**Full Stack Development with MERN**

**Project Report**

**1. Introduction**

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

1. Gopika. A (Team Leader)
2. Yogasandhiya Devi. M
3. Priyatharsini. P
4. Venmathi. R

**2. Project Overview**

* **purpose :**

**The purpose of a food delivery app is to offer a convenient and efficient platform that enables users to browse menus from local restaurants, place orders, and have their food delivered directly to their doorstep.**

* **Features:**

**Enhanced Convenience:** Enable users to browse menus, place orders, and receive food deliveries without leaving their homes or workplaces.

**Improved Accessibility:** Provide a wide range of restaurant options and cuisines, allowing users to discover diverse food choices tailored to their preferences.

**Streamlined Delivery:** Ensure fast and dependable food delivery through a user-friendly interface with real-time order tracking.

**Support for Local Businesses:** Connect local restaurants with a larger customer base, helping to boost their sales and visibility.

**Personalized Experience:** Offer tailored recommendations, save user preferences, and provide promotions or loyalty rewards based on ordering patterns and preferences.

**3. Architecture**

**Frontend:**  
 Building a solid and scalable frontend architecture for a food delivery app using React is essential to ensure a smooth user experience. Below is an outline of key frontend components:

**1. Footer:**  
 Displays essential information about the app and includes links or details relevant to users, such as contact information, restaurant info, and app policies.

**2. Login:**  
 Allows users to access personalized services, including order history, saved payment methods, and customized settings.

**3. Navbar:**  
 A crucial navigation element that enables users to move between different sections of the app, such as home, menu, cart, and profile. It should be intuitive, easily accessible, and optimized for mobile devices.

**4. Popular Restaurants:**  
 Highlights a selection of popular restaurants within the app, making it easier for users to quickly discover popular choices.

**5. Register:**  
 Provides a user-friendly sign-up form where users can enter details like name, email, password, and additional information, such as phone number and address.

**6. Restaurants:**  
 Displays a list of available restaurants with details such as name, ratings, cuisine type, estimated delivery time, and price range. Users can view menus, place orders, and read reviews.

**Backend:**  
 Below is an overview of the backend architecture for a food delivery app using Node.js and Express.js, covering key components like API routes, database structure, authentication, and scalability considerations.

Below is an outline of essential backend components:

1. **Node Modules:**  
 Various Node.js modules (packages) are used to manage tasks such as routing, authentication, database operations, real-time updates, and payment integration.

2. **Environment Configuration (.env):**  
 Stores sensitive information and configuration variables such as API keys and database credentials, providing security and flexibility.

3. **Index.js:**  
 Serves as the entry point for the server, setting up routing, middleware, and essential configurations.

4. **Package-lock.json:**  
 Ensures consistent and deterministic dependency installations across environments by locking specific versions of dependencies.

5. **package.json:**  
 Serves as the project manifest, defining metadata, dependencies, scripts, and configuration details for the Node.js application.

6. **Schema.js:**  
 Defines the structure of the data model using an ORM like Sequelize (for SQL databases) or Mongoose (for MongoDB). The schema outlines data attributes, types, relationships, validation rules, and other constraints.

4. **Database Schema:**  
 In a food delivery application, the following entities are central to the database design:

1. **User:** Represents a customer or admin within the system.

2. **MenuItem:** Represents an individual food item available on a restaurant’s menu.

3. **Order:** Represents a customer’s order, which may include multiple menu items.

4. **Payment:** Represents the payment details for an order.

5. **Cart (Optional):** Represents items a user has added to their cart before placing an order

5. **Folder Structure**

**Client:** The frontend structure for the food delivery app, built with React, includes following components:

1. **Footer:** Displays food-related details and app-specific information.

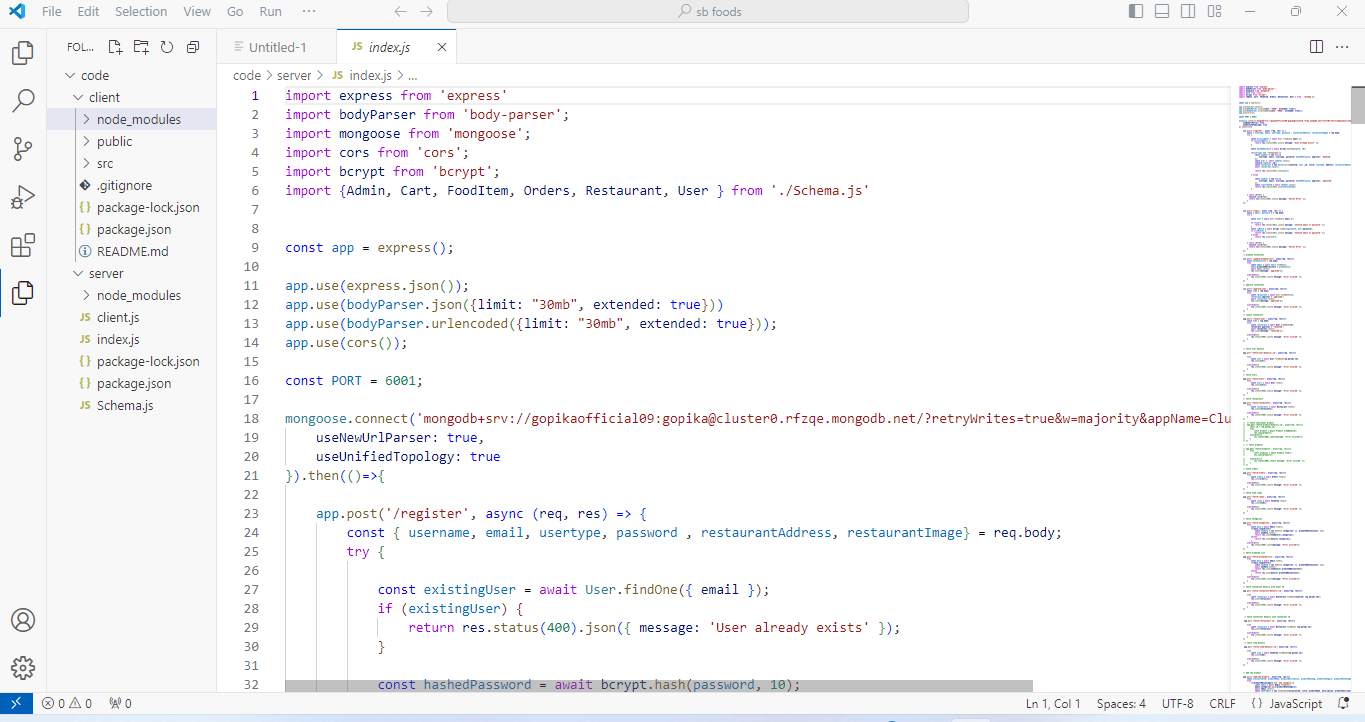
2. **Login:** Manages user access to personalized services.

3. **Navbar:** Provides easy navigation across sections like home, menu, cart, and profile, with a mobile-friendly design.

4. **Popular Restaurants:** Highlights popular dining options for easy discovery.

5. **Register:** A sign-up form where users enter essential account information.

6. **Restaurants Listing:** Displays available restaurants, allowing users to filter by name, rating, cuisine, delivery time, and price. Users can view detailed menus, place orders, and read reviews, enhancing their ordering experience.



**Server**: Below is an outline of the backend architecture for a food ordering app built with Node.js and Express.js. It includes the key components such as API routes, database structure, authentication mechanisms, and considerations for scalability.

1. **Node Modules**:  
 In a food ordering app built with Node.js and Express.js, you'll need various modules (or packages) to handle tasks like routing, authentication, database management, real-time communication, and payment integration. Common modules include:

* + **express** for handling HTTP requests and routing.
  + **mongoose** (for MongoDB) or **sequelize** (for SQL) for database interaction.
  + **jsonwebtoken** for authentication.
  + **dotenv** for managing environment variables.
  + **cors** for cross-origin resource sharing.

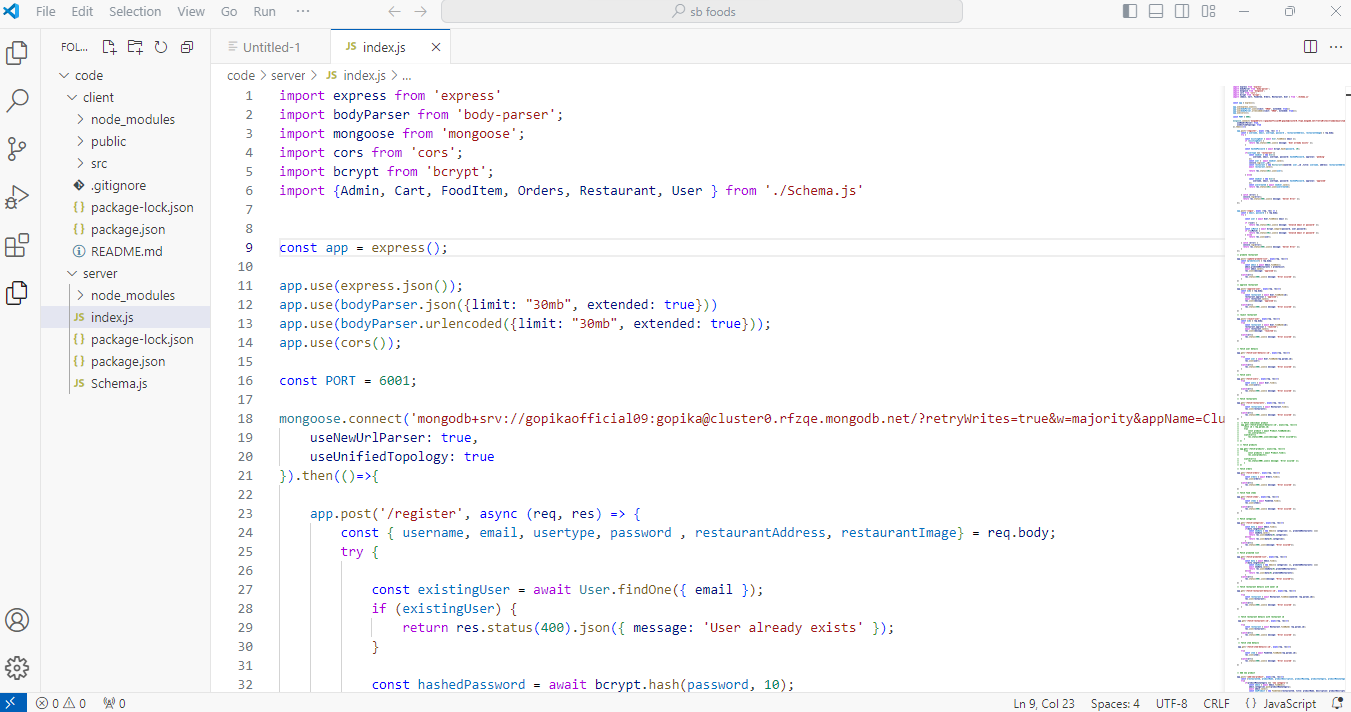
2. .**env File**:  
This file stores environment variables, such as database credentials, API keys, and other configuration settings, to keep sensitive information secure and separate from the codebase.

3. **index.js**:  
The entry point for your server. This file sets up the Express app, connects middleware, defines routes, and handles the server’s initialization.

4. **package-lock.json**:  
This file is generated automatically when you first install dependencies via npm install. It ensures that all developers and environments use the same versions of dependencies, making the app's setup consistent across systems.

5. **package.json**:  
This file is the heart of any Node.js project, listing the project’s metadata, dependencies, and scripts. It allows you to manage and install required packages using npm.

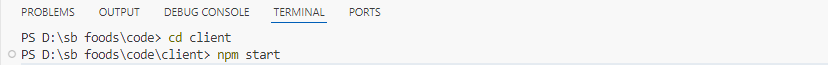
6. **schema.js**:  
Defines the structure of your database models. Using an ORM like Sequelize (for SQL) or Mongoose (for MongoDB), this file outlines the fields, data types, relationships, validation rules, and constraints of entities like **User**, **Order**, **MenuItem**, **Payment**, and **Cart**.



6**. Running the Application:**

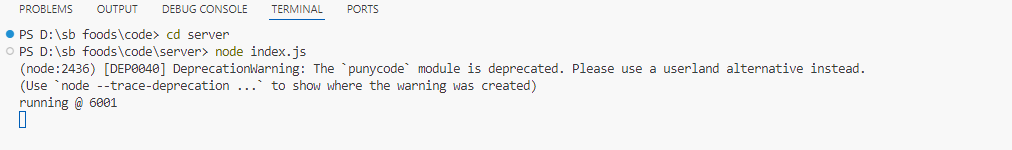
1. **Client:**

**Command:**



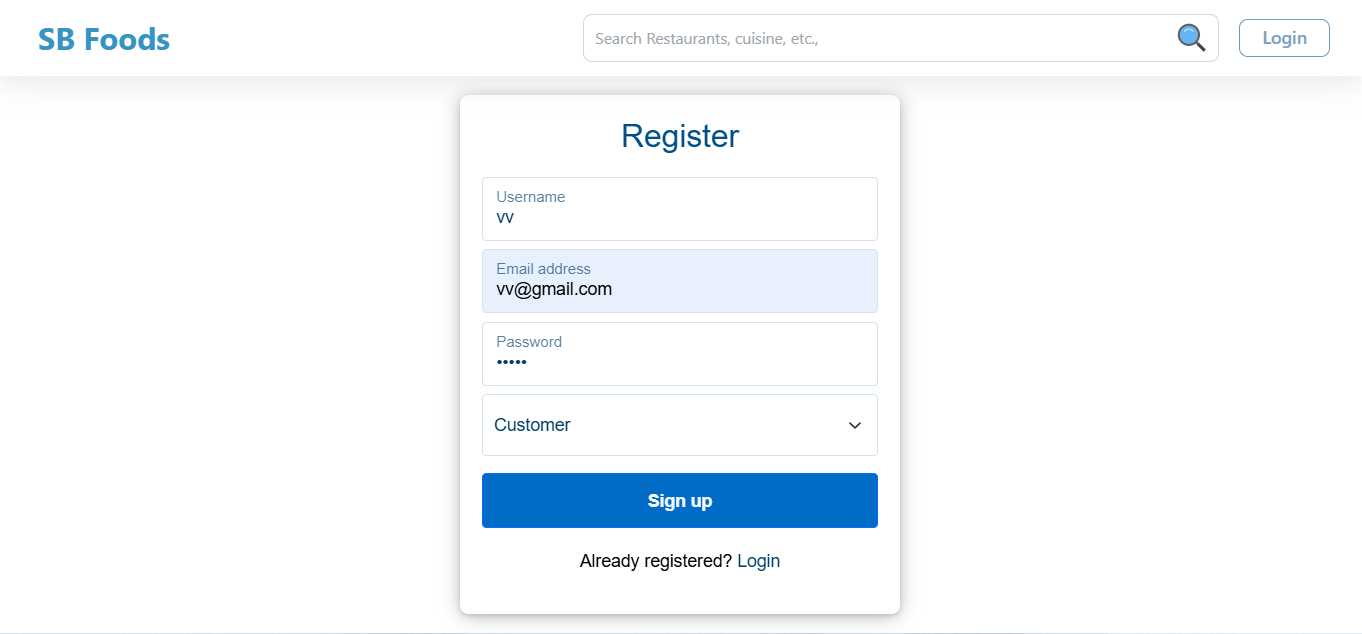
1. **Server:**

**Command:**

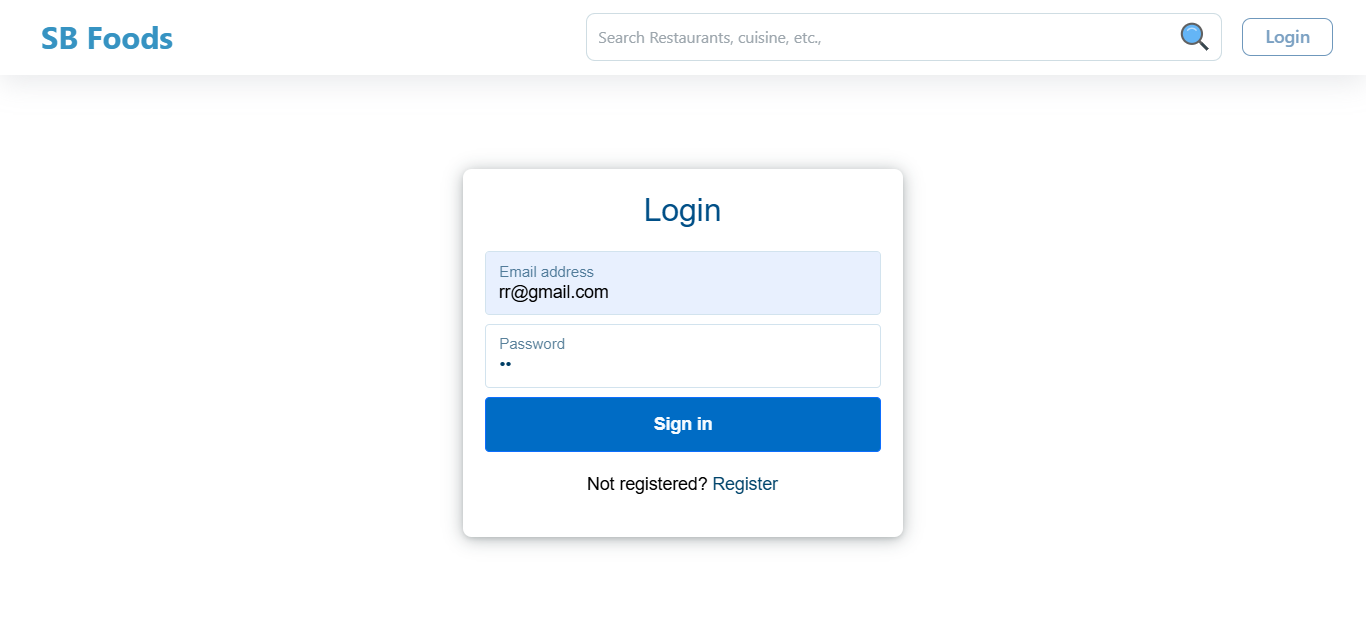
****

7 **.API Documentation:**

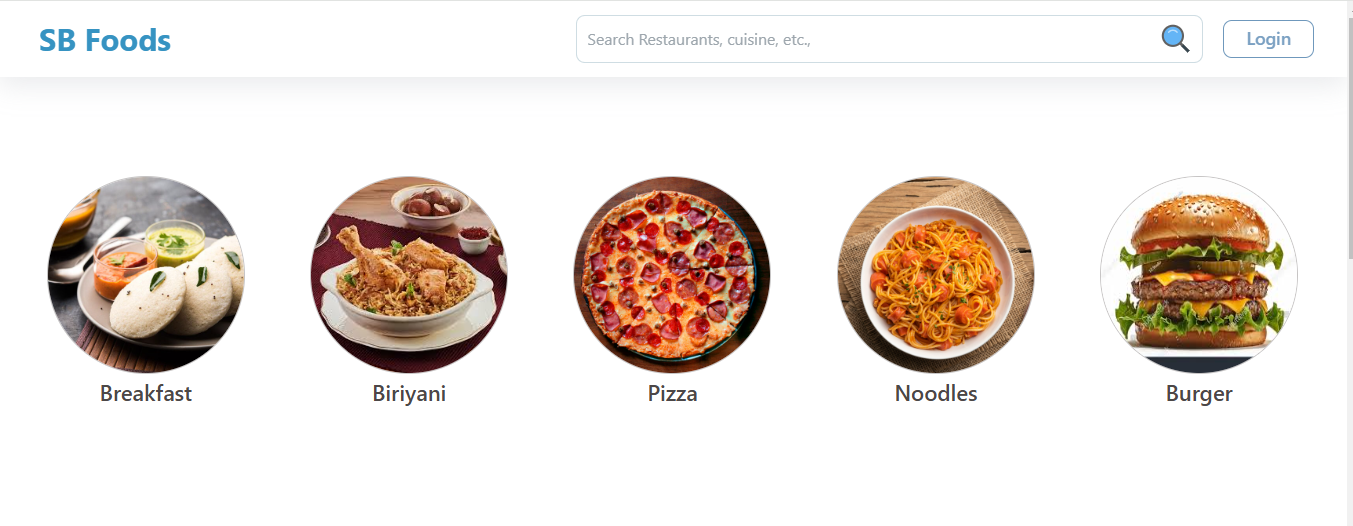
1**.Register user:**

****

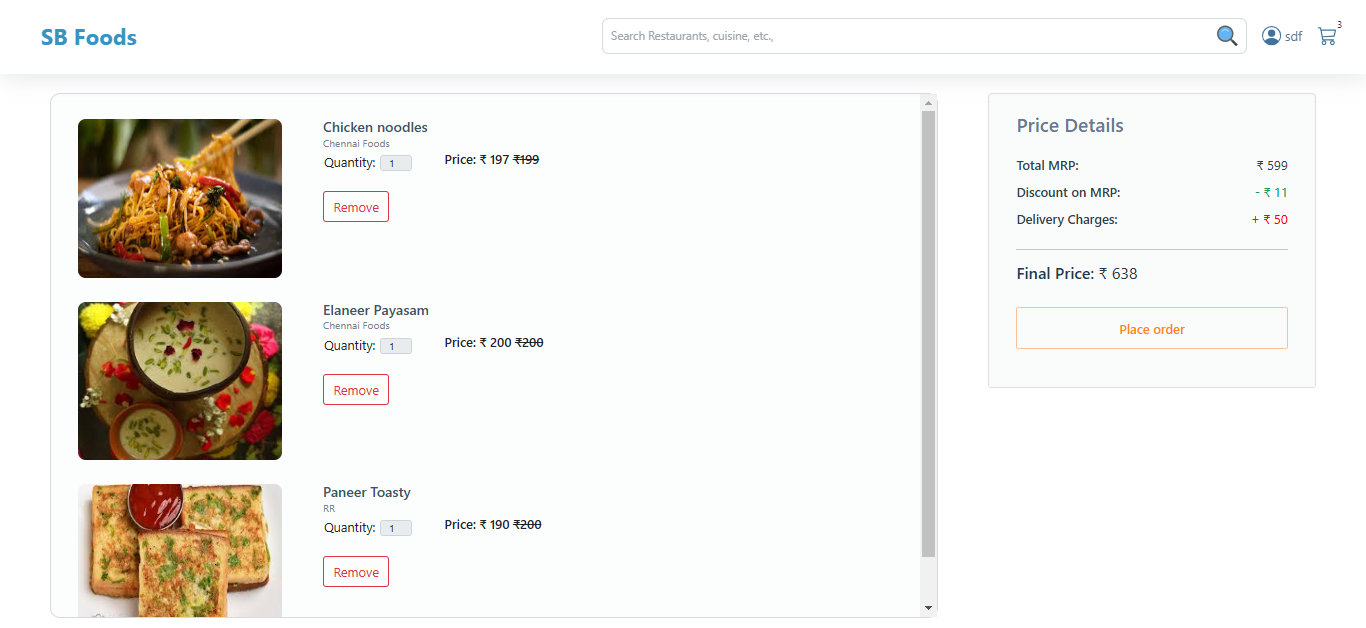
2. **Login:**

****

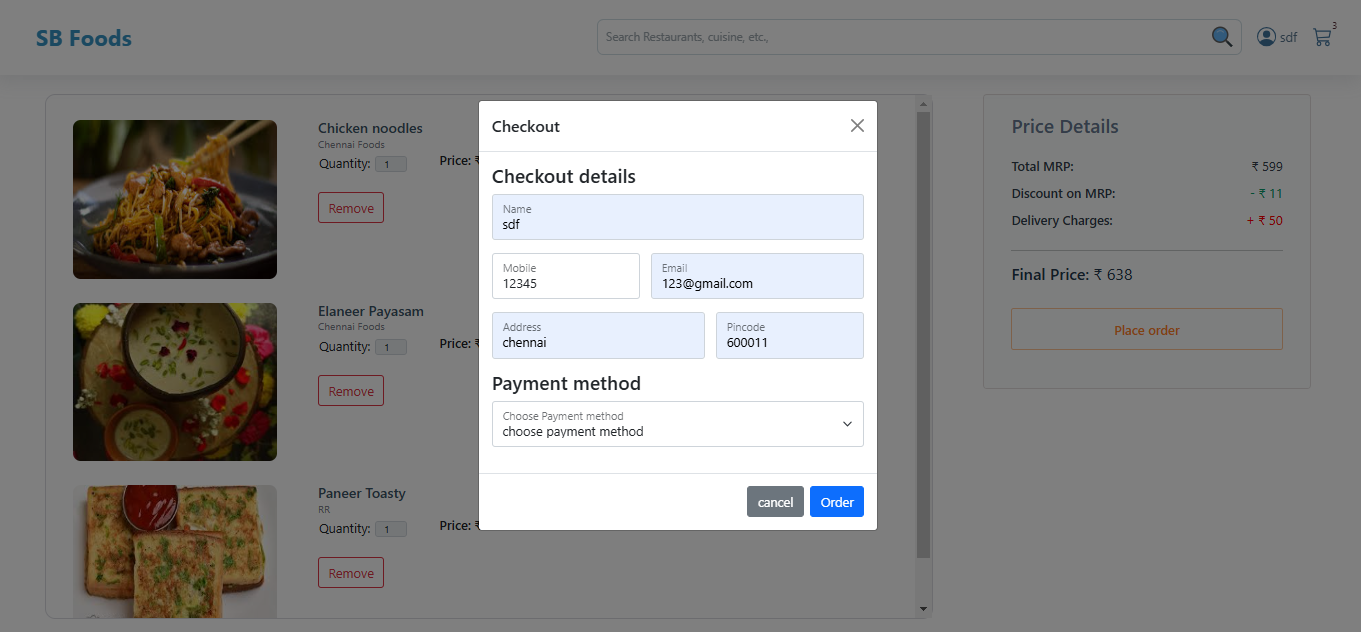
3. **Food Menu:**

****

4. **Order:**

****

5.**payment:**

****