

Full Stack Development with MERN

Project Documentation

1. Introduction

- **Project Title:** SB Foods – Food Ordering App
- **Team Members:** DANUSHIYA P (311421205015)

HEMAHARINI N (311421205026)

MANISHA M (311421205048)

DIVYA S (311421205021)

2. Project Overview

◆ Purpose:

The purpose of **SB Foods** is to provide a seamless and efficient online food ordering experience, connecting users with a variety of popular restaurants. Designed with both users and restaurant owners in mind, SB Foods aims to simplify food discovery, ordering, and management, making it easier for food lovers to find and enjoy their favourite dishes while empowering restaurant owners with tools to manage their offerings effectively.

◆ Features:

- **User-Friendly Interface:** A clean and intuitive interface for effortless browsing and ordering.
- **Comprehensive Dish Details:** View detailed descriptions, customer reviews, pricing, and active promotions for each dish, enabling informed decisions.
- **Streamlined Ordering Process:** Quick and easy checkout with minimal inputs, including name, delivery address, and payment method.

- **Admin Dashboard:** A powerful dashboard for administrators to manage users, products, orders, and restaurant approvals.
- **Restaurant Management:** Restaurant owners can manage their menu items, including adding, editing, and categorizing dishes, upon admin approval.
- **Cart and Order History:** Users can add items to a cart, place orders, and review past orders in their profile section.
- **Multi-Role Support:** Different flows and dashboards for users, restaurant owners, and administrators to facilitate specific needs.
- **Real-Time Order Confirmation:** Immediate confirmation for users after placing an order, enhancing the user experience.

3. Architecture

◆ Frontend:

The frontend of **SB Foods** is built using **React**, structured with a component-based architecture to ensure modularity and reusability.

Key components include:

- ★ **Authentication Components:** Handle user and admin registration, login, and logout functionality.
- ★ **Product Display and Details:** Dynamic components to display restaurants and menu items with detailed views for each item, including images, descriptions, and reviews.
- ★ **Cart and Checkout:** A shopping cart component that manages items selected by users and integrates a checkout process for order confirmation.
- ★ **Admin and User Dashboards:** Conditional rendering allows different interfaces for users, restaurant owners, and administrators, showing specific sections like order history, menu management, and user management.

React's state management (via Context API or Redux) helps maintain the flow of information across components, while **React Router** enables smooth navigation throughout the app. Additionally, UI libraries like Material-UI or Bootstrap may be used for consistent styling and responsive design.

◆ Backend:

The backend architecture leverages **Node.js** with **Express.js** as the web framework, organizing the server-side logic into distinct modules for scalability and maintainability:

- ★ **API Endpoints:** RESTful API endpoints serve data to the frontend and handle CRUD operations for users, orders, products, and restaurants. This includes:
 - 🌸 **User Routes:** Handles user registration, authentication, profile management, and order history.
 - 🌸 **Product Routes:** Manages product listings, categorization, and availability.
 - 🌸 **Order Routes:** Processes orders from cart to checkout, including confirmation and tracking.
 - 🌸 **Admin Routes:** Provides administrative functions, such as user management, order tracking, and restaurant approvals.
- ★ **Authentication & Authorization:** JWT (JSON Web Tokens) are used to secure routes and provide role-based access control for users, restaurant owners, and admins.
- ★ **Error Handling & Validation:** Middleware functions handle errors and validate data to ensure a robust and secure system.
- ◆ **Database:** MongoDB is used as the database, storing application data across various collections. Each collection has a well-defined schema, optimizing data retrieval and interactions
 - **Users Collection:** Stores user information (e.g., name, email, hashed password) along with role details (e.g., user, restaurant owner, admin).
 - **Restaurants Collection:** Holds restaurant details and their menu items, including information about each dish's description, category, price, and any promotions.
 - **Products Collection:** A dedicated collection for menu items, with fields like name, category, price, description, and associated restaurantId.
 - **Orders Collection:** Tracks each order with details such as userId, items, totalAmount, deliveryAddress, and status.
 - **Cart Collection:** Temporary storage of selected items for each user before checkout, including productId, quantity, and userId.

Interactions:

MongoDB communicates with the backend through Mongoose ORM, enabling schema enforcement and data modeling. Each operation (e.g., adding an item to the cart, updating order status) is translated into a MongoDB query, ensuring efficient database interactions.

4. Setup Instructions

- ◆ **Prerequisites:** Before setting up the SB Foods project, make sure you have the following dependencies installed:
- 🌈 **Node.js:** Download Node.js (Ensure it includes npm, the Node package manager).
- 🌈 **MongoDB:** Install MongoDB or set up a MongoDB Atlas account for a cloud-based instance.

◆ Installation:

1. Clone the Repository

Start by cloning the project repository to your local machine:

```
git clone https://github.com/Danushiya/SB-Foods
```

```
cd sb-foods
```

2. Backend Setup

- Navigate to the server directory:

```
cd server
```

- Install backend dependencies:

```
npm install
```

- Start the backend server:

```
npm start
```

- The backend server should now be running on <http://localhost:27017>

3. Frontend Setup

- Open a new terminal, navigate to the client directory:

```
cd client
```

- Install frontend dependencies:

```
npm install
```

- Start the frontend development server:

```
npm start
```

The frontend will run on `http://localhost:3000` by default.

4. Access the Application

- Open your web browser and navigate to `http://localhost:3000` to access the SB Foods application.
- Ensure that both the frontend and backend servers are running for full functionality.

5. Folder Structure

◆ **Client:** React Frontend

The **React frontend** is organized into a modular structure to ensure clean, maintainable code and efficient development.

- `public/`: Contains static files accessible by the browser, including the `index.html` file, icons, and images. This is where the React app is loaded.
- `src/`: The main source directory for the React application, containing all the components, styles, and logic.
- `components/`: Houses reusable UI components, such as buttons, forms, product cards, and navigation bars.
- `pages/`: Contains individual pages for routing, such as Home, ProductList, ProductDetail, Cart, Profile, and AdminDashboard.
- `services/`: Manages API requests to the backend using libraries like `axios`. This typically includes functions for user login, product retrieval, order submission, etc.
- `context/`: Contains React context files for managing global states, such as authentication and cart items.
- `styles/`: Holds global CSS or styled-component files to maintain consistent design across the app.
- `App.js`: The root component, defining routes and managing the flow between different pages.
- `index.js`: The entry point file that renders the App component and attaches it to the DOM.

◆ **Server:** Node.js Backend

The **Node.js backend** is structured to support scalability and maintainability, with clearly defined folders and files for different functionalities.

- **config/:** Contains configuration files, such as the MongoDB connection setup and environment variable handling.
- **controllers/:** Houses business logic for each feature, managing request handling, data processing, and response formatting.
- Examples include `UserController.js` (user-related operations), `productController.js` (product handling), and `orderController.js` (order management).
- **models/:** Defines the MongoDB schemas and models using Mongoose. Models include `User.js`, `Product.js`, `Order.js`, and `Restaurant.js`, representing different entities in the database.
- **routes/:** Contains route definitions that map HTTP requests to specific controller functions.
- Each route file is organized by entity, such as `userRoutes.js`, `productRoutes.js`, and `adminRoutes.js`.
- **middleware/:** Holds middleware functions for tasks like error handling, authentication (JWT verification), and request validation.
- **utils/:** Contains helper functions for common tasks, such as token generation, email validation, and other reusable utilities.
- **server.js:** The main entry point for the backend server, where Express is configured, routes are set up, middleware is applied, and the server is started.

6. Running the Application

To run the SB Foods application locally, follow these commands to start both the frontend and backend servers:

1. Start the Backend Server

- Open a terminal, navigate to the server directory:
cd server
- Install necessary dependencies (if not done previously):
npm install
- Start the backend server:
npm start
- The backend server will run on `http://localhost:5000` by default.

2. Start the Frontend Server

- Open a new terminal, navigate to the client directory:

cd client

- Install frontend dependencies (if not done previously):

npm install

- Start the frontend server:

npm start

- The frontend will be accessible on <http://localhost:3000> by default.

With both servers running, you can open <http://localhost:3000> in your browser to view and interact with the SB Foods application locally.

7. API Documentation

Below is a list of the key API endpoints exposed by the SB Foods backend. Each endpoint includes the request method, URL, parameters, and example responses.

User Authentication

1. Register User

- **Method:** POST
- **Endpoint:** `/api/users/register`
- **Request Body:**

```
{  
  
  "name": "Danushiya",  
  
  "email": "danushiya@gmail.com",  
  
  "password": "danu@123"  
}
```

- **Response:**

```
{  
  
  "message": "User registered successfully",  
}
```

```
"userId": "62e7e5a4e5b3d65d8b1c0c4e"
```

```
}
```

2. Login User

- **Method:** POST
- **Endpoint:** /api/users/login
- **Request Body:**

```
{
```

```
"email": "danushiya@gmail.com",
```

```
"password": "danu@123"
```

```
}
```

- **Response:**

```
{
```

```
"message": "Login successful",
```

```
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
```

```
}
```

3. Product Management

1. Get All Products

- **Method:** GET
- **Endpoint:** /api/products
- **Response:**

```
[
```

```
{
```

```
"productId": "6731e396cd4c496d939b250c",
```

```
"title": "French Fries",
```

```
"description": "with tasty crispy crust and soft crumby inside",
```

```
"itemImg": "https://i.postimg.cc/5yQB7c6h/fries.png",
```



```
"category": "Veg",  
  
"menuCategory": "Side dish",  
  
"restaurantId": "6731e1ffcd4c496d939b24de",  
  
"price": 99,  
  
"discount": 10,  
  
},  
  
{  
  
  "productId": "62e8b7f9f2d3a7b1e3d8c1a5",  
  
  "name": "Pepperoni Pizza",  
  
  "description": "Classic pizza with pepperoni and cheese",  
  
  "price": 199,  
  
  "category": "Pizza",  
  
  "restaurantId": "62e8a5d6f9b2c4b7e3d2b1b2"  
  
}  
  
]
```

2. Add a New Product (Restaurant Owner)

- **Method:** POST
- **Endpoint:** /api/products
- **Request Body:**

```
{  
  
  "name": "Chocolate Lava Cake",  
  
  "description": "Chocolate Goodness! Soft chocolate cake with lava oozing center",  
  
  "price": 108,  
  
  "category": "Desserts",  
  
  "restaurantId": "62e8a5d6f9b2c4b7e3d2b1b2"  
  
}
```

- **Response:**

```
{  
  
  "message": "Product added successfully",  
  
  "productId": "62e8a5d6f9b2c4b7e3d2b1a0"  
}
```

4. Cart Management

1. Add Item to Cart

- **Method:** POST
- **Endpoint:** /api/cart/add
- **Request Body:**

```
{  
  
  "userId": "62e7e5a4e5b3d65d8b1c0c4e",  
  
  "productId": "62e8a5d6f9b2c4b7e3d2b1a0",  
  
  "quantity": 2  
}
```

- **Response:**

```
{  
  
  "message": "Item added to cart",  
  
  "cartId": "62e9b1f5e5b3d67e3d2b2a7f"  
}
```

2. Get User Cart

- **Method:** GET
- **Endpoint:** /api/cart/{userId}
- **Response:**

```
[
```

```
{  
  "productId": "62e8a5d6f9b2c4b7e3d2b1a0",  
  "name": "Chocolate Lava Cake",  
  "quantity": 2,  
  "price": 108,  
  "total": 216  
}  
]
```

5. Order Management

1. Place Order

- **Method:** POST
- **Endpoint:** /api/orders
- **Request Body:**

```
{  
  "userId": "6731fcedcd4c496d939b281c",  
  "items": [  
    {  
      "productId": "6731e4bacd4c496d939b252a",  
      "quantity": 2  
    }  
  ],  
  "totalAmount": 768.97,  
  "address": "Nungambakkam",  
  "paymentMethod": "cod"  
}
```

- **Response:**

```
{  
  "message": "Order placed successfully",  
  "orderId": "62e9f1f5e5b3d67e3d2b2a7c"  
}
```

2. Get User Orders

- **Method:** GET
- **Endpoint:** /api/orders/{userId}
- **Response:**

```
[  
  {  
    "orderId": "62e9f1f5e5b3d67e3d2b2a7c",  
    "items": [  
      {  
        "productId": "62e8a5d6f9b2c4b7e3d2b1a0",  
        "name": "Chocolate Lava Cake",  
        "quantity": 2,  
        "price": 108  
      }  
    ],  
    "totalAmount": 216,  
    "status": "Pending",  
    "address": "123 Main Street, Cityville",  
    "paymentMethod": "Credit Card"  
  }  
]
```

6. Admin Management

1. Approve Restaurant

- **Method:** PATCH
- **Endpoint:** /api/admin/approveRestaurant/{restaurantId}
- **Response:**

```
{  
  
  "message": "Restaurant approved successfully",  
  
  "restaurantId": "62e8a5d6f9b2c4b7e3d2b1b2"  
}
```

2. Get All Users

- **Method:** GET
- **Endpoint:** /api/admin/users
- **Response:**

```
[  
  
  {  
  
    "userId": "62e7e5a4e5b3d65d8b1c0c4e",  
  
    "name": "Danushiya",  
  
    "email": "danushiya@gmail.com",  
  
    "role": "user"  
  },  
  
  {  
  
    "userId": "62e8a5f9f2d3a7b1e3d8c1a6",  
  
    "name": "Starbucks",  
  
    "email": "starbucks@example.com",  
  
    "role": "restaurantOwner"  
  }  
]
```

8. Authentication and Authorization

In the SB Foods project, **authentication** and **authorization** are managed using **JSON Web Tokens (JWT)** for secure and stateless sessions. JWT tokens provide a simple, efficient way to verify user identities and grant access based on roles (user, restaurant owner, or admin).

Authentication Flow

1. User Registration:

- When a new user registers, the backend validates and saves their information (name, email, password) in the database.
- Passwords are **hashed** using bcrypt before being stored for enhanced security.
- After successful registration, the user can log in to the application.

2. User Login:

- On login, the backend verifies the user's credentials (email and password).
- If authentication is successful, a **JWT token** is generated and sent to the client.
- The JWT contains the user's **ID** and **role**, allowing the system to identify the user without storing session data server-side.
- The client stores this token (typically in localStorage or sessionStorage) and includes it in future requests as part of the Authorization header.

JWT Token Structure

- **Header:** Specifies the signing algorithm and token type.
- **Payload:** Encodes the user's ID, role, and any other claims necessary for authorization.
- **Signature:** Ensures the token's integrity and authenticity, signed using a secret key on the server.

Authorization Flow

1. Token Verification:

- For any protected route (e.g., accessing the user profile, placing an order), the frontend sends the JWT token in the Authorization header.
- The backend checks the token's validity and decodes it using the secret key.
- If the token is valid, the request proceeds; if not, the server responds with an **authentication error**.

2. Role-Based Access Control (RBAC):

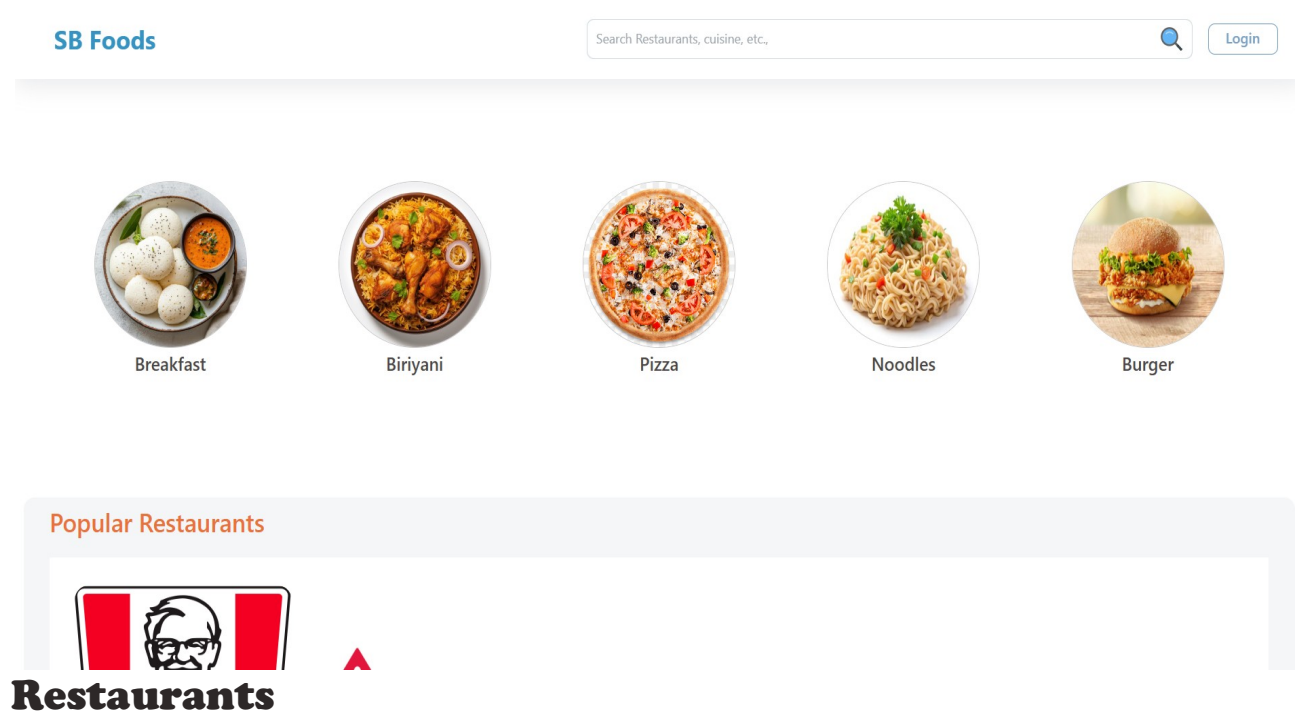
- The decoded token also includes the user's **role**, which determines access permissions:
- **User:** Can view and manage their profile, view products, add items to the cart, and place orders.
- **Restaurant Owner:** Can add or edit their products but must get admin approval.
- **Admin:** Has full access to manage users, approve restaurants, view all orders, and handle products across the platform.
- Middleware functions check the user's role before granting access to restricted endpoints. For example, only an admin role can access `/api/admin/approveRestaurant`.

Token Expiration and Refresh

- **Token Expiration:** Tokens are typically set with an expiration time (e.g., 1 hour), after which the user must re-authenticate.
- **Token Refresh:** If implementing token refresh, the frontend can periodically refresh the token without needing the user to re-login, improving user experience while maintaining security.

9. User Interface

Landing Page





All Restaurants



KFC
Virugambakkam



ibaco
Kodambakkam



Starbucks
Anna Nagar



Dindigul Thalappakatti Restaurant
K.K Nagar



Domino's Pizza

Restaurant Menu



KFC

Virugambakkam

Filters

Sort By

- ☐ Popularity
- ☐ low-price
- ☐ high-price
- ☐ Discount
- ☐ Rating

Food Type

- ☐ Veg
- ☐ Non Veg
- ☐ Beverages

Categories

- ☐ Side dish
- ☐ Main Entree
- ☐ MATCH DAY COMBOS
- ☐ Flavours
- ☐ Drinks
- ☐ Biryani Variety

All Items



French Fries
with tasty crispy crust a...

₹ 89 99

Add item



8 pc Hot & Crispy Chicken
Signature Hot & crispy ch...

₹ 653 760-97

Add item



Cricket Crunch Meal
Crunchy Savings of Rs. 75...

₹ 449 449

Add item



Ultimate Savings Chicken Bucket
Savings of 35% on signatu...

₹ 454 699-05

Add item



Authentication

Register

Username

Email address

Password

User type

User type

Admin

Restaurant

Customer

User Profile

Orders

Username: harini
Email: harini@gmail.com
Orders: 4
Logout

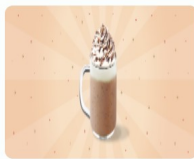


Cookie Creme Latte
Starbucks
Quantity: 1 Total Price: ₹ 430 ₹ 436 Payment mode: cod
Ordered on: 2024-11-11 Time: 12:51 status: In-transit
Cancel



Cricket Crunch Meal
KFC
Quantity: 1 Total Price: ₹ 449 ₹ 449 Payment mode: cod
Ordered on: 2024-11-11 Time: 12:51 status: delivered

Cart



Cookie Creme Latte

Starbucks

Quantity: 1

Price: ₹ 430 ~~₹430~~

Remove



CHOCOLATE OVERLOAD

ibaco

Quantity: 1

Price: ₹ 159 ~~₹199~~

Remove



Ultimate Savings Chicken Bucket

KFC

Quantity: 1

Price: ₹ 454 ~~₹699.05~~

Remove

Price Details

Total MRP: ₹ 2095.05

Discount on MRP: - ₹ 336

Delivery Charges: + ₹ 0

Final Price: ₹
1759.0500000000002

Place order

Admin Dashboard

Total users

7

View all

All Restaurants

5

View all

All Orders

12

View all

Popular Restaurants(promotions)

- ☒ KFC
- ☐ ibaco
- ☐ Starbucks
- ☐ Dindigul Thalappakatti Restaurant
- ☒ Domino's Pizza

Update

Approvals


No new requests...

All Orders

SB Foods (admin)

HomeUsersOrdersRestaurantsLogout

Orders



CHOCOLATE OVERLOAD

ibaco

Userid: 6731fdfbcd4c496d939b28d4Name: PriyaMobile: 9178689542Email: priya@gmail.com

Quantity: 1Total Price: ₹ 159 ₹ 199Payment mode: upi


Address: Ashok NagarPincode: 600025Ordered on: 2024-11-11 Time: 12:53

status: order placed

Update order status

Update

Cancel



VANILLA CHOCO CHIPS

ibaco


Userid: 6731fdfbcd4c496d939b28d4Name: PriyaMobile: 9178689542Email: priya@gmail.com

All Restaurants


SB Foods (admin)

HomeUsersOrdersRestaurantsLogout


All restaurants




KFC
Virugambakkam




ibaco
Kodambakkam



Starbucks
Anna Nagar

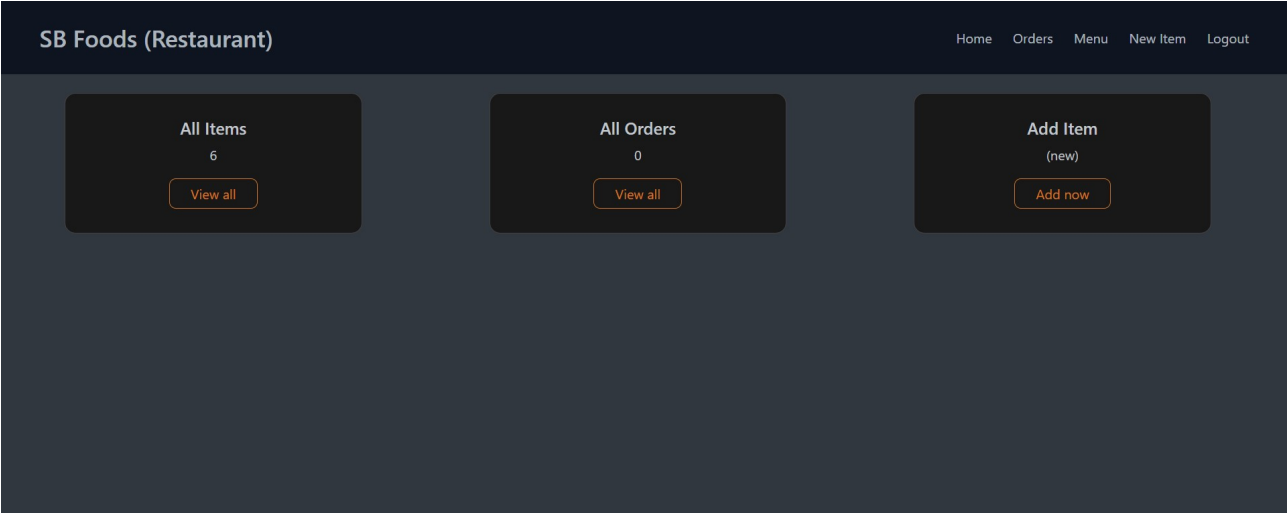


Dindigul Thalappakatti Restaurant
K.K Nagar

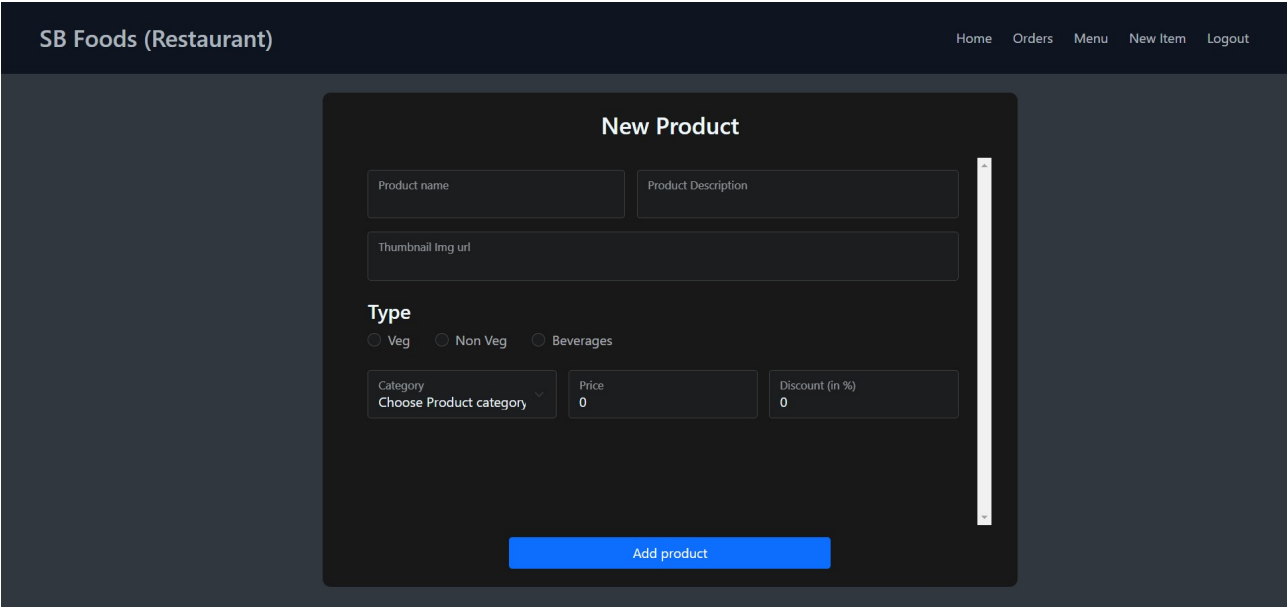


Domino's Pizza
Anna Nagar

Restaurant Dashboard



New Item



10. Testing

Summary of Testing Tools

Testing Tool	Purpose	Used For
Jest	Unit Testing	Frontend components
Mocha & Chai	Unit & Integration Testing	Backend API and services
Supertest	API Testing	Backend API endpoints
Cypress	End-to-End Testing	Full frontend user flows
Postman	API Testing	Manual endpoint verification
Manual Testing	UI/UX and Compatibility	UI, cross-browser, and responsiveness

11. Demo Video

Demo Video - https://drive.google.com/file/d/1fem-9O4VOITK_oQrgjlpQmvRWyAU8_E/view?usp=sharing

12. Known Issues

● Database Connection Stability

● **Issue:** Occasionally, there may be delays or failures when connecting to the MongoDB database, especially during heavy traffic or high concurrency.

● **Solution:** Optimize the database connection and introduce retry mechanisms or connection pooling to improve stability under load.

● Token Expiration Handling

● **Issue:** The token expiration may not be well-handled in some cases, which can cause users to be logged out unexpectedly.

● **Solution:** Implement a refresh token mechanism to automatically renew JWT tokens before expiration.

13. Future Enhancements

To further improve the SB Foods project and enhance user experience, the following features and improvements are planned for future versions:

1. User Review and Rating System

● **Feature:** Allow customers to leave reviews and ratings for products and restaurants. This feature would help improve customer feedback and aid other users in decision-making.

● **Potential Benefit:** Better product discovery and increased customer engagement.

2. Payment Gateway Integration

- **Feature:** Integrate a payment gateway (e.g., Stripe, PayPal) to allow users to securely pay for their orders directly within the app.
- **Potential Benefit:** Streamlined ordering process with secure transactions, enhancing the user experience.

3. Multi-language and Currency Support

- **Feature:** Add the ability to change the language and currency based on the user's location or preferences.
- **Potential Benefit:** Attract international users and create a more inclusive experience.

4. Restaurant Rating and Review System

- **Feature:** Allow customers to rate and review restaurants in addition to products.
- **Potential Benefit:** Create a more robust feedback system and improve restaurant visibility.