

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

## 1.Introduction:

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

Team ID: LTVIP2025TM1D56988

Team Members:

- 1.Ramakrishna Tejasti (Team Leader)
- 2.k. Guna shekar(Team member)
- 3.K. Indrani(Team member)
- 4.K. chandusri(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.Prerequisites

Make sure the following are installed:

Node.js v14 or higher

MongoDB v4.4 or higher

npm (included with Node.js)

Git & VS Code (recommended)

### 4.Application Flow

1. User visits the landing page
2. Registers or logs in
3. Menu items are dynamically rendered from JSON
4. Authenticated users can view their profile
5. Auth token stored in browser (localStorage)
6. Logout clears token and redirects

### 5.Project Structure

```
orderonthego/  
├── client/           # React frontend  
│   ├── public/  
│   └── src/  
│       ├── components/ # Navbar, Login, MenuItem, etc.  
│       ├── data/       # JSON data (menuItems.js)  
│       └── App.js  
└── server/           # Node/Express backend  
    ├── models/       # Mongoose schemas  
    ├── routes/       # API routes  
    ├── controllers/  # Business logic  
    ├── middleware/   # Auth, validation  
    └── server.js      # Main backend entry
```

### 6. Project Flow

Frontend handles routing, API calls, and UI state  
Backend serves REST API with JWT-secured routes  
MongoDB stores users and (optionally) orders  
JSON data in frontend handles menu display

## **7.Project Setup & Configuration**

### **Backend Setup**

```
cd server
npm install
npm start
.env file in /server:
PORT=5000
JWT_SECRET=your-secret-key-here
MONGODB_URI=mongodb://localhost:27017/
```

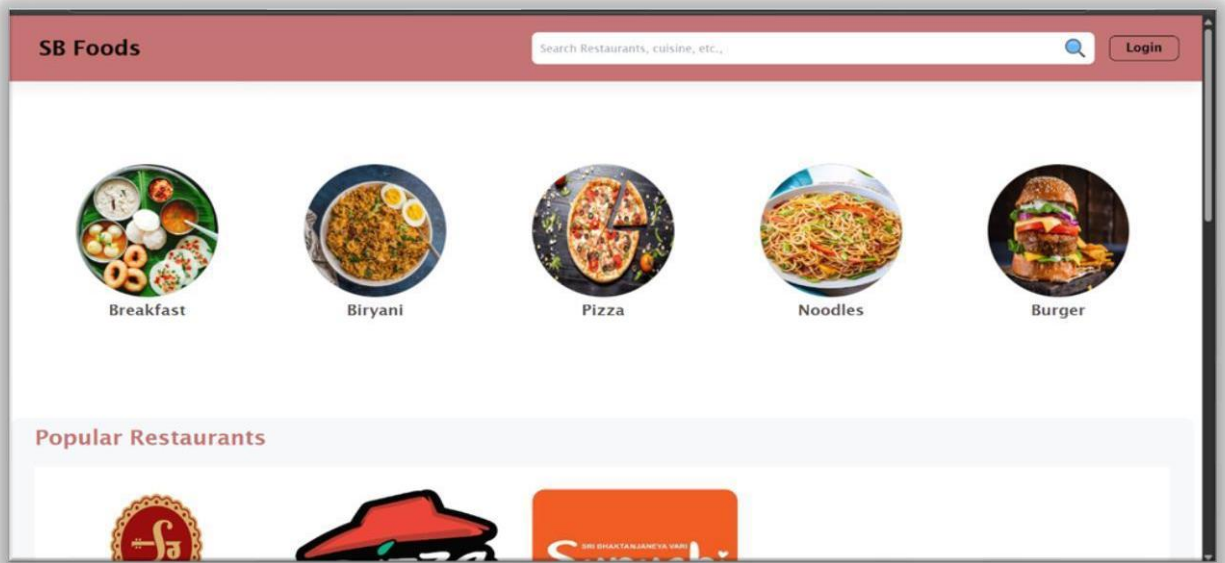
### **Frontend Setup**

```
cd client
npm install
npm start
Runs on: http://localhost:3000
```

## **8.Database Development (Mongoose)**

```
models/User.js
const mongoose = require('mongoose');
const userSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true, minlength: 3 },
  email: { type: String, required: true, unique: true, lowercase: true },
  password: { type: String, required: true, minlength: 6 },
  createdAt: { type: Date, default: Date.now }
});
module.exports = mongoose.model('User', userSchema);
```

## **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)

HomeUsersOrdersRestaurantsLogout

Total users

10

View all

All Restaurants

6

View all

All Orders

12

View all

Popular Restaurants List

☒ Sampradaya Restaurant

☐ Akshaya Restaurant

☐ Kashish Restaurant

☒ Pizza Hut

☐ Sahasra Restaurant

☒ Suruchi Foods

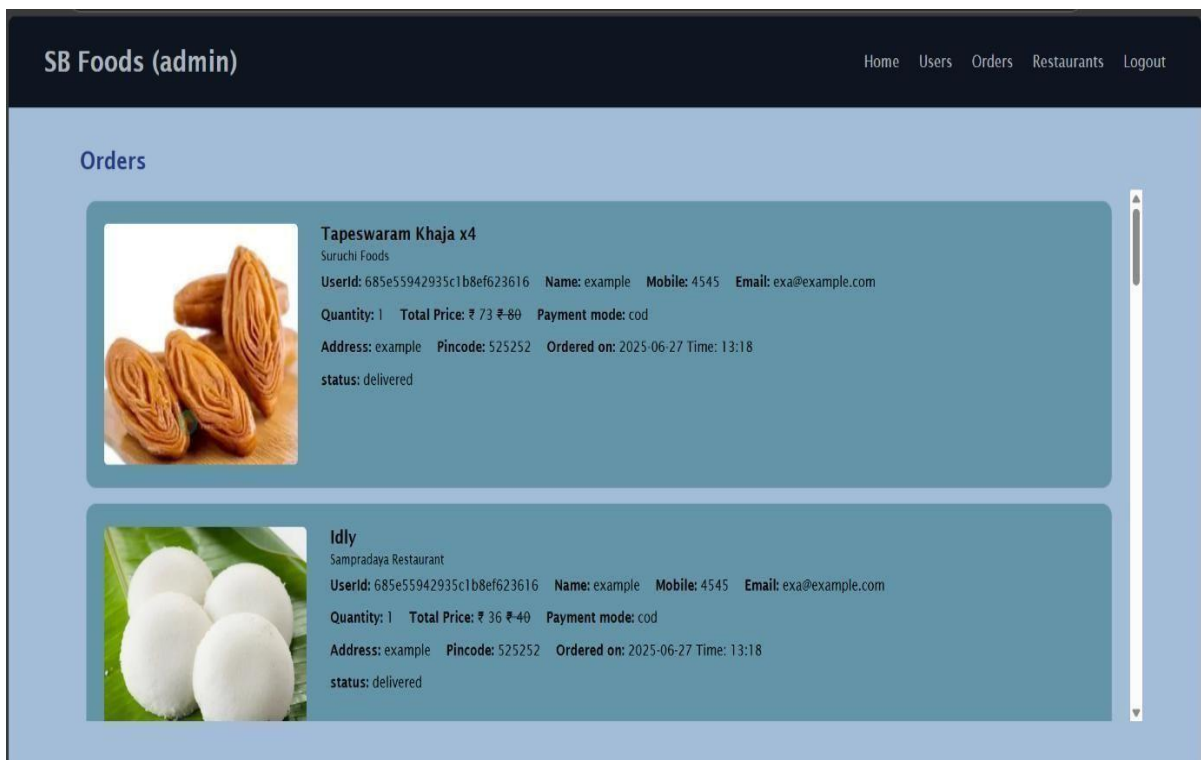
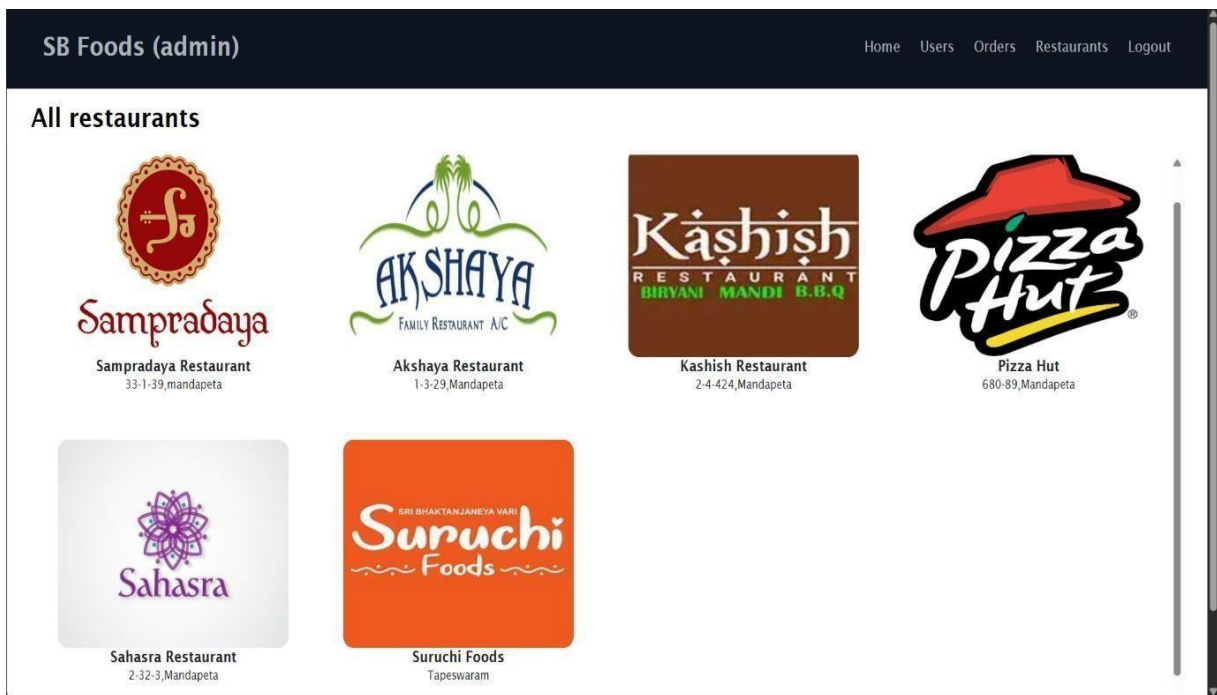
Update

Seeking Approval

No new requests...

9.3.Admin Page

9.4.All Restaurants Page



**9.5.All Orders Page**

**9.6.Individual Restaurant Page**

Welcome to SB Foods

Search Restaurants, cuisine, etc.,

user

0

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...

₹ 155 469

Add item


## 9.7.Cart Page

Welcome to SB Foods

Search Restaurants, cuisine, etc.,

user

1



Crispy Dosa

Sampradaya Restaurant

Quantity: 2 Price: ₹ 46 ₹60

Remove

Price Details

Total MRP: ₹ 120

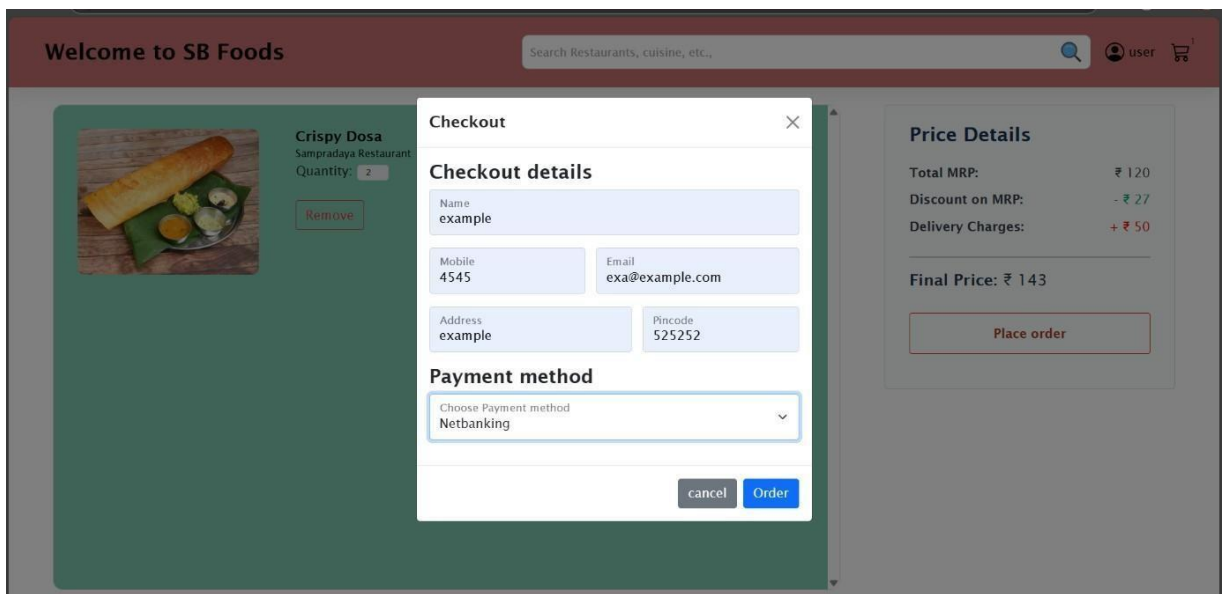
Discount on MRP: - ₹ 27

Delivery Charges: + ₹ 50

Final Price: ₹ 143

Place order

## 9.8.Checkout Page



## 10. Testing:

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

## 11. Known Issues:

- Order cancelation logic incomplete
- Limited mobile responsiveness

Restaurant ratings and reviews not yet implemented



## 12.Future Enhancements:

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

## 13.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');
}
});
```

## 14.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},    foodItemImage:
{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},
  foodItemImage: {type: String},
  quantity: {type: Number},    price:
{type: Number},
  discount: {type: Number}
})
```