**Full Stack Development with MERN**

**Project Document**

**1. Introduction**

* **Project Title:** SB Foods - Food Ordering App
* **Team Members:** Ashwanth R (2021503503)  
   Sandeep Raman R (2021503551)  
   Mohamed Afri Habeeb Mohamed (2021503029)  
   Taufiq Aziz A R (2021503565)  
   Anurudh J (2021503005)

**2. Project Overview**

* **Purpose:** The SB Foods Management System is designed to streamline and enhance the user experience for an online food ordering platform, catering to three different perspectives: Admin, Customer, and Restaurant. The main goal of the project is to create a seamless and efficient interface for managing food orders, restaurant operations, and customer interactions. The system aims to provide a comprehensive solution for order management, inventory control, and user engagement, ensuring a smooth end-to-end process from placing orders to delivery.
* **Features:**

**Streamline Ordering Process**: Provide customers with an easy and intuitive platform to browse, order, and track their food.

**Efficient Restaurant Management:** Enable restaurants to manage their menus, process orders, and update inventory in real-time.

**Comprehensive Admin Control:** Allow the admin to oversee and manage the entire platform, including user management, order tracking, and data analytics.

**Enhanced User Experience:** Offer a user-friendly interface that provides a pleasant experience for customers, restaurant staff, and administrators.

**Secure Transactions:** Ensure secure and reliable payment processing for customers.

**3. Architecture**

* **Frontend:** React Architecture
* The frontend of the SB Foods Management System is developed using React, a popular JavaScript library for building user interfaces. The architecture follows the component-based design pattern, allowing for reusable, maintainable, and scalable UI components.
* Component-Based Structure: The application is divided into reusable components like Header, Footer, Menu, OrderCard, RestaurantList, etc.
* State Management: Uses React's Context API or Redux for global state management, enabling the synchronization of user data and order status across components.
* Routing: Utilizes React Router for managing navigation and creating a seamless single-page application (SPA) experience.
* API Integration: Uses Axios or Fetch API for handling asynchronous HTTP requests to interact with the backend services.
* Styling: CSS-in-JS (e.g., Styled Components) or frameworks like Bootstrap or TailwindCSS are used for responsive design.
* **Backend:** Node.js and Express.js Architecture
* The backend is built using Node.js and Express.js, providing a robust, scalable, and efficient server-side environment.
* RESTful API: The backend exposes a set of RESTful APIs to interact with the frontend, handling CRUD operations for users, orders, and restaurant data.
* Authentication & Authorization: Uses JWT (JSON Web Token) for secure authentication and user sessions.
* Middleware: Implements custom middleware for error handling, logging, and request validation using Express.js.
* Security: Implements security best practices such as CORS, helmet.js, and rate limiting to protect against common vulnerabilities.
* **Database:** MongoDB Schema and Interactions
* The system uses MongoDB, a NoSQL database, for its flexibility and scalability. The schema is defined using Mongoose.
* User Schema: Contains user details like username, email, password (hashed), role (customer, restaurant owner, admin), and address.
* Restaurant Schema: Stores restaurant details, including name, menu items (with sub-documents for each item), ratings, and operational details.
* Order Schema: Captures order data, including user ID, restaurant ID, order items, total price, and status (e.g., pending, delivered).
* Menu Schema: Manages menu items with fields like item name, description, price, and availability.

**4. Setup Instructions**

* **Prerequisites:**   
  Node.js (v18+)  
  MongoDB (v6+)  
  Git for version control

NPM for package management

* **Installation:**

1. Install Dependencies  
 Frontend:  
 cd client  
 npm install

Backend:  
 cd server  
 npm install

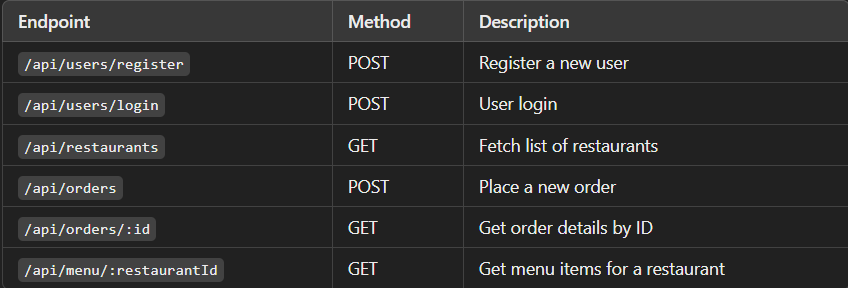
**5. Folder Structure**

* **Client:**   
  client/
* ├── src/
* │ ├── components/
* │ │ ├── Header.jsx
* │ │ ├── Footer.jsx
* │ ├── pages/
* │ │ ├── Home.jsx
* │ │ ├── Menu.jsx
* │ │ ├── Cart.jsx
* │ ├── context/
* │ │ └── AuthContext.jsx
* │ ├── utils/
* │ │ └── api.js
* │ └── App.jsx
* ├── public/
* │ └── index.html
* └── package.json
* **Server:**
* **server/**
* ├── controllers/
* │ ├── userController.js
* │ ├── orderController.js
* │ ├── restaurantController.js
* ├── models/
* │ ├── User.js
* │ ├── Order.js
* │ ├── Restaurant.js
* ├── routes/
* │ ├── userRoutes.js
* │ ├── orderRoutes.js
* │ ├── restaurantRoutes.js
* ├── middleware/
* │ ├── authMiddleware.js
* │ └── errorHandler.js
* ├── config/
* │ └── db.js
* ├── index.js
* └── package.json

**6. Running the Application**

* Commands to start the frontend and backend servers locally.  
  + **Frontend:**  
    cd client  
    npm start
  + **Backend:**  
    cd server  
    node index.js

**7. API Documentation**

* 
* Example Response for GET/api/restaurants  
  [  
   {

"\_id": "634b8f4e1d2e4a321b8e",

"name": "Spicy Delight",

"address": "123 Main St",

"rating": 4.5

}

]

**8. Authentication**

* The system uses JWT for authentication:
* Login: Users receive a JWT token upon login.
* Protected Routes: Middleware verifies the token for accessing protected endpoints.
* Authorization: Role-based access control is implemented (Admin, Customer, Restaurant Owner).

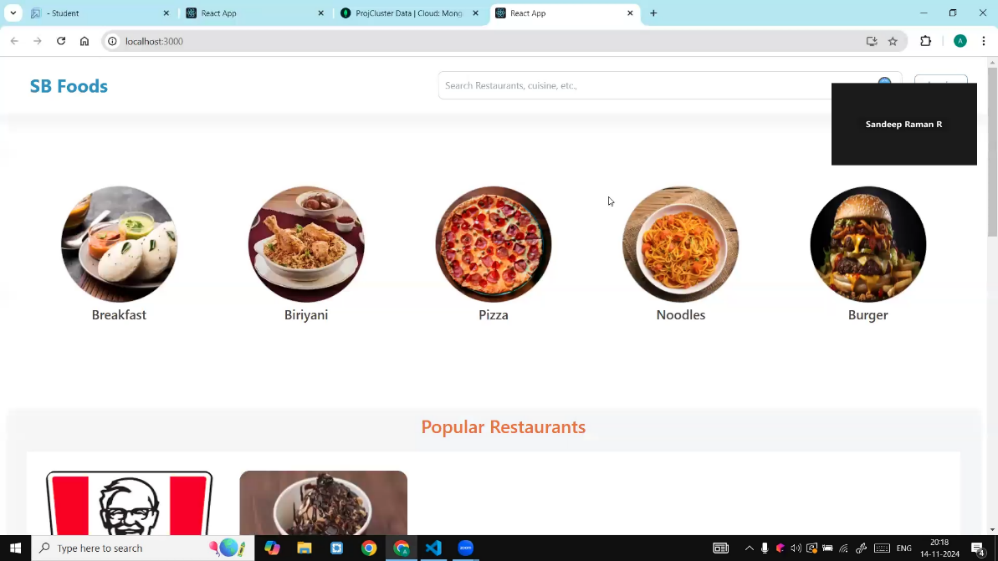
**9. User Interface**

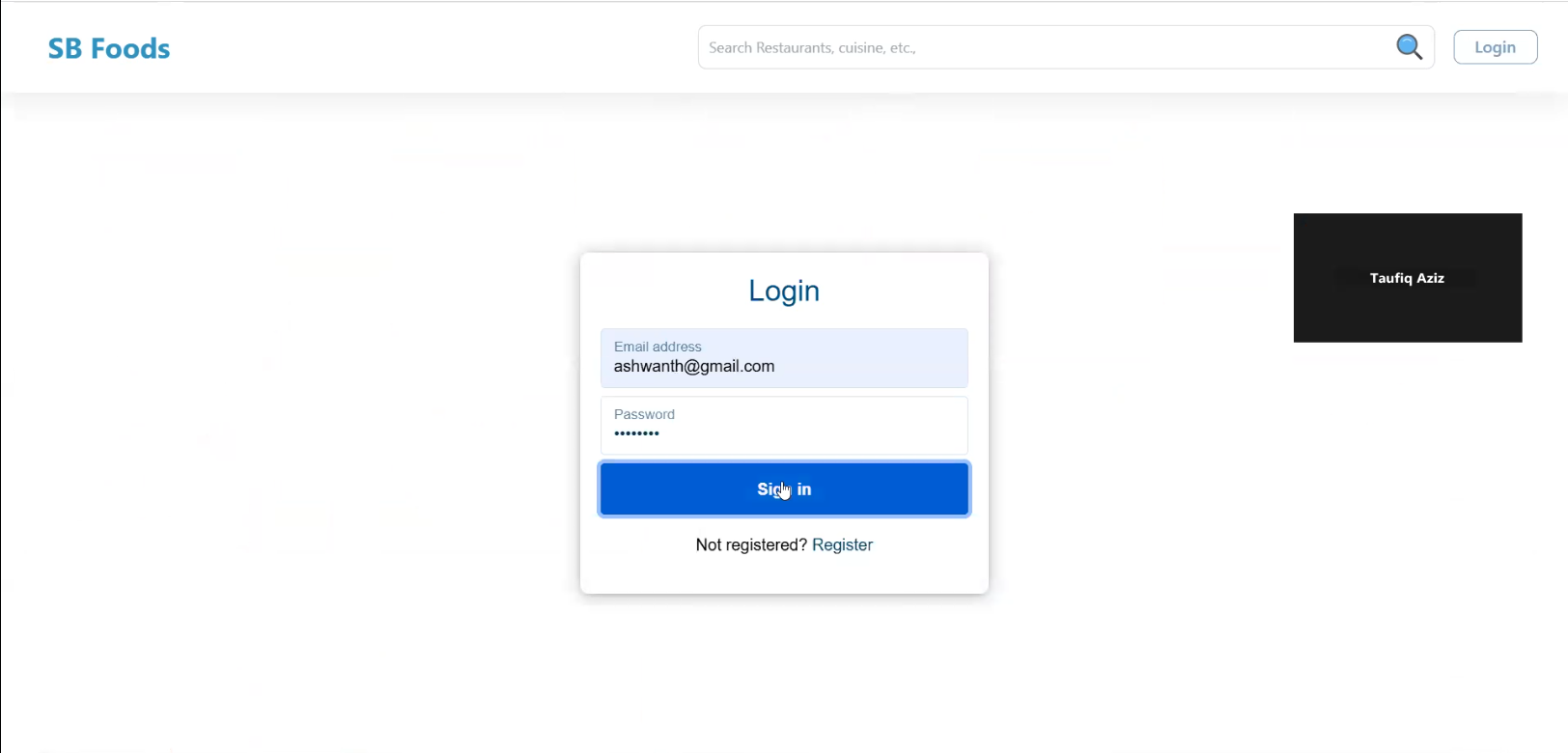
* Dashboard: Admin dashboard with analytics and controls.
* Order Page: Customer order placement and status tracking.
* Restaurant Page: Menu management and order processing.

**10. Testing**

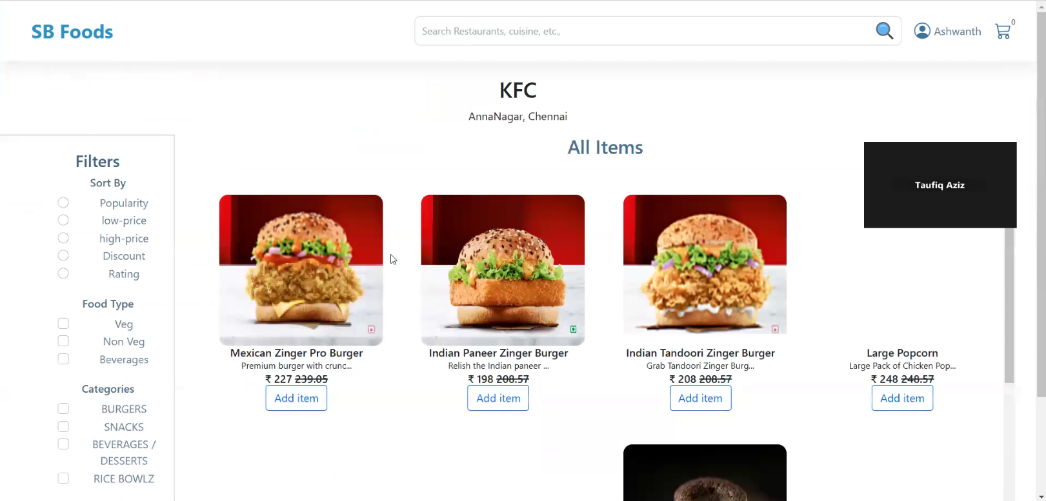
* Tools: Jest for unit tests, Supertest for API testing, and React Testing Library for frontend components.
* Strategy:
* Unit Testing for individual functions and components.
* Integration Testing for API endpoints.
* End-to-End Testing using Cypress.

**11. Screenshots or Demo**

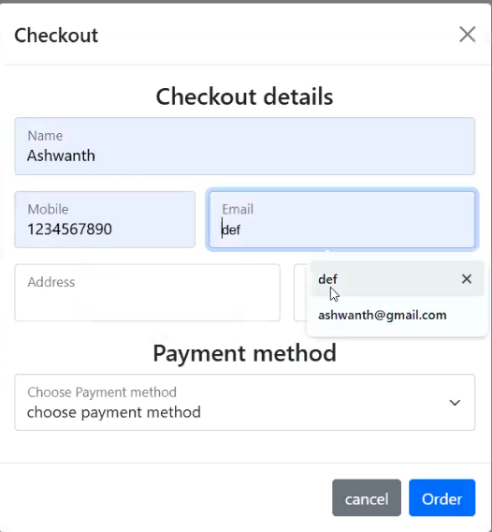
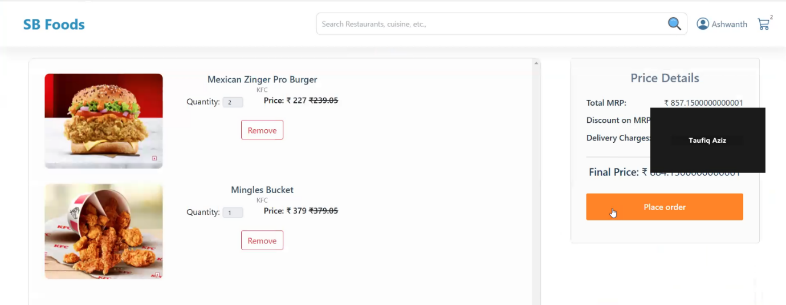
* Customer UI  
  
* Login Page

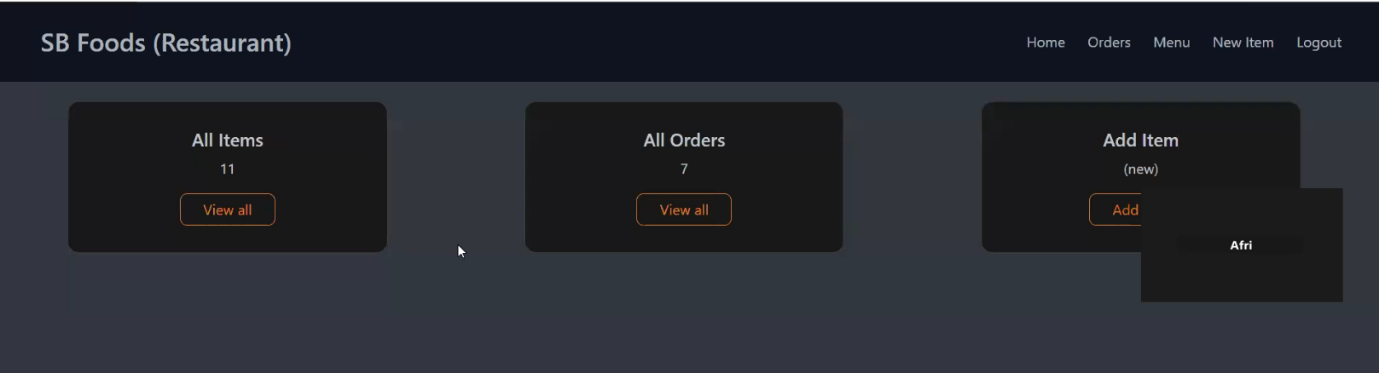


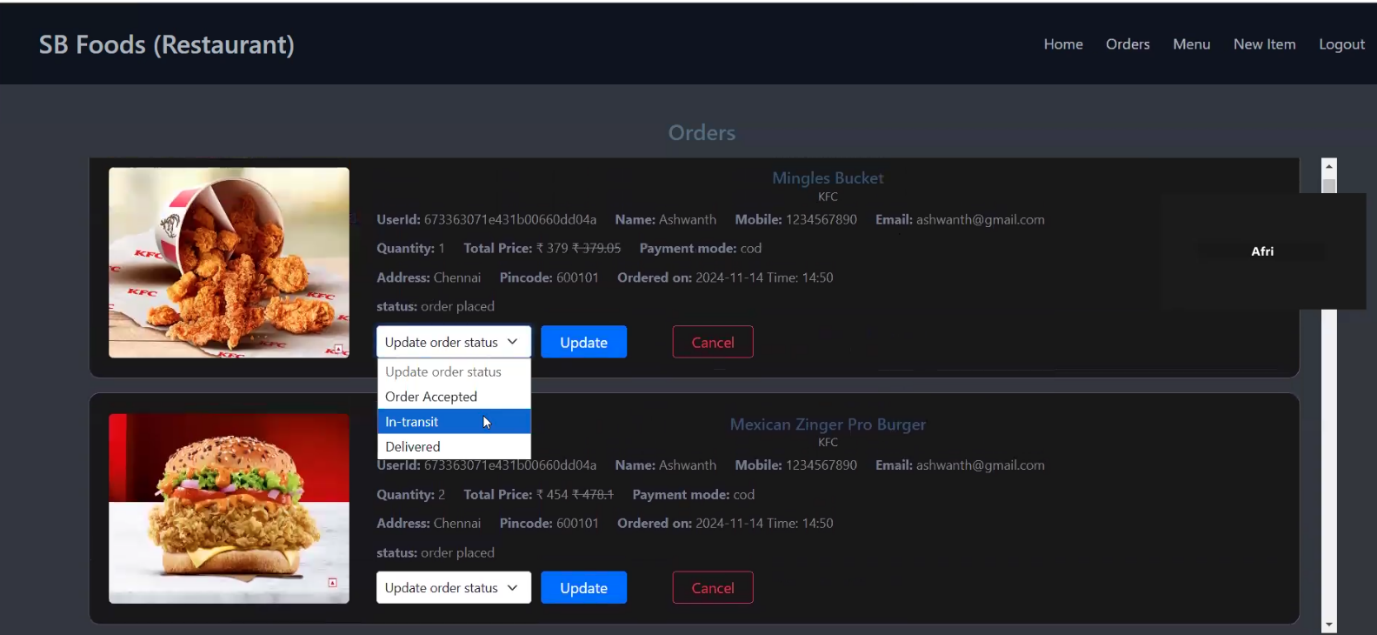
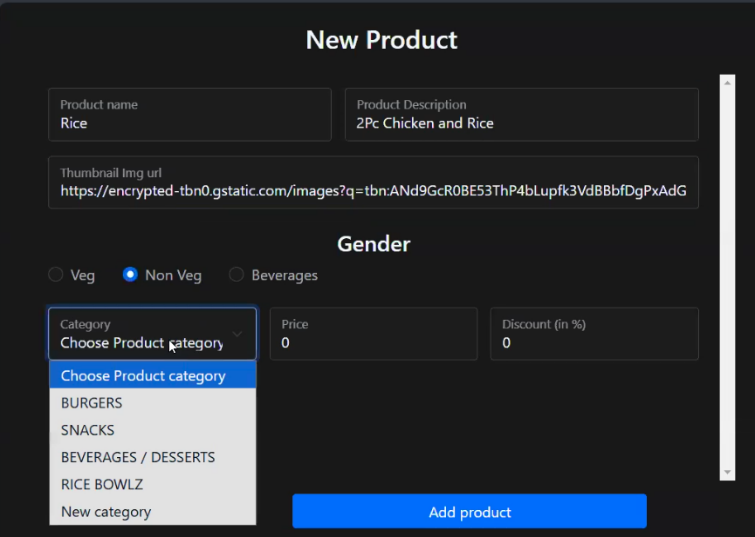
* Order Page



* Payment Page



* Restaurant Page



**12. Known Issues**

* Session Expiry: Users may need to re-login after long periods of inactivity.
* Order Synchronization: Occasional delay in reflecting order status changes in the customer UI.

**13. Future Enhancements**

* Real-Time Notifications: Using Socket.io for real-time updates on order status.
* Mobile App: Extend the platform to a mobile application using React Native.
* Payment Gateway Integration: Support for additional payment methods like Apple Pay and Google Pay.
* AI-based Recommendations: Implement personalized dish recommendations based on user preferences and order history.