**Full Stack Development with MERN**

**Project Documentation**

**1. Introduction**

* **Project Title:**SB Foods - Food Ordering App
* **Team Members:**
  + Sree Varshini S (2021503563) – Developer
  + Sree Vaasini S (2021503562) – Developer
  + Monish R (2021503030) – Developer
  + Faheema Gullnaas (2021503016) - Developer

**2. Project Overview**

**Purpose:**

SB Foods is a full-stack food-ordering application designed to revolutionize online food ordering with a seamless, efficient, and user-friendly experience. The platform empowers users to browse, customize, and order food effortlessly while enabling restaurants and administrators to manage listings and orders effectively.

**Features**:

* **User Flow**: Registration, login, product browsing, adding to cart, placing orders, and tracking via profile.
* **Restaurant Flow**: Restaurant authentication, admin approval, and food listing management.
* **Admin Flow**: Managing users, products, orders, and restaurant approvals.

**3. Architecture**

* **Frontend**:  
  The frontend is built using **React.js**, ensuring a responsive and interactive user experience. Features include user authentication, product browsing, cart management, and admin dashboards.
* **Backend**:  
  The backend is developed with **Node.js** and **Express.js**. It handles server-side logic, routing, API endpoints for users, orders, products, and authentication mechanisms.
* **Database**:  
  **MongoDB** serves as the database, organized into schemas for users, orders, products, carts, admins, and restaurants. The database is connected using **Mongoose** for Object-Document Mapping (ODM).

**4. Setup Instructions**

* **Prerequisites:**
  + Node.js with npm
  + MongoDB
  + Express.js
  + React.js
  + Git
* **Installation:**  
  + Clone the repository:
    - git clone <https://github.com/harsha-vardhan-reddy-07/Food-Ordering-App-MERN>
    - cd Food-Ordering-App-MERN
  + Install dependencies:
    - npm install
  + Start the development server:
    - npm run dev
  + Access the app at http://localhost:3000.

**5. Folder Structure**

* Client:
  + src/components: Contains UI components for the frontend.
  + src/pages: Defines pages like login, profile, and product listing.
* Server:
  + routes: Defines API endpoints for users, products, orders, and authentication.
  + models: Contains MongoDB schemas for users, orders, and other collections.

**6. Running the Application**

* Provide commands to start the frontend and backend servers locally.
  + **Frontend:** npm start in the client directory.
  + **Backend:** npm start in the server directory.

**7. API Documentation**

Endpoints:

User API:

POST /api/users/register: Registers a new user.

POST /api/users/login: Authenticates a user.

Product API:

GET /api/products: Fetches available products.

Order API:

POST /api/orders: Places a new order.

GET /api/orders/:userId: Retrieves user-specific orders.

Example Response:

GET /api/products

json

Copy code

[

{

"id": "1",

"name": "Pizza",

"price": 12.99,

"description": "Delicious cheese pizza",

"rating": 4.5

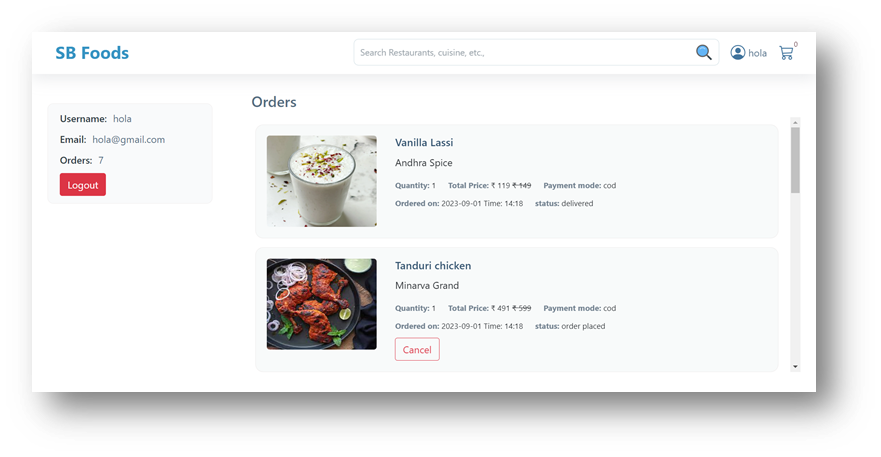
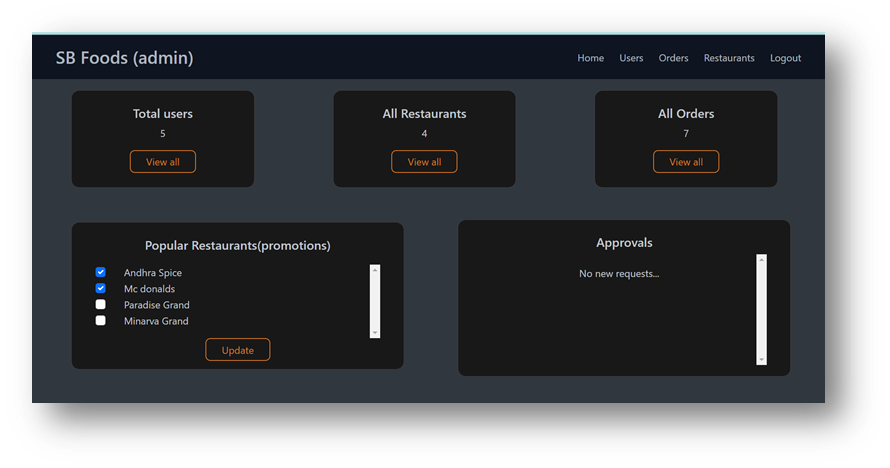
}

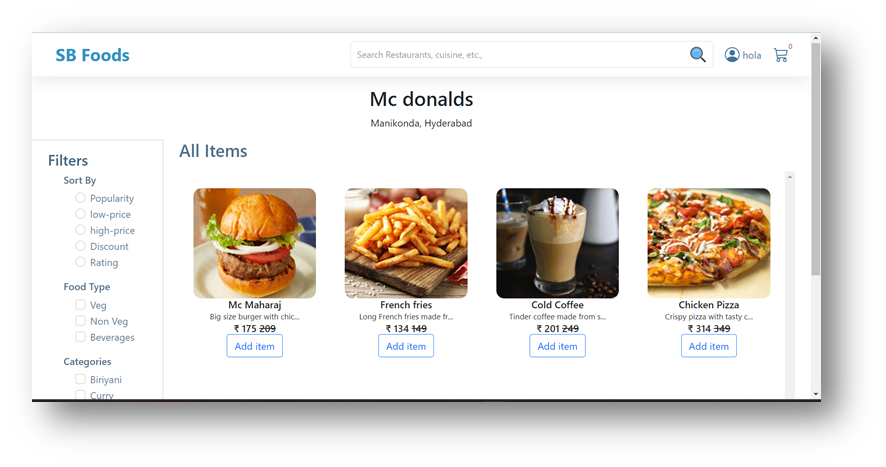
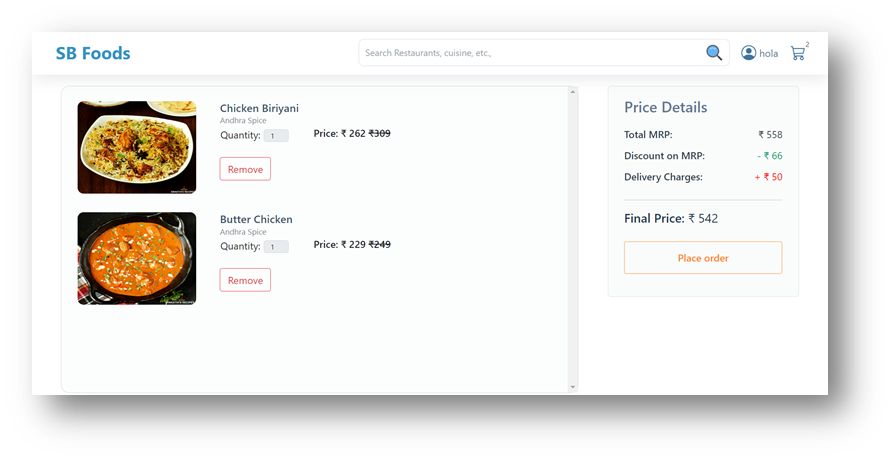
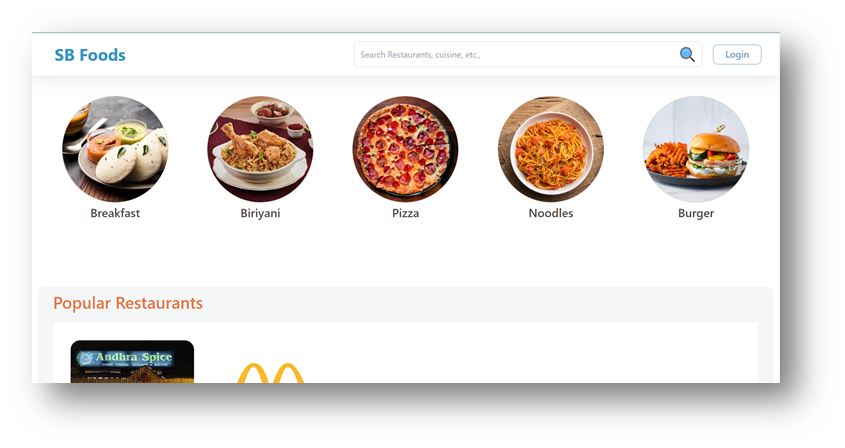
]

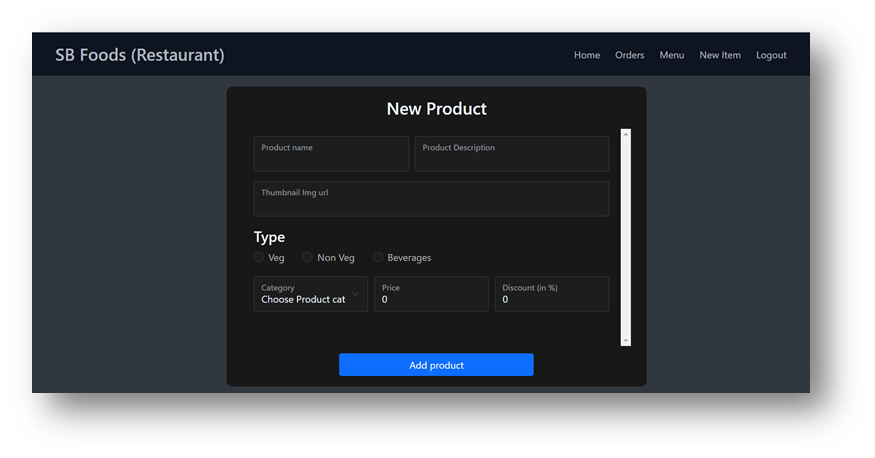
**8. Authentication**

**Token-based Authentication**:

* Users and restaurants are authenticated using JSON Web Tokens (JWT).
* Tokens are issued during login and validated for secure routes.

**9. User Interface**





**10. Testing**

Strategy:Unit testing with Jest for frontend components.

API testing with Postman: Integration testing to ensure seamless communication between frontend and backend.

**11. Known Issues**

* Limited payment gateway options.
* No push notifications for order updates.

**12. Future Enhancements**

* Mobile App: Expand to Android and iOS platforms.
* AI Recommendations: Suggest dishes based on user preferences.
* Enhanced Analytics: Advanced reporting for admins and restaurants.