OrderOnTheGo – Online Food Ordering System

1. Introduction
   - **Project Title**: **OrderOnTheGo – Online Food Ordering System**
   - **Team Members:**

Kalapureddi Bala veeranjaneyulu (Role : Frontend Development)

Raghu Rama Raju Indukuri (Role : Backend Development)

Gottapu Bharath kumar(Role : Database Developmet)

Ramesh Balabhadhra(Role : Database Development)

2. Project Overview

Purpose of the Project

The purpose of this project is to design and develop a full-stack online food ordering web application using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The system enables customers to browse restaurants, explore food menus, add items to cart, and place orders online in a seamless and efficient manner.

The application also provides separate dashboards for restaurants and administrators to manage products, orders, and users.

Objectives

- To build a dynamic web application using MERN stack.
- To implement secure user authentication.
- To manage restaurant menus and food categories.
- To allow customers to place and track orders.
- To maintain structured database collections using MongoDB.
- To implement RESTful APIs for communication between frontend and backend.

Key Features

Customer Features

1. User Registration & Login

User Registration:
This feature allows new customers to create an account by providing basic details such as name, email, phone number, and password. The system stores this information securely in the database. Registration ensures that each user has a unique identity in the system.

User Login:

Registered users can log in using their email/username and password. The system verifies the credentials and grants secure access. Authentication ensures that only authorized users can place orders and view personal data.

2. Browse Restaurants

This feature allows customers to view a list of available restaurants registered in the system. Customers can:

- View restaurant names
- See restaurant details (location, contact, description)
- Filter or search restaurants (if implemented)

This helps users easily choose their preferred restaurant.

3. View Food Items

After selecting a restaurant, customers can view the list of food items offered by that restaurant.

Details displayed may include:

- Food name
- Description
- Price
- Food category (Veg/Non-Veg, etc.)
- Availability status

This feature helps customers review menu items before ordering.

4. Add to Cart

Customers can select desired food items and add them to their cart.

The cart:

- Stores selected items temporarily
- Displays item quantity
- Calculates total price automatically
- Allows users to remove or update item quantity

This feature helps customers review and modify their order before confirming.

5. Place Order

After reviewing the cart, customers can confirm and place the order.

This feature:

- Saves order details in the database
- Generates an order ID

- Updates restaurant order dashboard
- Clears cart after successful order

It ensures smooth transaction processing between customer and restaurant.

6. View Order History

Customers can view their past orders.

Order history includes:

- Order ID
- Date and time
- Ordered items
- Total amount
- Order status (Pending/Completed/Cancelled)

This feature helps users track previous purchases and reorder if needed.

Restaurant Features

1. Restaurant Registration

Restaurants can register in the system by providing:

- Restaurant name
- Owner details
- Contact information
- Address
- Login credentials

After registration, the restaurant account remains pending until admin approval.

2. Admin Approval System

To maintain system quality and security:

- Admin reviews new restaurant registrations
- Admin can approve or reject requests
- Only approved restaurants can log in and add food items

This ensures authenticity and prevents fake registrations.

3. Add New Food Items

Approved restaurants can add new menu items.

They can provide:

- Food name
- Description
- Price

- Category
- Image (optional)

This allows restaurants to update their menu dynamically.

4. Update or Delete Food Items

Restaurants can:

- Modify price
- Update description
- Change availability status
- Delete food items

This helps restaurants keep their menu updated and accurate.

5. Manage Orders

Restaurants can view customer orders in real time.

They can:

- View order details
- Update order status (Accepted, Preparing, Completed, Cancelled)
- Track daily orders

This feature ensures smooth communication between customer and restaurant.

Admin Features

The Admin plays a crucial role in managing and controlling the entire food ordering system. The Admin has full access to monitor restaurants, users, and orders to ensure smooth system operations.

1. Approve / Reject Restaurants

After a restaurant registers in the system, its account remains in a **pending state** until verified by the Admin.

Explanation:

- Admin reviews restaurant registration details.
- Admin checks authenticity and required information.
- Admin can:
    - Approve the restaurant (activates account)
    - Reject the restaurant (denies access)

Purpose:

- Prevents fake or duplicate restaurant registrations.
- Maintains system quality and security.
- Ensures only verified restaurants can sell food.

2. Manage Categories

The Admin can create and manage food categories that restaurants use to classify their items.

Admin Capabilities:

- Add new categories (e.g., Starters, Main Course, Desserts, Beverages)
- Update category names
- Delete unwanted categories

Purpose:

- Keeps menu structure organized.
- Helps customers easily filter food items.
- Maintains consistency across all restaurants.

3. Promote Restaurants

The Admin can promote selected restaurants to increase their visibility on the platform.

Explanation:

- Mark restaurants as "Featured" or "Top Rated"
- Display promoted restaurants at the top of the homepage
- Highlight special offers or discounts

Purpose:

- Encourages healthy competition.
- Helps new or high-performing restaurants gain visibility.
- Improves customer engagement.

4. View All Users

The Admin has access to the list of all registered users (customers and restaurants).

Information Available:

- User name
- Email
- Contact details
- Registration date
- Account status

Purpose:

- Monitor user activity.
- Identify suspicious accounts.
- Maintain system transparency.

5. View All Orders

Admin can monitor all orders placed in the system.

Admin Can:

- View order details
- Track order status
- Check total revenue
- Analyze daily/monthly orders

Purpose:

- Monitor system performance.
- Handle disputes or issues.
- Generate business insights and reports.

3. System Architecture

The project follows a **Three-Tier Architecture**:

1. Frontend Layer (Client Side)

Technology Used:

- React.js
- Axios
- CSS
- JavaScript

Responsibilities:

- User Interface Rendering
- API Calls to Backend
- State Management
- Form Handling and Validation

React components are used to create reusable UI elements such as Navbar, Login Form, Register Form, Restaurant Dashboard, Cart, etc.

2. Backend Layer (Server Side)

Technology Used:

- Node.js
- Express.js
- bcrypt (for password hashing)
- CORS

- Body-parser

Responsibilities:

- Handle HTTP Requests
- Perform CRUD Operations
- Manage Authentication
- Connect to MongoDB
- Process Business Logic

The backend exposes RESTful APIs such as:

- POST /register
- POST /login
- GET /fetch-restaurants
- GET /fetch-items
- POST /add-new-product
- POST /add-to-cart
- POST /place-cart-order

3. Database Layer (MongoDB)

Database Name:

Foodordering

Collections Used:

users

Stores user credentials and roles.

Fields:

- username
- email
- password (hashed)
- usertype
- approval

restaurants

Stores restaurant information.

Fields:

- ownerId
- title

- address
- mainImg
- menu

fooditems

Stores food item details.

Fields:

- restaurantId
- title
- description
- itemImg
- category
- menuCategory
- price
- discount
- rating

orders

Stores order information.

Fields:

- userId
- restaurantId
- foodItemId
- quantity
- orderStatus
- paymentMethod
- address
- orderDate

cart

Stores temporary cart data before order placement.

admin

Stores:

- categories
- promotedRestaurants

4. Setup Instructions

Prerequisites

- Node.js installed
- MongoDB installed and running
- Git installed
- VS Code

Installation Steps

1.Clone repository:

git clone <repository-link>

2. Install frontend dependencies:

cd client

npm install

npm start

3.Install backend dependencies:

cd server

npm install

node index.js

4. Ensure MongoDB is running locally.

Application URL:

http://localhost:3000

Backend runs on:

http://localhost:6001

5. Folder Structure

Client Folder

- src/components → UI components
- src/pages → Page-level components
- src/context → State management
- App.js → Routing

Server Folder

- index.js → Main server file
- Schema.js → Database schema definitions

- package.json → Dependencies

6. Running the Application

Frontend:

npm start

Backend:

node index.js

MongoDB must be active.

7. API Documentation

Authentication APIs

POST /register

Registers new user.

Request Body:

```
{
  username,
  email,
  password,
  usertype
}
```

POST /login

Authenticates user credentials.

Food APIs

GET /fetch-items
GET /fetch-restaurants
GET /fetch-orders

Cart APIs

POST /add-to-cart
PUT /remove-item

Order APIs

POST /place-cart-order
PUT /update-order-status

8. Authentication & Security

The system implements robust authentication and security mechanisms to ensure data protection and secure user access.

Password Security

- All user passwords are securely hashed using the **bcrypt** algorithm before being stored in the database.
- Plain text passwords are never saved.
- During login, passwords are verified using bcrypt.compare() to securely match the entered password with the hashed password stored in the database.

User Authentication

- Secure login functionality is implemented to validate registered users.
- Authentication ensures that only authorized users can access protected routes and features.

Admin Approval for Restaurant Users

- Restaurant users must receive **admin approval** before gaining full access to the platform.
- Until approved, restaurant accounts remain restricted.
- This ensures authenticity and prevents unauthorized listings.

Role-Based Access Control (RBAC)

- The system supports multiple user roles such as:
  - Admin
  - Restaurant Owner
  - Customer
- Access to routes and functionalities is restricted based on user roles.
- Middleware is implemented to verify user roles before granting access to protected endpoints.

CORS Configuration

- Cross-Origin Resource Sharing (CORS) is properly configured to allow secure communication between frontend and backend servers.
- Only authorized origins are permitted to access the API.
- This prevents unauthorized cross-origin requests and enhances application security.

9. User Interface

The UI includes:

- Landing Page
- Login & Register Pages
- Restaurant Dashboard
- Customer Dashboard
- Cart Page
- Order Summary Page

- Admin Panel

The interface is responsive and user-friendly.

10. Testing Strategy

The Food Ordering Application was thoroughly tested to ensure reliability, accuracy, and smooth user experience. Both functional and manual testing approaches were used to validate system behavior.

Manual Testing

Manual testing was performed to verify core functionalities of the system:

1. User Registration

- Verified successful registration with valid inputs.
- Checked validation messages for empty or invalid fields.
- Ensured password hashing before storing in database.
- Confirmed duplicate email prevention.

2. Login Validation

- Tested login with correct credentials.
- Verified error handling for incorrect email/password.
- Checked role-based redirection after login.
- Confirmed JWT/token generation (if implemented).

3. Restaurant Approval

- Verified restaurant registration process.
- Tested admin approval functionality.
- Confirmed restricted access before approval.
- Ensured approved restaurants can access dashboard features.

4. Add Product Functionality

- Tested adding new food items with valid inputs.
- Verified image upload (if available).
- Checked database entry creation.
- Tested update and delete operations.

5. Cart Operations

- Verified adding items to cart.
- Tested quantity increase/decrease.
- Checked cart total calculation accuracy.
- Confirmed item removal functionality.

6. Order Placement

- Tested successful order placement process.
- Verified order data stored in database.
- Confirmed order history display.
- Checked status updates (if implemented).

Testing Methods Used

Browser Testing

- Tested application in multiple browsers (Chrome, Edge, etc.).
- Verified responsive design and UI consistency.
- Checked form validations and button actions.

Console Log Monitoring

- Used browser developer tools to monitor errors.
- Debugged API responses and request status codes.
- Verified frontend-backend communication.

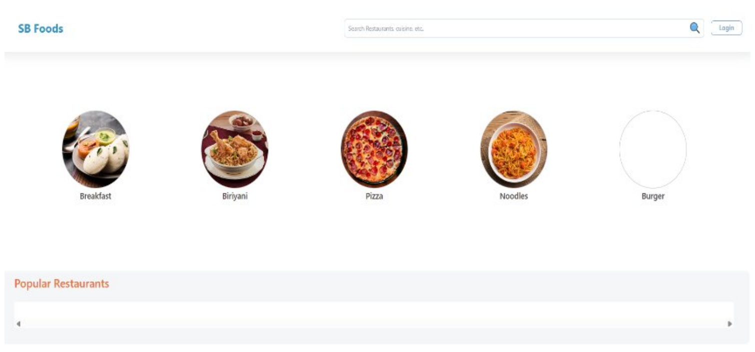
MongoDB Data Verification

- Directly checked MongoDB collections to confirm:
  - User data storage
  - Product entries
  - Cart details
  - Order records
- Ensured proper schema validation and data consistency.

11. Screenshots / Demo

Include screenshots of:

- **Landing Page**



Description:

The Landing Page is the home interface of the SB Foods web application. It provides users with an overview of available food categories and restaurants.

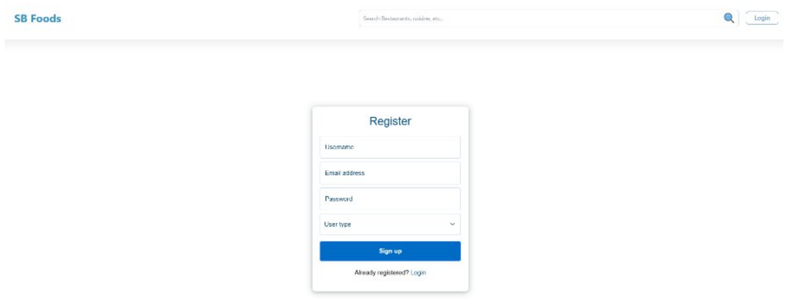
Features:

- Application logo and branding (SB Foods)
- Search bar for restaurants and food items
- Food categories (Breakfast, Biryani, Pizza, Noodles, Burger)
- Popular Restaurants section
- All Restaurants listing
- Footer with food categories and copyright

Purpose:

This page allows users to browse available food options and explore restaurants before logging in.

- ## Register Page



Description:

The Registration Page allows new users to create an account.

Fields:

- Username
- Email
- Password
- User Type (Admin / Restaurant / Customer)

Restaurant:

- Restaurant Address
- Restaurant Image

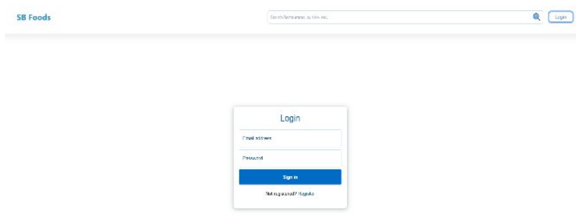
Functionality:

- Password is encrypted using bcrypt before storing in MongoDB.

- Restaurant users require admin approval.
- Customers are automatically approved.

Purpose:
Enables new users and restaurants to join the platform.

- ## Login Page



Description:

The Login Page allows registered users to access their accounts by entering valid credentials.

Fields:

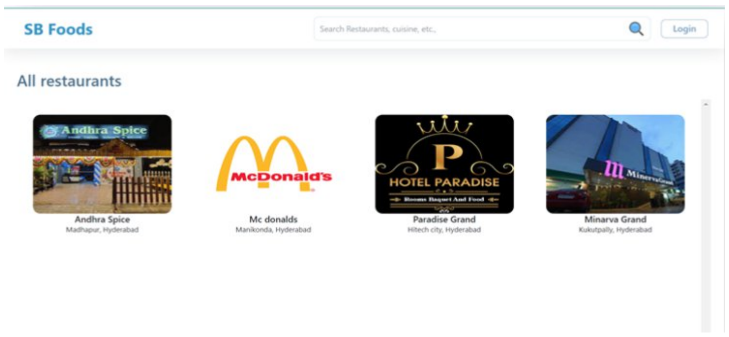
- Email Address
- Password

Functionality:

- Validates user credentials using backend API.
- Authenticates using bcrypt password comparison.
- Redirects user based on user type (Admin / Restaurant / Customer).

Purpose:
Provides secure authentication for users.

- ## Restaurant Dashboard



Description:

Displays the list of all available restaurants stored in MongoDB.

Features:

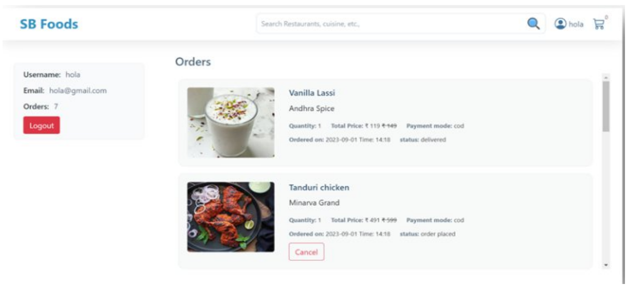
- Restaurant Name
- Restaurant Address
- Restaurant Image

Functionality:

- Data fetched from /fetch-restaurants API.
- Dynamically rendered using React frontend.

Purpose:
Allows users to select and view restaurant menus.

- User Profile



Username: hola
Email: hola@gmail.com
Orders: 7

Vanilla Lassi
Andhra Spice
Quantity: 1 Total Price: ₹ 119 ~~₹ 149~~ Payment mode: cod
Ordered on: 2023-09-01 Time: 14:18 status: delivered

Tanduri chicken
Minarva Grand
Quantity: 1 Total Price: ₹ 491 ~~₹ 599~~ Payment mode: cod
Ordered on: 2023-09-01 Time: 14:18 status: order placed

Description:

The User Profile page displays the logged-in user's details and order history.

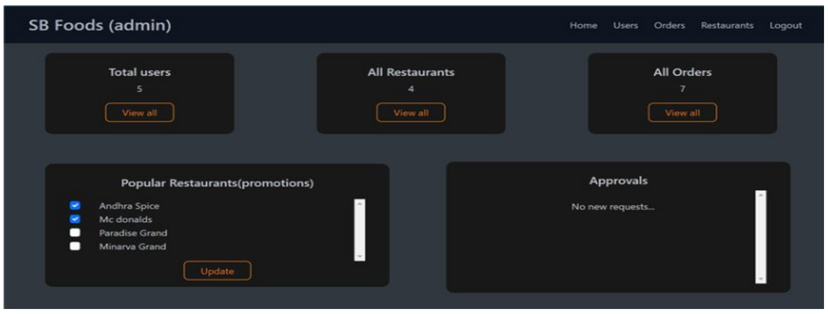
Features:

- Shows Username and Email
- Displays total number of orders
- Logout option
- Lists all placed orders with:
  - Item name and image
  - Restaurant name
  - Quantity and total price
  - Payment mode
  - Order date and status
  - Cancel option for pending orders

Purpose:

This page allows users to track their orders, check delivery status, and cancel orders if not delivered.

- Admin Dashboard



Description:

The Admin Dashboard provides complete control over the food ordering system. It displays overall platform statistics and management options.

Features:

- Total Users count
- Total Restaurants count
- Total Orders count
- View All buttons for detailed lists
- Popular Restaurants (Promotions) management
- Restaurant Approval section
- Logout option

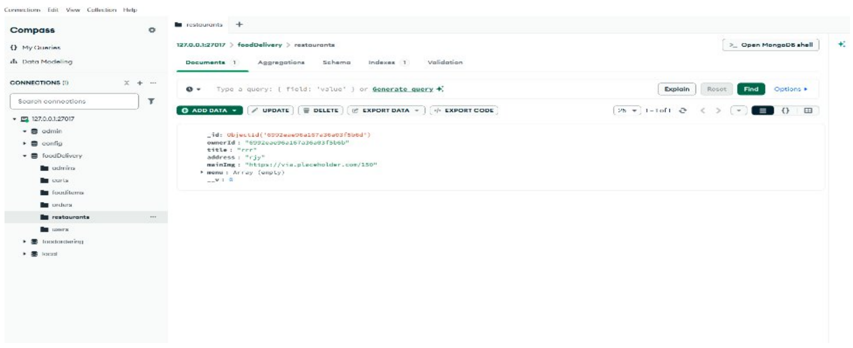
Functionality:

- Admin can view all users, restaurants, and orders.
- Admin can promote restaurants.
- Admin can approve or reject restaurant registration requests.

Purpose:

The Admin Dashboard helps manage users, restaurants, promotions, and system activities efficiently.

- ## MongoDB – Restaurants Collection



```
_id: ObjectId('6992eae96a167a36a03f5b6d')
ownerId : "6992eae96a167a36a03f5b6b"
title : "rrr"
address : "rjy"
mainImg : "https://via.placeholder.com/150"
▸ menu : Array (empty)
__v : 0
```

Description:

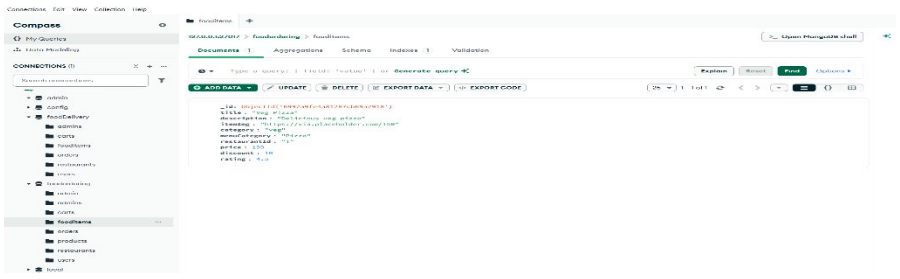
This screenshot shows the restaurants collection in MongoDB.

Fields:

- _id
- ownerId
- title
- address
- mainImg
- menu

Purpose:
Stores restaurant details registered by restaurant users.

- ## MongoDB – Food Items Collection



```
_id: ObjectId('[illegible]')
title : "Veg Pizza"
description : "Delicious veg pizza"
itemImg : "https://via.placeholder.com/150"
category : "veg"
menuCategory : "Pizza"
restaurantId : "1"
price : 150
discount : 10
rating : 4.5
```

Description:

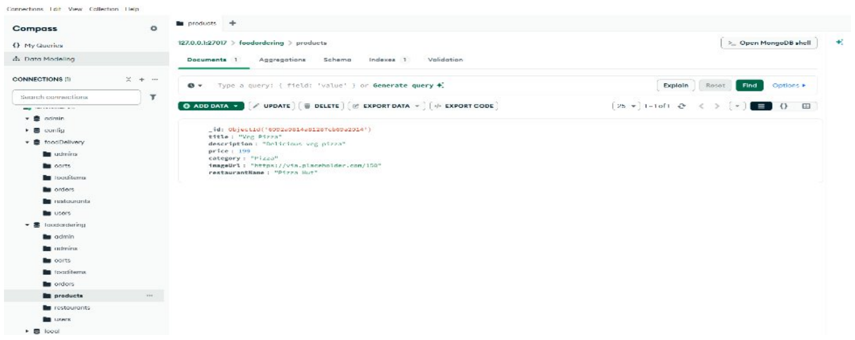
This screenshot shows the fooditems collection.

Fields:

- _id
- title
- description
- itemImg
- category
- menuCategory
- restaurantId
- price
- discount
- rating

Purpose:
Stores menu items added by restaurants.

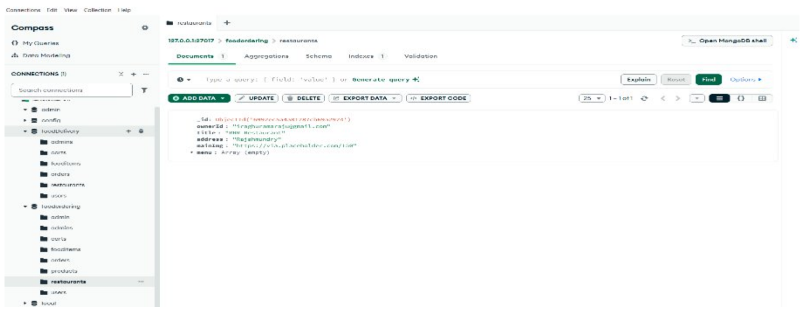
- ## Products Collection



```
_id: ObjectId('6952e0814e81287cb69a2014')
title : "Veg Pizza"
description : "Delicious veg pizza"
price : 199
category : "Pizza"
imageUrl : "https://via.placeholder.com/150"
restaurantName : "Pizza Hut"
```

This screenshot shows the Products collection in MongoDB. It stores food item details such as title, description, price, category, image URL, and restaurant name. Each product is uniquely identified using an ObjectId.

- ## Restaurants Collection



```
_id: ObjectId('[illegible]')
ownerId : "iraghuramaraju@gmail.com"
title : "[illegible] Restaurant"
address : "Rajahmundry"
mainImg : "https://via.placeholder.com/150"
▸ menu : Array (empty)
```

This screenshot shows the **Restaurants collection** in MongoDB. It contains restaurant details including owner ID, restaurant name, address, main image, and menu categories. Each restaurant is stored as a separate document.

<u>12. Known Issues & Limitations</u>

Although the Food Ordering Application is fully functional for core operations, there are certain limitations and areas for future enhancement.

<u>1. No Online Payment Gateway Integration</u>

- Currently, the system supports only basic order placement without integrated online payment processing.
- Payment gateways such as credit/debit cards, UPI, or digital wallets are not implemented.
- Orders are assumed to be Cash on Delivery (COD).
- Future improvement: Integration with secure payment gateways like Razorpay or Stripe.

<u>2. No Real-Time Order Tracking</u>

- The application does not provide live order tracking functionality.
- Customers cannot track delivery status in real time.
- Order status updates are limited to manual status changes (if implemented).
- Future improvement: Implement real-time tracking using WebSockets or real-time database updates.

3. User Interface Improvements

- The current UI is functional but can be enhanced for better user experience.
- Advanced UI features such as animations, improved layout design, and modern styling can be added.
- Mobile responsiveness can be further optimized.
- Future improvement: Implement enhanced UI using advanced CSS frameworks or improved design components.

4. Manual Admin Approval Process

- Restaurant approval requires manual action from the admin.
- No automated verification system is available.
- This may cause delays in restaurant activation.
- Future improvement: Implement automated email verification or document validation system.

13. Future Enhancements

To improve scalability, security, and user experience, the following enhancements can be implemented in future versions of the application:

1. Payment Gateway Integration (Razorpay / Stripe)

- Integrate secure online payment systems such as **Razorpay** or **Stripe**.
- Support multiple payment options including UPI, debit/credit cards, and net banking.
- Implement secure transaction validation and payment status verification.
- This will enhance user trust and enable seamless digital payments.

2. JWT-Based Authentication System

- Replace basic session-based authentication with **JSON Web Token (JWT)** authentication.
- Generate secure tokens upon login.
- Implement token expiration and refresh mechanisms.
- Improve API security and enable scalable authentication for large applications.

3. Real-Time Order Tracking

- Implement real-time order status updates (Order Placed → Preparing → Out for Delivery → Delivered).
- Use WebSockets or real-time database updates.
- Allow customers to track their order progress live.
- Improve customer engagement and transparency.

4. Rating and Review System

- Allow users to rate restaurants and food items.

- Enable customers to leave written reviews.
- Display average ratings on restaurant pages.
- Improve credibility and help users make informed decisions.

5. UI/UX Enhancement

- Improve overall application design with modern styling frameworks.
- Enhance responsiveness for mobile and tablet devices.
- Add smooth animations and interactive components.
- Provide better user experience and professional appearance.

6. Search and Filtering Functionality

- Implement search functionality for restaurants and food items.
- Add filtering options based on:
    - Category (Veg / Non-Veg / Beverages)
    - Price range
    - Ratings
- Improve navigation and reduce time required to find desired items.

Conclusion

The OrderOnTheGo project successfully demonstrates the implementation of a full-stack web application using the MERN stack. It integrates frontend, backend, and database components efficiently. The system provides role-based functionality for customers, restaurants, and administrators, ensuring smooth food ordering operations.

This project enhanced understanding of:

- REST API development
- MongoDB database design
- React component-based architecture
- Authentication and password security
- Full-stack application deployment