NAAN MUDHALVAN FULL STACK MERN DEVELOPMENT- “SB-FOODS FOOD ORDERING APP”

A PROJECT REPORT

Submitted by

ALICE A - 310821104009
HARINI M - 310821104034
JULIYA A- 310821104043
RAJESH KANNA - 310821104306

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

JEPPIAAR ENGINEERING COLLEGE, CHENNAI

ANNA UNIVERSITY: CHENNAI 600 025

NOVEMBER 2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**SB FOODS FOOD ORDERING APP USING MERN"** is the Bonafide work of _____who carried out the project work under my supervision.

SIGNATURE
HEAD OF THE DEPARTMENT

SIGNATURE
MENTOR

DATE:

SIGNATURE
INTERNAL

Table of Contents

1) Introduction:..... 4
2) Project Overview 4
3) Features:..... 4
4) Architecture: 6
5) ER Diagram: 6
6) Setup Instructions: 7
7) Folder Structure 8
8) Running the Application 10
9) API Documentation: 12
10) Authentication:.....12
11) User Interface:.....12
12) Testing: 18
13) Technical Issues:.....19
14) Future Improvements for SB Food Ordering App:.....20
15) Conclusion.....22
16)Appendices.....22
17) Result.....25.

1) Introduction:

This is the detailed report of the project of a food ordering app which is done with the help of MERN Stack for the whole application and with my team members. It took 3 to 6 days of teamwork without rest and handled so many errors while running the project. But finally, we have done the project.

2) Project Overview

The **"SB FOOD ORDERING APP"** is a food ordering app which can be used by all users for ordering your favorite food from home and getting it delivered to have a fun meal with your friends and family. This app is free for all to use and it shows varieties of food from varieties of restaurant.

Purpose of the project:

This app have so many purposes and some of these are:

- ➢ Simplifies and streamlines the food ordering and delivery process.
- ➢ Connects customers with a wide range of restaurants and their menus.
- ➢ Allows users to browse and search for food items easily.
- ➢ Facilitates seamless order placement and payment.
- ➢ Provides realtime order tracking for customers.

.

3) Features:

1. User Registration and Authentication:

- o New users can register using their email and password.
- o Secure login mechanism using JWT (JSON Web Tokens).
- o Password hashing for security.

2. User Roles and Permissions:

- o Admin and user roles with differentiated permissions.
- o Admins have access to manage orders, menus, and users.
- o Users can browse menus, place orders, and manage their account details.

3. Browse and Search Menus:

- o View menus from various restaurants.
- o Search for food items using keywords

- Filter options available based on cuisine, price, and rating.

4. Add to Cart and Place Orders:

- Users can select and add food items to their cart.
- Easy checkout process to review orders before placing them.
- Option to modify cart items before finalizing the order.

5. Payment Integration:

- Multiple payment methods supported (e.g., credit card, PayPal, digital wallets).
- Secure payment gateway integration for safe transactions.
- Realtime payment status updates.

6. Order Management and Tracking:

- Users can track their order status from preparation to delivery.
- Realtime updates on order status (e.g., confirmed, in progress, out for delivery).
- Notification system to alert users about order updates.

7. Admin Panel:

- Dashboard for managing orders, users, and menus.
- Add, update, or delete menu items.
- View order statistics and reports.

8. Responsive User Interface:

- Optimized UI for both desktop and mobile devices.
- Seamless navigation and interactive elements.
- Intuitive design for user convenience.

9. User Profile Management:

- Users can view and edit their profile information.
- Manage saved addresses for faster checkout.
- View past orders and reorder favourite items easily

4) Architecture:

The SB Food Ordering App has a simple architecture which shows a perfect flow of the application. 1. Frontend:

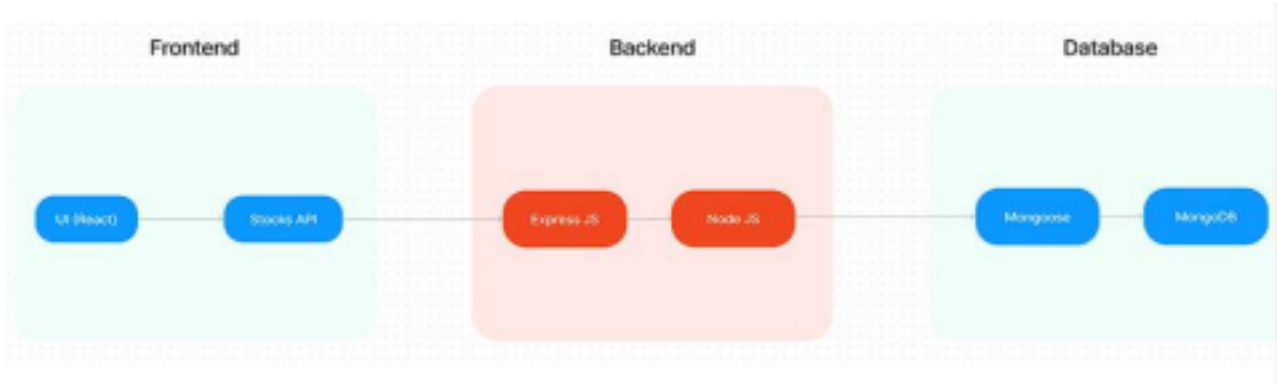
The frontend is developed using React.js, ensuring a dynamic and responsive user interface with efficient rendering of components and stock Api.

2. Backend:

The backend is built using Node.js and Express.js to provide RESTful APIs for various features such as user authentication, order processing, and payment handling.

3. Database:

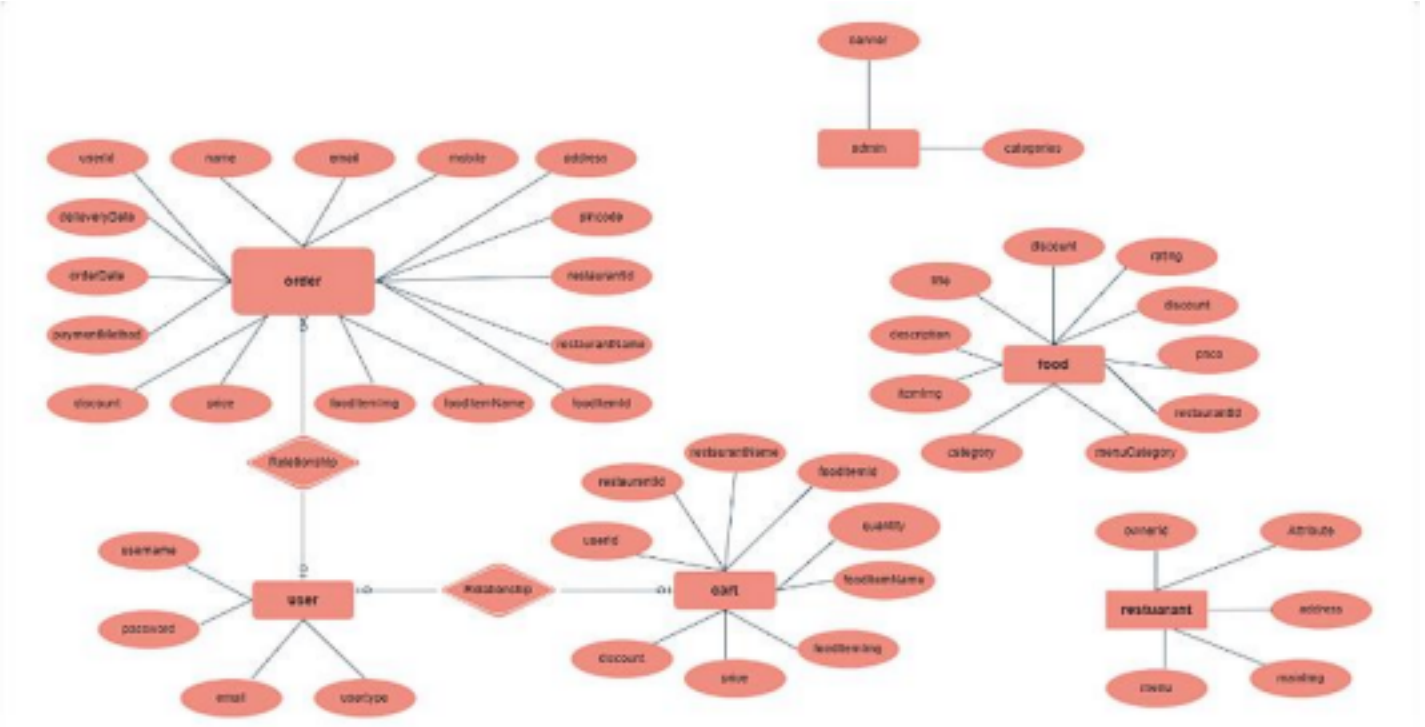
The database used for the system uses MongoDB to store and manage user data, menu items, orders, and payment information of the app.



5) ER Diagram:

The SB Foods ER Diagram represents the entities and relationships involved in a food ordering ecommerce system. It illustrates how users, restaurants, products, carts, and orders are interconnected. Here is a breakdown of the entities and their relationships:

- **User:** Represents the individuals or entities who are registered in the platform.
- **Restaurant**: This represents the collection of details of each restaurant in the platform.
- **Admin:** Represents a collection with important details such as promoted restaurants and categories.
- **Products:** Represents a collection of all the food items available in the platform.
- **Cart:** This collection stores all the products that are added to the cart by users. Here, the elements in the cart are differentiated by the user Id.
- **Orders:** This collection stores all the orders that are made by the users in the platform.



6) Setup Instructions:

Pre-Requisite:

To develop a full stack food ordering app using React JS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

a) Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: https://nodejs.org/en/download/
• Installation instructions: https://nodejs.org/en/download/packagemanager/

b) MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: https://www.mongodb.com/try/download/community
• Installation instructions: https://docs.mongodb.com/manual/installation/

c) Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server side routing, middleware, and API development.

• Installation: Open your command prompt or terminal and run the following command:
"npm install express"

d) React.js: React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. To install React.js, a JavaScript library for building user interfaces, follow the installation guide: "https://reactjs.org/docs/createanewreactapp.html"

e) HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client side interactivity is essential.

f) Database Connectivity: Use a MongoDB driver or an Object Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

g) Frontend Framework: Utilize Angular to build the user facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

h) Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• Git: Download and installation instructions can be found at: https://git scm.com/downloads

I. **Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download

• Sublime Text: Download from https://www.sublimetext.com/download

• WebStorm: Download from https://www.jetbrains.com/webstorm/download

II. **To Connect the Database with Node JS go through the below provided link:**

Link: https://www.section.io/engineeringeducation/nodejs mongoosejsmongodb/

III. **To run the existing SB Foods App project downloaded from GitHub:**

Follow below steps:

1. Clone the repository:

• Open your terminal or command prompt.

• Navigate to the directory where you want to store the ecommerce app.

• Execute the following command to clone the repository:

2. Git clone:

https://github.com/harinimurugadoss/SB-Foods-Food-Ordering-App/tree/main/food-del

3. Install Dependencies:

• Install the required dependencies by running the following command:

“npm install”

4. Start the Development Server:

• To start the development server, execute the following command:

“npm run dev” or “npm run start”

• The ecommerce app will be accessible at http://localhost:5176/ by default. You can change the port configuration in the .env file if needed.

5. Access the App:

• Open your web browser and navigate to http://localhost:5176/.

• You should see the flight booking app's homepage, indicating that the installation and setup were successful.

7) Folder Structure

SB-Foods/

```
│
├── Projectphotos/
│   └── All images for the project
│
├── Client/
│   ├── public/
│   │   ├── favicon.ico
│   │   ├── index.html
```

```
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
│
├── src/
│   ├── components/
│   │   ├── Footer.jsx
│   │   ├── Login.jsx
│   │   ├── Navbar.jsx
│   │   ├── PopularRestaurants.jsx
│   │   ├── Register.jsx
│   │   └── Restaurants.jsx
│   │
│   ├── Context/
│   │   └── GeneralContext.js
│   │
│   ├── images/
│   │   ├── homebanner2.png
│   │   └── homebanner1.png
│   │
│   ├── pages/
│   │   ├── admin/
│   │   │   ├── Admin.jsx
│   │   │   ├── AllOrders.jsx
│   │   ├── AllProducts.jsx
├── AllRestaurants.jsx
└── AllUsers.jsx
│   │   │
│   │   ├── customer/
│   │   │   ├── Cart.jsx
│   │   │   ├── CategoryProducts.jsx
│   │   ├── IndividualRestaurant.jsx
│   └── Profile.jsx
│   │   │
│   │   ├── restaurant/
│   │   │   ├── EditProduct.jsx
├── NewProduct.jsx
RestaurantHome.jsx
RestaurantMenu.jsx
RestaurantOrders.jsx
│   │   ├── Authentication.jsx
│   └── Home.jsx
│   │
│   ├── styles/
│   │   ├── Admin.css
│   │   ├── AllOrders.css
│   │   ├── AllProducts.css
│   │   ├── AllUsers.css
```

```
│   │   ├── Authentication.css │
│   ├── Cart.css
│   │   ├── CategoryProducts.css │
│   ├── CheckOutPage.css
│   │   ├── Footer.css
│   │   ├── Home.css
│   │   ├── IndividualRestaurant.css
│   │   ├── Navbar.css
│   │   ├── NewProducts.css
│   │   ├── PopularRestaurants.css
│   │   ├── Products.css
│   │   ├── Profile.css
│   │   ├── RestaurantHome.css
│   │   └── Restaurants.css
│   │
│   ├── App.css
│   ├── App.js
│   ├── App.test.js
│   ├── index.css
│   ├── index.js
│   ├── logo.svg
│   ├── reportWebVitals.js
│   └── setupTests.js
│
├── Server/
│   ├── packagelock.json
│   └── package.json
```

8) Running the Application

To set up and run the SB FOOD ORDERING APP locally, follow the steps below:

Prerequisites:

- Ensure that Node.js and npm (Node Package Manager) are installed on your system.
- Confirm that MongoDB is installed and running if you are using a local database.
- Clone the project repository to your local machine.

Step 1: Set Up the Frontend:

1. Navigate to the Client Directory:
   - Open a terminal window.
   - Change your current directory to the client folder: "cd SB-Foods/Client"

2. Install Frontend Dependencies:

Run the following command to install all required npm packages for the frontend:

"npm install"

3. Start the Frontend Development Server:
Once dependencies are installed, start the frontend server by running:
"npm run dev"

4. Verify the Frontend:
- Check that the UI loads correctly, and all components render without errors.
- Ensure there are no errors in the browser console.

Step 2: Set Up the Backend

1. Navigate to the Server Directory:
- Open a new terminal window.
- Change your current directory to the server folder:

"cd SB-Foods/Server"

2. Install Backend Dependencies:
Run the following command to install all necessary npm packages for the backend:
"npm install"

3. Configure Environment Variables:
If your application requires environment variables, create a `.env` file in the `Server` directory with necessary configurations like MongoDB URI, JWT secret, and other API keys. Example:
"PORT=3000
MONGODB_URI=mongodb://localhost:5176/s food ordering
JWT_SECRET=your_jwt_secret_key"

4. Start the Backend Server:
Start the backend server by running:
"npm start"

5. Verify the Backend:
- Check that the backend server starts without errors.
- You should see a message in the terminal indicating that the server is running and connected to the database.

Step 3: Access the Application

- Frontend (React Application):
Open http://localhost:5176/in your web browser to interact with the frontend UI.

- Backend (API Server):
Ensure that the backend server is running and can be accessed via http://localhost:5173/(or another port if specified).

Step 4: Testing End-to-end Functionality

1. Register or log in as a user to verify the authentication and authorization features.
2. Browse restaurants, add items to the cart, and place an order to test the core functionalities. 3. Use the admin account (if applicable) to manage orders, users, and menu items.

6.Troubleshooting:

Port Conflicts:
If you encounter port conflicts, you can change the frontend or backend port in the respective configurations.

Database Connection Issues:
Ensure that MongoDB is running and accessible. If you are using a cloud database, verify that the URI in the ".env" file is correct.

9). API Documentation:

Endpoints Overview:
`POST /api/users/register` User registration
`POST /api/users/login` User login
`GET /api/menu` Get menu items
`POST /api/orders` Place an order
`GET /api/orders/: userId` Retrieve user orders

Example Response:
json
{
"success": true,
"message": "Order has been placed",
"orderId": "585"
}

10).Authentication:

- The app uses JWT (JSON Web Tokens) for user authentication and authorization.
- Tokens are generated upon login and stored in the client's local storage for subsequent API calls.

11).User Interface:

The User Interface or, UI, is the layout or window of the application. The UI of the app consists of:

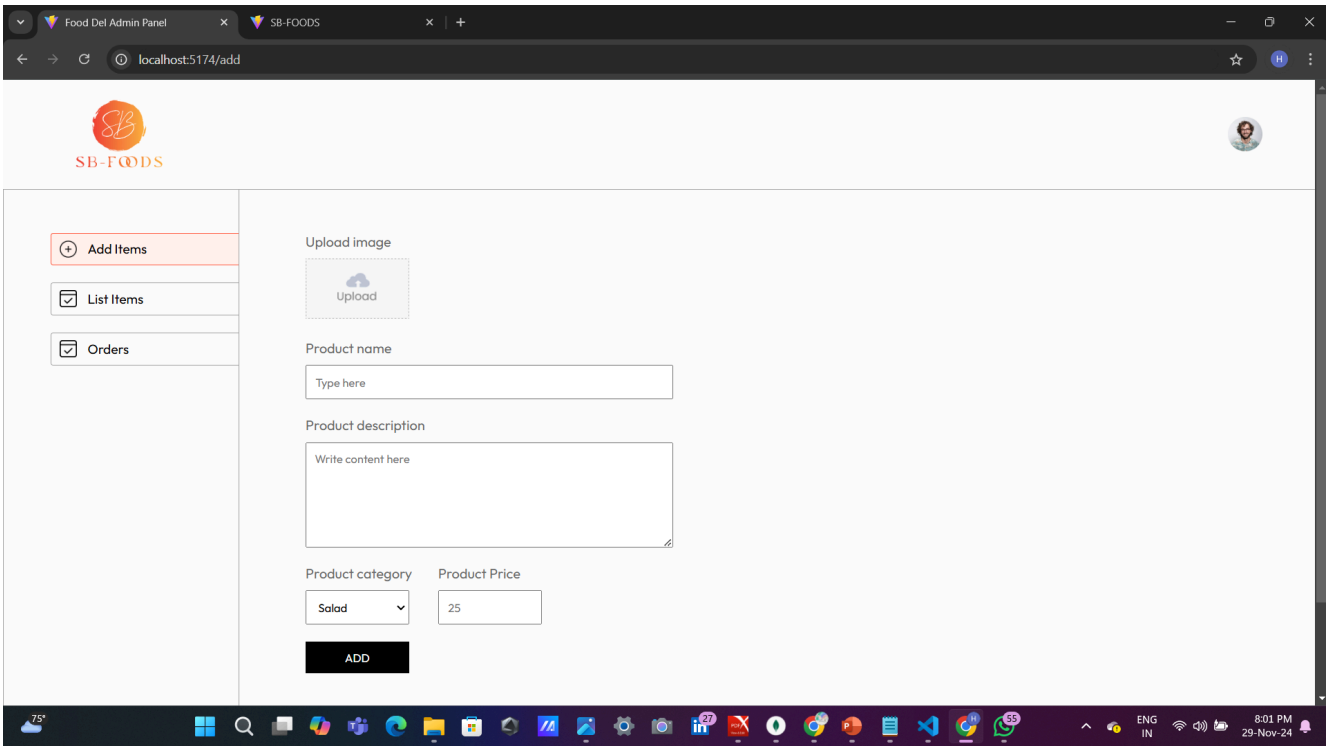
- ➢ A homepage has feature of our restaurant
- ➢ Menu pages with filtering options
- ➢ Cart and checkout pages

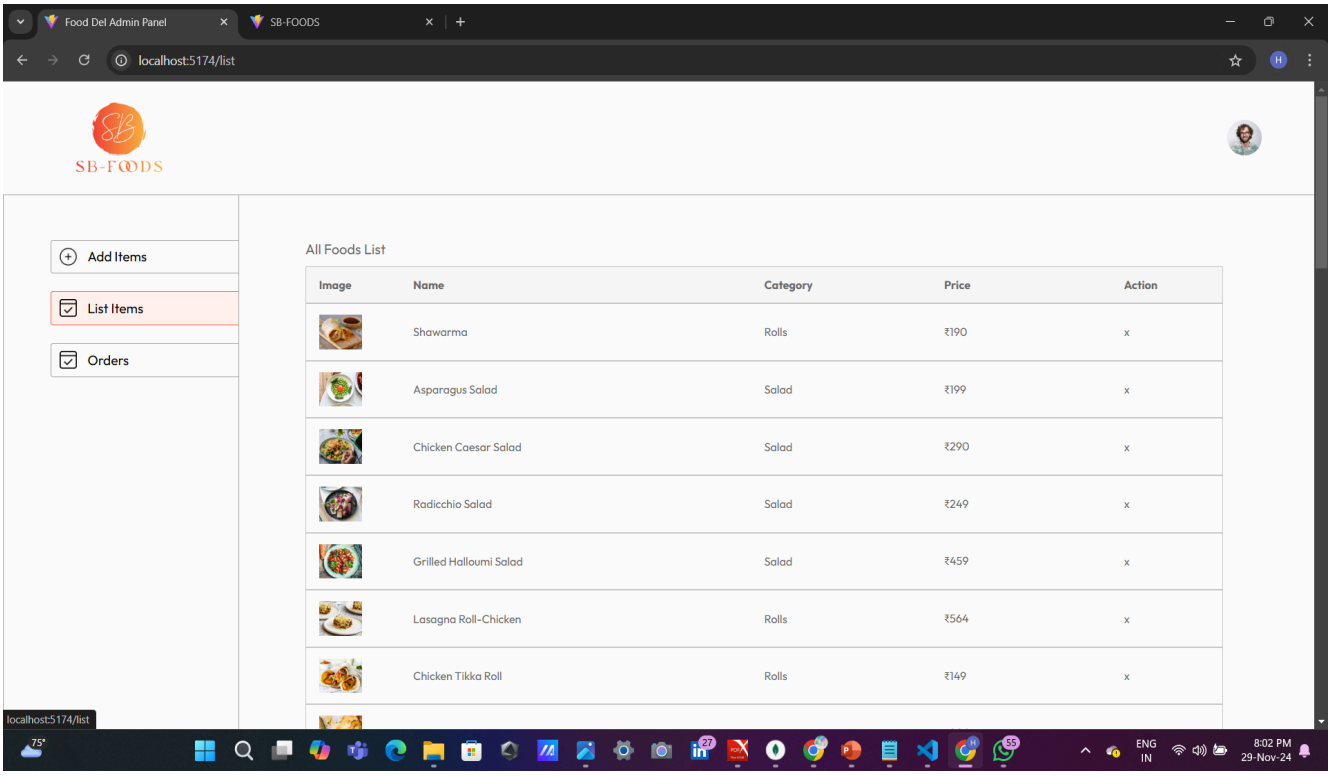
Screenshots:

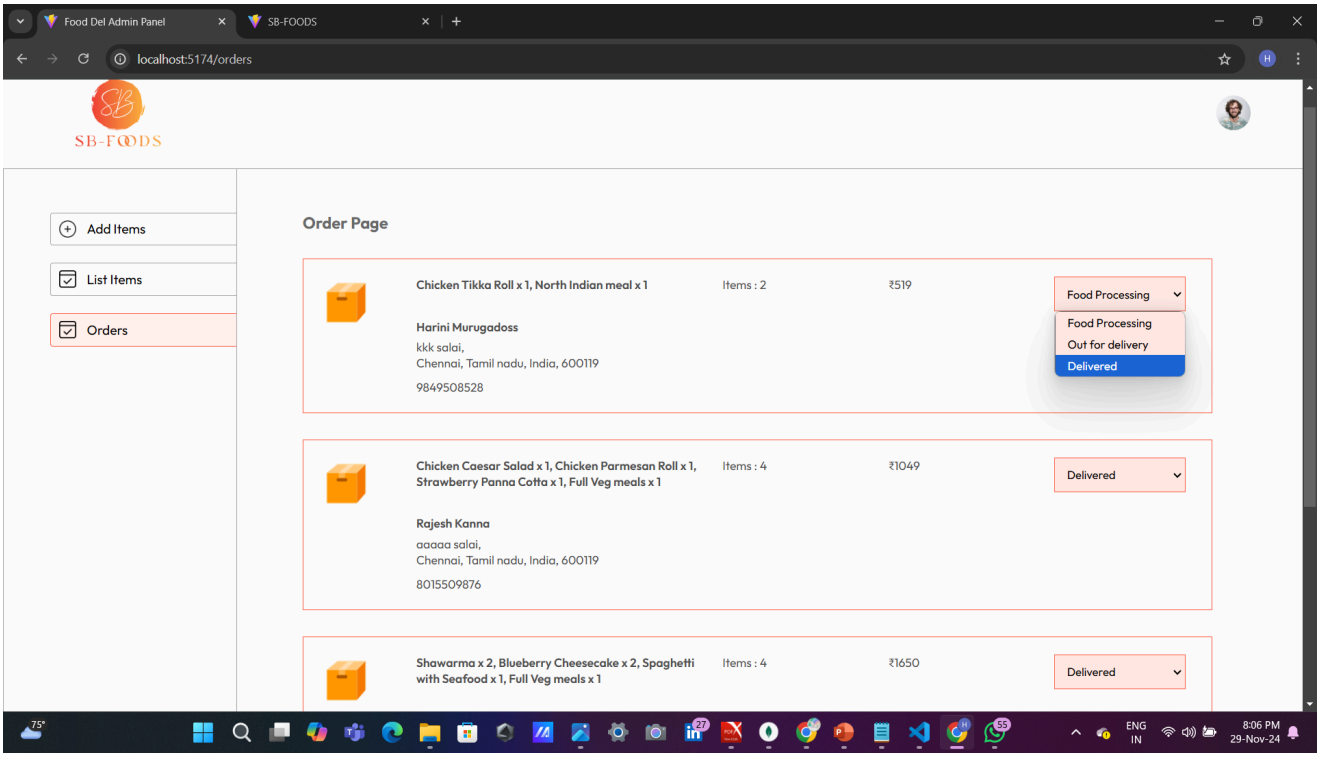
The screenshots of the app will give you a clear vision about the appearance and layout of the application. It includes the screenshots of:

1. Admin
2. Customer
3. Restaurants and
4. App's UI

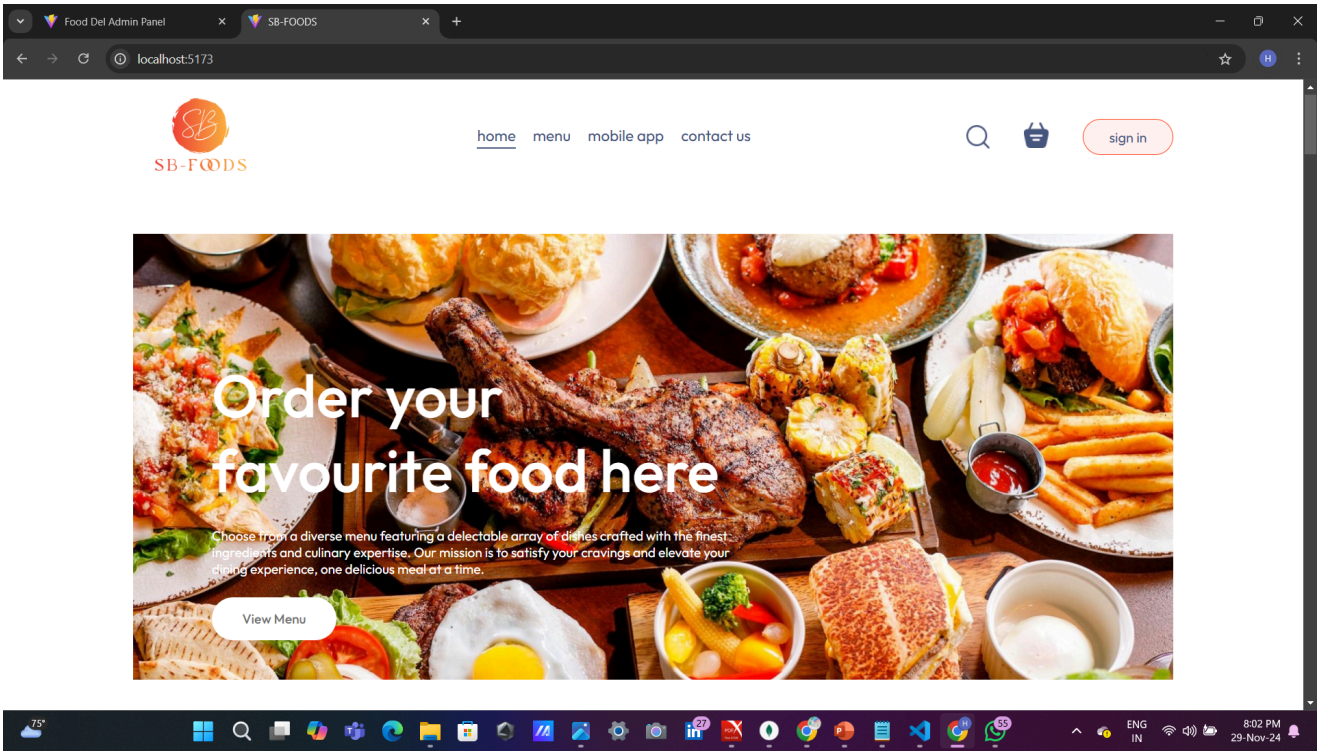
Admin:

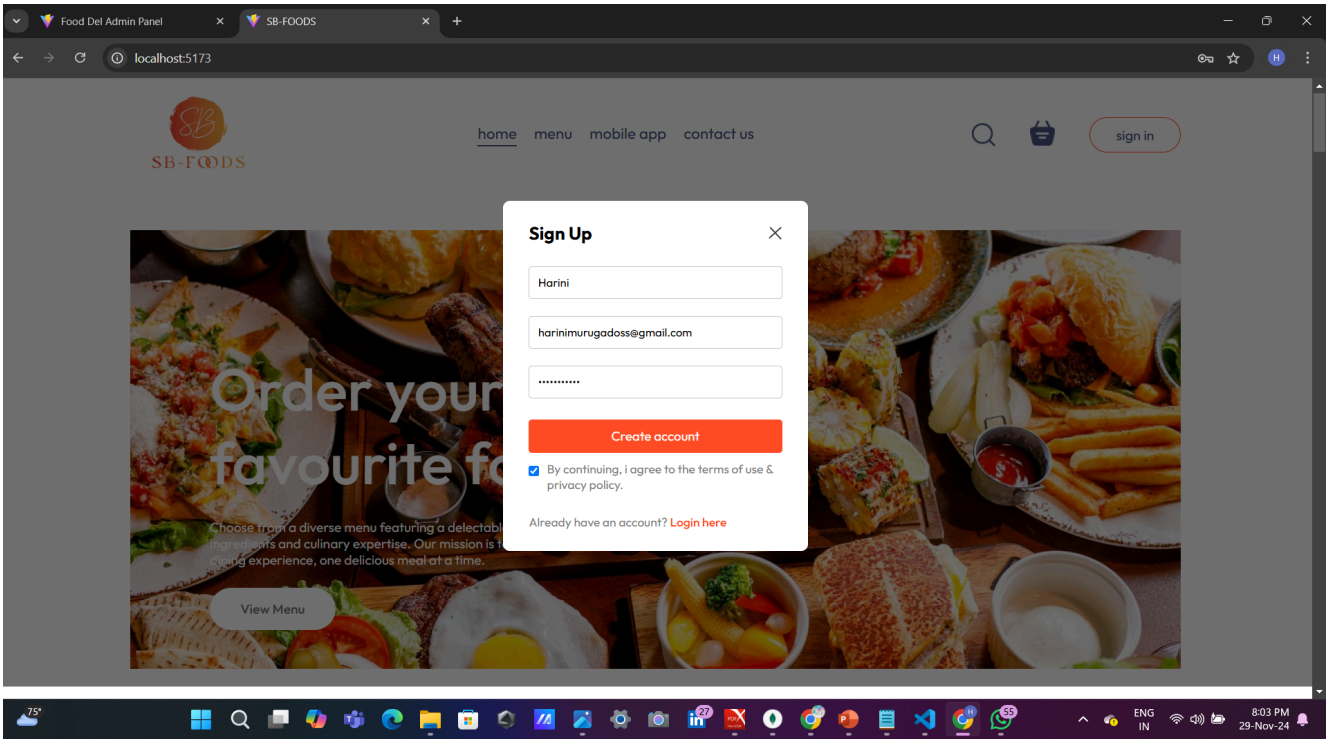


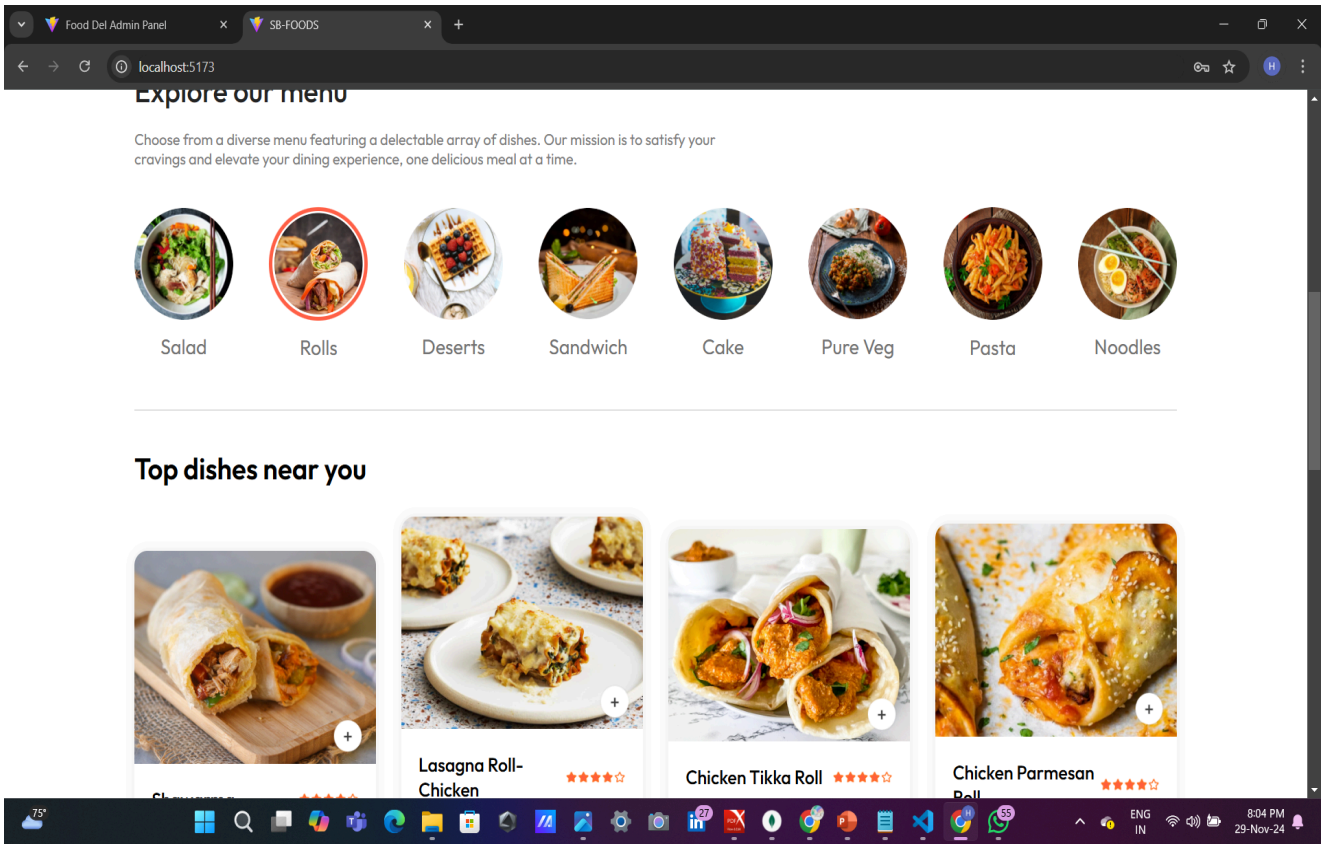

Food Del Admin Panel
SB-FOODS
localhost:5174/list
SB-FOODS
Add Items
List Items
Orders
All Foods List
Image
Name
Category
Price
Action
Shawarma
Rolls
₹190
Asparagus Salad
Salad
₹199
Chicken Caesar Salad
Salad
₹290
Radicchio Salad
Salad
₹249
Grilled Halloumi Salad
Salad
₹459
Lasagna Roll-Chicken
Rolls
₹564
Chicken Tikka Roll
Rolls
₹149
localhost:5174/list

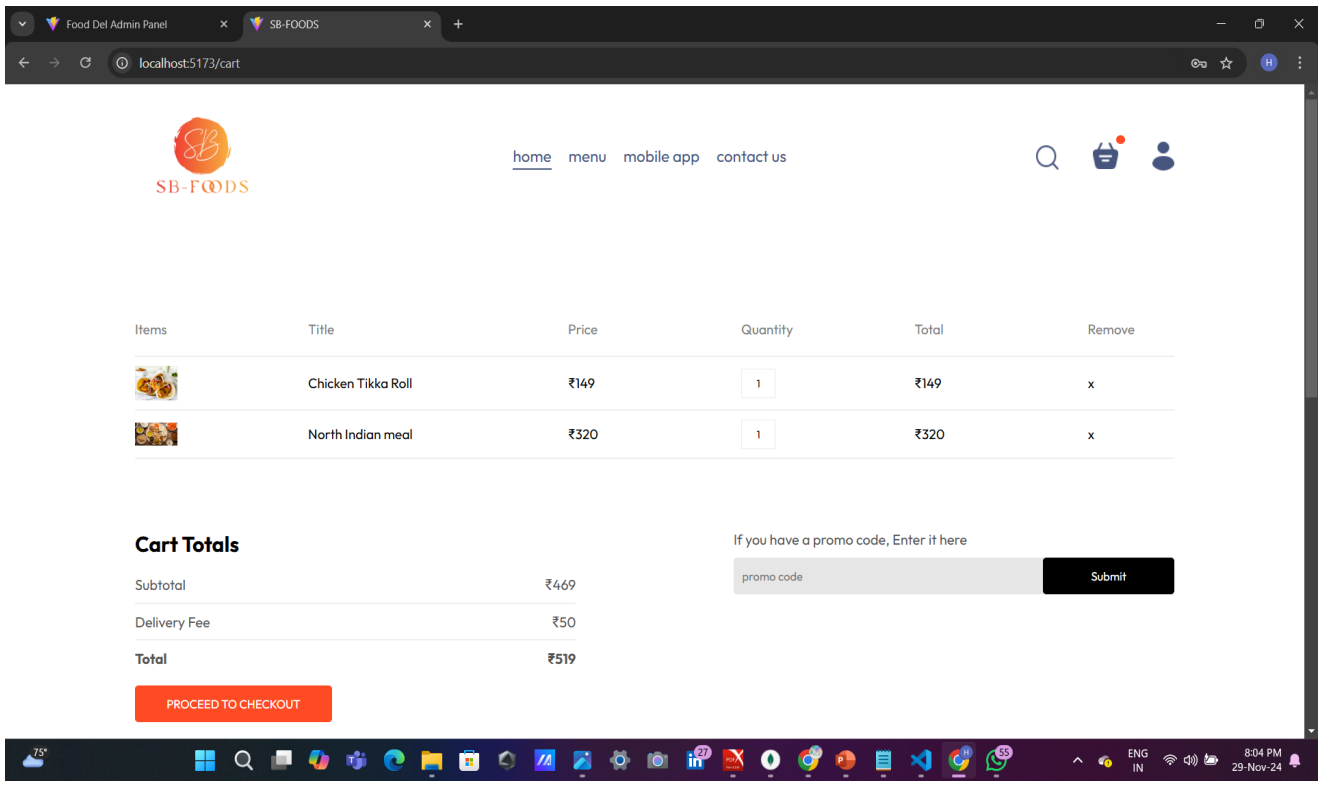

Food Del Admin Panel
SB-FOODS
localhost:5174/orders
SB-FOODS
Add Items
List Items
Orders
Order Page
Chicken Tikka Roll x 1, North Indian meal x 1
Items : 2
₹519
Food Processing
Harini Murugadoss
kkk salai,
Chennai, Tamil nadu, India, 600119
9849508528
Food Processing
Out for delivery
Delivered
Chicken Caesar Salad x 1, Chicken Parmesan Roll x 1, Strawberry Panna Cotta x 1, Full Veg meals x 1
Items : 4
₹1049
Delivered
Rajesh Kanna
aaaaa salai,
Chennai, Tamil nadu, India, 600119
8015509876
Shawarma x 2, Blueberry Cheesecake x 2, Spaghetti with Seafood x 1, Full Veg meals x 1
Items : 4
₹1650
Delivered

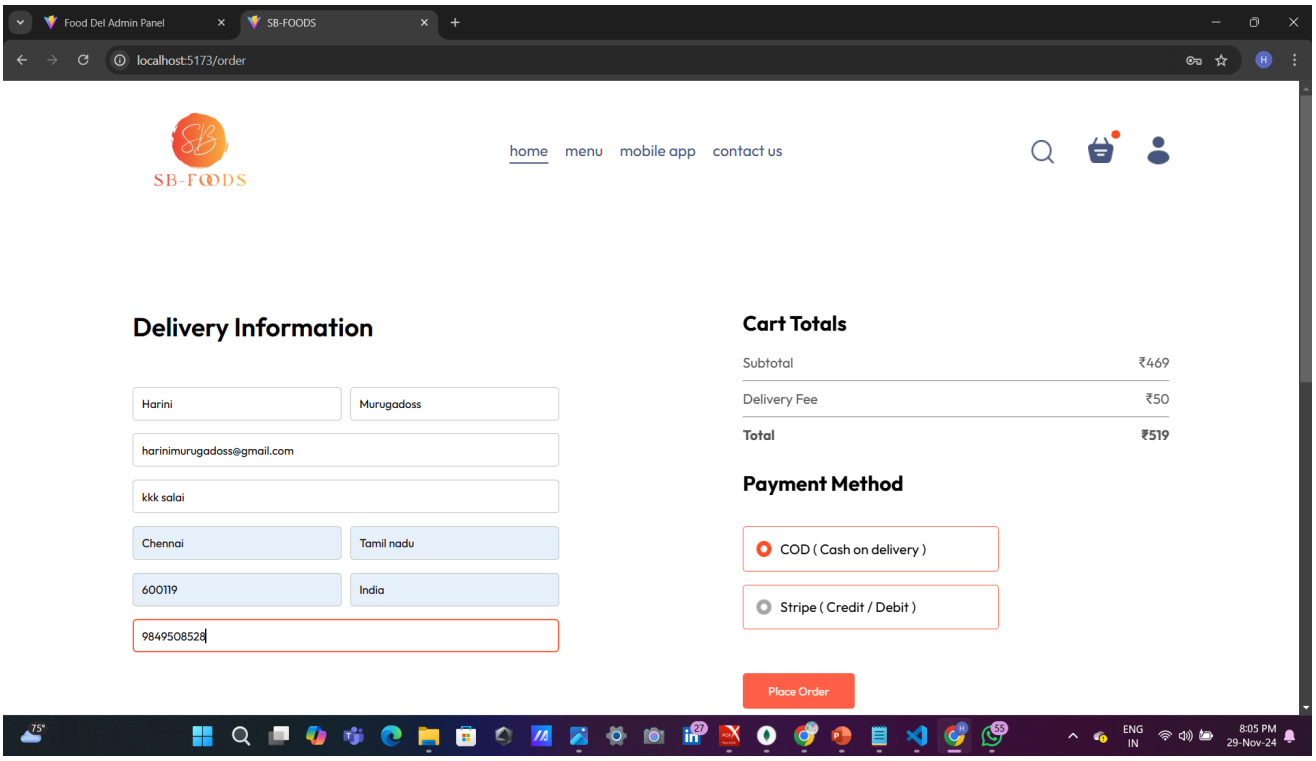
Customer:

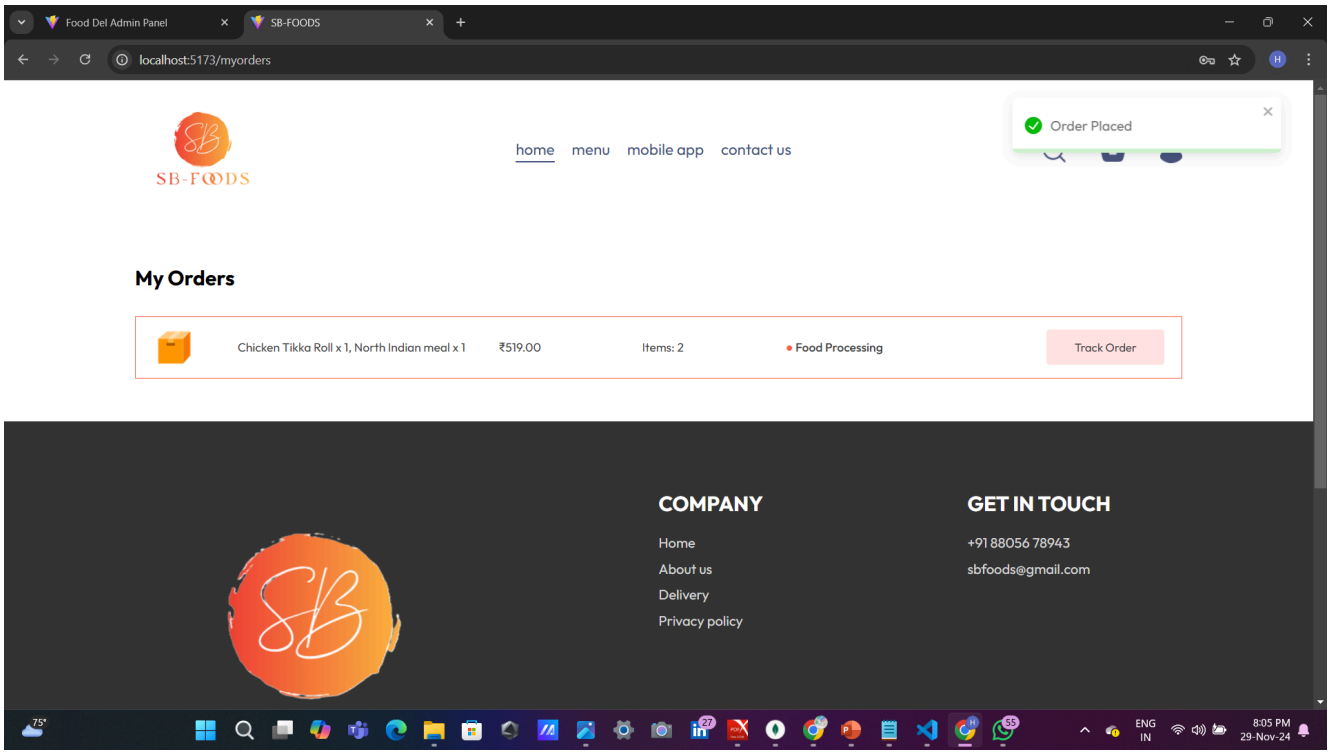

Food Del Admin Panel
SB-FOODS
localhost:5173
SB-FOODS
home
menu
mobile app
contact us
sign in
Order your favourite food here
Choose from a diverse menu featuring a delectable array of dishes crafted with the finest ingredients and culinary expertise. Our mission is to satisfy your cravings and elevate your dining experience, one delicious meal at a time.
View Menu


Food Del Admin Panel
SB-FOODS
localhost:5173
SB-FOODS
home
menu
mobile app
contact us
sign in
Sign Up
Harini
harinimurugadoss@gmail.com
Create account
By continuing, i agree to the terms of use & privacy policy.
Already have an account? Login here
View Menu


Food Del Admin Panel
SB-FOODS
localhost:5173
Explore our menu
Choose from a diverse menu featuring a delectable array of dishes. Our mission is to satisfy your cravings and elevate your dining experience, one delicious meal at a time.
Salad
Rolls
Deserts
Sandwich
Cake
Pure Veg
Pasta
Noodles
Top dishes near you
Lasagna Roll-Chicken
Chicken Tikka Roll
Chicken Parmesan


Food Del Admin Panel
SB-FOODS
localhost:5173/cart
SB-FOODS
home
menu
mobile app
contact us
Items
Title
Price
Quantity
Total
Remove
Chicken Tikka Roll
₹149
1
₹149
x
North Indian meal
₹320
1
₹320
x
Cart Totals
Subtotal
₹469
Delivery Fee
₹50
Total
₹519
PROCEED TO CHECKOUT
If you have a promo code, Enter it here
promo code
Submit


Food Del Admin Panel
SB-FOODS
localhost:5173/order
SB-FOODS
home
menu
mobile app
contact us
Delivery Information
Harini
Murugadoss
harinimurugadoss@gmail.com
kkk salai
Chennai
Tamil nadu
600119
India
9849508528
Cart Totals
Subtotal
₹469
Delivery Fee
₹50
Total
₹519
Payment Method
COD (Cash on delivery)
Stripe (Credit / Debit)
Place Order


Food Del Admin Panel
SB-FOODS
localhost:5173/myorders
SB-FOODS
home
menu
mobile app
contact us
Order Placed
My Orders
Chicken Tikka Roll x 1, North Indian meal x 1
₹519.00
Items: 2
Food Processing
Track Order
COMPANY
Home
About us
Delivery
Privacy policy
GET IN TOUCH
+91 88056 78943
sbfoods@gmail.com

17

12) Testing:

Tools:

• **Jest**:

This comprehensive testing framework is widely used with React applications, providing a powerful combination of assertion, mocking, and snapshot testing. Jest offers efficient test running, built-in mocking capabilities for functions and modules, and detailed test result outputs, which are critical for React component testing.

• **Mocha**:

An adaptable testing framework for Node.js, often utilized for server-side and backend testing. Mocha allows for asynchronous testing, supports hooks like before(), after(), beforeEach(), and afterEach() for controlling the testing lifecycle, and pairs well with assertion libraries like Chai.

Strategy:

1. **Unit Testing**:

   • **Objective**: Test individual units such as functions, components, and services to ensure they behave correctly in isolation.

   • **Key Activities**:

      ▪ **Testing React Components**: Verify that React components render correctly given a set of props or state changes.

      ▪ **Function Validation**: Test utility functions, input validation, business logic, and transformations, ensuring expected outcomes for various inputs.

      ▪ **Mocking**: Employ Jest's mock functions to simulate dependent services, APIs, or module behaviours during testing.

2. **Integration Testing**:

   • **Objective**: Test the interaction between different components, modules, or services, ensuring that they work together as expected.

   • **Key Activities**:

      ▪ **API Testing**: Use Mocha to test server-side RESTful API endpoints for correct HTTP responses, payloads, and error handling.

      ▪ **Database Queries**: Validate integration with database operations to ensure queries return expected results and handle edge cases correctly.

      ▪ **User Interaction Flows**: Test user paths that involve multiple components, simulating real-world interactions (e.g., registering, logging in, ordering food).

Technical Details:

- **Mocking and Stubbing**: Dependency injection and mocking allow isolation of units and controlled environments for reliable testing.

- **Assertions and Matchers**: Leverage Jest's and Mocha's built-in assertions to ensure expected values and states, checking for elements, API status codes, etc.

- **Automated Test Suites**: Organized suites for both unit and integration tests, with continuous integration hooks for automated test execution on code commits.

- **Code Coverage Analysis**: Employ coverage tools like Istanbul (integrated with Jest) to measure the extent of tested code paths, identifying potential gaps in test coverage.

13) Technical Issues:

Even though the app is working perfectly, we won't forget the issues faced during this. As Thomas Alva Edison tried 1000 times to build a light bulb, we tried 4 times to create and make our app work. Let's see about that now:

1. **State Management**:
   - **Complexity**: Managing and synchronizing application state across numerous components, especially in large applications, can be difficult due to prop drilling and maintaining UI consistency.

   - **Solution**:

     - Utilize **Redux** or **React Context** for predictable state updates, centralized state storage, and global state access.

     - Apply middleware like **Redux Thunk** or **Redux Saga** for asynchronous actions and side-effects management.

2. Database Schema Complexity:

   - **Challenge**:

     - Structuring complex data relationships, such as one-to-many or many-to-many, within **MongoDB** requires thoughtful modelling.

     - Nesting documents can lead to large data sizes, while referencing can cause slower joins.

   - **Solution**:

     - Optimize by determining when to **embed** (for denormalized, frequently-read data) vs. **reference** (for more normalized data with less redundancy).

     - Utilize **Mongoose's schema design capabilities**, such as schema validation, population (for joins), and indexing to enhance data query efficiency

3. Handling Edge Cases:

• **Challenge**:
Scenarios such as user input validation, unexpected API response errors, race conditions, server outages, and invalid data can lead to application crashes or poor user experience.

• **Solution**:

▪ Implement robust input validation using libraries like **Joi** and enforce type checks.

▪ Use **try-catch blocks** and error-handling middleware in **Express** to manage exceptions..

▪ Integrate retry mechanisms, fallbacks, and circuit breakers for network errors, and perform comprehensive testing (unit, integration, and boundary testing) to handle edge cases

.

4. API Design:

• **Challenge**:
Ensuring scalable, secure, and maintainable API design is crucial for the backend architecture in **Node.js/Express**.

• **Solution**:

▪ Follow **RESTful design principles** by structuring routes cleanly (using versioning and semantic endpoints).

▪ Implement **JWT (JSON Web Token)** for secure authentication and authorization mechanisms.

▪ Optimize data fetching using **pagination** and **filtering** to minimize payload size and latency.

▪ Use **middleware** for input validation, logging, and rate limiting to enforce security best practices.

▪ Testing and documentation tools, such as **Postman** or **Swagger**, ensure proper API functionality and ease of consumption for developers.

14) Future Improvements for SB Food Ordering App:

1.AI-Based Recommendations

Implement AI algorithms to provide personalized food recommendations based on user preferences, order history, and trending dishes.

2.Real-Time Order Tracking

Integrate GPS tracking to enable customers to monitor their orders in real time, from preparation to delivery.

3.Multiple Payment Gateways
Expand payment options by integrating multiple gateways, including UPI, credit/debit cards, and wallet services for enhanced user convenience.

4.Subscription Plans
Offer subscription plans for frequent customers, such as meal combos, monthly packages, or discounts for repeat orders.

5.Multi-Language Support
Introduce support for regional languages to cater to a diverse customer base, ensuring inclusivity**.**

6.Dynamic Feedback System
Implement a feedback system that prompts customers to rate and review specific menu items or services after each order.

7.Nutritional Information
Add nutritional data for each dish, helping health-conscious customers make informed decisions.

8.Integration with Smart Devices
Develop a mobile app compatible with smart home assistants like Alexa or Google Assistant for hands-free ordering.

9.Corporate Catering Services
Extend the app to include options for bulk and corporate food orders with customizable menus and delivery schedules.

Detailed Overview:

1. Location-Based Services
By integrating GPS and geolocation services, the app will recognize the user's location and display a list of nearby restaurants. This will allow customers to choose from a wider variety of options, based on proximity and delivery availability. Furthermore, location-based promotions can be implemented, providing users with special deals in their area, encouraging them to order more frequently.

2. Real-Time Order Tracking
The addition of real-time tracking will make it easy for users to stay informed about their order's status. Notifications will update users on key stages such as "Order Confirmed," "Being Prepared," "Out for Delivery," and "Delivered." A live map feature can show the exact location of the delivery driver, offering precise delivery times. This will reduce uncertainty and improve the overall experience.

3. Credit Points and Discounts:

To retain customers and foster loyalty, the app will reward users with credit points for every order placed. These points can be accumulated and redeemed for discounts on future orders. The introduction of tier-based rewards can also motivate customers to place larger or more frequent orders to unlock greater benefits. Additionally, personalized discount offers based on order history can be provided, ensuring that users feel valued and incentivized to continue using the app.

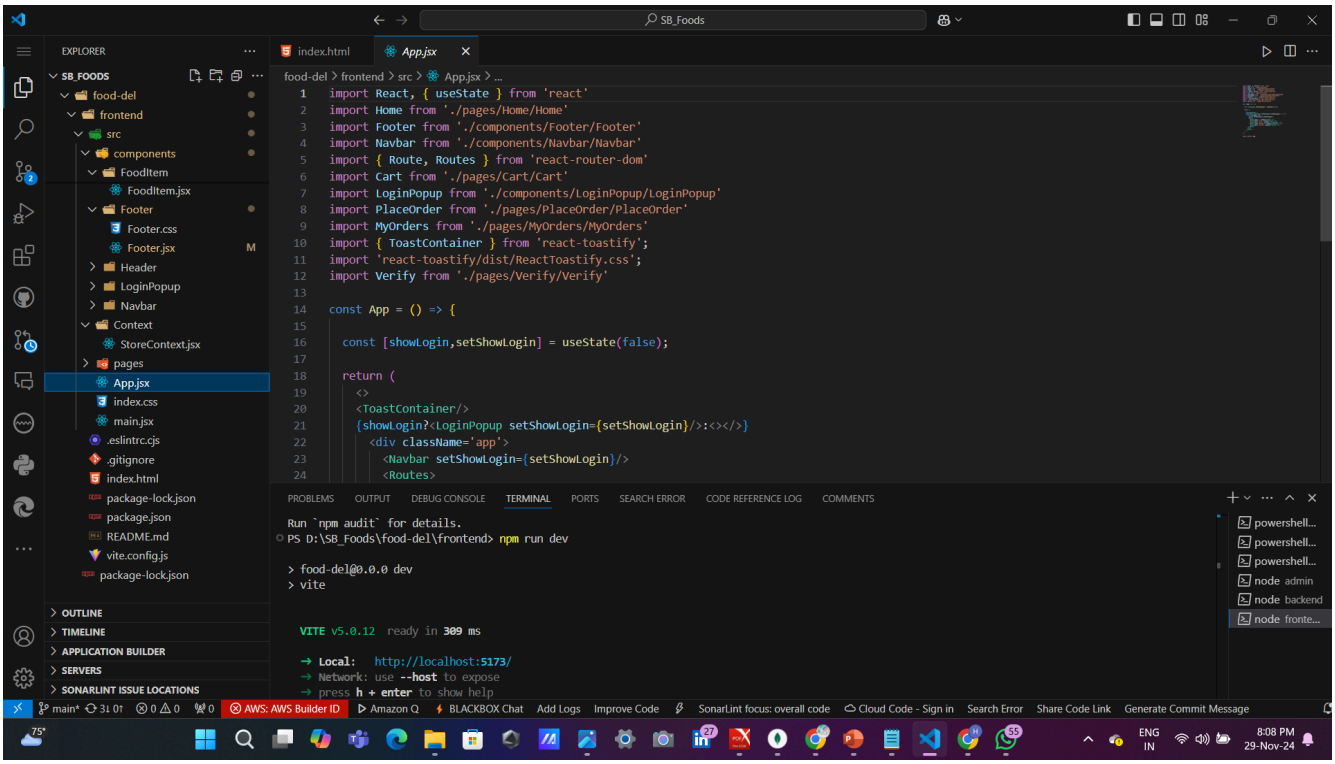
15). Conclusion:

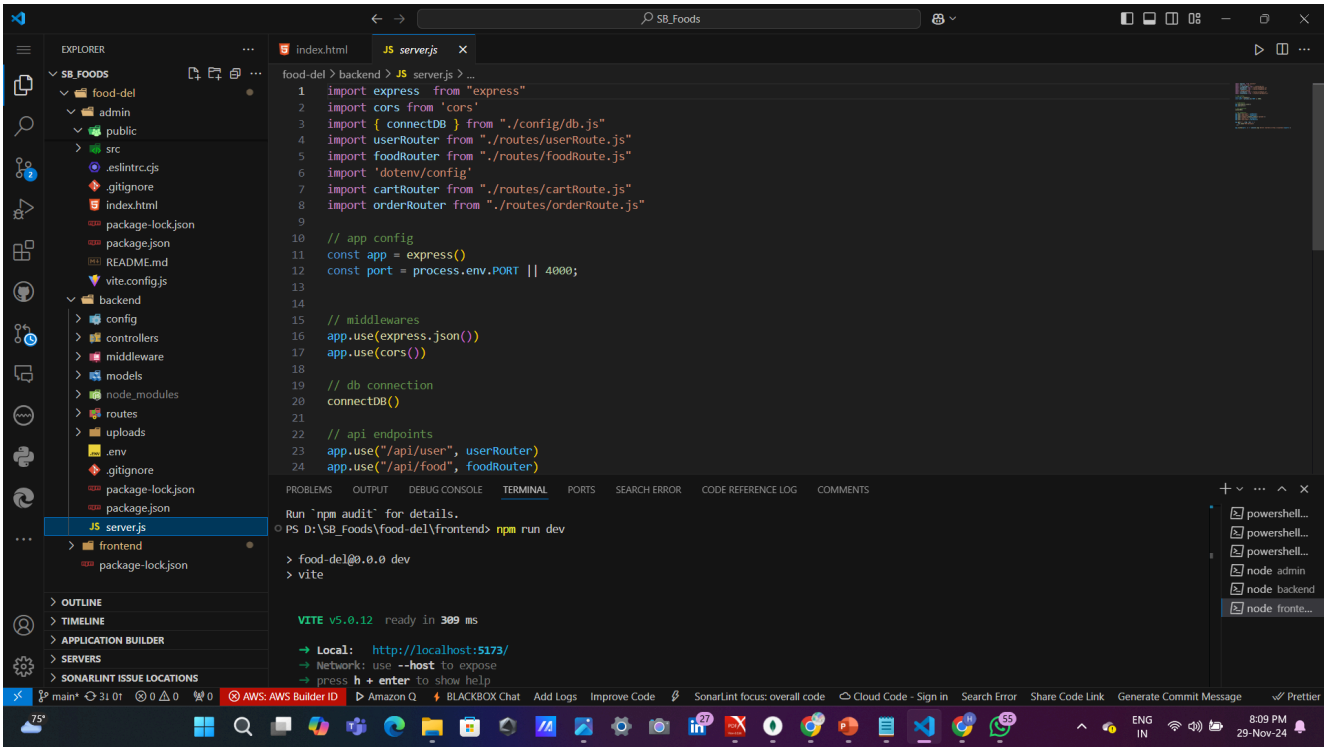
The **SB-Foods Food Ordering App** simplifies the food ordering process, making it convenient and efficient for customers to explore, customize, and enjoy their favorite meals. Through a user-friendly interface and seamless integration of key features, the app enhances customer satisfaction and supports local food businesses. The development process provided valuable insights into the implementation of real-world software solutions and user-centric design principles.

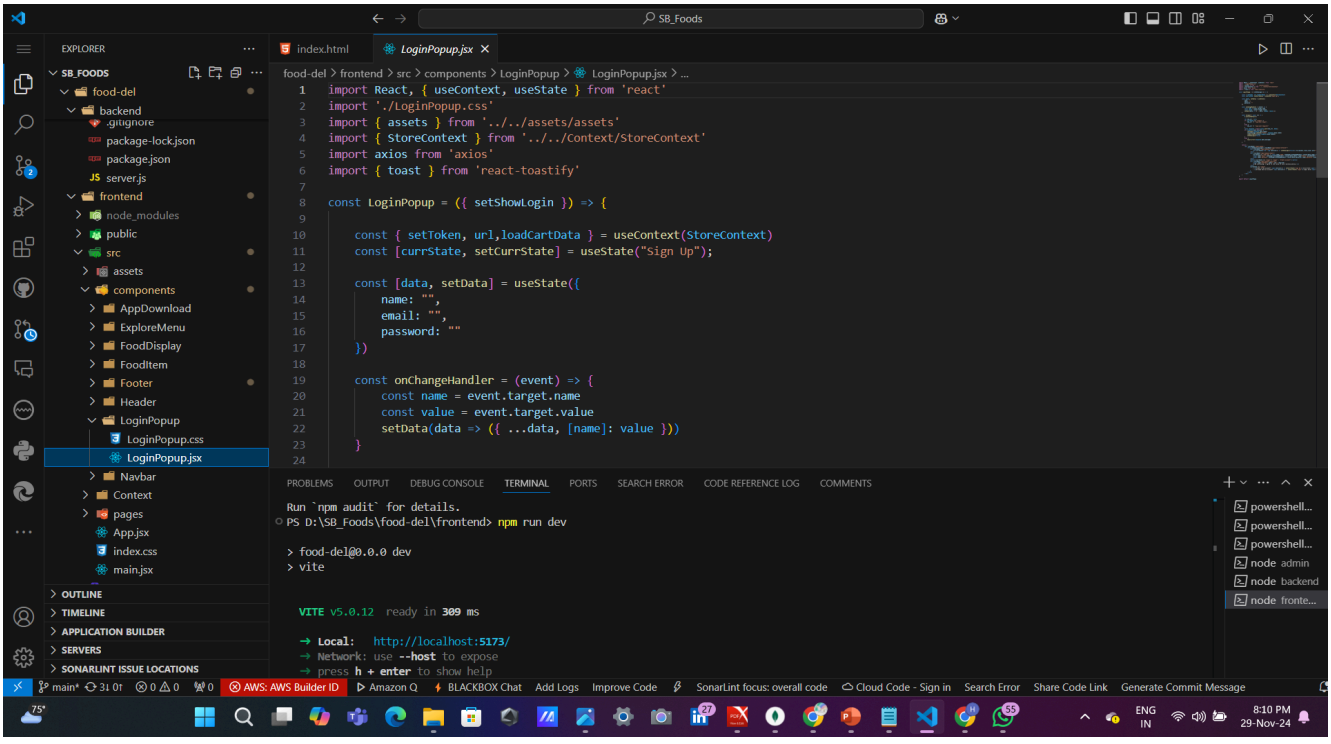
With further enhancements, the app has the potential to become a market leader in the food-tech industry, offering innovative features that cater to the evolving needs of customers.

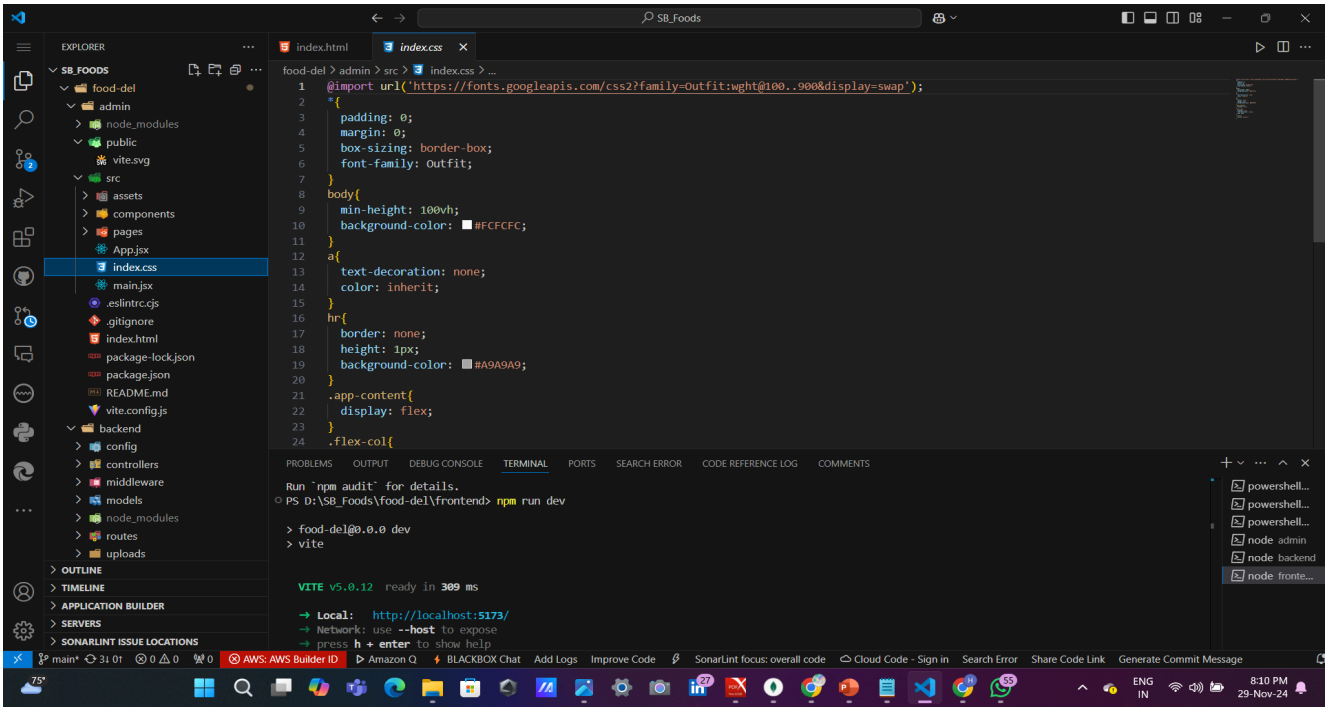
Appendices

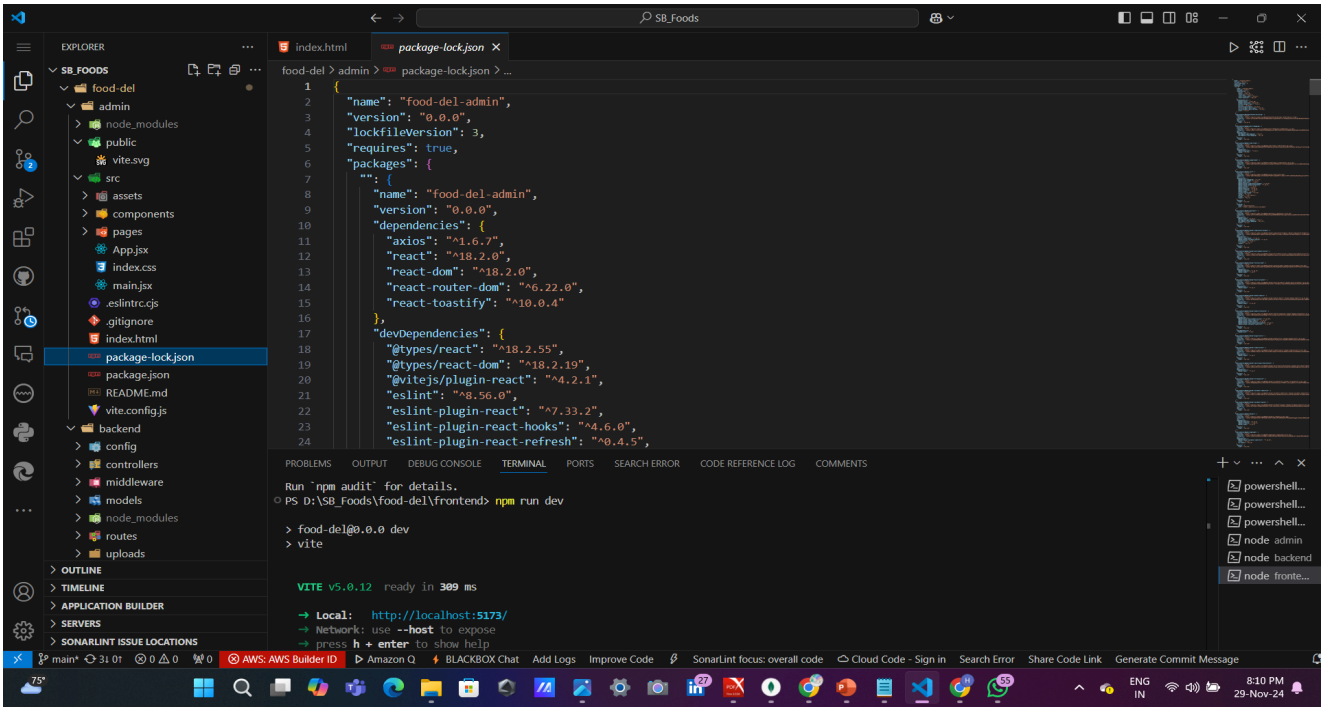
Code snippets:




SB_Foods
EXPLORER
SB_FOODS
food-del
admin
public
src
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
vite.config.js
backend
config
controllers
middleware
models
node_modules
routes
uploads
.env
.gitignore
package-lock.json
package.json
server.js
frontend
package-lock.json
index.html
server.js
food-del > backend > server.js > ...
1 import express from "express"
2 import cors from 'cors'
3 import { connectDB } from "./config/db.js"
4 import userRouter from "./routes/userRoute.js"
5 import foodRouter from "./routes/foodRoute.js"
6 import 'dotenv/config'
7 import cartRouter from "./routes/cartRoute.js"
8 import orderRouter from "./routes/orderRoute.js"
9
10 // app config
11 const app = express()
12 const port = process.env.PORT || 4000;
13
14
15 // middlewares
16 app.use(express.json())
17 app.use(cors())
18
19 // db connection
20 connectDB()
21
22 // api endpoints
23 app.use("/api/user", userRouter)
24 app.use("/api/food", foodRouter)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR CODE REFERENCE LOG COMMENTS
Run `npm audit` for details.
PS D:\SB_Foods\food-del\frontend> npm run dev
> food-del@0.0.0 dev
> vite
VITE v5.0.12 ready in 309 ms
➜ Local: http://localhost:5173/
➜ Network: use --host to expose
➜ press h + enter to show help
powershell...
powershell...
powershell...
node admin
node backend
node fronte...
OUTLINE
TIMELINE
APPLICATION BUILDER
SERVERS
SONARLINT ISSUE LOCATIONS
main* 31↓ 0↑ 0 0 0 AWS: AWS Builder ID Amazon Q BLACKBOX Chat Add Logs Improve Code SonarLint focus: overall code Cloud Code - Sign in Search Error Share Code Link Generate Commit Message Prettier
75°
ENG IN
8:09 PM 29-Nov-24


SB_Foods
EXPLORER
SB_FOODS
food-del
backend
.gitignore
package-lock.json
package.json
server.js
frontend
node_modules
public
src
assets
components
AppDownload
ExploreMenu
FoodDisplay
FoodItem
Footer
Header
LoginPopup
LoginPopup.css
LoginPopup.jsx
Navbar
Context
pages
App.jsx
index.css
main.jsx
index.html
LoginPopup.jsx
food-del > frontend > src > components > LoginPopup > LoginPopup.jsx > ...
1 import React, { useContext, useState } from 'react'
2 import './LoginPopup.css'
3 import { assets } from '../../assets/assets'
4 import { StoreContext } from '../../Context/StoreContext'
5 import axios from 'axios'
6 import { toast } from 'react-toastify'
7
8 const LoginPopup = ({ setShowLogin }) => {
9
10 const { setToken, url,loadCartData } = useContext(StoreContext)
11 const [currState, setCurrState] = useState("Sign Up");
12
13 const [data, setData] = useState({
14 name: "",
15 email: "",
16 password: ""
17 })
18
19 const onChangeHandler = (event) => {
20 const name = event.target.name
21 const value = event.target.value
22 setData(data => ({ ...data, [name]: value }))
23 }
24
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR CODE REFERENCE LOG COMMENTS
Run `npm audit` for details.
PS D:\SB_Foods\food-del\frontend> npm run dev
> food-del@0.0.0 dev
> vite
VITE v5.0.12 ready in 309 ms
➜ Local: http://localhost:5173/
➜ Network: use --host to expose
➜ press h + enter to show help
powershell...
powershell...
powershell...
node admin
node backend
node fronte...
OUTLINE
TIMELINE
APPLICATION BUILDER
SERVERS
SONARLINT ISSUE LOCATIONS
main* 31↓ 0↑ 0 0 0 AWS: AWS Builder ID Amazon Q BLACKBOX Chat Add Logs Improve Code SonarLint focus: overall code Cloud Code - Sign in Search Error Share Code Link Generate Commit Message
75°
ENG IN
8:10 PM 29-Nov-24


SB_Foods
EXPLORER
SB_FOODS
food-del
admin
node_modules
public
vite.svg
src
assets
components
pages
App.jsx
index.css
main.jsx
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
vite.config.js
backend
config
controllers
middleware
models
node_modules
routes
uploads
OUTLINE
TIMELINE
APPLICATION BUILDER
SERVERS
SONARLINT ISSUE LOCATIONS
index.html
index.css
food-del > admin > src > index.css > ...
@import url('https://fonts.googleapis.com/css2?family=Outfit:wght@100..900&display=swap');
*{
padding: 0;
margin: 0;
box-sizing: border-box;
font-family: Outfit;
}
body{
min-height: 100vh;
background-color: #FCFCFC;
}
a{
text-decoration: none;
color: inherit;
}
hr{
border: none;
height: 1px;
background-color: #A9A9A9;
}
.app-content{
display: flex;
}
.flex-col{
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
CODE REFERENCE LOG
COMMENTS
Run `npm audit` for details.
PS D:\SB_Foods\food-del\frontend> npm run dev
> food-del@0.0.0 dev
> vite
VITE v5.0.12 ready in 309 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
powershell...
powershell...
powershell...
node admin
node backend
node fronte...
main*
AWS: AWS Builder ID
Amazon Q
BLACKBOX Chat
Add Logs
Improve Code
SonarLint focus: overall code
Cloud Code - Sign in
Search Error
Share Code Link
Generate Commit Message
ENG
IN
8:10 PM
29-Nov-24


SB_Foods
EXPLORER
SB_FOODS
food-del
admin
node_modules
public
vite.svg
src
assets
components
pages
App.jsx
index.css
main.jsx
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
vite.config.js
backend
config
controllers
middleware
models
node_modules
routes
uploads
OUTLINE
TIMELINE
APPLICATION BUILDER
SERVERS
SONARLINT ISSUE LOCATIONS
index.html
package-lock.json
food-del > admin > package-lock.json > ...
{
"name": "food-del-admin",
"version": "0.0.0",
"lockfileVersion": 3,
"requires": true,
"packages": {
"": {
"name": "food-del-admin",
"version": "0.0.0",
"dependencies": {
"axios": "^1.6.7",
"react": "^18.2.0",
"react-dom": "^18.2.0",
"react-router-dom": "^6.22.0",
"react-toastify": "^10.0.4"
},
"devDependencies": {
"@types/react": "^18.2.55",
"@types/react-dom": "^18.2.19",
"@vitejs/plugin-react": "^4.2.1",
"eslint": "^8.56.0",
"eslint-plugin-react": "^7.33.2",
"eslint-plugin-react-hooks": "^4.6.0",
"eslint-plugin-react-refresh": "^0.4.5",
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
SEARCH ERROR
CODE REFERENCE LOG
COMMENTS
Run `npm audit` for details.
PS D:\SB_Foods\food-del\frontend> npm run dev
> food-del@0.0.0 dev
> vite
VITE v5.0.12 ready in 309 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help
powershell...
powershell...
powershell...
node admin
node backend
node fronte...
main*
AWS: AWS Builder ID
Amazon Q
BLACKBOX Chat
Add Logs
Improve Code
SonarLint focus: overall code
Cloud Code - Sign in
Search Error
Share Code Link
Generate Commit Message
ENG
IN
8:10 PM
29-Nov-24

16) Result:

The **SB-Foods Food Ordering App** successfully achieved the following objectives:

1. Customers can easily browse the menu and add items like **Curd Rice** and **Sambar Rice** to their cart.
2. A seamless ordering and checkout process was implemented, enabling customers to place orders with minimal effort.
3. The backend integration ensures efficient data storage and retrieval, maintaining a responsive app experience.
4. The app lays a strong foundation for further expansion with modular and scalable code architecture.

This project demonstrates the practical application of full-stack development skills and showcases the feasibility of creating an intuitive and functional food ordering platform.

****