OrderOnTheGo: Your On-Demand Food Ordering Solution

Team ID :LTVIP2025TMID55780

MEMBER1: KAVETI BHAVANA

MEMBER2: KOLUKULA GNANA SIRISHA

MEMBER3: KODALI MADHU KANTH

MEMBER4: KOKA SARATH MAHESH

Introduction:

Welcome to SB Foods, your ultimate destination for convenient food ordering and delivery, bringing diverse cuisines from local restaurants right to your doorstep. It allows users to browse food categories and restaurants, view detailed menus, add items to a shopping cart with transparent pricing, and track their orders. The platform features secure user authentication for customers, restaurants, and administrators. Restaurant owners can manage their menu items and orders through a dedicated dashboard, while administrators have overall control to manage users, restaurants, and all orders. This system streamlines the entire food ordering and delivery process for all involved.

Description:

SB Foods is a complete system for online food ordering. It facilitates the connection between users and various restaurants, providing access to their menus. The platform enables users to easily browse different food categories, select desired items, place their orders, and monitor the delivery status. Key functionalities include displaying popular food types and restaurants, allowing users to view detailed menus from specific eateries, managing orders, and offering a clear shopping cart experience with price breakdowns. It also features user authentication for secure access and dedicated dashboards for both restaurant owners to manage their items and orders, and administrators to oversee users, restaurants, and overall orders. Business & Technology Focused: SB Foods leverages technology to reshape food ordering and delivery. Our intelligent, reliable, and user-friendly platform empowers customers, restaurants, and delivery partners to connect effortlessly, ensuring fresh food, quick service, and a seamless dining experience.

Scenario Based Case Study:

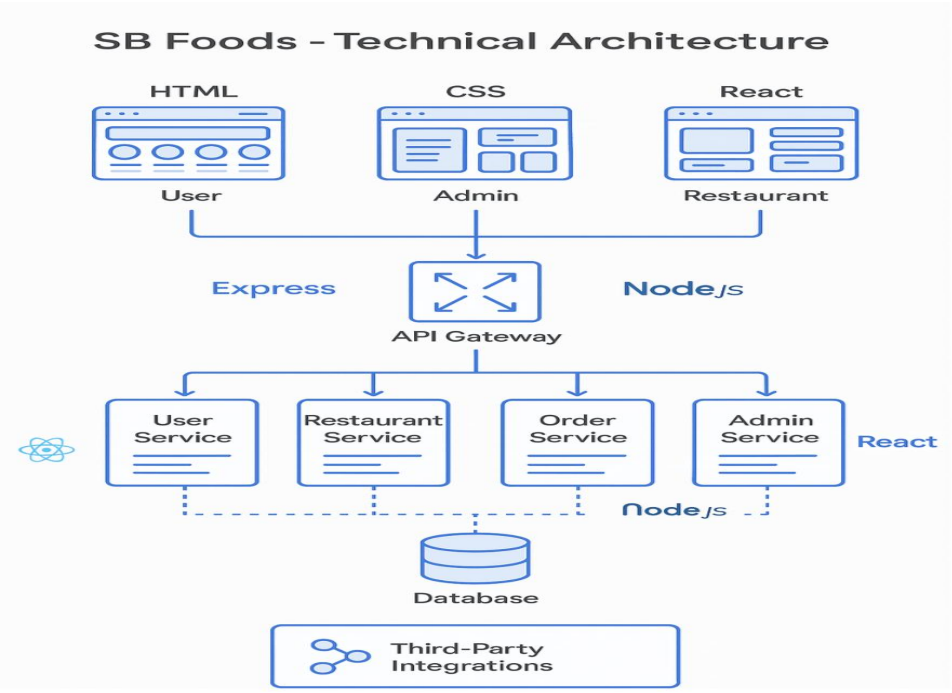
Meet Hola, a busy professional in Hyderabad. Hola often works late and needs a convenient way to order dinner without cooking. She decides to try SB Foods, signs up on the platform, and quickly sets up her profile.

- **User Registration and Authentication:** Hola registers on the platform, verifies her account, and sets up her basic user profile.

- **Restaurant Browse:** She explores available restaurants and cuisines using category filters (like "Biriyani" or "Pizza") and location tags to find dinner options near her office in Madhapur.
- **Smart Suggestions:** Based on her previous orders or Browse history, SB Foods recommends restaurants and dishes aligned with her preferences (e.g., suggesting "Minarva Grand" after she frequently orders Indian food).
- **Order Placement:** She browses the menu of "Minarva Grand," selects "Chicken Biriyani" and "Butter Chicken," customizes her order, and proceeds to checkout, confirming her delivery address.
- **Secure Payments:** After reviewing her cart and the final price (including discounts and delivery charges), she makes a secure payment through SB Foods' integrated payment system.
- **Order Tracking & Delivery:** Hola tracks her order in real-time, receiving updates as the restaurant prepares the food and the delivery executive is on the way. She successfully receives her order.
- **Rating & Review:** Hola enjoys her meal and gives "Minarva Grand" a positive rating and review on the SB Foods app, contributing to the restaurant's visibility and helping other users make informed choices.

Thanks to SB Foods, Hola can easily enjoy delicious meals delivered directly to her doorstep, saving time and effort on cooking or picking up food, and always finding reliable options from her favorite local restaurants.

Technical Architecture:-

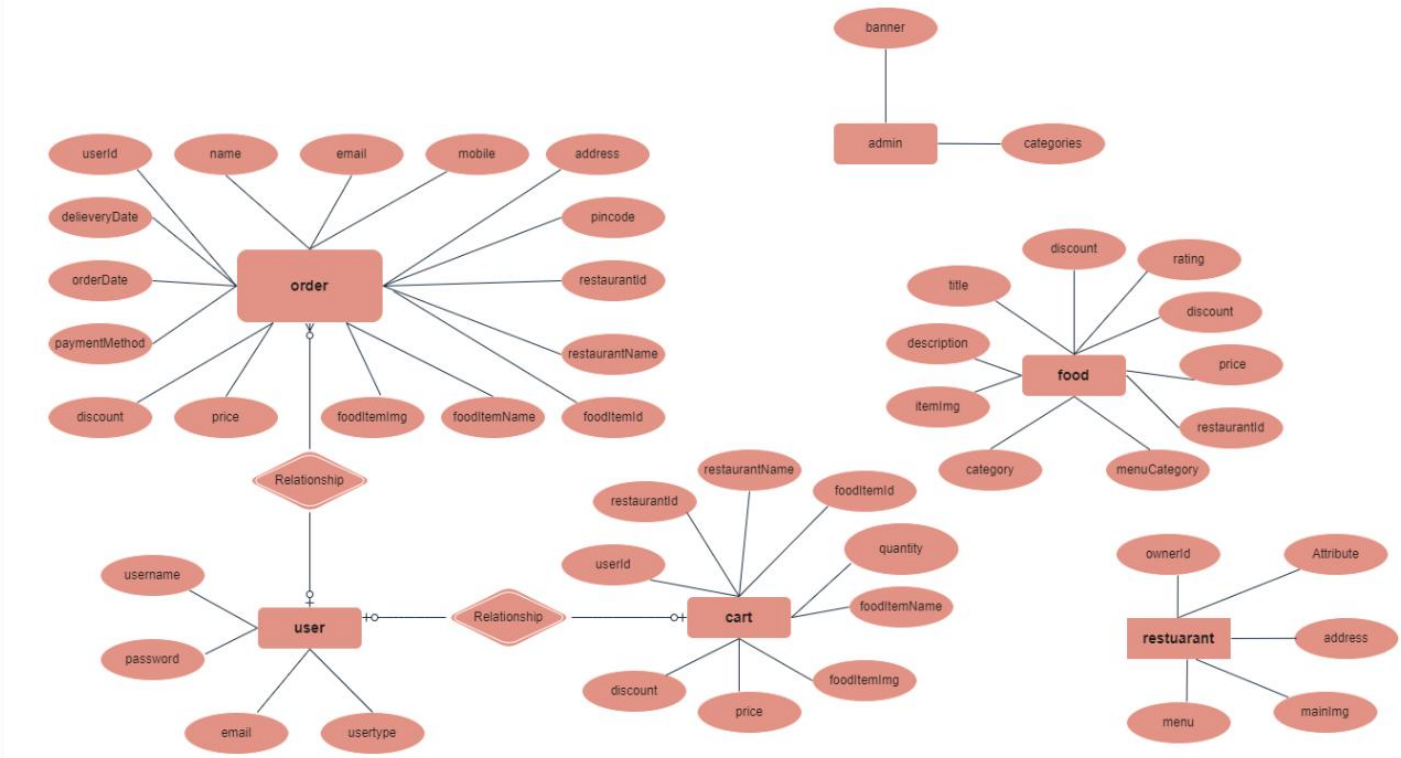


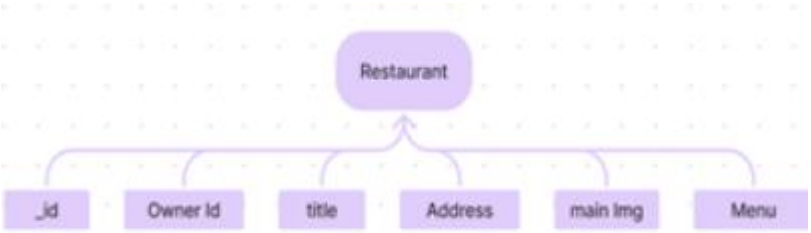
SB Foods follows a scalable web architecture built on the **MERN stack** (MongoDB, Express.js, React.js, Node.js) designed to support online food ordering and restaurant management.

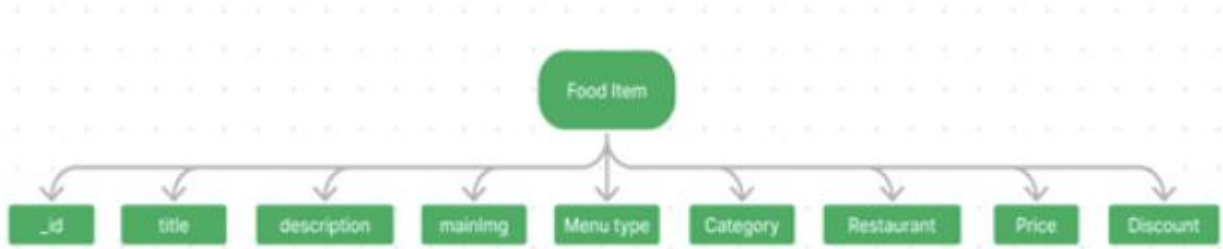
- **Frontend:** Developed using **React.js** to deliver a fast, interactive, and user-friendly experience for customers, restaurants, and admins.
- **Backend:** Powered by **Node.js** and **Express.js**, handling user authentication, restaurant management, menu operations, order processing, and admin functionalities.
- **Database:** Utilizes **MongoDB** to efficiently store user profiles, restaurant data, food items, orders, and reviews.
- **Communication:** Implements **RESTful APIs** for smooth interaction between the frontend and backend, with support for real-time features such as order status updates using **WebSocket** (optional enhancement).
- **Security:** Ensures robust protection through **JWT-based authentication**, encrypted password storage, and secure handling of sensitive operations like payments and file uploads.

This architecture provides a reliable, secure, and responsive platform for seamless food ordering and management.

ER-Diagram:







ER-Diagram Overview:

- Overall view
- Restaurant
- Food Item
- Admin
- Users

Key Features:

- **Profile Management:** Customers, restaurants, and admins can manage and personalize their profiles based on their roles.
- **Restaurant Listings:** Restaurants can register and showcase their menus, locations, and offers for customers to browse.
- **Smart Search & Filters:** Users can easily find restaurants, food items, or cuisines using smart search, categories, and price-based filters.
- **Secure Orders & Payments:** Customers can place food orders with confidence, with secure order management and payment handling (COD/Online Payment).
- **Order Tracking & Notifications:** Real-time order status updates and notifications keep customers informed throughout the ordering process.
- **Cart & Price Breakdown:** Users can manage their cart, view itemized price details, discounts, and delivery charges before checkout.
- **Admin Panel:** Admins can manage restaurants, users, promotions, and orders, ensuring smooth platform operation and quality control.

PRE REQUISITES:

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.
Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.
Download: https://nodejs.org/en/download/
Installation instructions: https://nodejs.org/en/download/package-manager/

Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.
Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.
Installation: Open your command prompt or terminal and run the following command:
npm install express

MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.
Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community
Installation instructions: https://docs.mongodb.com/manual/installation/

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.
Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

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.

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

Front-end Framework: Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

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

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

Install Dependencies:

• Navigate into the cloned repository directory:
cd freelancer-app-MERN

• Install the required dependencies by running the following commands:

cd client
npm install
../cd server
npm install

Start the Development Server:
• To start the development server, execute the following command:
npm start
• This app will be accessible at http://localhost:3000

You have successfully installed and set up the SB application on your local machine. You can now proceed with further customization, development, and testing as needed.

Roles and Responsibility:

Customer Responsibilities:

- **Order Placement:** Customers are responsible for placing accurate orders, reviewing item details, quantities, and delivery information before confirming.
- **Timely Communication:** Respond to calls or messages from delivery personnel or restaurants to ensure smooth order processing.
- **Payment Obligations:** Complete payments for orders as per the selected payment method, whether online or cash on delivery.
- **Feedback & Ratings:** Provide honest feedback and ratings for restaurants and food quality, helping improve service standards.
- **Responsible Use:** Use the platform ethically, respecting restaurant partners and delivery staff.

Restaurant Responsibilities:

- **Menu Management:** Keep menu items, prices, availability, and offers updated to ensure customers have accurate information.
- **Order Fulfillment:** Prepare and deliver food orders in a timely manner, ensuring quality, packaging, and accuracy.
- **Communication:** Stay responsive to customer inquiries or special instructions provided with the order.
- **Compliance:** Adhere to platform guidelines related to food safety, hygiene, and operational standards.
- **Professionalism:** Maintain a professional and customer-friendly attitude to build reputation and trust on the platform.

Admin Responsibilities:

- **Platform Oversight:** Monitor platform operations, ensuring smooth functioning for customers and restaurants.
- **Policy Enforcement:** Enforce platform policies, guidelines, and quality standards for both customers and restaurants.
- **Conflict Resolution:** Address complaints, disputes, or technical issues fairly and promptly to maintain user satisfaction.
- **User Support & Guidance:** Provide assistance and support to customers and restaurants regarding platform usage.
- **Platform Maintenance & Improvement:** Continuously improve platform features, security, and performance to enhance user experience.

PROJECT FLOW:-

Before starting to work on this project, let's see the demo.

Demo link:-
https://drive.google.com/file/d/18-gnCVAZdojBhl7QM0_yShH1JL5A6v32/view?usp=drive_link

Use the code in:-

https://drive.google.com/drive/folders/1gSc8RWmM7yqdvYh525n5DoLjZxS_N-F4

or follow the videos below for better understanding.

Milestone 1: Project Setup and Configuration:

1. Install required tools and software:

- Node.js.
- MongoDB.
- Create-react-app.

2. Create project folders and files:

- Client folders.
- Server folders.

3. Install Packages:

Frontend npm Packages

- Node
- React
- Bootstrap.

Backend npm Packages

- Express.

- Mongoose.
- Body-parser
- Cors.

Milestone 2: Backend Development:

- **Setup express server**
    1. Create index.js file in the server (backend folder).
    2. Create a .env file and define port number to access it globally.
    3. Configure the server by adding cors, body-parser.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Define API Routes:**
    - Create separate route files for different API functionalities such as users orders, and authentication.
    - Define the necessary routes for listing products, handling user registration and login,managing orders, etc.
    - Implement route handlers using Express.js to handle requests and interact with the database.

- **Implement Data Models:**
    - Define Mongoose schemas for the different data entities like products, users, and orders.
    - Create corresponding Mongoose models to interact with the MongoDB database.
    - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Error Handling:**
    - Implement error handling middleware to catch and handle any errors that occur during the API requests.
    - Return appropriate error responses with relevant error messages and HTTP status codes.

Milestone 3: Database:

1. Configure MongoDB:

- Install Mongoose.
- Create database connection.
- Create Schemas & Models.

2. Connect database to backend:

Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.

```
                                    const mongoose = require("mongoose");

const db= 'mongodb://127.0.0.1:27017/grocery'
// Connect to MongoDB using the connection string

mongoose.connect(db, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => {
  console.log(`Connection successful`);
}).catch((e) => {
  console.log(`No connection: ${e}`);
});
```

3. Configure Schema:

Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The schemas are looks like for the Application.

```
import mongoose, { Schema, mongo } from "mongoose";

const userSchema = mongoose.Schema({
    username: {
        type: String,
        require: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    usertype:{
        type: String,
        require: true
    }
})

const freelancerSchema = mongoose.Schema({
    userId: String,
    skills: {
        type: Array,
        default: []
```

```
  },
  description: {
    type: String,
    default: ""
  },
  currentProjects: {
    type: Array,
    default: []
  },
  completedProjects: {
    type: Array,
    default: []
  },
  applications: {
    type: Array,
    default: []
  },
  funds: {
    type: Number,
    default: 0
  },
})

const projectSchema = mongoose.Schema({
  clientId: String,
  clientName: String,
  clientEmail: String,
  title: String,
  description: String,
  budget: Number,
  skills: Array,
  bids: Array,
  bidAmounts: Array,
  postedDate: String,
  status: {
    type: String,
    default: "Available"
  },
  freelancerId: String,
  freelancerName: String,
  deadline: String,
  submission: {
    type: Boolean,
    default: false
  },
  submissionAccepted: {
    type: Boolean,
    default: false
  },
  projectLink: {
    type: String,
    default: ""
  },
  manulaLink: {
    type: String,
    default: ""
  },
  submissionDescription: {
    type: String,
    default: ""
  },
})

const applicationSchema = mongoose.Schema({
```

```
  projectId: String,
  clientId: String,
  clientName: String,
  clientEmail: String,
  freelancerId: String,
  freelancerName: String,
  freelancerEmail: String,
  freelancerSkills: Array,
  title: String,
  description: String,
  budget: Number,
  requiredSkills: Array,
  proposal: String,
  bidAmount: Number,
  estimatedTime: Number,
  status: {
    type: String,
    default: "Pending"
  }
})

const chatSchema = mongoose.Schema({
  _id: {
    type: String,
    require: true
  },
  messages: {
    type: Array
  }
})

export const User = mongoose.model('users', userSchema);
export const Freelancer = mongoose.model('freelancer', freelancerSchema);
export const Project = mongoose.model('projects', projectSchema);
export const Application = mongoose.model('applications', applicationSchema);
export const Chat = mongoose.model('chats', chatSchema);
```

Milestone 4: Frontend Development:

1. Setup React Application:

- Create React application.
- Configure Routing.
- Install required libraries.

2. Design UI components:

- Create Components.
- Implement layout and styling.
- Add navigation.

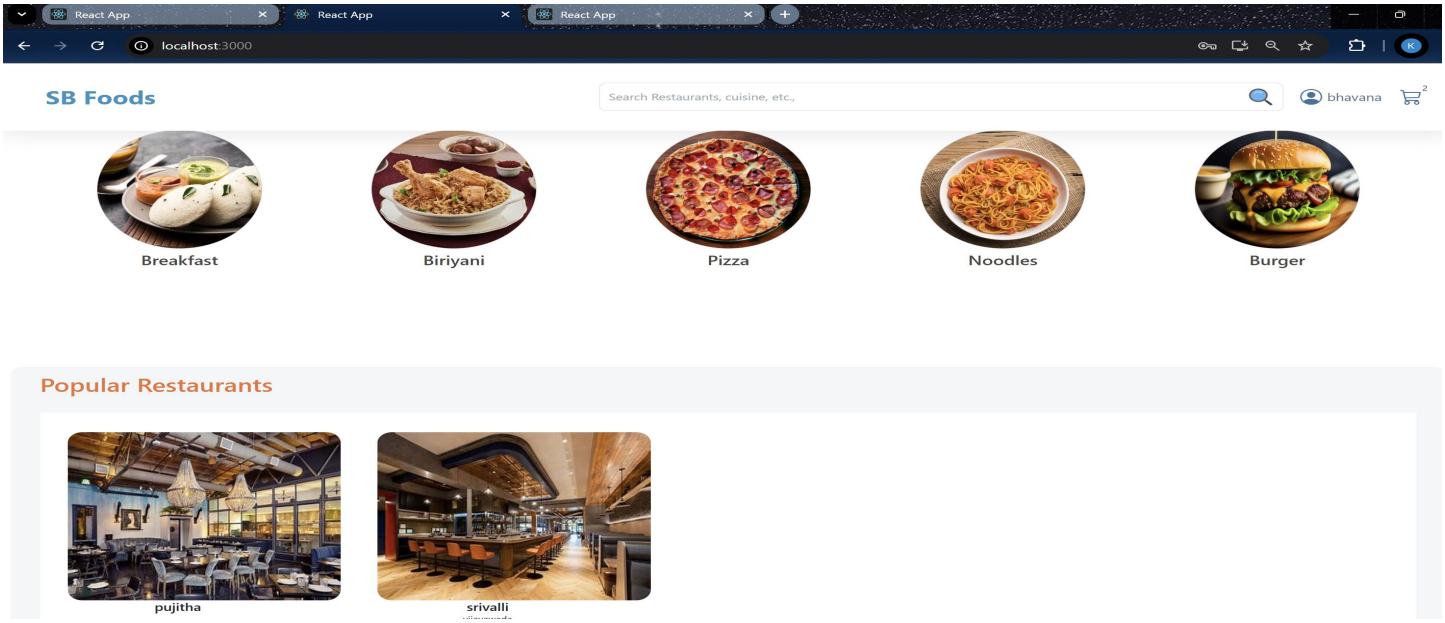
3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

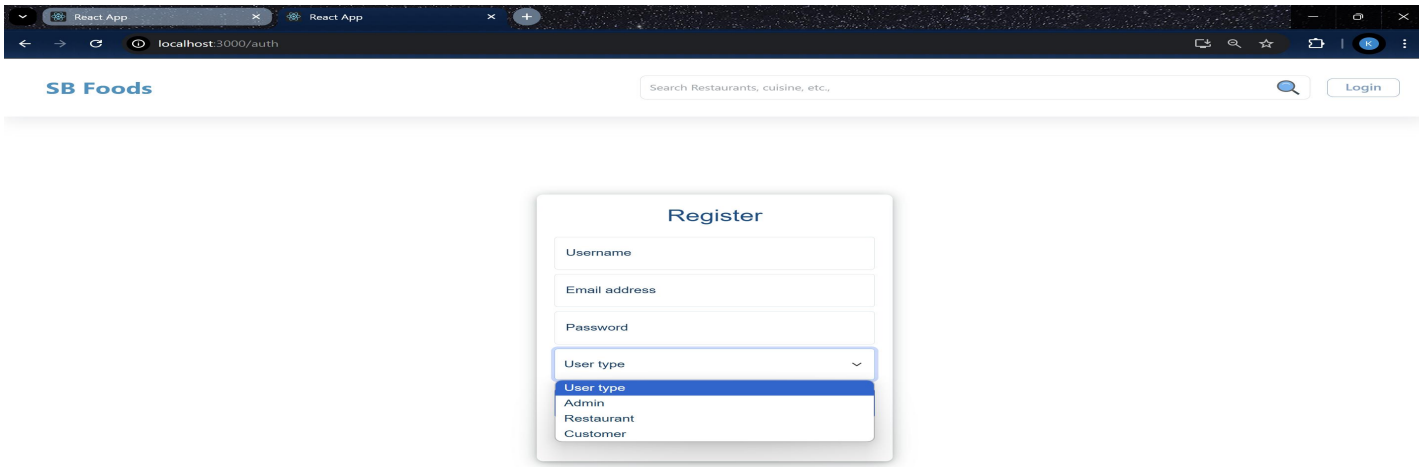
Milestone 5: Project Implementation:

Finally, after finishing coding the projects we run the whole project to test it's working process and look for bugs. Now, let's have a final look at the working of our Darshan Ease.

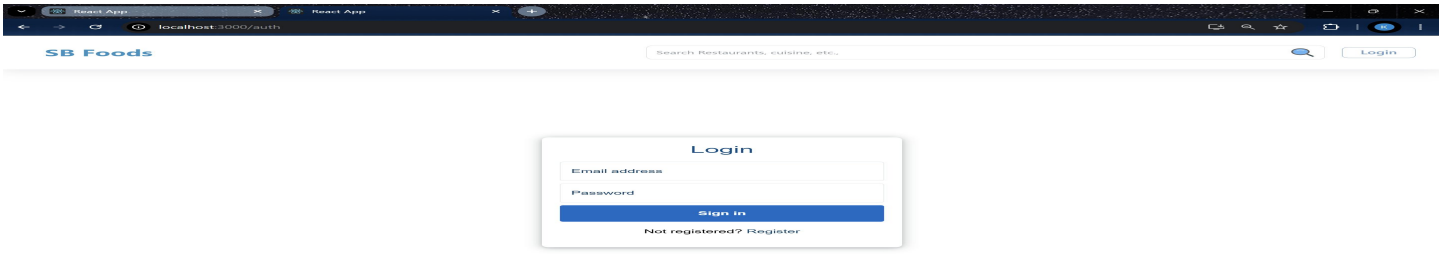
Landing page:-



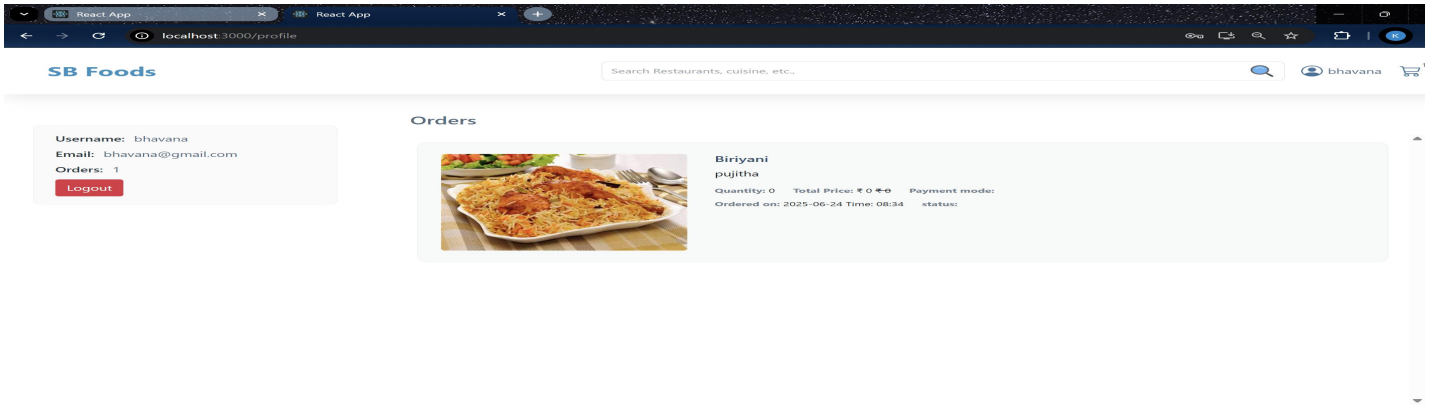
Register Page:



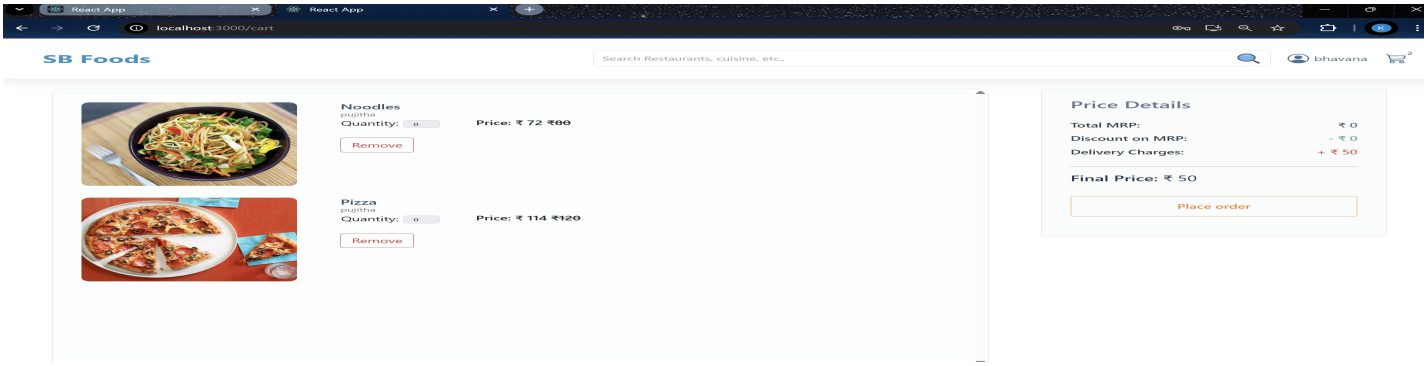
Login Page:



Customer page:

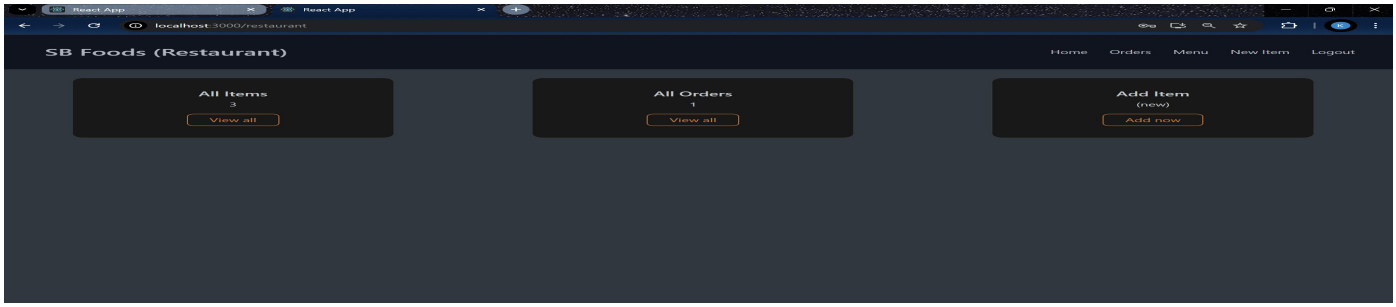


Cart Page:

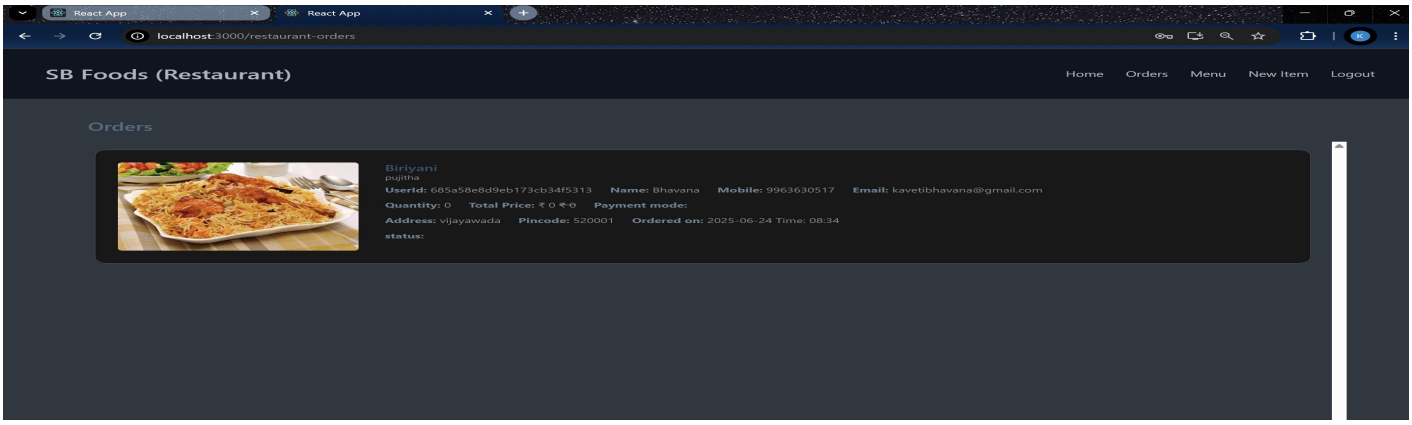


Restaurant page:

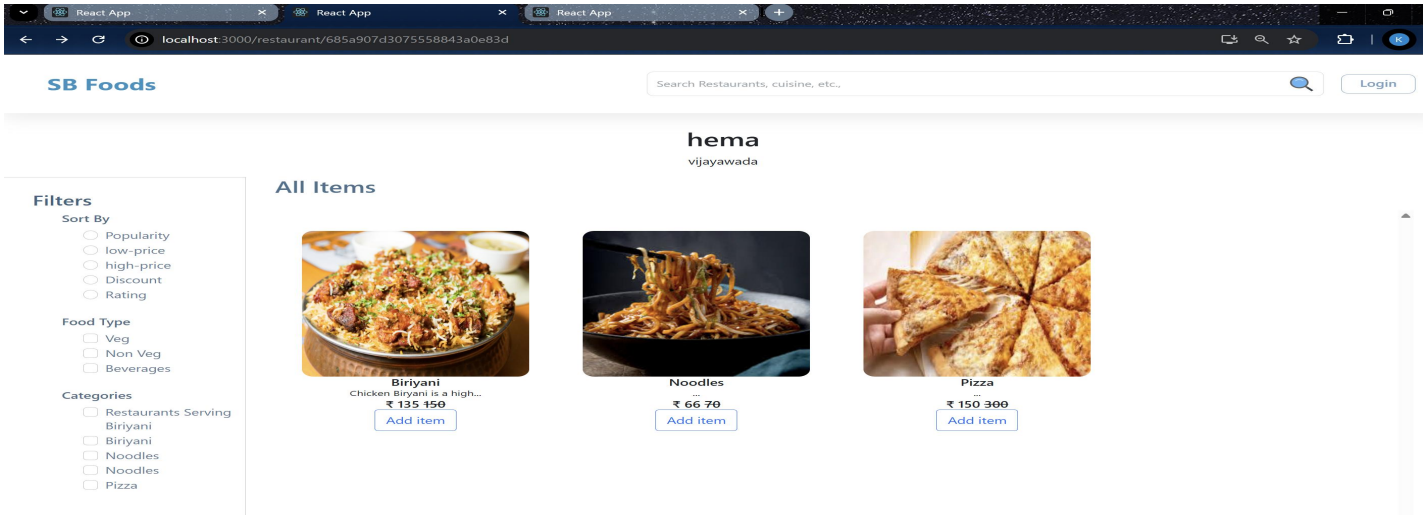
Home page



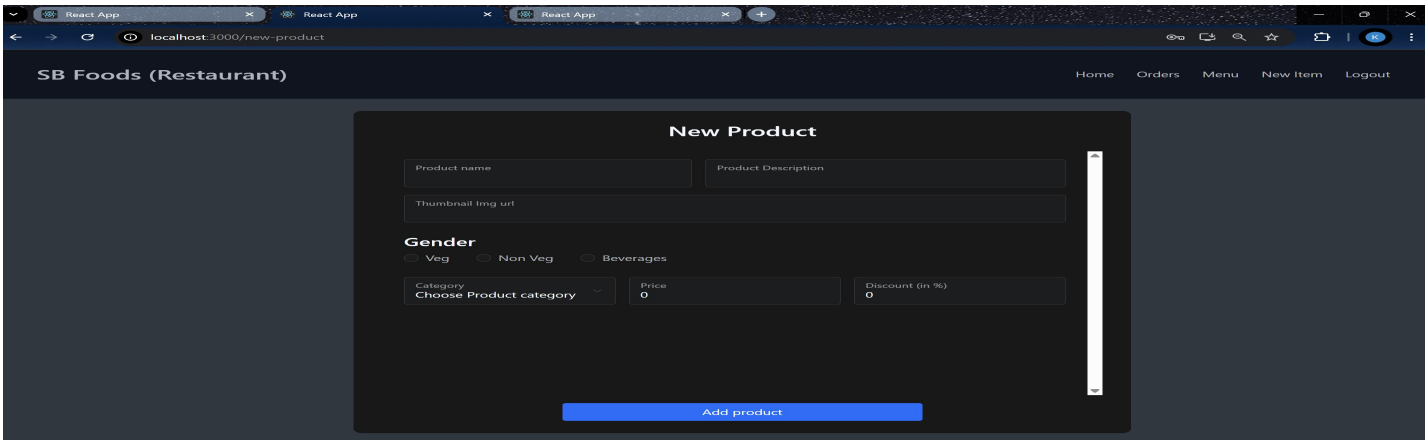
Orders page



Menu Page

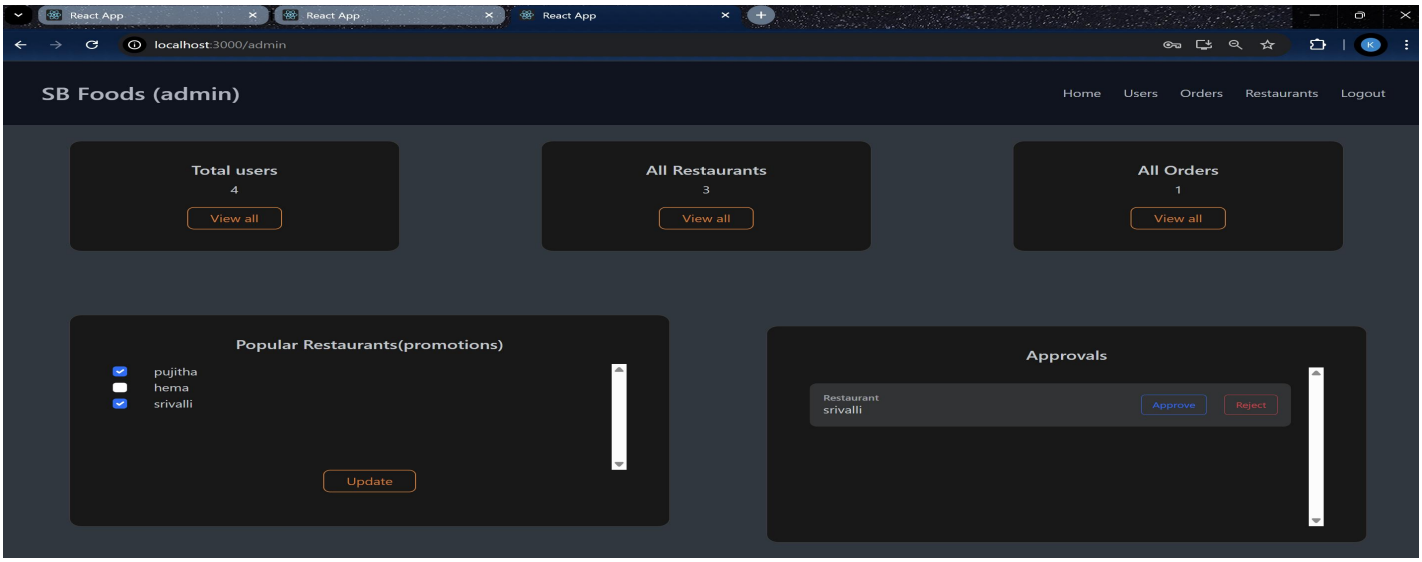


New Item to be added:

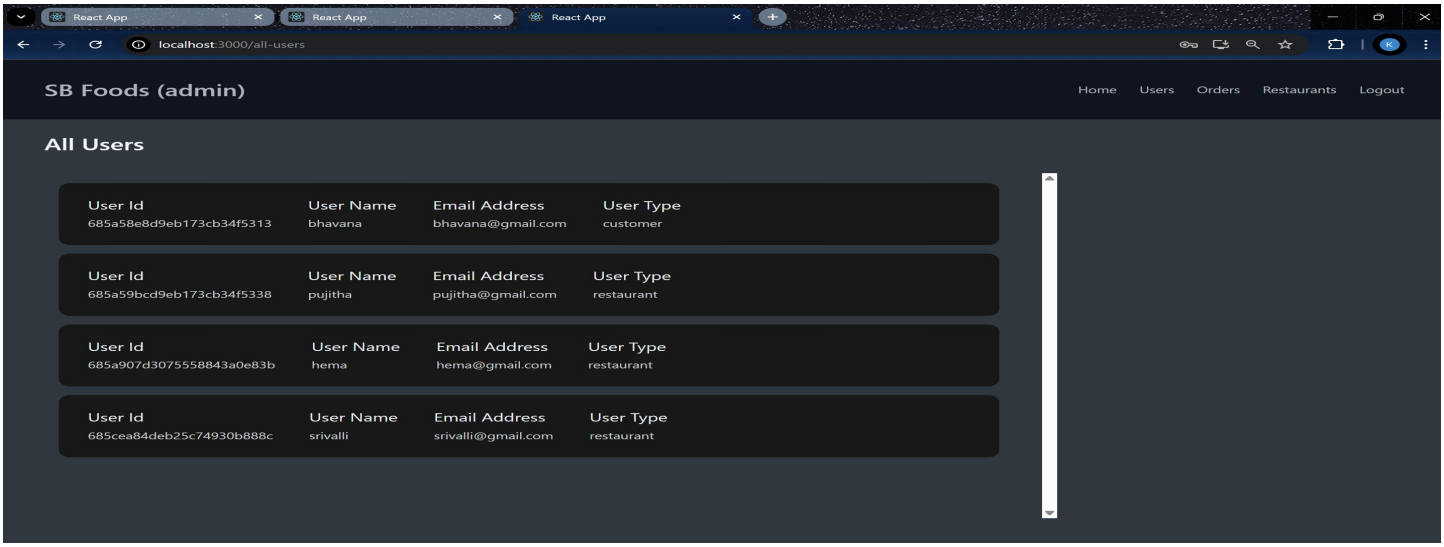


Admin page:

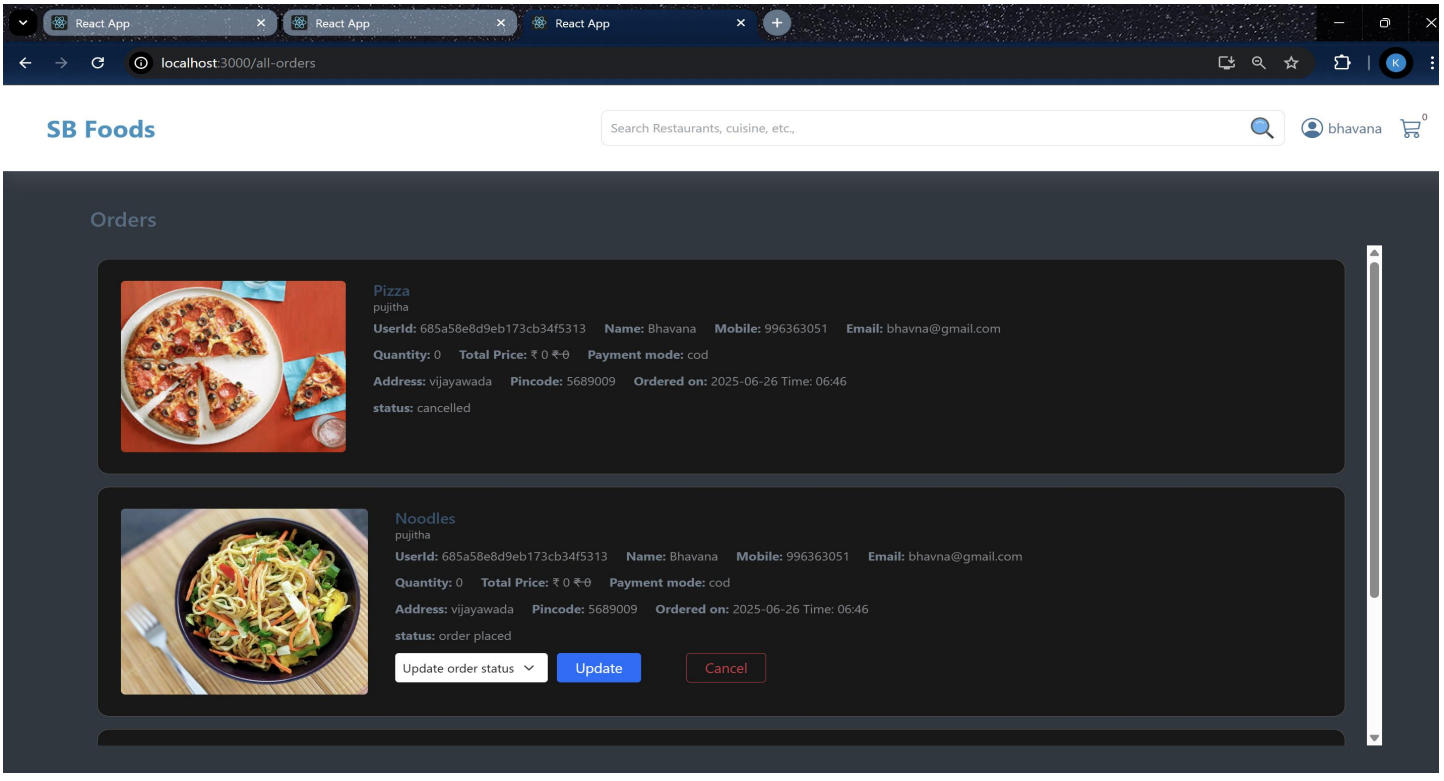
Home page:



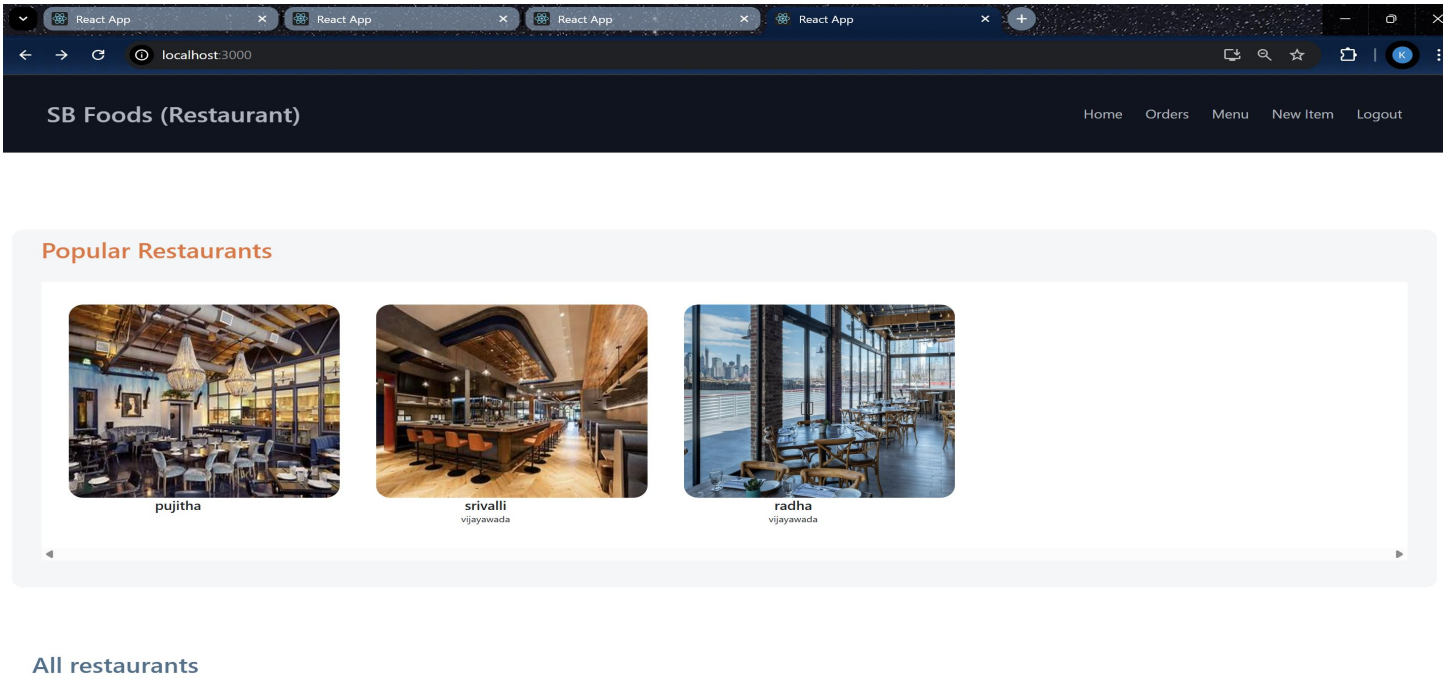
Users page:



Orders page:



Restaurants Page:



Conclusion:

SB Foods empowers individuals to enjoy hassle-free food ordering and restaurant discovery, one meal at a time. By providing an intelligent, secure, and user-friendly platform, we aim to revolutionize the way people experience food, connecting customers with their favorite restaurants quickly and conveniently.We believe that ordering food should be effortless, reliable, and enjoyable. With SB Foods, users can explore a wide range of restaurants, browse detailed menus, and place orders with just a few taps. Our platform is designed to ensure secure transactions, real-time order tracking, and prompt delivery, making every meal experience smooth and satisfying. By continuously innovating and listening to our customers' needs, we are committed to building a food ordering ecosystem that prioritizes convenience, quality, and trust. Whether it's breakfast, lunch, dinner, or a late-night craving, SB Foods is here to serve — anytime, anywhere.