

# Full Stack Development with MERN

## 1.Introduction

- **Project Title:** OrderOnTheGo: Your On-Demand Food Ordering Solution
- **Team Members:** Vasamsetti Sri Mahalakshmi [Team Leader]  
Pentapati Vishnu Ratha [Team member]  
Vanum Karthik [Team member]

## 2. Project Overview

- **Purpose:**  
SB Foods is a MERN-stack based food ordering web application designed to offer users a seamless experience of browsing restaurant menus, managing carts, placing orders, and tracking order status—all in real time. It also provides restaurant and admin dashboards for business-side operations and management.
- **Key Features:**
  - Users, restaurants, and admins can securely register and log in with dedicated access. Role-based authorization ensures protected and personalized experiences.
  - Users can browse a wide range of food items with detailed images, descriptions, pricing, and categories. The catalog is visually rich and easy to navigate.
  - Real-time search and smart filters allow users to find meals based on cuisine, food type, restaurant, or keyword. This enhances discovery and convenience.
  - Restaurants can manage their menu by adding, editing, or deleting items. They also track order history and control pricing and availability from their dashboard.
  - Users can add products to their cart with dynamic quantity control. The cart persists across sessions and displays a clear pricing summary before checkout.
  - The checkout process is simple and secure, allowing users to enter delivery details and select payment methods. A confirmation is shown instantly after placing an order.
  - Users receive real-time order confirmation and can track the status from their profile. Order history, payment method, and delivery time are all accessible.
  - Admins can monitor the platform from a centralized dashboard, managing users, orders, restaurants, and products. They can also approve and promote restaurants.
  - The user interface is responsive across all screen sizes with smooth navigation. The design ensures a consistent and engaging experience across devices.
  - Late-night food options are available through special filters for restaurants open during odd hours. This helps users satisfy cravings even past midnight.

### 3. System Architecture

- **Client Layer (Frontend)**

**Technology:** React.js

**Users:** Customers, Restaurants, Admin

**Role:** Provides an interactive and responsive user interface for browsing restaurants, registering, logging in, adding to cart, placing orders, and managing dashboards. Manages user sessions and role-based navigation using Context API and local storage. Interacts with backend APIs via Axios for all data operations. Responsive UI components styled with Tailwind CSS for cross-device compatibility.

- **Server Layer (Backend API)**

**Technology:** Node.js + Express.js

**Responsibilities:** Handles all RESTful API endpoints for user authentication, product management, order processing, cart handling, and admin controls.

Implements JWT-based authentication for secure route access.

Middleware functions are used for validation, authorization, and error handling.

API routing is modularized and separated by user role (customer, restaurant, admin).

- **Database Layer**

**Technology:** MongoDB + Mongoose

**Functions:** Stores persistent data such as users, restaurants, products, orders, carts, and admin metadata.

Mongoose is used to define and enforce schemas with relationships (e.g., userId in orders, restaurantId in products).

Efficiently handles CRUD operations, data validation, and indexing for fast query performance.

Ensures data integrity and scalability for a growing food-ordering ecosystem.

### 4. Setup Instructions

- **Install Prerequisites:** Node.js, MongoDB or MongoDB Atlas, Git, Visual Studio Code
- **Clone Repository:** git clone <https://github.com/your-username/Food-Ordering-App.git>
- **Start MongoDB:** Run mongod (or connect Atlas DB)
- **Setup Backend:** cd server npm install npm start
- **Setup Frontend:** cd client npm install npm start
- **Access the App:** <http://localhost:3000>
- **Backend API:** <http://localhost:3001>
- **Environment Variables:** .env file in server directory

## 5. Folder Structure

- **Client (React Frontend)**

The client directory contains the complete frontend application built using **React.js**. It is structured to ensure modularity, reusability, and scalability

**public/** – Static assets such as index.html and image files used across the UI.

**src/** – Main source code folder containing application logic:

**components/** – Shared UI components such as Navbar, Footer, Login, Register, etc.

**pages/** – Page-level views grouped by user type (admin, customer, restaurant).

**context/** – React Context API implementation for managing global state like auth, user info, and cart.

**styles/** – CSS and Tailwind-based styling files for pages and components.

**App.js** – Root component managing app-wide routes and structure.

**package.json** – Lists all frontend dependencies like React, Axios, TailwindCSS, React Router, etc.

- **Server (Node.js + Express Backend)**

The server directory contains the backend logic written using **Node.js**, **Express.js**, and **Mongoose** (for MongoDB).

**Schema.js** – All Mongoose models for collections like User, Admin, Restaurant, Product, Cart, and Orders are defined in this file.

**routes/** – API endpoint definitions for user login/register, orders, cart operations, restaurant and admin-specific actions. (optional split for larger apps)

**controllers/** – (Optional if separated) Contains the logic for handling requests (e.g., placing orders, managing products).

**index.js** – Main server entry point which sets up Express app, MongoDB connection, and routes for all entities.

**.env** – Stores sensitive environment variables such as PORT, JWT\_SECRET, and MongoDB URI.

**package.json** – Lists backend dependencies like Express, Mongoose, dotenv, bcrypt, etc.

## 6. Running the Application

To run the project locally, follow these steps for both the frontend and backend servers:

- **Frontend (React):**

- Open a terminal.
- Navigate to the client folder: `cd client`
- Start the React development server: `npm start`
- The frontend will run at: <http://localhost:3000>

- **Backend (Node.js + Express):**

- Open another terminal window.

- Navigate to the server folder: `cd server`
- Start the backend server: `npm start`
- The backend will run at: <http://localhost:3001>

## 7. API Documentation

- **User APIs**

POST /register

Registers a new user (customer, restaurant, or admin).

Request Body:

```
{
  "name": "Amit Kumar",
  "email": "amit@example.com",
  "password": "123456",
  "userType": "customer"
}
```

POST /login

Authenticates a user and returns a JWT token.

Request Body:

```
{
  "email": "amit@example.com",
  "password": "123456"
}
```

- **Product APIs**

POST /add-product

Allows restaurants or admins to add new food items to the platform.

Request Body:

```
{
  "name": "Chicken Biryani",
  "price": 250,
  "category": "Biryani",
  "restaurantId": "res123",
  "description": "Spicy and aromatic biryani with tender chicken pieces"
}
```

GET /products

Fetches all available food products listed on the platform.

GET /products/:restaurantId

Returns all food items for a specific restaurant.

- **Cart APIs**

POST /add-to-cart

Adds a product to the user's cart.

Request Body:

```
{
  "userId": "cus456",
  "productId": "prod789",
  "quantity": 2
}
```

GET /cart/:userId

Returns all cart items for the specified user.

DELETE /cart/remove/:itemId

Removes a specific item from the user's cart.

- **Order APIs**

POST /place-order

Places an order with all cart items and delivery details.

Request Body:

```
{
  "userId": "cus456",
  "items": [
    { "productId": "prod789", "quantity": 2 }
  ],
  "address": "123, MG Road, Bengaluru",
  "paymentMethod": "Online"
}
```

GET /orders/:userId

Returns all orders placed by a specific user.

GET /admin/all-orders

Admin-only route to fetch all orders placed across the platform.

- **Restaurant APIs**

POST /restaurant/register

Registers a new restaurant for approval.

Request Body:

```
{
  "name": "The Spice Hub",
  "owner": "Rahul Joshi",
  "email": "spicehub@example.com",
  "location": "Hyderabad",
  "password": "res1234"
}
```

GET /restaurant/orders/:restaurantId

Returns all orders placed for a specific restaurant's food items.

GET /restaurant/products/:restaurantId

Returns all menu items added by a specific restaurant.

- **Admin APIs**

GET /admin/users

Returns a list of all users registered on the platform.

GET /admin/restaurants

Returns a list of all restaurant accounts.

POST /admin/approve-restaurant/:restaurantId

Approves a pending restaurant so it can list products.

POST /admin/mark-popular

Marks selected restaurants as "Popular".

Request Body:

```
{
  "restaurantIds": ["res123", "res456"]
}
```

}

## 8. Authentication

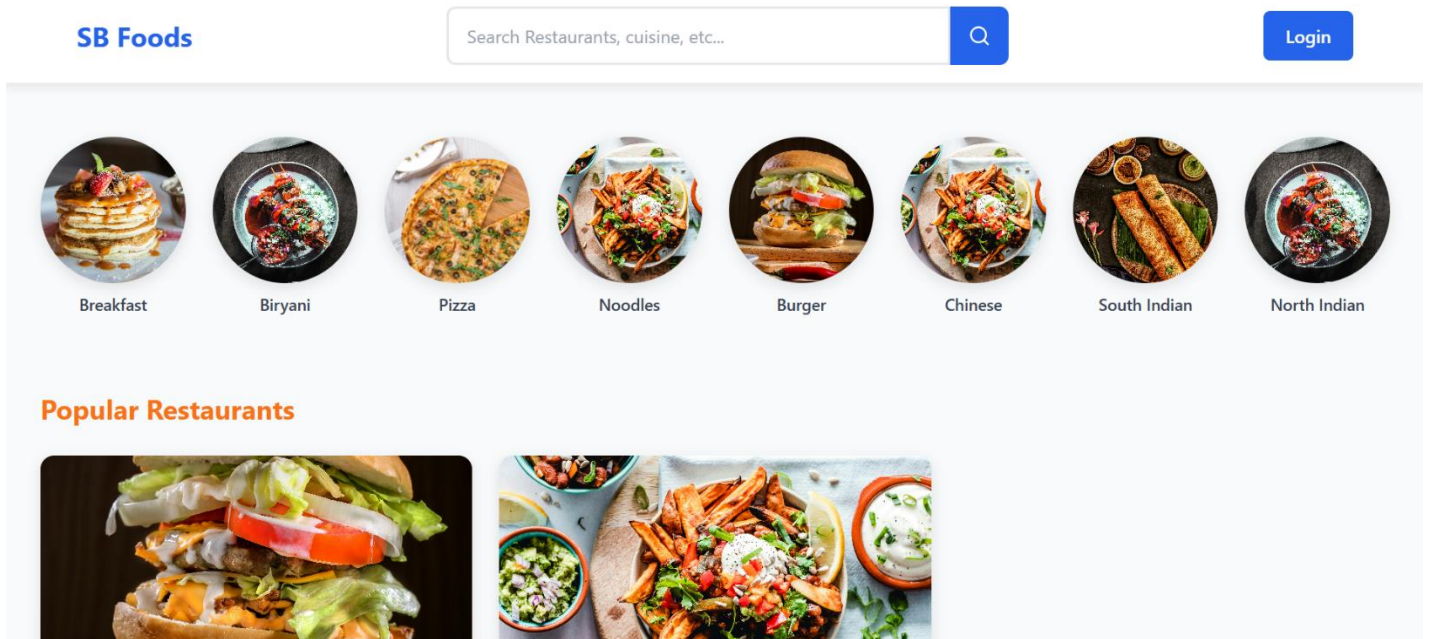
- **User Registration (POST /register)**  
New users (customers, restaurants, or admins) register by submitting their name, email, password, and user type.  
User details are securely stored in the MongoDB database with passwords hashed using bcrypt for encryption.
- **User Login (POST /login)**  
When a user logs in, their email and password are validated against stored records. If successful, the backend responds with a JWT token, user ID, and role (either "customer", "restaurant", or "admin").  
This data is stored on the frontend using localStorage or Context API for session management.
- **Protected Routes (JWT Middleware)**  
Backend routes like placing orders, adding products, or accessing admin dashboards are protected using JWT middleware.  
The token must be included in the headers (Authorization: Bearer token) for access to secured endpoints.
- **Logout**  
When a user logs out, the frontend clears the saved JWT token and user information from localStorage and Context, effectively ending the session.

## 9. Testing

**Testing Strategy:** The project follows a manual and functional testing strategy to validate complete user journeys and features across all roles (Customer, Restaurant, and Admin).

- **Unit Testing:** Core backend functions like authentication, cart logic, and price calculations were tested manually during development to ensure correctness.
- **Integration Testing:** Backend API integration with frontend was tested for seamless data flow, including product listing, cart operations, and order placement. Admin dashboard functionality was validated for data control and accuracy.
- **End-to-End Testing:** Complete ordering workflows were tested, including:  
User registration → Login → Browse restaurants → Add to cart → Place order → View order history.  
Restaurant flows like Add product → View orders → Update menu were also tested from start to finish.
- **UI/UX Testing:** Manual testing was conducted on multiple devices (mobile and desktop) to ensure responsive design, proper layout rendering, intuitive navigation, and usability across pages.

## 10. Screenshots / Demo



## SB Foods

### Register

Register as:

User

Full Name:

Email:

Password:

Address:

# SB Foods

## Login

Login as:

User

Email:

Password:

Login

Don't have an account? Register

SB Foods

Search Restaurants, cuisine, etc...



John Doe



Logout



Breakfast



Biryani



Pizza



Noodles



Burger



Chinese



South Indian



North Indian

### Popular Restaurants





## All Restaurants



## McDonald's

Manikonda, Hyderabad

Fast Food

★ 4.5



## Andhra Spice

Gachibowli, Hyderabad

South Indian

★ 4.3



## Paradise Grand

Banjara Hills, Hyderabad

Biryani

★ 4.7



## Filters

## Sort By

- ☒ Popularity
- ☐ Low-price
- ☐ High-price
- ☐ Discount
- ☐ Rating

## Food Type

- ☐ Veg
- ☐ Non Veg
- ☐ Beverages

## Categories

## All Items



## Mc Maharaj

Big size burger with chicken tikka and special sauce

₹ 175

+ Add Item



## French Fries

Long French fries made from potato

₹ 134

+ Add Item



My Profile

Personal Information

Name: John Doe

Email: john@user.com

Phone: +91 9876543210

Address: Manikonda, Hyderabad

Order History

Order #1751023482475

Pending

Total: ₹ 1081.99

Items: 4

Date: 6/27/2025

Order #1751029120286

Pending

Total: ₹ 201

Items: 1

Date: 6/27/2025

Order #1751029795102

Pending

Total: ₹ 175

Items: 1

Date: 6/27/2025

Your Cart



Mc Maharaj

Big size burger with chicken tikka and special sauce

₹ 175

- 1 +



French Fries

Long French fries made from potato

₹ 134

- 1 +



Order Summary

Subtotal: ₹ 309

Delivery Fee: ₹ 50

Total: ₹ 359

Place Order

Restaurant Dashboard

HomeOrdersMenuNew ItemLogout

Restaurant Dashboard

Total Orders

3

Total Revenue

₹ 1457.99

Menu Items

4

Recent Orders

Order #1751023482475

Total: ₹ 1081.99

Items: 4

Pending

Date: 6/27/2025

Restaurant Dashboard

HomeOrdersMenuNew ItemLogout

Orders

Order ID	Customer	Items	Total	Status	Date	Actions
#1751023482475	3	4 items	₹ 1081.99	Pending	6/27/2025	<div>ViewAccept</div>
#1751029120286	3	1 items	₹ 201	Pending	6/27/2025	<div>ViewAccept</div>
#1751029795102	3	1 items	₹ 175	Pending	6/27/2025	<div>ViewAccept</div>

## Menu Management



**Mc Maharaj**  
Big size burger with chicken tikka and special sauce  
₹ 175



**French Fries**  
Long French fries made from potato  
₹ 134



**Cold Coffee**  
Tender coffee made from milk  
₹ 201



## Add New Menu Item

Item Name:

Description:

Price (₹):

Image URL:

SB Foods (admin)

Home

Users

Orders

Restaurants

Logout

Total users

3

View all

All Restaurants

4

View all

All Orders

3

View all

Popular Restaurants (promotions)

☒ McDonald's

☒ Andhra Spice

☐ Paradise Grand

Approvals

No new requests...

SB Foods (admin)

Home

Users

Orders

Restaurants

Logout

All Users

Name	Email	Phone	Orders	Actions
John Doe	john@user.com	+91 9876543210	5	<div>ViewDelete</div>
Jane Smith	jane@user.com	+91 9876543211	3	<div>ViewDelete</div>
Mike Johnson	mike@user.com	+91 9876543212	8	<div>ViewDelete</div>

All Orders

Order ID	Customer	Restaurant	Total	Status	Date	Actions
#1751023482475	3	1	₹ 1081.99	Pending	6/27/2025	<button>View</button>
#1751029120286	3	1	₹ 201	Pending	6/27/2025	<button>View</button>
#1751029795102	3	1	₹ 175	Pending	6/27/2025	<button>View</button>

All Restaurants

Name	Address	Cuisine	Rating	Popular	Status	Actions
McDonald's	Manikonda, Hyderabad	Fast Food	★ 4.5	Popular	Approved	<button>View</button> <button>Edit</button>
Andhra Spice	Gachibowli, Hyderabad	South Indian	★ 4.3	Popular	Approved	<button>View</button> <button>Edit</button>
Paradise Grand	Banjara Hills, Hyderabad	Biryani	★ 4.7	Regular	Approved	<button>View</button> <button>Edit</button>
Minerva Grand	Kondapur, Hyderabad	Multi-cuisine	★ 4.2	Regular	Approved	<button>View</button> <button>Edit</button>

## 11. Known Issues

- **Restaurant Approval Delay**

**Description:** When a restaurant registers, admin approval is required before they can list products. In some cases, there's a delay in the approval reflecting on the frontend due to caching or refresh lag, causing confusion for new restaurant users.

- **Cart Not Persisting on Page Reload**

**Description:** The items added to the cart are stored only in local context and not persisted in localStorage. This causes the cart to reset if the user refreshes the page or navigates away, potentially leading to frustration or rework.

- **No Image Upload Validation**

**Description:** When restaurants add new food items, there's no strong validation for image uploads. Users can submit invalid or broken URLs, resulting in blank images in the menu UI.

- **Inconsistent Order Status Sync**

**Description:** After an order is placed, the real-time update of order status (e.g., "Preparing", "Delivered") sometimes lags between the admin/restaurant side and the customer view, especially under slow network conditions.

- **Basic Error Feedback on Forms**

**Description:** Some input forms like registration and add product show only generic error messages (e.g., "Something went wrong"), making it difficult for users to understand the exact issue (e.g., missing field, duplicate email).

## 12. Future Enhancements

- **Image Upload for Products**

Allow restaurants and admins to upload food item images directly via file input instead of providing external URLs. This will enhance data consistency and reduce broken or invalid image issues.

- **Advanced Filters for Menu & Restaurants**

Implement advanced filters on restaurant and product listing pages (e.g., price range, food type, cuisine, popularity) to help users quickly narrow down options based on their preferences.

- **Frontend Role-Based Route Guarding**

Add stricter role-based access control in the frontend so that only customers, restaurants, or admins can access pages meant for their roles, improving security and navigation clarity.

- **In-App Notifications**

Introduce real-time notifications for users regarding order confirmations, status

changes, restaurant approvals, and admin updates to enhance engagement and responsiveness.

- **Order Progress Tracker**  
Allow users to track their order status with a visual timeline, showing phases like "Order Placed", "Preparing", "Out for Delivery", and "Delivered" for better transparency.
- **Restaurant Review & Rating System**  
Enable customers to rate restaurants and write reviews after receiving their orders. This would help build trust and guide new users in their decision-making.
- **Admin Analytics Dashboard**  
Expand the admin panel to include visual charts showing metrics like total orders, popular food categories, top restaurants, and user engagement over time.
- **Mobile Application Support**  
Build dedicated mobile apps for Android and iOS to allow customers and restaurant partners to manage orders, menus, and profiles more conveniently.
- **Popular Items Carousel**  
Add a carousel for trending or most-ordered items on the landing page to increase visibility of high-demand products and improve upselling.
- **Payment Gateway Integration**  
Integrate secure online payment options (e.g., Razorpay, Stripe) for smooth and safe order transactions, replacing or supplementing the current cash-on-delivery model.