OrderOnTheGo : Your On-Demand Food Ordering Solution - Project Documentation

1.Introduction:

Project Title: OrderOnTheGo:Your On-Demand Food Ordering Solution.

Team ID: LTVIP2025TMID42697

Team Members:

1.Ganisetti Satya Ajay (Team Leader),

2.Antarvedi Kushal Sai Kumar (Team Member),

3.Barla Anu Pranita Sai Ramya Sree (Team Member),

4.B.Thulasi Devi (Team Member).

2.Project Overview:

2.1.Purpose:

OrderOnTheGo is a web-based food ordering platform that connects users with restaurants for quick and easy meal ordering. It streamlines the ordering process, provides real-time order tracking, and offers an intuitive user interface for both customers and restaurant owners.

2.2.Features:

- User authentication (Customers & Restaurants)
- Restaurant and Menu management
- Real-time food order placement and status tracking
- Admin panel for managing users and orders
- Search/filter by cuisine or restaurant

- Dashboards for both users and restaurants

3.Architecture:

3.1.Frontend:

Built using **React.js** with dynamic routing, responsive UI, and component-based design.

3.2.Backend:

Developed with **Node.js** and **Express.js**, providing RESTful APIs for frontend interaction.

3.3.Database:

MongoDB with **Mongoose** for schema and data modelling.

4.Setup Instructions

4.1. Prerequisites:

- Node.js
- MongoDB

4.2. Installation:

1. Clone the repository
2. Install dependencies using npm install.
3. Create a .env file and add environment variables.

4.3.Environment Variables:

```
MONGO_URI=my_mongo_connection_URL
```

```
JWT_SECRET=my_secret
PORT=6001
```

5.Folder Structure:

5. 1.Client:

- src/components/ – Reusable components like Navbar, MenuList, etc.
- src/pages/ – Views like Login, Dashboard, Orders
- App.js – Routing setup.

5. 2.Server:

- routes/ – API endpoints for users, orders, menus
- controllers/ – Route logic
- models/ – MongoDB schemas (User, Order, MenuItem)
- server.js – Entry point of the backend .

6.Running the Application

6. 1.Frontend:

```
cd client && npm start
```

6.2.Backend:

```
cd server
&& node index.js
```

7.API Documentation:

- POST /api/register – Register new user

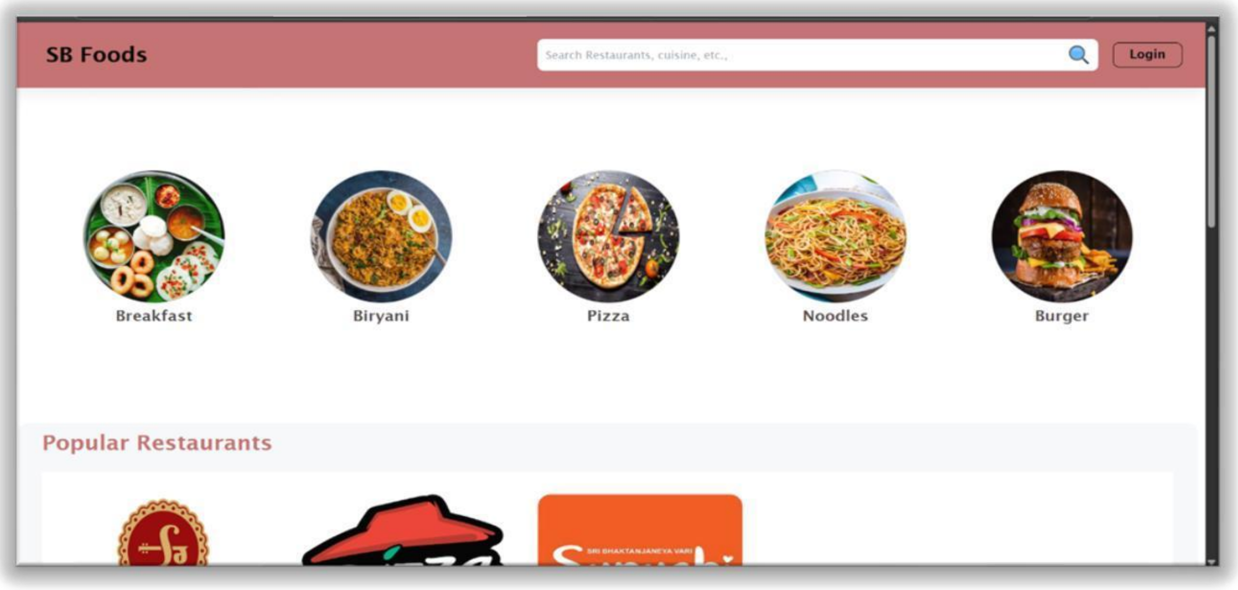
- POST /api/login – Login and receive JWT
- GET /api/restaurants – List available restaurants
- GET /api/menu/:restaurantId – Get menu for a restaurant
- POST /api/orders – Place an order
- GET /api/orders/:id – Get user's order history

8.Authentication

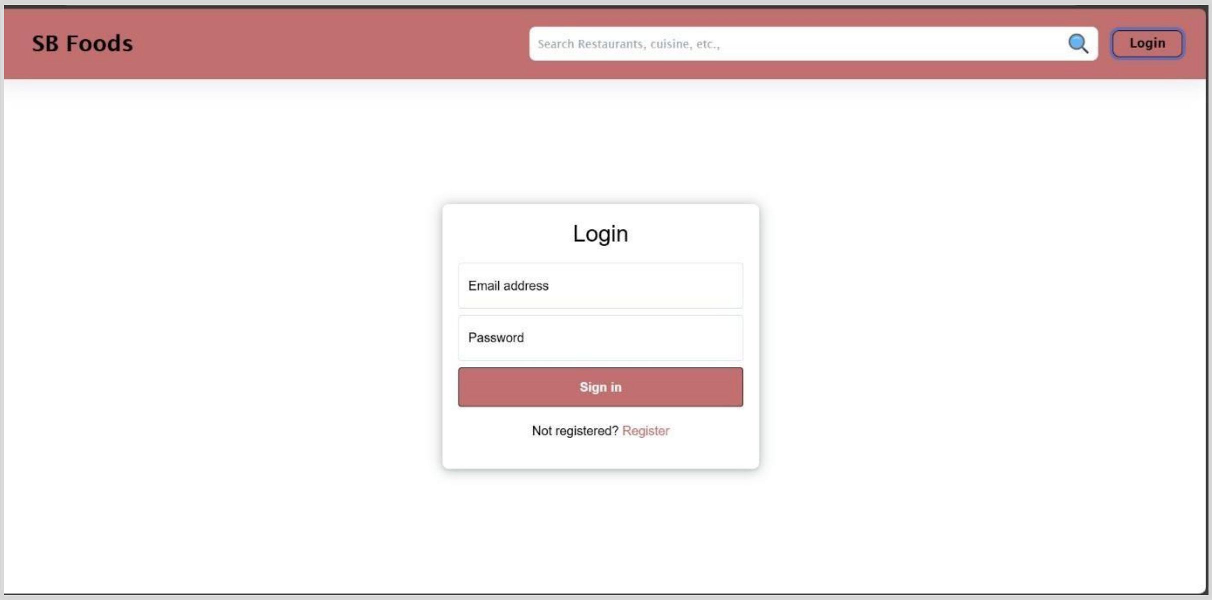
8. 1.JWT (JSON Web Token)-based authentication.

- Tokens generated at login
- Middleware verifies token for protected routes
- Role-based access for customer, restaurant, and admin

9.User Interface:



9. 1.Home Page 9.2.Login Page



SB Foods

Search Restaurants, cuisine, etc.,

Login

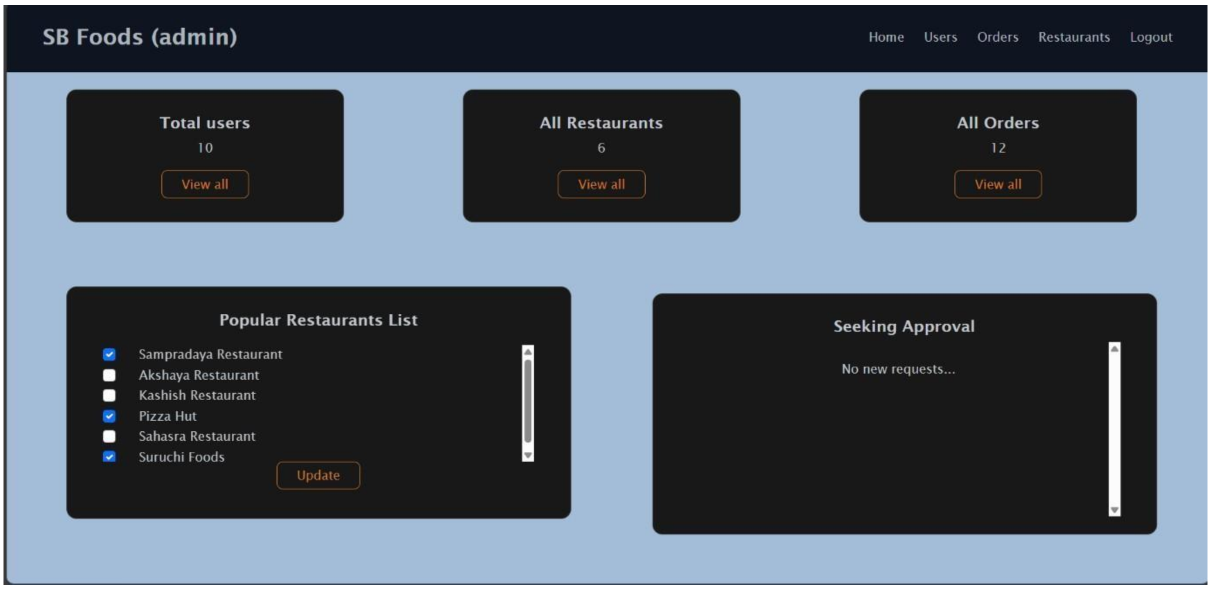
Login

Email address

Password

Sign in

Not registered? Register



SB Foods (admin)

Home Users Orders Restaurants Logout

Total users	All Restaurants	All Orders
10	6	12
View all	View all	View all

Popular Restaurants List

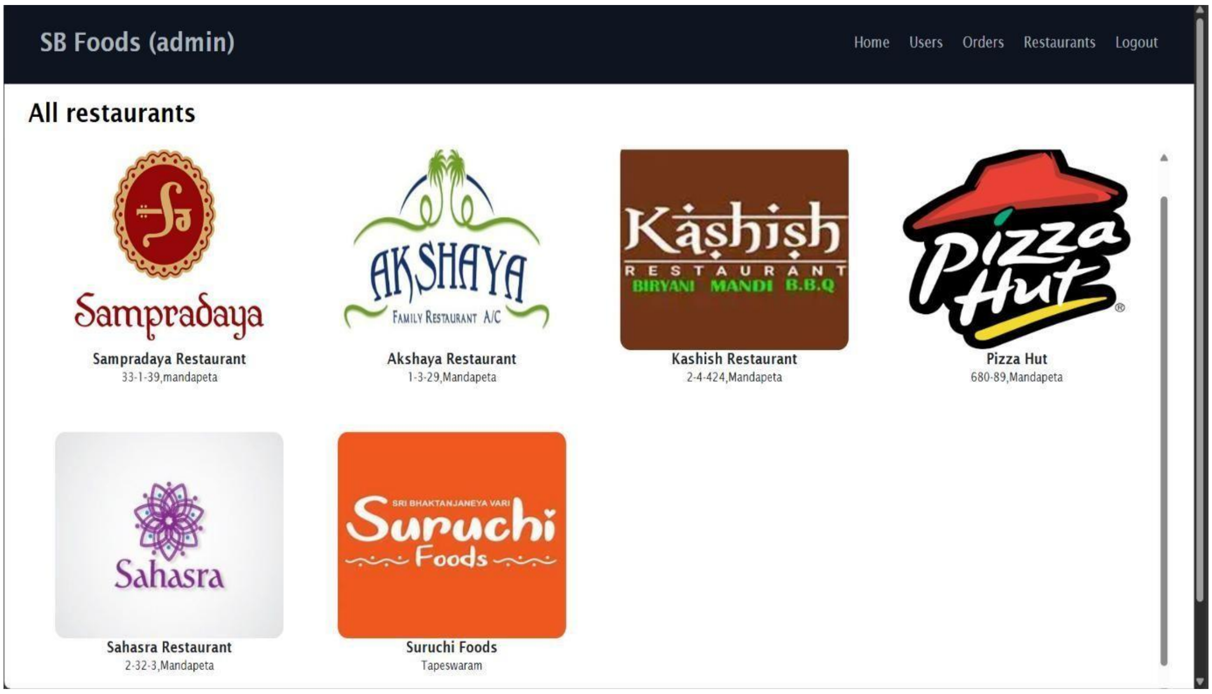
- ☑ Sampradaya Restaurant
- ☐ Akshaya Restaurant
- ☐ Kashish Restaurant
- ☑ Pizza Hut
- ☐ Sahasra Restaurant
- ☑ Suruchi Foods

Update

Seeking Approval

No new requests...

9.3.Admin Page 9.4.All Restaurants Page



All restaurants

Sampradaya Restaurant
33-1-39,mandapeta

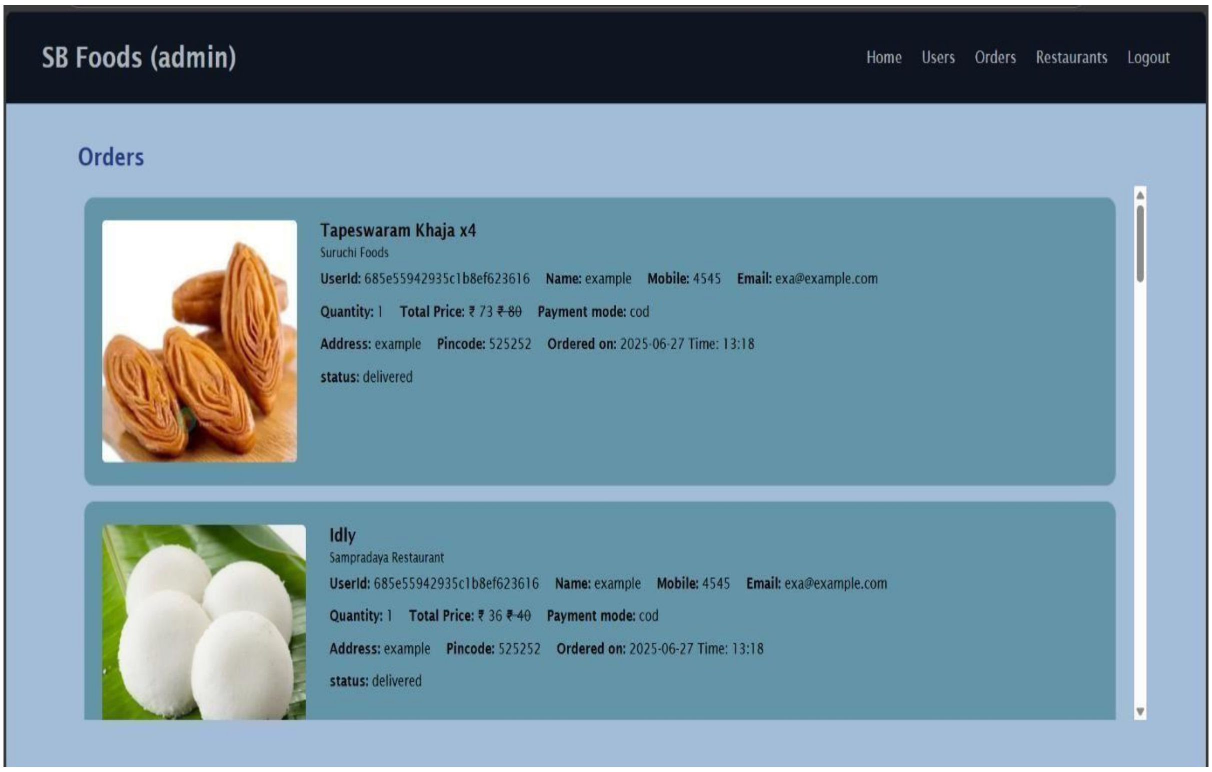
Akshaya Restaurant
1-3-29,Mandapeta

Kashish Restaurant
2-4-424,Mandapeta

Pizza Hut
680-89,Mandapeta

Sahasra Restaurant
2-32-3,Mandapeta

Suruchi Foods
Tapeswaram

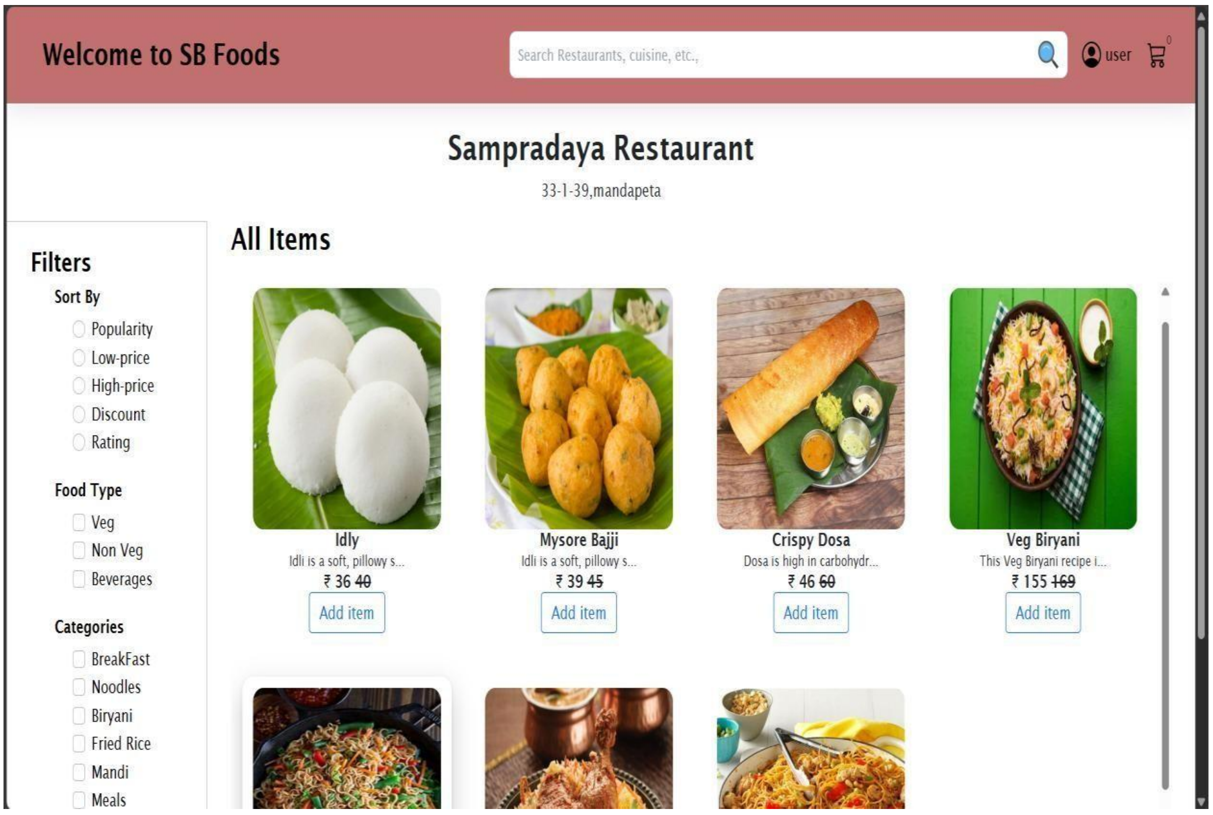


Orders

Tapeswaram Khaja x4
Suruchi Foods
UserId: 685e55942935c1b8ef623616 **Name:** example **Mobile:** 4545 **Email:** exa@example.com
Quantity: 1 **Total Price:** ₹ 73 ~~₹ 80~~ **Payment mode:** cod
Address: example **Pincode:** 525252 **Ordered on:** 2025-06-27 Time: 13:18
status: delivered

Idly
Sampradaya Restaurant
UserId: 685e55942935c1b8ef623616 **Name:** example **Mobile:** 4545 **Email:** exa@example.com
Quantity: 1 **Total Price:** ₹ 36 ~~₹ 40~~ **Payment mode:** cod
Address: example **Pincode:** 525252 **Ordered on:** 2025-06-27 Time: 13:18
status: delivered

9.5.All Orders Page 9.6.Individual Restaurant Page


Welcome to SB Foods

Search Restaurants, cuisine, etc.,

user

Sampradaya Restaurant

33-1-39,mandapeta

Filters

Sort By
- Popularity
- Low-price
- High-price
- Discount
- Rating

Food Type
- Veg
- Non Veg
- Beverages

Categories
- BreakFast
- Noodles
- Biryani
- Fried Rice
- Mandi
- Meals

All Items

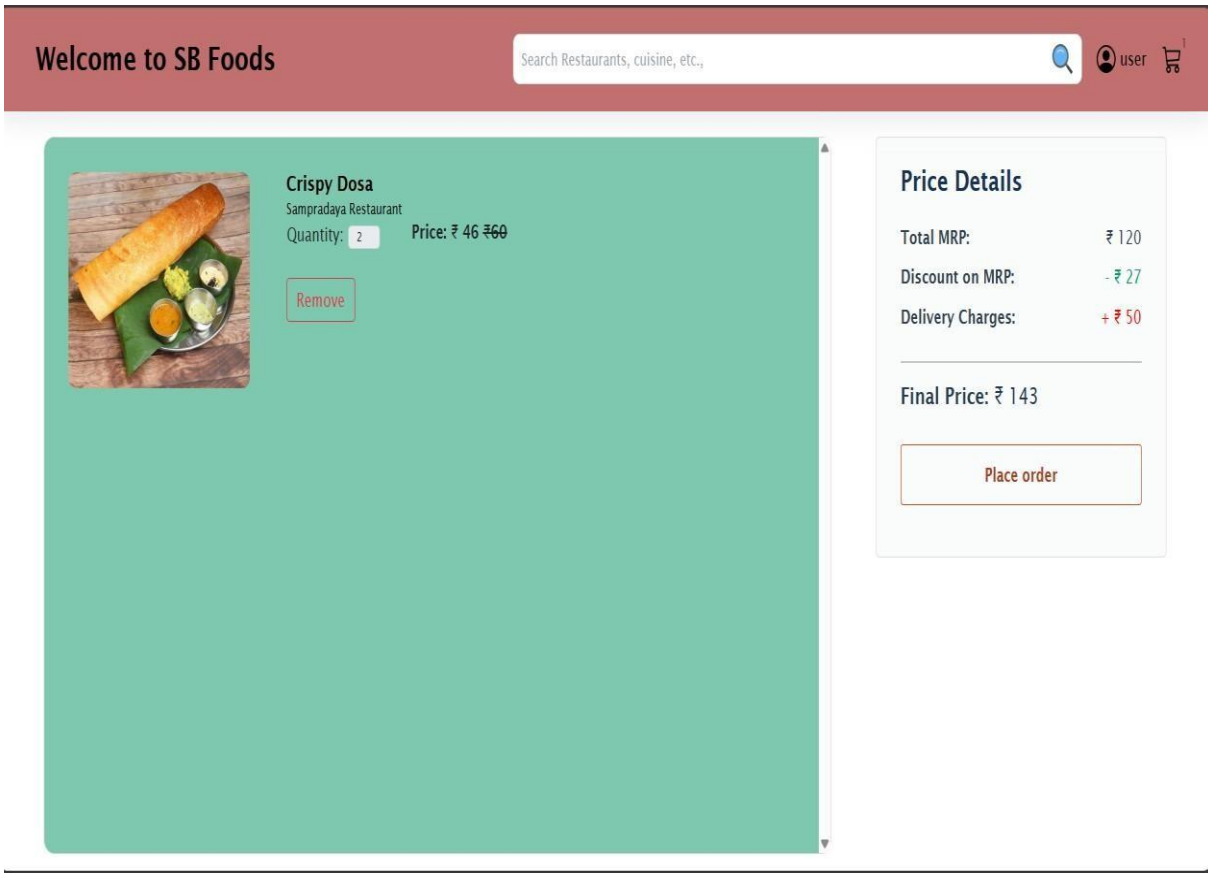
Idly
Idli is a soft, pillowy s...
₹ 36 ~~40~~
Add item

Mysore Bajji
Idli is a soft, pillowy s...
₹ 39 ~~45~~
Add item

Crispy Dosa
Dosa is high in carbohydr...
₹ 46 ~~60~~
Add item

Veg Biryani
This Veg Biryani recipe i...
₹ 155 ~~169~~
Add item

9.7.Cart Page


Welcome to SB Foods

Search Restaurants, cuisine, etc.,

user

Crispy Dosa
Sampradaya Restaurant
Quantity: 2 Price: ₹ 46 ~~₹60~~

Remove

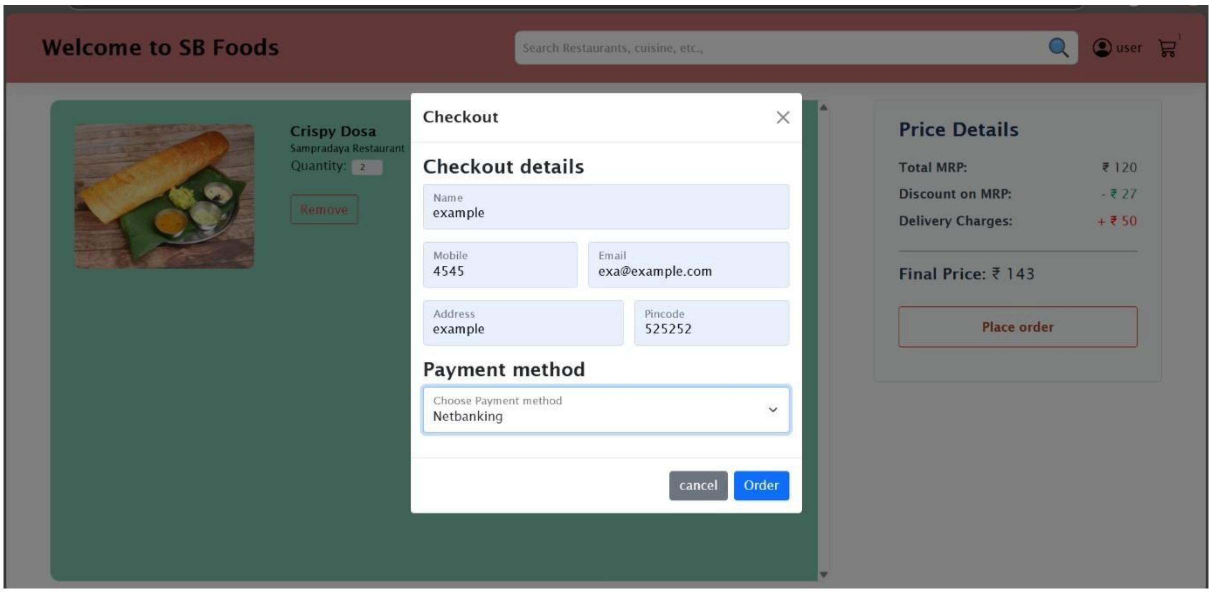
Price Details

Total MRP:	₹ 120
Discount on MRP:	- ₹ 27
Delivery Charges:	+ ₹ 50

Final Price: ₹ 143

Place order

9.8.Checkout Page


Welcome to SB Foods

Crispy Dosa
Sampradaya Restaurant
Quantity: 2

Checkout

Checkout details

Name
example

Mobile
4545

Email
exa@example.com

Address
example

Pincode
525252

Payment method

Choose Payment method
Netbanking

Price Details

Total MRP: ₹ 120
Discount on MRP: - ₹ 27
Delivery Charges: + ₹ 50

Final Price: ₹ 143

10.Testing:

Manual testing using Postman and browser. Tested routes, forms, and user flows.

11.Demo Link

View project demo: https://drive.google.com/file/d/1d-iKOXpUW5f91MFefyPuy56ezWWo1jHd/view?usp=sharing

12.Known Issues:

- Order cancelation logic incomplete
- Limited mobile responsiveness

Restaurant ratings and reviews not yet implemented

13.Future Enhancements:

- Online payment integration (e.g., NetBanking)
- Feedback and rating system
- Order delivery tracking
- Push/email/SMS notifications

14.Sample Code: Order Placement:

```
router.post('/order', async (req, res) => {  const
{ userId,  restaurantId,  items,  totalPrice } =
req.body;
try {
const order = new Order
        ({ userId, restaurantId, items, totalPrice, status: 'Placed' });
await order.save();  res.status(201).send('Order placed
successfully');
}  catch (err)
{
res.status(5
00).send('E
rror placing
order');
  }
});
```

15.Sample Code: MongoDB Schema

```
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: Array, default: []}
})

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},     restaurantId:
{type: String},     restaurantName:
{type: String},     foodItemId:
{type: String},     foodItemName:
{type: String},     foodItemImg:
{type: String},     quantity: {type:
Number},     price: {type:
Number},     discount: {type:
Number},     paymentMethod:
{type: String},     orderDate: {type:
String},
    orderStatus: {type: String, default: 'order placed'}
})

const cartSchema = new mongoose.Schema({
    userId: {type: String},
restaurantId: {type: String},
restaurantName: {type: String},
foodItemId: {type: String},
foodItemName: {type: String},
foodItemImg: {type: String},
quantity: {type: Number},     price:
{type: Number},
    discount: {type: Number}
})
```