# **Full Stack Development with MERN**

**SB FOODS – Food ordering online app**

**1. Introduction**

* **Project Title: SB Foods**
* **Team Members:**   
    
   1.**UBIKA.U**- Front end development  
   2.**YUVAN SHANKAR RAJA.A**- Back end development  
   3.**SAIVIKRAM.K**:UI/UX designing

2. **Project Overview**

The SB Foods Online Ordering App is a digital platform aimed at revolutionizing the food ordering experience for customers, enabling them to explore and order from a diverse menu with ease. The app will provide a user-friendly interface, seamless order management, secure payment processing, and convenient delivery tracking, all aimed at enhancing customer satisfaction. SB Foods aims to meet the demands of a modern food market, focusing on speed, personalization, and convenience.

**Objectives**

* **Enhanced Customer Experience**: Provide a seamless, intuitive interface that simplifies food ordering for all users.
* **Increased Order Efficiency**: Streamline the ordering process from browsing to checkout.
* **Expanded Reach**: Attract a broader customer base by offering a convenient alternative to in-person ordering.
* **Data Collection and Analytics**: Gain insights into customer preferences and ordering habits for better menu optimization and promotions.

**Key Features**

1. **User Registration & Profile Management**:
   * Easy account creation and secure login.
   * Options to save payment details and address for future orders.
2. **Menu Browsing and Search Functionality**:
   * Searchable, categorized menu (e.g., starters, mains, desserts).
   * Detailed descriptions, photos, and prices for each item.
3. **Personalization and Recommendations**:
   * AI-based suggestions based on past orders and preferences.
   * "Most Popular" and "Chef's Specials" sections.
4. **Order Customization**:
   * Options to customize items (e.g., spice level, toppings).
   * Special instructions for each item.
5. **Order Tracking**:
   * Real-time order status updates and estimated delivery time.
   * GPS tracking of delivery personnel.
6. **Payment Processing**:
   * Secure payment gateway supporting credit/debit cards, digital wallets, and cash-on-delivery.
7. **Push Notifications & Alerts**:
   * Notifications for order confirmation, preparation, out-for-delivery status, and special offers.
8. **Review & Feedback System**:
   * Option for customers to review items and provide feedback on their experience.
9. **Admin Dashboard**:
   * Centralized control for managing menu, prices, orders, and promotions.
   * Insights into sales data, popular items, and customer demographics.

**Frontend Architecture**

**Technology**: React (React Native for mobile)

The app’s frontend is built with React, which makes it easy to create reusable components and manage the app’s state.

* **Components**: The app will have separate components for pages like the Menu, Cart, Order Tracking, and Profile.
* **State Management**: We’ll use tools like React’s Context API or Redux to keep track of the user’s information and their cart.
* **Navigation**: React Router (for web) or React Navigation (for mobile) will help users move between screens easily.
* **Data Handling**: Axios or Fetch API will handle communication with the backend, allowing users to view menu items, place orders, and check order status.

**Backend Architecture**

**Technology**: Node.js and Express.js

The backend is organized to handle requests from the frontend and perform actions like fetching menu items and processing orders.

* **API Structure**: The backend provides endpoints like /menu for fetching items, /order for placing orders, and /profile for managing user details.
* **Authentication**: We’ll use JSON Web Tokens (JWT) to securely keep users logged in.
* **Business Logic**: Key functions like managing orders and user sessions are separated into services to keep code organized.
* **Real-Time Updates**: WebSockets or Firebase will provide live updates to track order status.

**Database Architecture**

**Database**: MongoDB

MongoDB is used for storing user profiles, menu items, and order details in collections that allow fast access to relevant data.

* **User Collection**: Stores user details, including their name, email, address, and order history.
* **Menu Collection**: Stores categories and items with details like name, price, description, and customization options.
* **Order Collection**: Tracks each order with its items, total cost, status (e.g., pending, delivered), and delivery address.
* **Real-Time Tracking**: Keeps the current location of orders in a collection that updates as the delivery progresses.

**4. Setup Instructions**

**Prerequisites**

* **Node.js** - for running the app.
* **MongoDB** - for the database.
* **Git** - for cloning the repository.

**Installation**

**1.Clone the Repository**:

git clone <repository-url>

cd <repository-name>

2. **Install Dependencies**:

**Backend**:

cd server

npm install

**Frontend**:

cd client

npm install

**Set Up Environment Variables**:

PORT=5000

MONGODB\_URI=<your-mongodb-connection-string>

JWT\_SECRET=<your-secret-key>

**5. Folder Structure**

**Client (Frontend)**

* **/src/components**: UI components.
* **/src/pages**: Main pages (e.g., Menu, Cart).
* **/src/App.js**: Main app component.

**Server (Backend)**

* **/src/models**: Database models for users, orders, etc.
* **/src/routes**: API routes.
* **/src/server.js**: Starts the backend server.

**6. Running the Application**

**Frontend**:

cd client

npm start

**Backend**:

cd server

npm start

**7. API Documentation**

* **POST /api/users/register**: Register a user.
* **POST /api/users/login**: Login and get a token.
* **GET /api/menu**: Get menu items.
* **POST /api/orders**: Place an order.

**8. Authentication**

* **JWT Tokens**: Users log in and get a token, which they use for secure access to routes.

**10. Testing**

* **Jest** for unit testing.
* **Postman** for API testing.
* **Cypress** for end-to-end testing.

**11. Known Issues**

* **Loading Delays**: Occasional delays when fetching menu items if the server load is high. This can be optimized with caching in the future.
* **Login Persistence**: Users may get logged out after closing the app, as token persistence across sessions hasn’t been fully implemented yet.
* **Order Tracking Accuracy**: Real-time location tracking may sometimes show a delay due to limited update frequency.
* **Error Handling**: Limited feedback for network errors; currently, only basic error messages are shown to users.
* **Compatibility**: Some minor display issues on older devices and screen sizes, particularly for mobile.

**12. Future Enhancements**

* **Improved Caching**: Add caching for menu items and user data to reduce load times and server requests.
* **Enhanced Order Customization**: Allow more options for customizing items (e.g., additional toppings or substitutions).
* **Multi-Language Support**: Add support for additional languages to reach a broader audience.
* **Offline Mode**: Allow users to browse the menu offline, with the ability to sync orders once back online.
* **Advanced Analytics**: Integrate advanced analytics to better understand user behavior and preferences.
* **Loyalty Program**: Implement a points-based loyalty program for frequent customers.
* **Enhanced Error Messages**: Provide more specific feedback and troubleshooting for network or server issues.
* **Order History Improvements**: Allow users to re-order past orders with a single click and view more detailed order history.