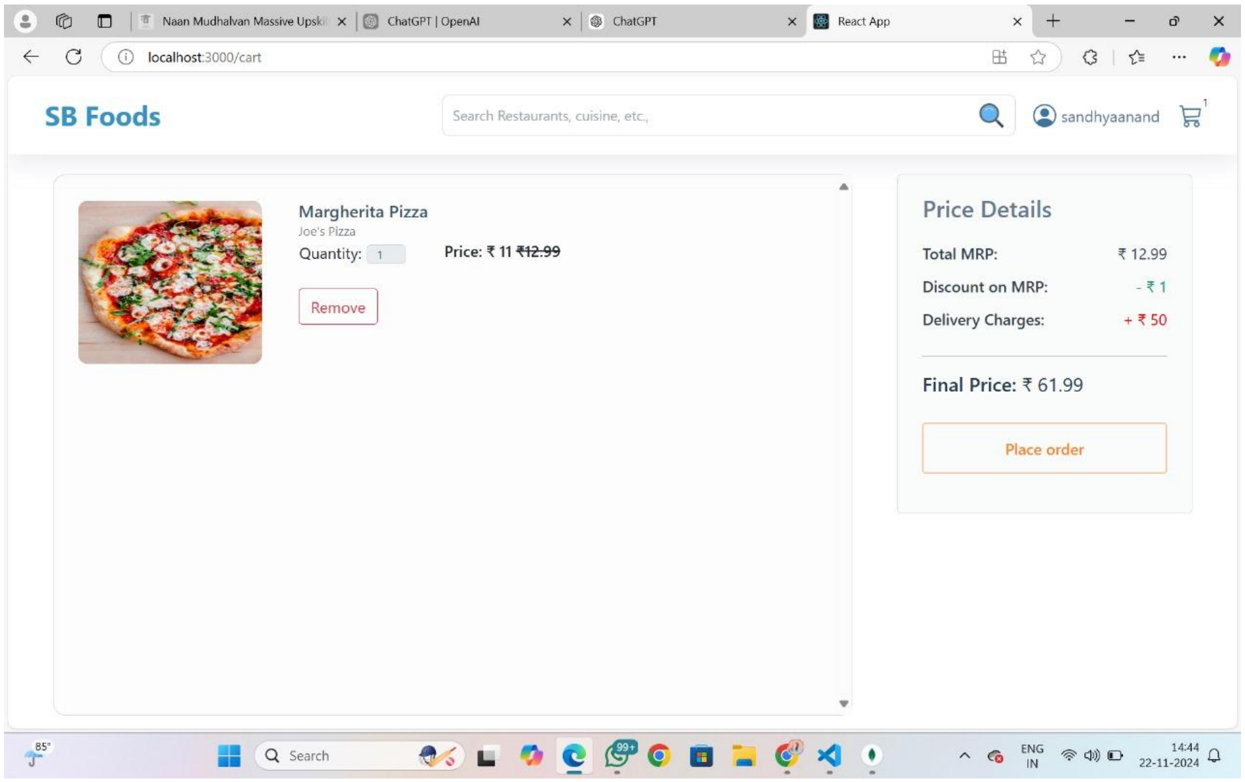
Login Page: For user authentication.



Menu Page: Displays food items with filter and search options.



Cart Page: Allows users to view selected items and total price.


Naan Mudhalvan Massive Upskil | ChatGPT | OpenAI | ChatGPT | React App

localhost:3000/cart

SB Foods

Search Restaurants, cuisine, etc.,

sandhyaanand 1

Margherita Pizza
Joe's Pizza
Quantity: 1 Price: ₹ 11 ~~₹12.99~~

Remove

Price Details

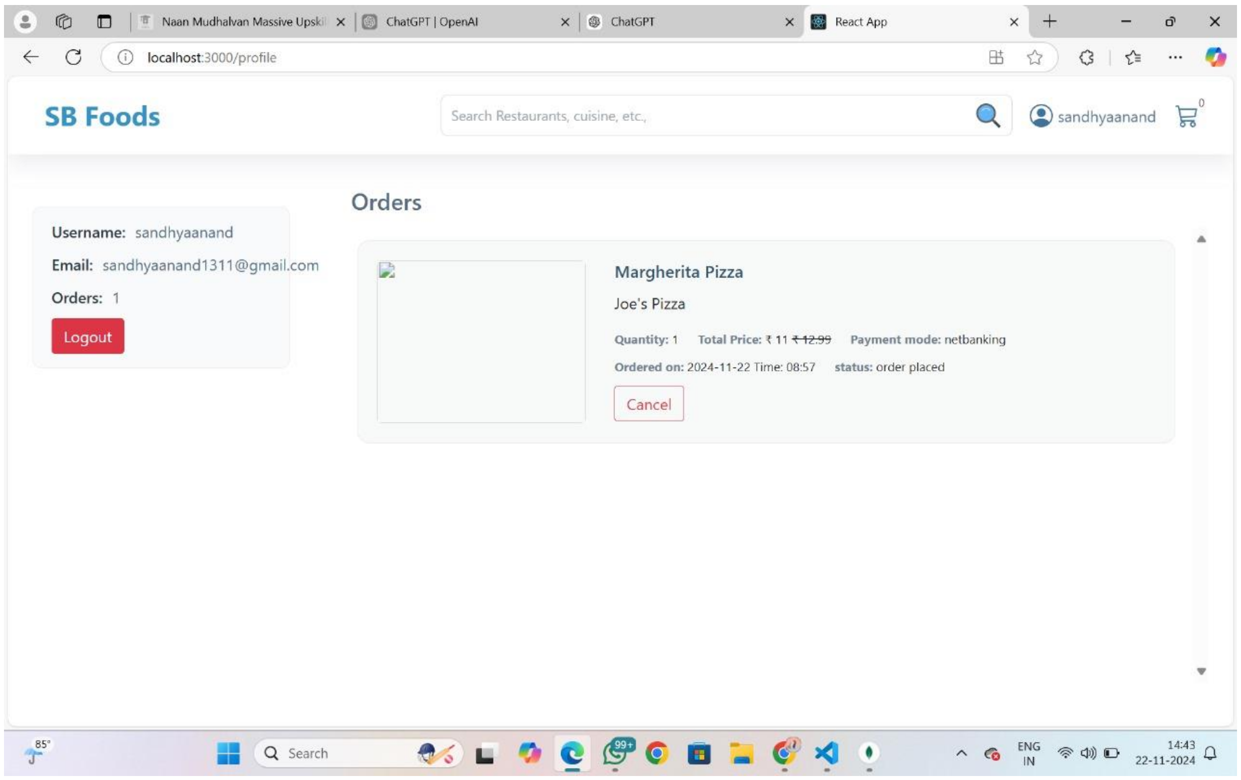
Total MRP: ₹ 12.99
Discount on MRP: - ₹ 1
Delivery Charges: + ₹ 50

Final Price: ₹ 61.99

Place order

85° Search ENG IN 14:44 22-11-2024

Order Tracking Page: Displays the order status.


Naan Mudhalvan Massive Upskil | ChatGPT | OpenAI | ChatGPT | React App

localhost:3000/profile

SB Foods

Search Restaurants, cuisine, etc.,

sandhyaanand 0

Username: sandhyaanand
Email: sandhyaanand1311@gmail.com
Orders: 1

Logout

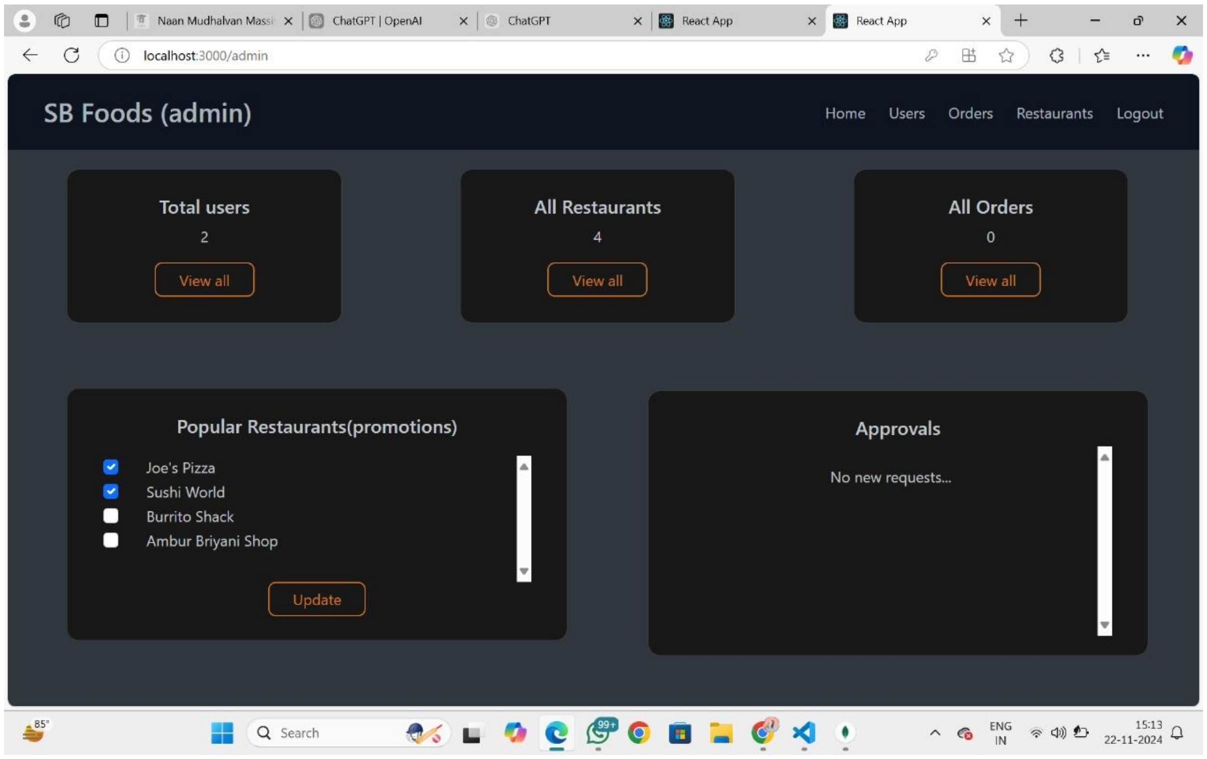
Orders

Margherita Pizza
Joe's Pizza
Quantity: 1 Total Price: ₹ 11 ~~₹ 12.99~~ Payment mode: netbanking
Ordered on: 2024-11-22 Time: 08:57 status: order placed

Cancel

85° Search ENG IN 14:43 22-11-2024

Admin Panel: For managing menu and order



TESTING

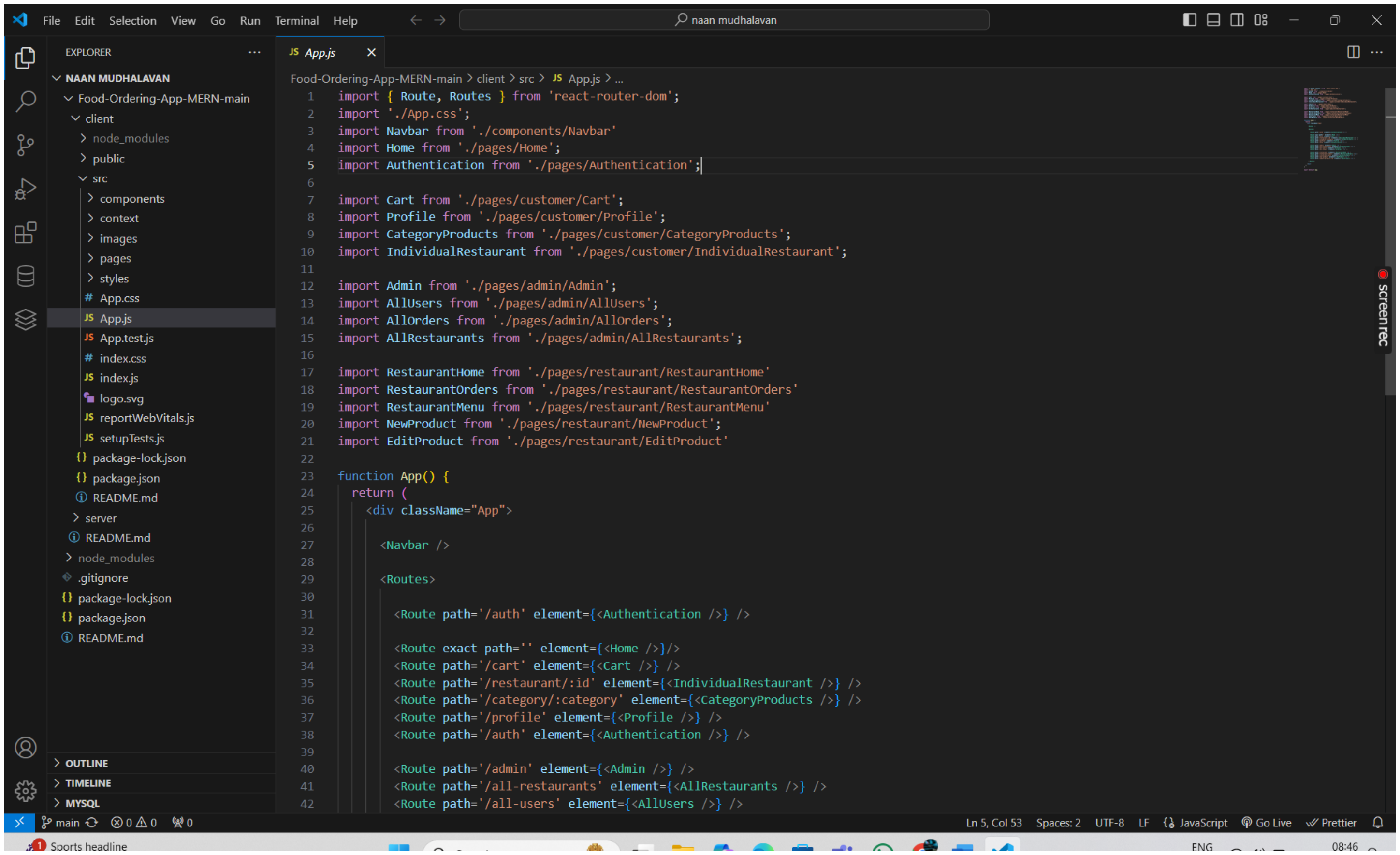
The testing strategy for the Food Delivery App focuses on ensuring functionality, security, and performance through multiple testing approaches:

1. **Frontend Testing (React)**:
   - **Unit Testing**: Using **Jest** and **React Testing Library** to test individual components.
   - **End-to-End Testing**: Using **Cypress** to simulate real user interactions and test workflows.
2. **Backend Testing (Node.js/Express)**:
   - **Unit Testing**: Using **Jest** and **Supertest** to test individual API routes and backend logic.
   - **Integration Testing**: Ensuring modules and services work together, including database interactions.
3. **Security Testing**:
   - Automated vulnerability scans using **OWASP ZAP** to detect common security issues like XSS or SQL injection.

4. **Performance Testing**:
   - **Load and Stress Testing**: Using **Artillery** or **Apache JMeter** to test app performance under high traffic.
5. **CI/CD Integration**:
   - Automated testing and deployment using **GitHub Actions** or **Jenkins** to ensure continuous code quality.

SCREENSHOTS

Client Folder Structure

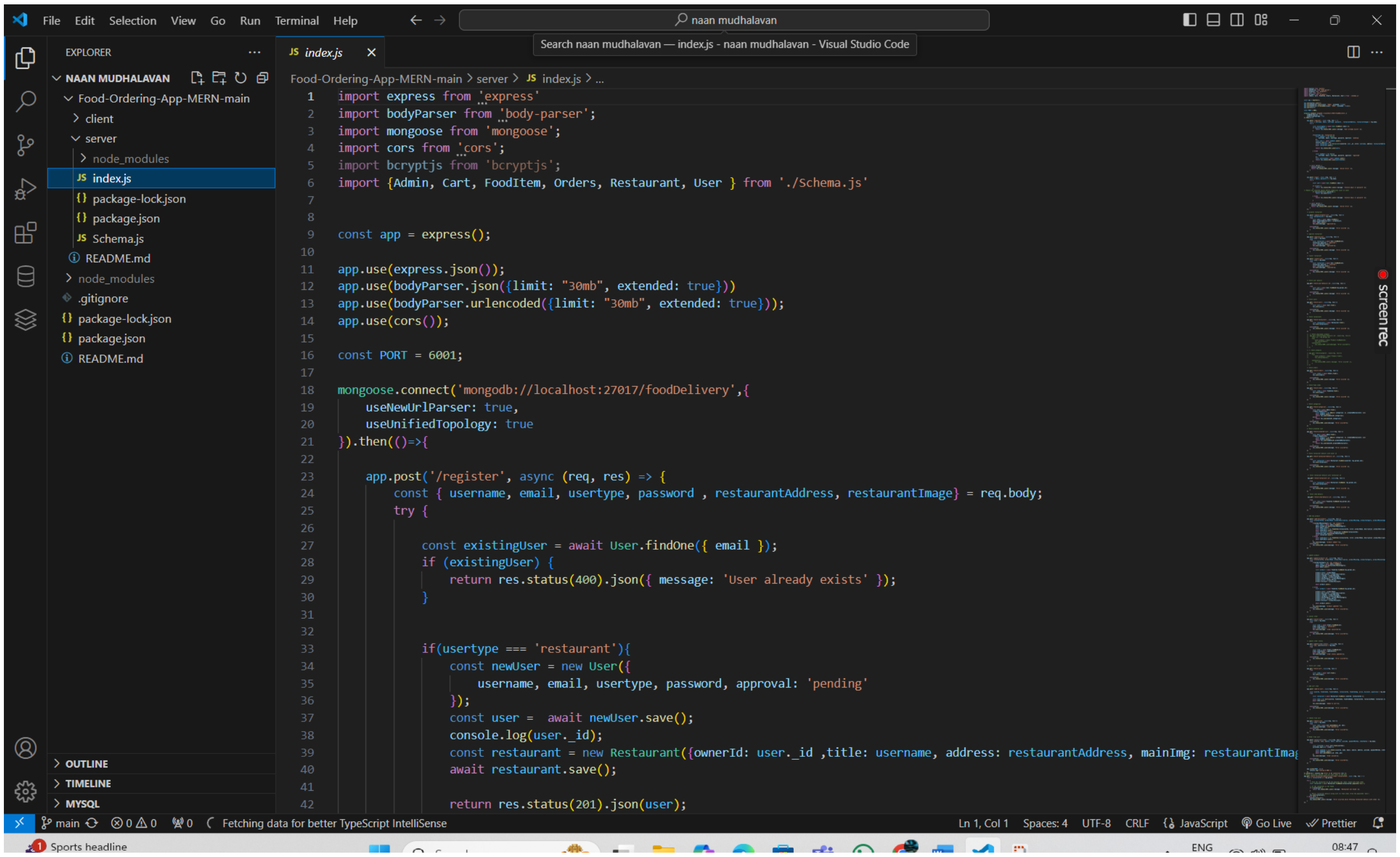

File Edit Selection View Go Run Terminal Help
naan mudhalavan
EXPLORER
App.js
OUTLINE
TIMELINE
MYSQL
main
Ln 5, Col 53 Spaces: 2 UTF-8 LF JavaScript Go Live Prettier
screenrec
Sports headline
ENG
08:46

```
NAAN MUDHALAVAN
  Food-Ordering-App-MERN-main
    client
      node_modules
      public
      src
        components
        context
        images
        pages
        styles
        App.css
        App.js
        App.test.js
        index.css
        index.js
        logo.svg
        reportWebVitals.js
        setupTests.js
      package-lock.json
      package.json
      README.md
    server
    README.md
  node_modules
  .gitignore
  package-lock.json
  package.json
  README.md
```

Food-Ordering-App-MERN-main > client > src > App.js > ...

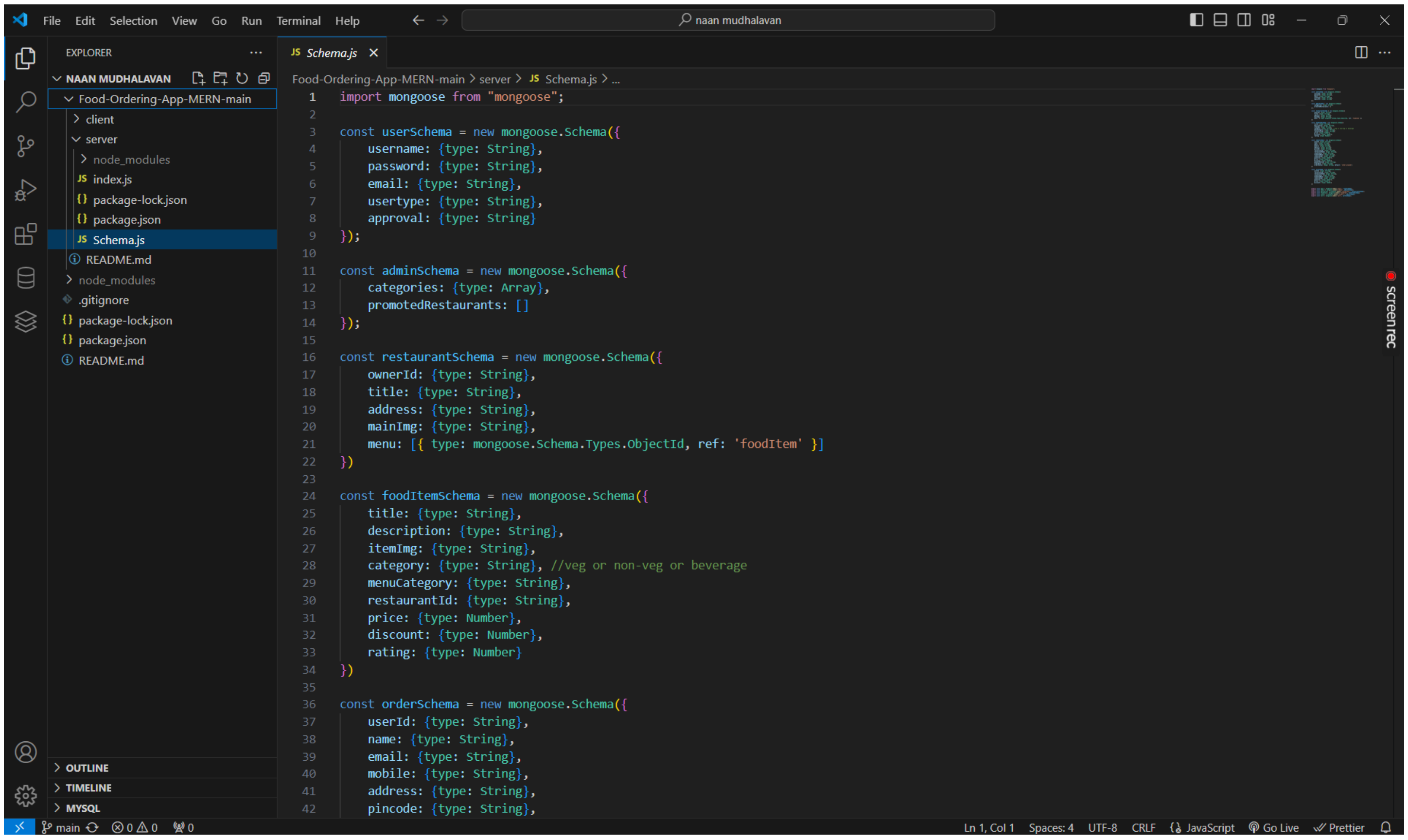
```
import { Route, Routes } from 'react-router-dom';
import './App.css';
import Navbar from './components/Navbar'
import Home from './pages/Home';
import Authentication from './pages/Authentication';

import Cart from './pages/customer/Cart';
import Profile from './pages/customer/Profile';
import CategoryProducts from './pages/customer/CategoryProducts';
import IndividualRestaurant from './pages/customer/IndividualRestaurant';

import Admin from './pages/admin/Admin';
import AllUsers from './pages/admin/AllUsers';
import AllOrders from './pages/admin/AllOrders';
import AllRestaurants from './pages/admin/AllRestaurants';

import RestaurantHome from './pages/restaurant/RestaurantHome'
import RestaurantOrders from './pages/restaurant/RestaurantOrders'
import RestaurantMenu from './pages/restaurant/RestaurantMenu'
import NewProduct from './pages/restaurant/NewProduct';
import EditProduct from './pages/restaurant/EditProduct'

function App() {
  return (
    <div className="App">

      <Navbar />

      <Routes>

        <Route path='/auth' element={<Authentication />} />

        <Route exact path='' element={<Home />}/>
        <Route path='/cart' element={<Cart />} />
        <Route path='/restaurant/:id' element={<IndividualRestaurant />} />
        <Route path='/category/:category' element={<CategoryProducts />} />
        <Route path='/profile' element={<Profile />} />
        <Route path='/auth' element={<Authentication />} />

        <Route path='/admin' element={<Admin />} />
        <Route path='/all-restaurants' element={<AllRestaurants />} />
        <Route path='/all-users' element={<AllUsers />} />
```

Server Folder Structure


File Edit Selection View Go Run Terminal Help
naan mudhalavan
Search naan mudhalavan — index.js - naan mudhalavan - Visual Studio Code
EXPLORER
NAAN MUDHALAVAN
Food-Ordering-App-MERN-main
client
server
node_modules
index.js
package-lock.json
package.json
Schema.js
README.md
node_modules
.gitignore
package-lock.json
package.json
README.md
OUTLINE
TIMELINE
MYSQL
Food-Ordering-App-MERN-main > server > index.js > ...
main ⊗ 0 ⚠ 0 Fetching data for better TypeScript IntelliSense
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF JavaScript Go Live Prettier
Sports headline
ENG 08:47

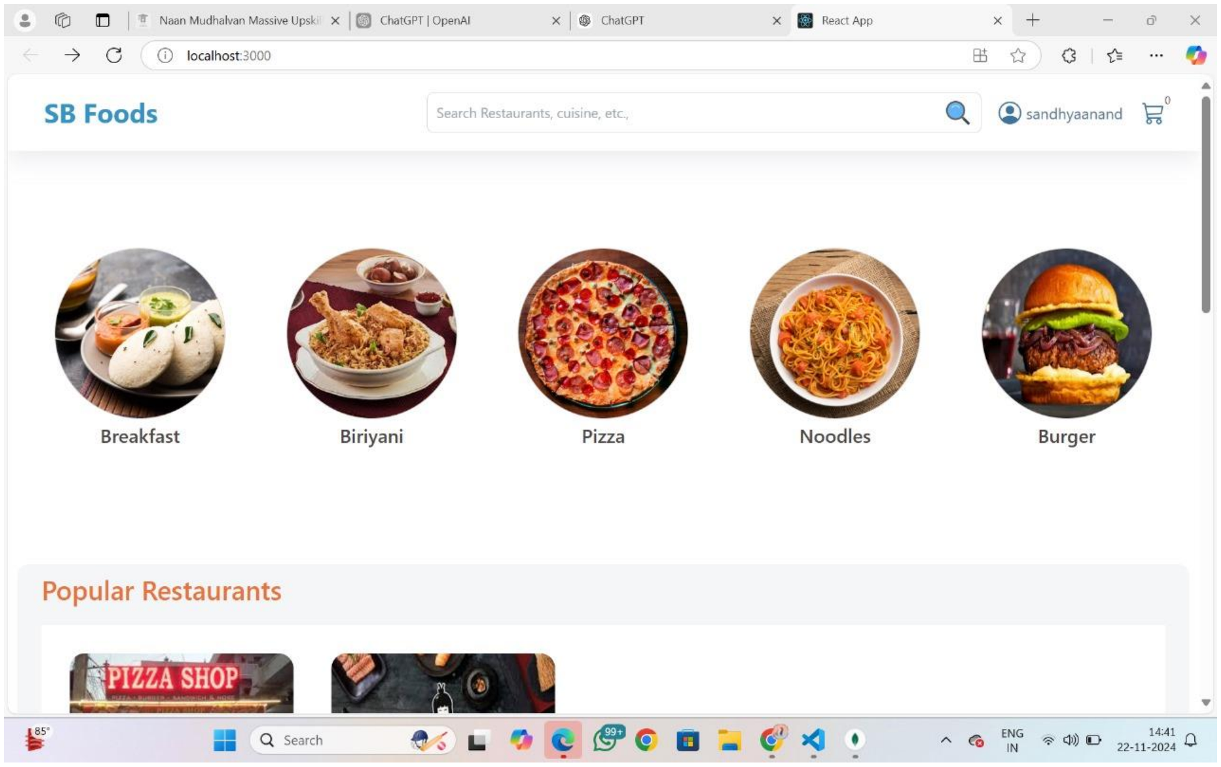
```
import express from 'express'
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import cors from 'cors';
import bcryptjs from 'bcryptjs';
import {Admin, Cart, FoodItem, Orders, Restaurant, User } from './Schema.js'


const app = express();

app.use(express.json());
app.use(bodyParser.json({limit: "30mb", extended: true}))
app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
app.use(cors());

const PORT = 6001;

mongoose.connect('mongodb://localhost:27017/foodDelivery',{
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{

    app.post('/register', async (req, res) => {
        const { username, email, usertype, password , restaurantAddress, restaurantImage} = req.body;
        try {

            const existingUser = await User.findOne({ email });
            if (existingUser) {
                return res.status(400).json({ message: 'User already exists' });
            }


            if(usertype === 'restaurant'){
                const newUser = new User({
                    username, email, usertype, password, approval: 'pending'
                });
                const user =  await newUser.save();
                console.log(user._id);
                const restaurant = new Restaurant({ownerId: user._id ,title: username, address: restaurantAddress, mainImg: restaurantImag
                await restaurant.save();

                return res.status(201).json(user);
```

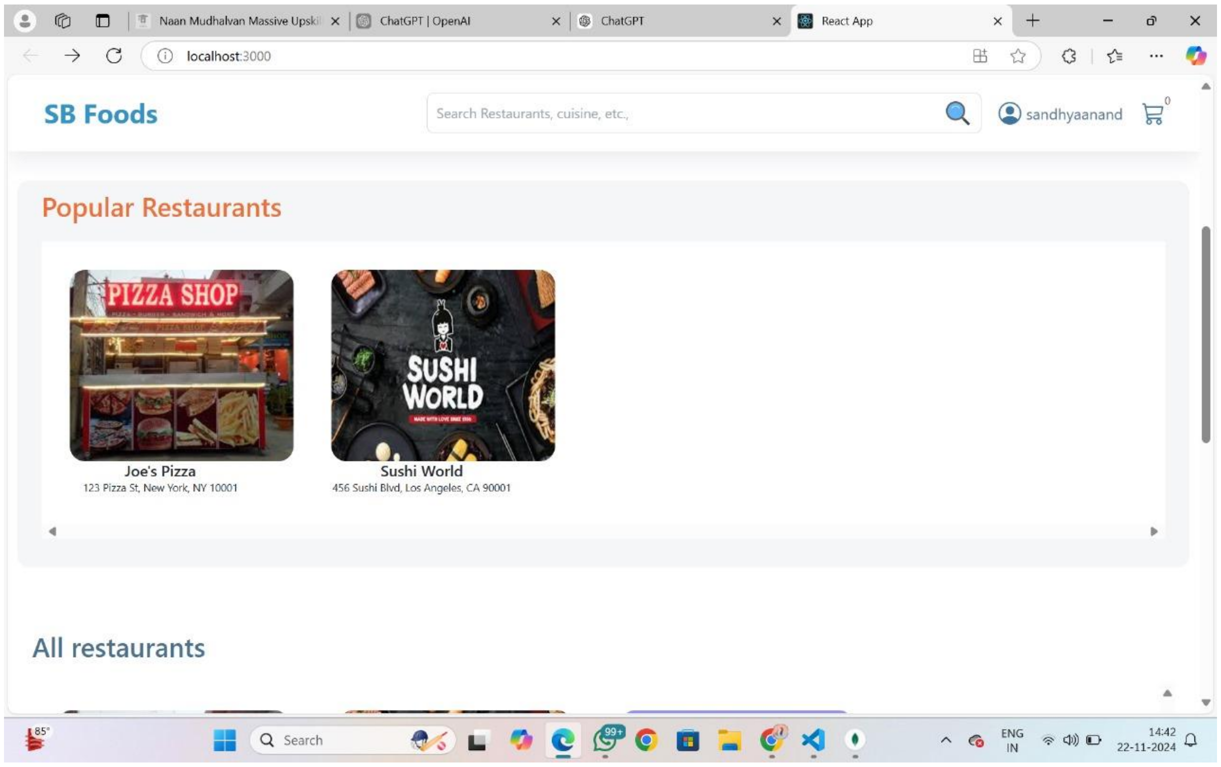
Schema

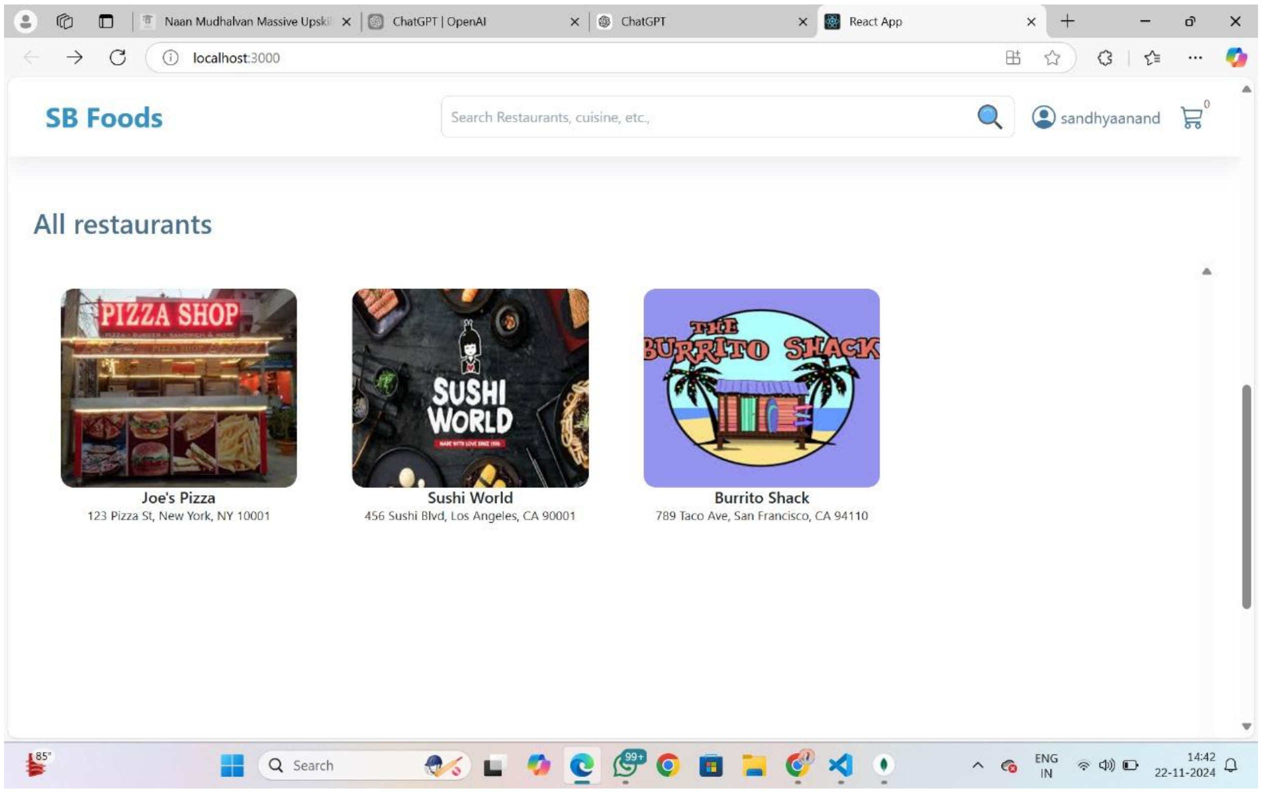


```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema({
    username: {type: String},
    password: {type: String},
    email: {type: String},
    usertype: {type: String},
    approval: {type: String}
});

const adminSchema = new mongoose.Schema({
    categories: {type: Array},
    promotedRestaurants: []
});

const restaurantSchema = new mongoose.Schema({
    ownerId: {type: String},
    title: {type: String},
    address: {type: String},
    mainImg: {type: String},
    menu: [{ type: mongoose.Schema.Types.ObjectId, ref: 'foodItem' }]
})

const foodItemSchema = new mongoose.Schema({
    title: {type: String},
    description: {type: String},
    itemImg: {type: String},
    category: {type: String}, //veg or non-veg or beverage
    menuCategory: {type: String},
    restaurantId: {type: String},
    price: {type: Number},
    discount: {type: Number},
    rating: {type: Number}
})

const orderSchema = new mongoose.Schema({
    userId: {type: String},
    name: {type: String},
    email: {type: String},
    mobile: {type: String},
    address: {type: String},
    pincode: {type: String},
```

User Page:


Naan Mudhalvan Massive Upskil
ChatGPT | OpenAI
ChatGPT
React App
localhost:3000
SB Foods
Search Restaurants, cuisine, etc.,
sandhyaanand
Breakfast
Biriyani
Pizza
Noodles
Burger
Popular Restaurants
Search
14:41
22-11-2024

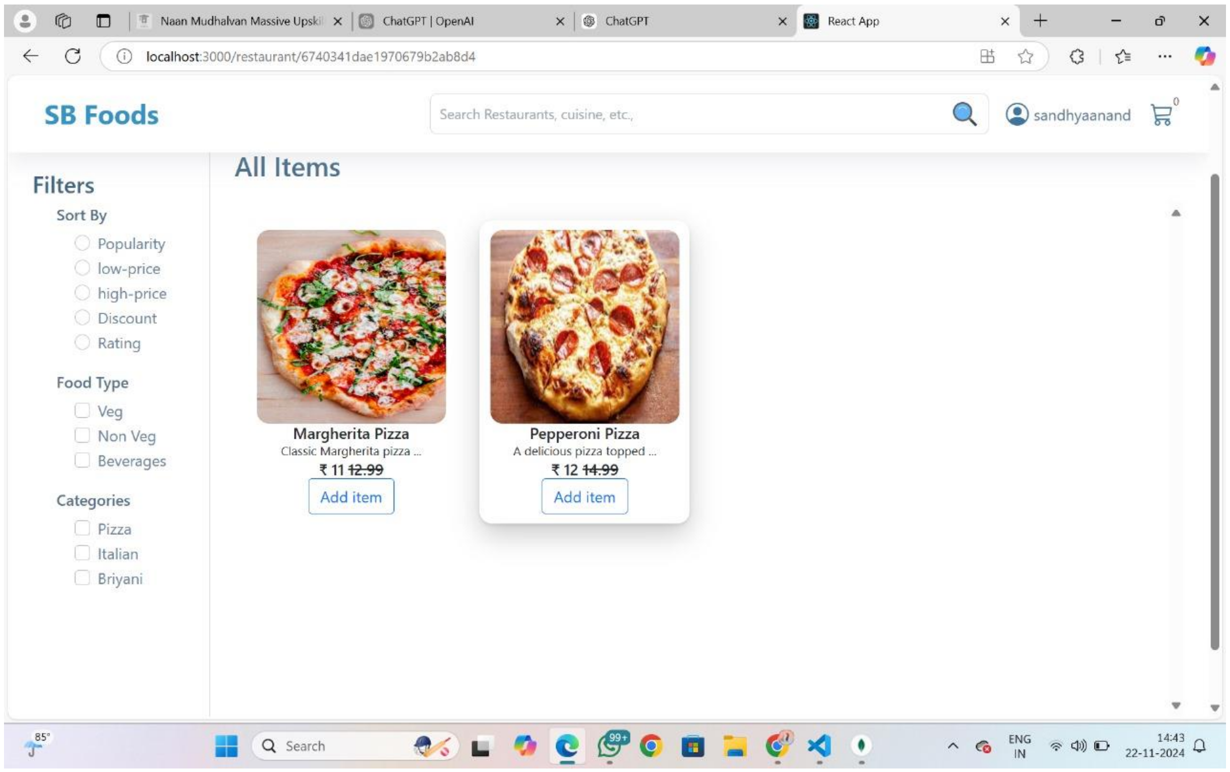

Naan Mudhalvan Massive Upskil
ChatGPT | OpenAI
ChatGPT
React App
localhost:3000
SB Foods
Search Restaurants, cuisine, etc.,
sandhyaanand
Popular Restaurants
Joe's Pizza
123 Pizza St, New York, NY 10001
Sushi World
456 Sushi Blvd, Los Angeles, CA 90001
All restaurants
Search
14:42
22-11-2024



All restaurants

Joe's Pizza	123 Pizza St, New York, NY 10001
Sushi World	456 Sushi Blvd, Los Angeles, CA 90001
Burrito Shack	789 Taco Ave, San Francisco, CA 94110

Menu Page: Displays food items with filter and search options.

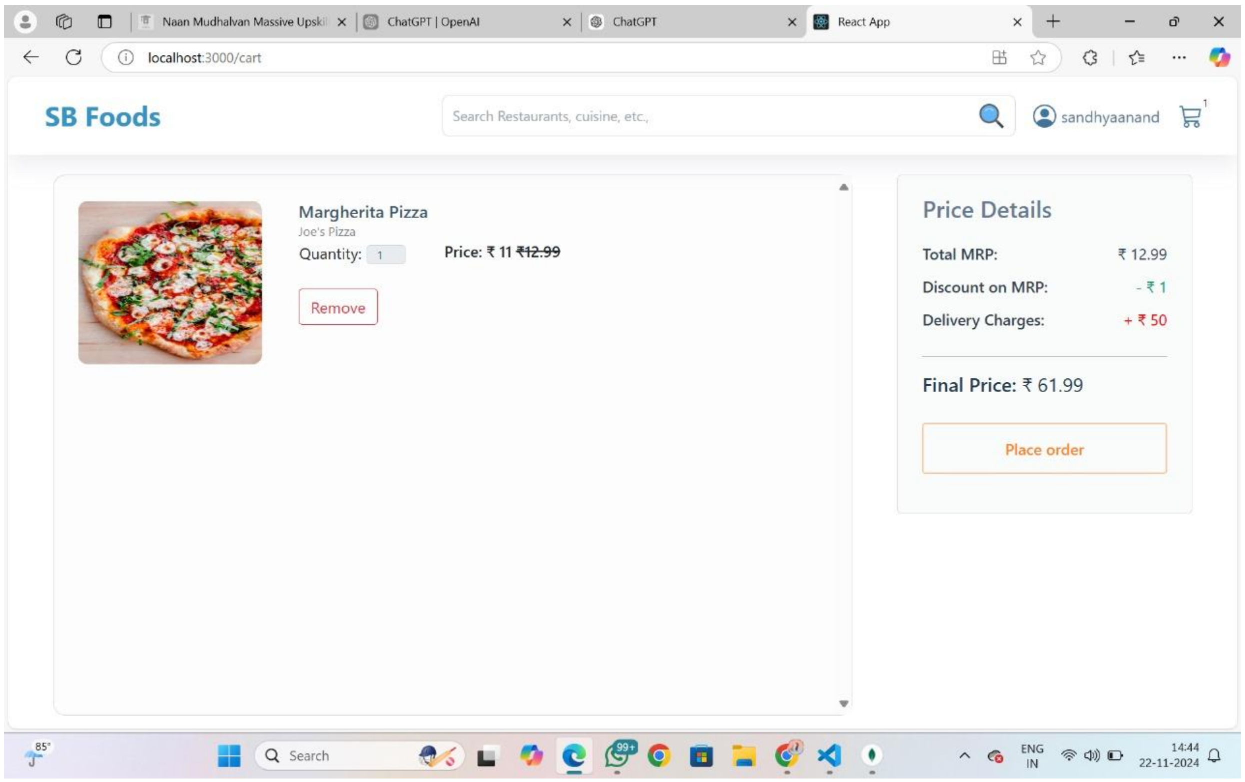


All Items

Margherita Pizza
Classic Margherita pizza ...
₹ 11 ~~12.99~~
Add item

Pepperoni Pizza
A delicious pizza topped ...
₹ 12 ~~14.99~~
Add item

Cart Page: Allows users to view selected items and total price.


localhost:3000/cart

SB Foods

Search Restaurants, cuisine, etc.,

sandhyaanand

Margherita Pizza
Joe's Pizza
Quantity: 1 Price: ₹ 11 ~~₹12.99~~

Remove

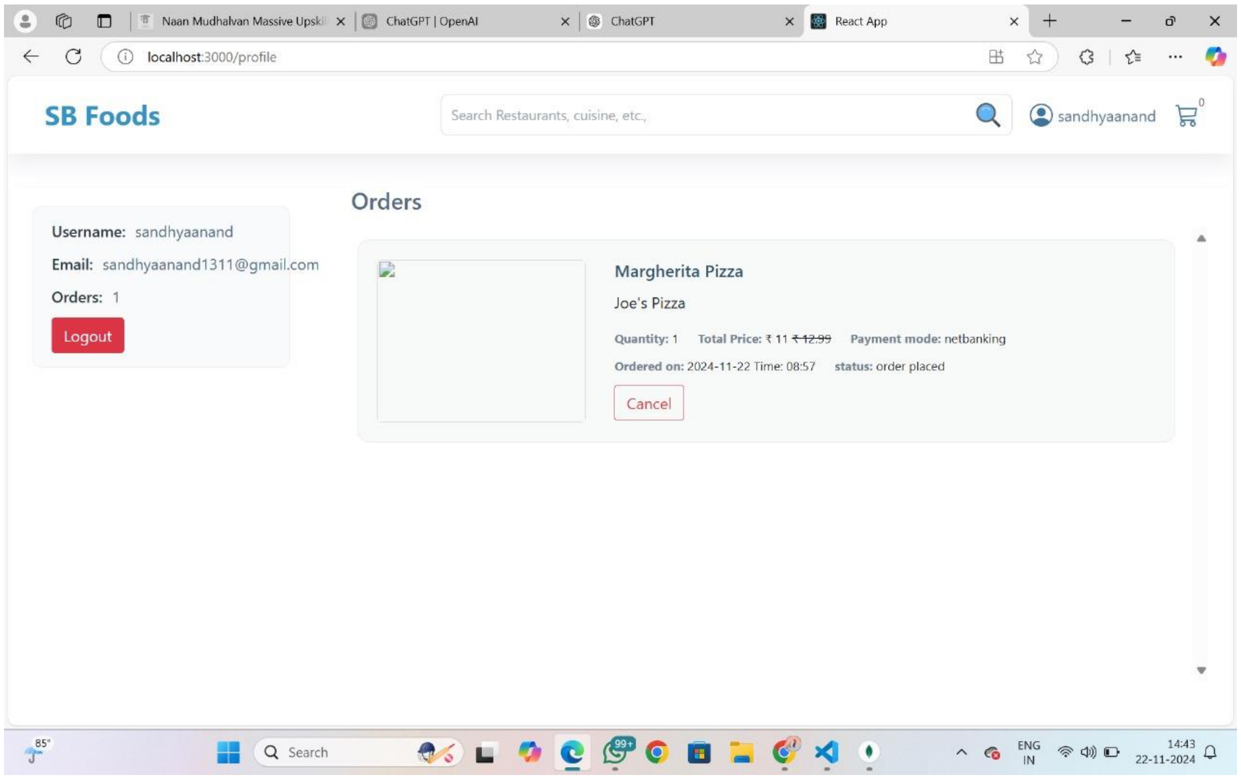
Price Details

Total MRP:	₹ 12.99
Discount on MRP:	- ₹ 1
Delivery Charges:	+ ₹ 50

Final Price: ₹ 61.99

Place order

Order Tracking Page: Displays the order status.


localhost:3000/profile

SB Foods

Search Restaurants, cuisine, etc.,

sandhyaanand

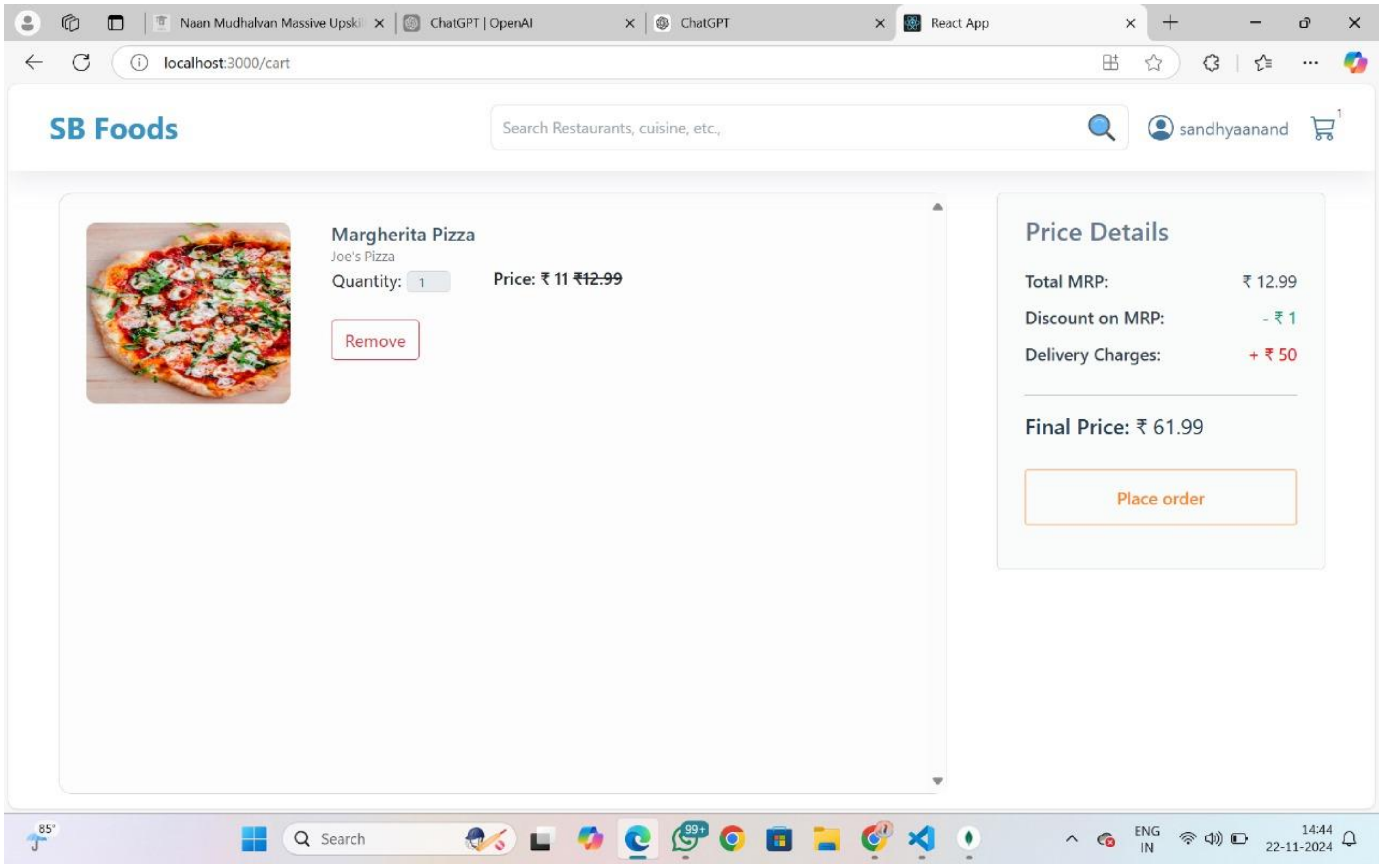
Username: sandhyaanand
Email: sandhyaanand1311@gmail.com
Orders: 1

Logout

Orders

Margherita Pizza
Joe's Pizza
Quantity: 1 Total Price: ₹ 11 ~~₹ 12.99~~ Payment mode: netbanking
Ordered on: 2024-11-22 Time: 08:57 status: order placed

Cancel



Margherita Pizza

Joe's Pizza

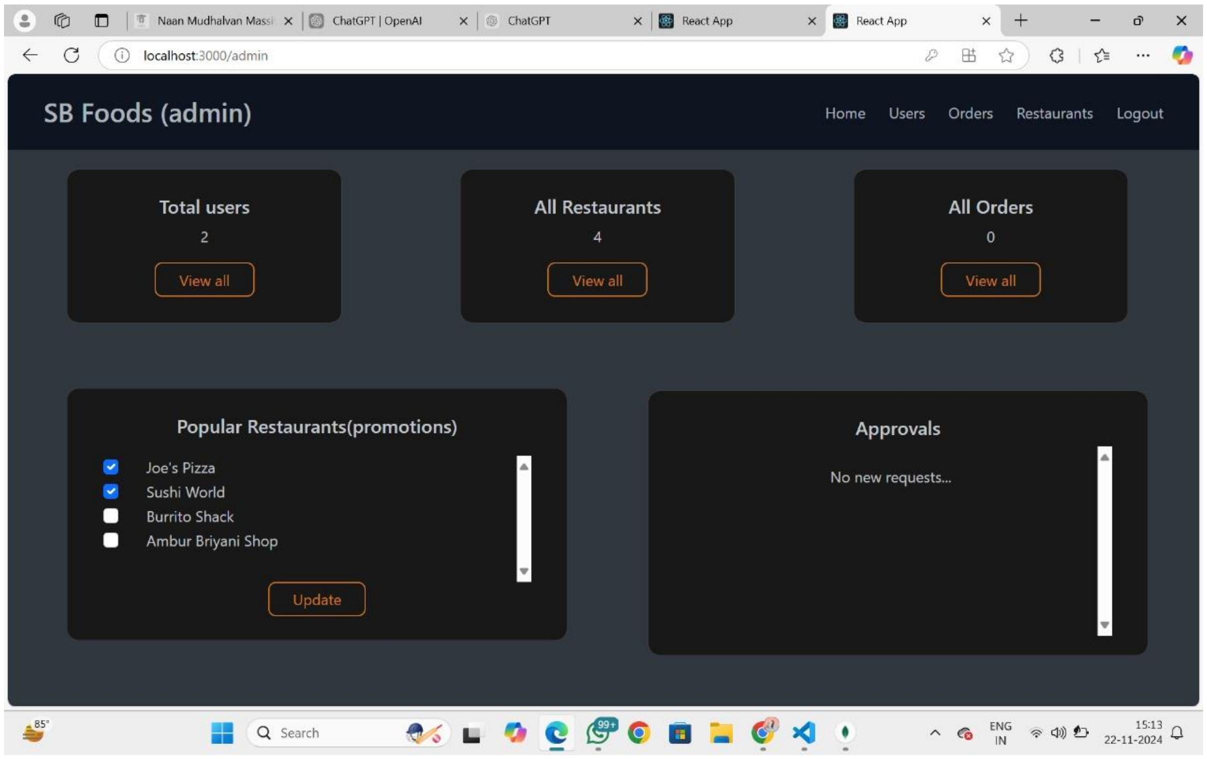
Quantity: 1 Price: ₹ 11 ~~₹12.99~~

Price Details

Total MRP:	₹ 12.99
Discount on MRP:	- ₹ 1
Delivery Charges:	+ ₹ 50

Final Price: ₹ 61.99

Admin Panel



Total users 2

All Restaurants 4

All Orders 0

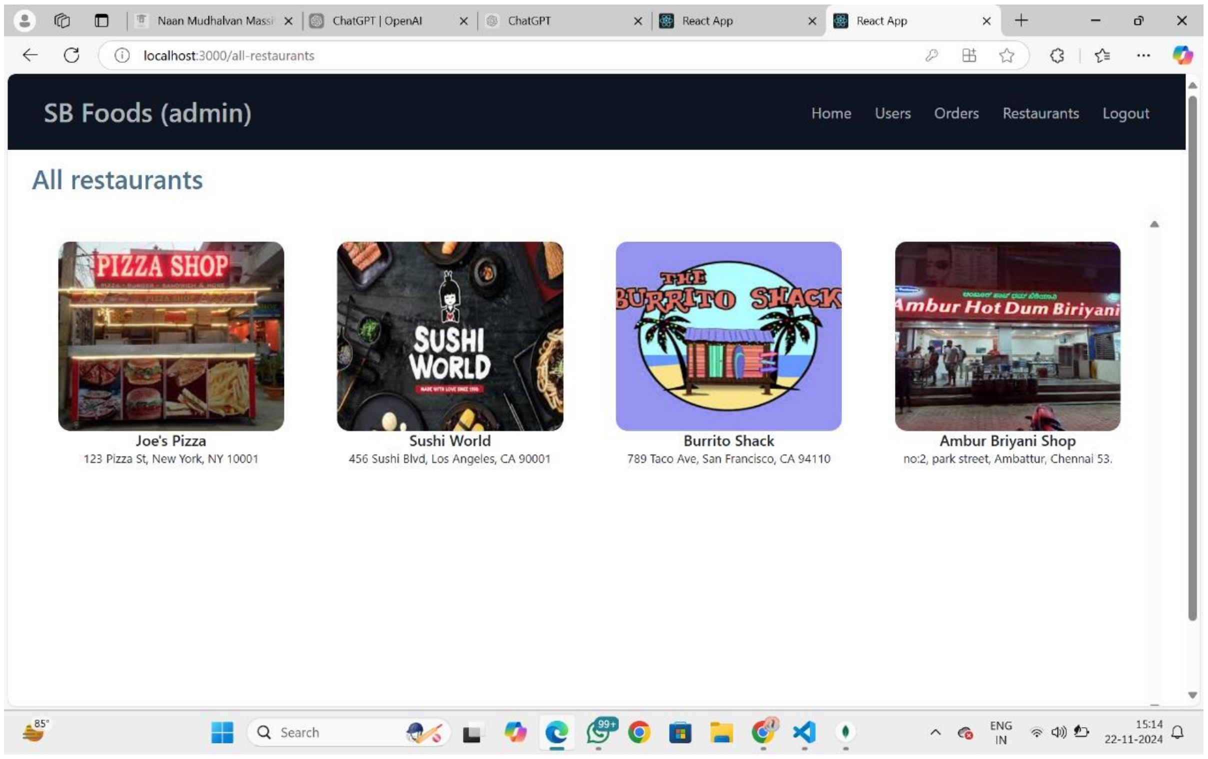
Popular Restaurants(promotions)

- [✓] Joe's Pizza
- [✓] Sushi World
- [ ] Burrito Shack
- [ ] Ambur Briyani Shop

Approvals

No new requests...

Restaurant Panel



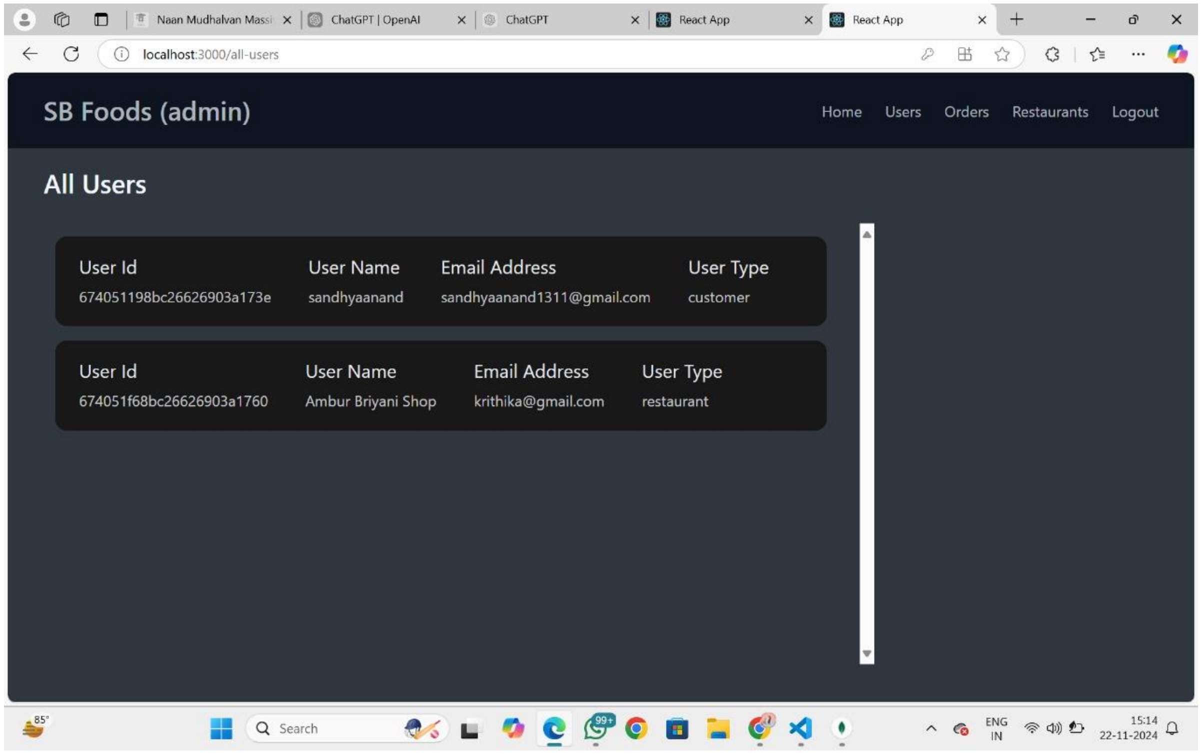
All restaurants

Joe's Pizza
123 Pizza St, New York, NY 10001

Sushi World
456 Sushi Blvd, Los Angeles, CA 90001

Burrito Shack
789 Taco Ave, San Francisco, CA 94110

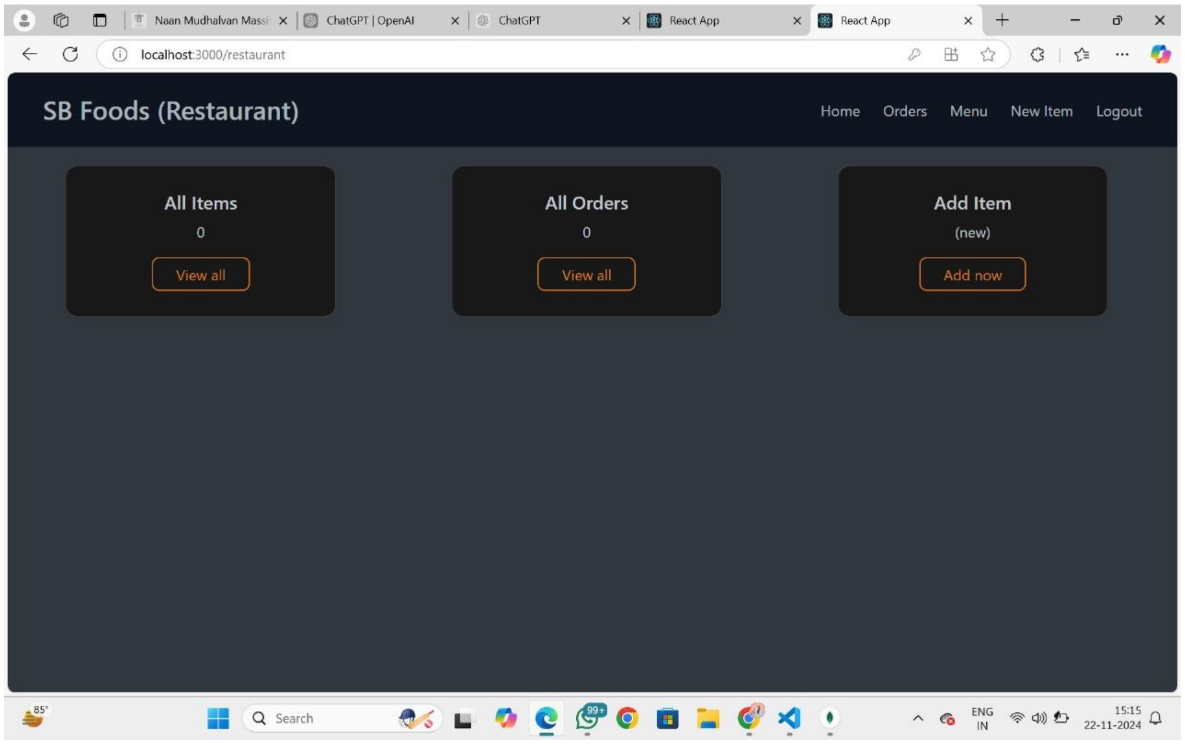
Ambur Briyani Shop
no:2, park street, Ambattur, Chennai 53.

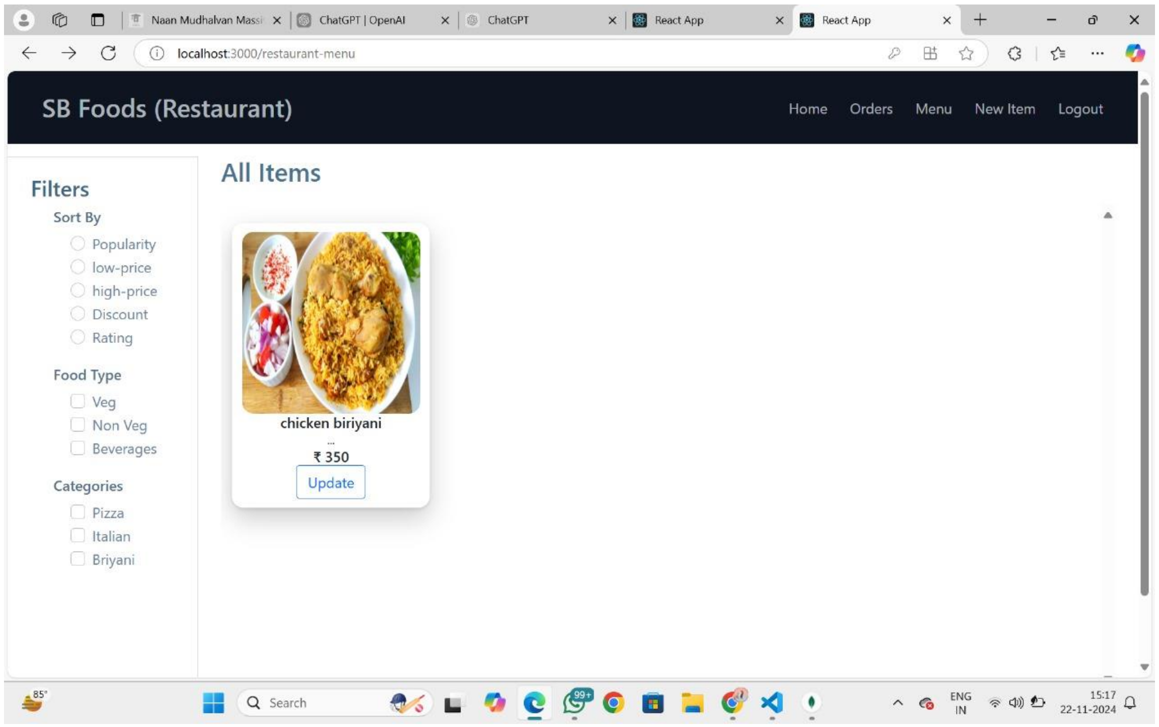


All Users

User Id	User Name	Email Address	User Type
674051198bc26626903a173e	sandhyaanand	sandhyaanand1311@gmail.com	customer
674051f68bc26626903a1760	Ambur Briyani Shop	krithika@gmail.com	restaurant

Restaurant Panel





DEMO LINK:

https://drive.google.com/file/d/1syematqUKJBgp7F8Ck1Zo0vfPzPxcy1E/view?usp=sharing

KNOWN ISSUES

1. **Payment Gateway Errors**: Occasional delays or failures during payment processing.

2. **Order Tracking Delays**: Real-time updates may be slow during peak times.
3. **Mobile Responsiveness**: Minor layout issues on older devices.
4. **Language Support**: The app currently supports only English, with potential inconsistencies in translations.
5. **Authentication Issues**: Users may experience session timeouts or login problems with expired tokens.
6. **Admin Dashboard Performance**: Slow loading times when displaying large datasets.

FUTURE ENHANCEMENTS

- **AI Recommendations**: Personalized food suggestions based on user preferences.
- **Multi-Language Support**: Adding support for additional languages.
- **Advanced Search Filters**: Allow users to filter by dietary restrictions and cuisine.
- **Loyalty Programs**: Reward points for frequent orders and referrals.
- **Voice Ordering**: Enable hands-free ordering using voice commands.
- **Dynamic Delivery Pricing**: Adjust delivery fees based on factors like time and weather.
- **Social Media Login**: Allow login via social media accounts.
- **Real-Time Customer Support**: In-app live chat for instant assistance.
- **Analytics for Restaurants**: Provide deeper insights into customer behavior and sales.
- **Subscription Plans**: Offer discounts or priority service with subscription models.

CONCLUSION:

In conclusion, the **SB Food Ordering App** stands out as a highly efficient and scalable solution for both customers and restaurant operators, harnessing the full potential of the MERN stack. Its user-friendly interface ensures that customers can easily navigate the app to browse menus, place orders, and track deliveries in real time, while restaurants benefit from streamlined order management, real-time updates, and detailed reporting.

The app's secure authentication system guarantees data protection, providing users with a safe and reliable experience. Meanwhile, the backend's structured and comprehensive admin tools offer restaurant operators the ability to manage their menus, track orders, and monitor business performance with ease.

Rigorous testing—ranging from unit and integration testing to performance validation—ensures that the app functions flawlessly under varying conditions, while its CI/CD pipeline facilitates smooth and timely deployments. The app is hosted on reliable platforms like **Heroku**, **AWS**, and **Netlify**, ensuring high availability, performance, and scalability in different environments.

Looking to the future, the SB Food Ordering App is well-positioned to meet the ever-growing demand in the online food delivery industry. Its focus on user satisfaction, business efficiency, and future-proof features, such as AI-driven recommendations and multi-language support, will continue to enhance the experience for both consumers and businesses. The app is not just a tool for today's needs but a forward-thinking solution set to shape the future of food delivery services, driving convenience and innovation for years to come.