**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

* **Project Title:** SB Food Delivery app
* **Team Members:**

1. Janani. R (Team Leader)
2. B. Lokesh Kumar
3. C. Santhosh
4. M. Dhivakar

**2. Project Overview**

* **purpose :**

**The purpose of a food delivery app is to provide a convenient and efficient platform for users to order food from local restaurants and have it delivered directly to their location.**

* **Features:**

1. Enhance Convenience: Make it easy for users to browse menus, place orders, and receive food without leaving their homes or offices.

2. Improve Accessibility: Offer a wide selection of restaurants and cuisines, allowing users to explore different food options based on their preferences.

3. Streamline Delivery: Ensure quick and reliable food delivery through a user-friendly interface that tracks orders in real-time.

4. Support Local Businesses: Connect restaurants with a broader customer base, helping them increase sales and visibility.

5. Provide Personalization: Offer recommendations, save preferences, and offer promotions or loyalty rewards based on user behavior and ordering habits.

**3. Architecture**

**1.Frontend:**Building the frontend architecture for a food delivery app using **React** requires a solid and scalable design to ensure a seamless user experience, and Below is a breakdown of how you could structure the frontend architecture.

1. Footer: It is an essential element that shares the details of the food in the app.
2. Login: A **login** feature is a critical part of any food delivery app, allowing users to access personalized services like order history, preferences, and saved payment methods.
3. Navbar: A **navbar** (navigation bar) is a key element in any web app, including a food delivery app, as it allows users to navigate between different sections (home, menu, cart, profile, etc.). It should be easily accessible, intuitive, and mobile-friendly.
4. Popular restaurants:It feature or showcase a list of popular restaurants within a **food delivery app**
5. Register: To implement a **register** (sign-up) feature in a food delivery app using **React**, you would typically need a form where users can enter their details, such as **name**, **email**, **password**, and perhaps other details like **phone number** or **address**.
6. Restaurants: **Restaurants** feature in a food delivery app, you're likely talking about a listing of available restaurants that users can browse, filter, and interact with. This could include displaying details like restaurant name, ratings, cuisine type, delivery time, price range, and more. Additionally, users should be able to view menus, place orders, and maybe even read reviews.

**2.Backend:**  Below is an outline of a backend architecture for a food ordering app using **Node.js** and **Express.js**. This outline covers the essential components, including API routes, database structure, authentication, and scaling considerations.

**Components:**

1. **node modules:**  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
2. **env:**  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
3. **Index.**js:  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
4. Package-lock.json:  This file is automatically generated when you first install dependencies using npm install.  It is crucial for ensuring consistent and deterministic dependency installations across different environments or among different developers on a project.
5. package.json: This file is a crucial part of a Node.js project. It serves as the manifest for your project, defining metadata, dependencies, scripts, and other configuration details for your application.
6. Schema.js:This file typically defines the structure of your data model, especially if you're using an ORM (Object-Relational Mapping) library like **Sequelize** (for SQL databases) or **Mongoose** (for MongoDB). The schema defines how data is structured in your database and often includes the attributes, data types, relationships, validation rules, and any other constraints on that data.

**3.Database Schema:**

In a food ordering application, the following entities are important,

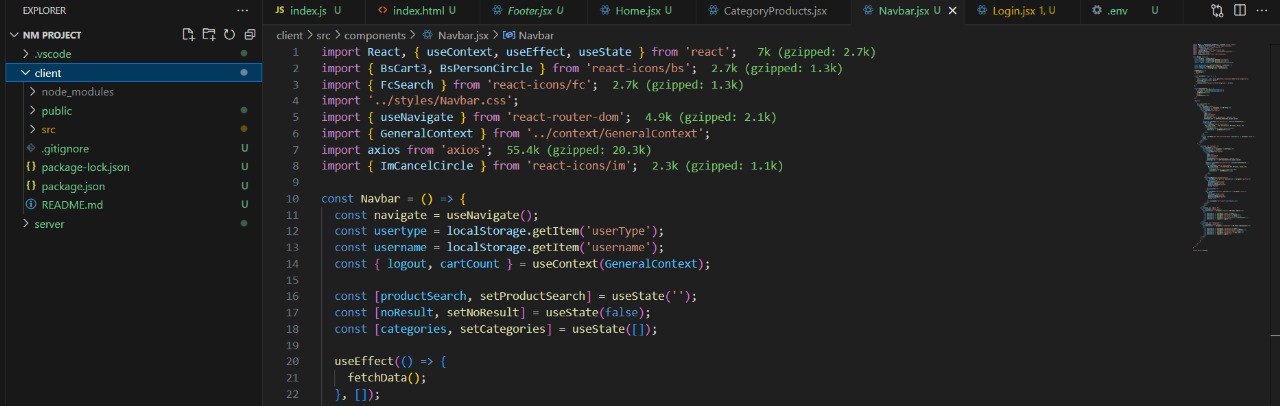
1. User: Represents a customer or admin of the system.
2. MenuItem: Represents a food item on the restaurant’s menu.
3. Order: Represents a customer’s order, which includes multiple menu items.
4. Payment: Represents a payment for an order.
5. Cart (Optional): Represents items that a user has added to their cart before placing an order.

**5.Folder Structure:**

**1.Client:**Building the frontend architecture for a food delivery app using **React** requires a solid and scalable design to ensure a seamless user experience, and Below is a breakdown of how you could structure the frontend architecture.

Components:

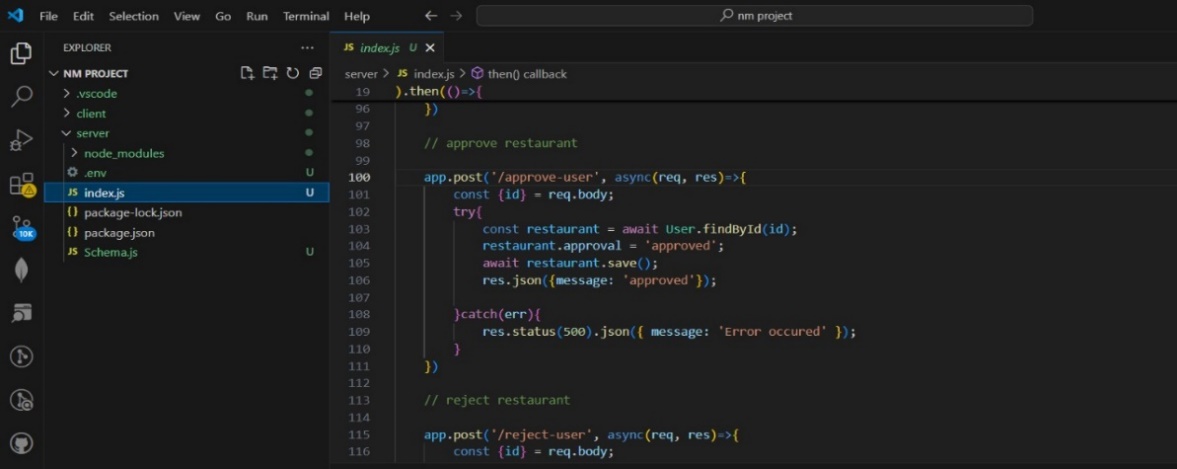
1. Footer: It is an essential element that shares the details of the food in the app.
2. Login: A **login** feature is a critical part of any food delivery app, allowing users to access personalized services like order history, preferences, and saved payment methods.
3. Navbar: A **navbar** (navigation bar) is a key element in any web app, including a food delivery app, as it allows users to navigate between different sections (home, menu, cart, profile, etc.). It should be easily accessible, intuitive, and mobile-friendly.
4. Popular restaurants:It feature or showcase a list of popular restaurants within a **food delivery app**
5. Register: To implement a **register** (sign-up) feature in a food delivery app using **React**, you would typically need a form where users can enter their details, such as **name**, **email**, **password**, and perhaps other details like **phone number** or **address**.
6. Restaurants: **Restaurants** feature in a food delivery app, you're likely talking about a listing of available restaurants that users can browse, filter, and interact with. This could include displaying details like restaurant name, ratings, cuisine type, delivery time, price range, and more. Additionally, users should be able to view menus, place orders, and maybe even read reviews.



**2.server:**  Below is an outline of a backend architecture for a food ordering app using **Node.js** and **Express.js**. This outline covers the essential components, including API routes, database structure, authentication, and scaling considerations.

**Components:**

1. **node modules:**  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
2. **env:**  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
3. **Index.**js:  building a **food ordering app** using **Node.js** and **Express.js**, you'll need several Node.js modules (or packages) to handle various tasks like routing, authentication, database management, real-time communication, and payment integration.
4. Package-lock.json:  This file is automatically generated when you first install dependencies using npm install.  It is crucial for ensuring consistent and deterministic dependency installations across different environments or among different developers on a project.
5. package.json: This file is a crucial part of a Node.js project. It serves as the manifest for your project, defining metadata, dependencies, scripts, and other configuration details for your application.
6. Schema.js:This file typically defines the structure of your data model, especially if you're using an ORM (Object-Relational Mapping) library like **Sequelize** (for SQL databases) or **Mongoose** (for MongoDB). The schema defines how data is structured in your database and often includes the attributes, data types, relationships, validation rules, and any other constraints on that data.

****

**6. Running the Application:**

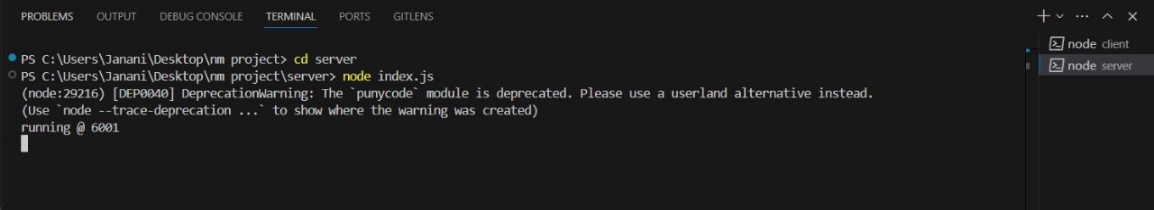
1. **Client:**

**Command:**

****

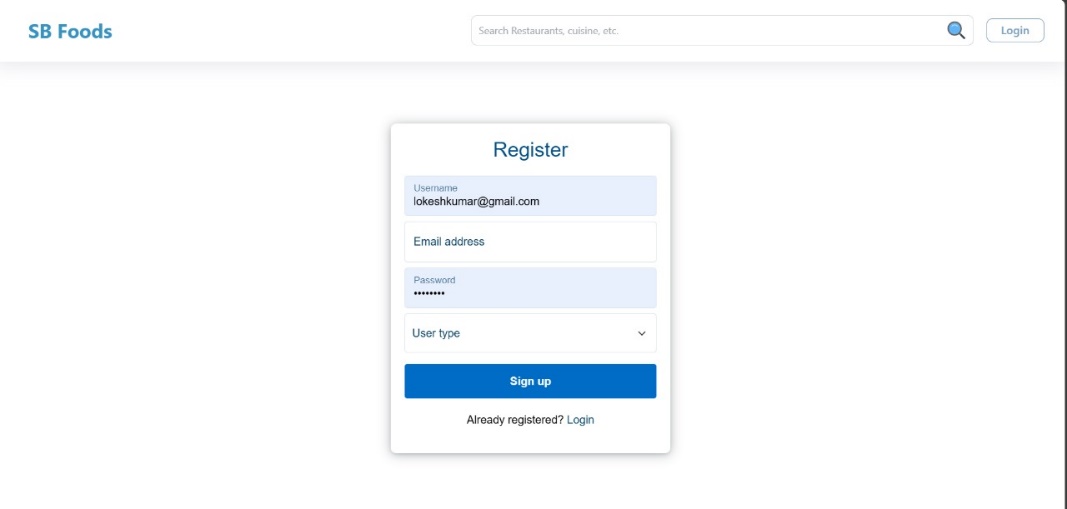
1. **Server:**

**Command:**

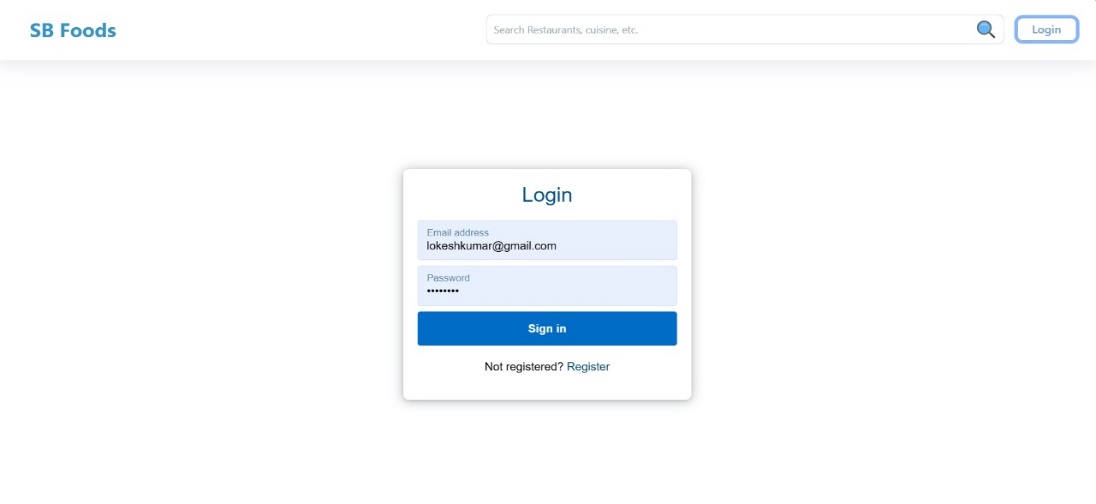
****

7.API DOCUMENTATION:

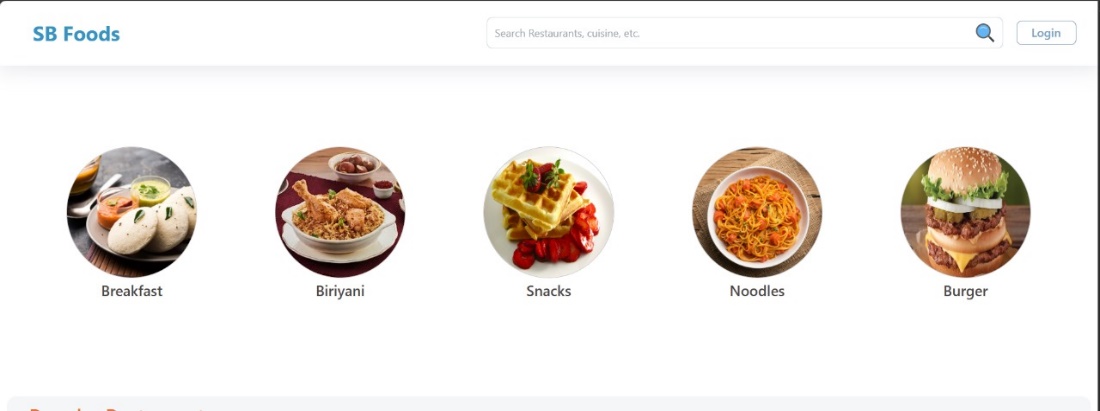
1.Register user:



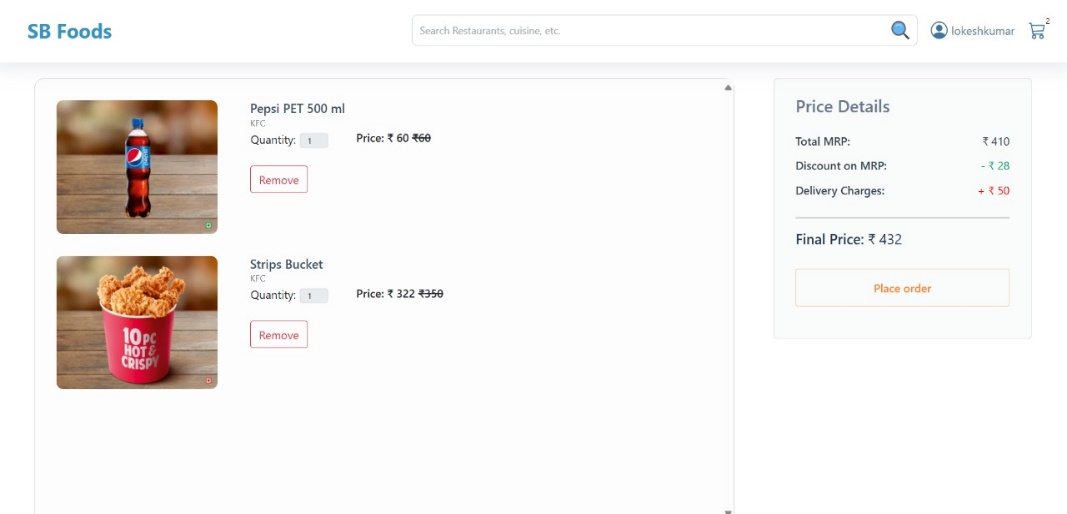
2. Login:



1. Food Menu:



1. Order:



5.payment:

