**NAAN MUDHALVAN PROJECT**



**PROJECT TITLE**

**SB Foods - Food Ordering App**

**Submitted by team members:**

**SYED SIDDIQ C – 311421205095**

**SAMEERUDIN S – 311421205303**

**GOWTHAM S - 311421205301**

**VINOTH L – 311421205304**



**MEENAKSHI COLLEGE OF ENGINEERING**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**12, VEMBULIAMMAN KOVIL STREET, WEST K.K. NAGAR, CHENNAI – 600078, TAMILNADU,**

**(AFFILATED TO ANNA UNIVERSITY)**

SB Foods - Food Ordering App

**Hardware Required:**

Windows 8 machine

**Software Required:**

Install with two web browser

**System Required:**

Bandwidth of 30mbps

• Introducing SB Foods, the cutting-edge digital platform poised to revolutionize the way you order food online. With SB Foods, your food ordering experience will reach unparalleled levels of convenience and efficiency.

• Our user-friendly web app empowers foodies to effortlessly explore, discover, and order dishes tailored to their unique tastes. Whether you're a seasoned food enthusiast or an occasional diner, finding the perfect meals has never been more straightforward.

• Imagine having comprehensive details about each dish at your fingertips. From dish descriptions and customer reviews to pricing and available promotions, you'll have all the information you need to make well-informed choices. No more second-guessing or uncertainty – SB Foods ensures that every aspect of your online food ordering journey is crystal clear.

• The ordering process is a breeze. Just provide your name, delivery address, and preferred payment method, along with your desired dishes. Once you place your order, you'll receive an instant confirmation. No more waiting in long queues or dealing with complicated ordering processes – SB Foods streamlines it, making it quick and hassle-free.

**SCENARIO:**

**Late-Night Craving Resolution**

Meet Lisa, a college student burning the midnight oil to finish her assignment. As the clock strikes midnight, her stomach grumbles, reminding her that she skipped dinner. Lisa doesn't want to interrupt her workflow by cooking, nor does she have the energy to venture outside in search of food.

Solution with Food Ordering App:

1. Lisa opens the Food Ordering App on her smartphone and navigates to the late-night delivery section, where she finds a variety of eateries still open for orders.

2. She scrolls through the options, browsing menus and checking reviews until she spots her favorite local diner offering comfort food classics.

3. Lisa selects a hearty bowl of chicken noodle soup and a side of garlic bread, craving warmth and satisfaction in each bite.

4. With a few taps, she adds the items to her cart, specifies her delivery address, and chooses her preferred payment method.

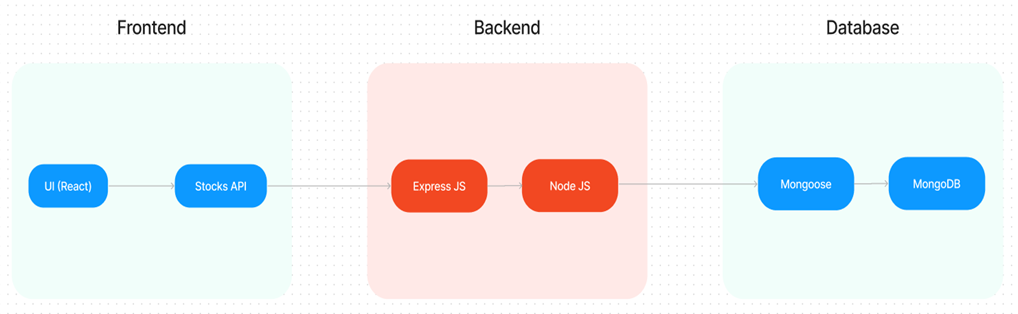
5. Lisa double-checks her order details on the confirmation page, ensuring everything looks correct, before tapping the "Place Order" button.

6. Within minutes, she receives a notification confirming her order and estimated delivery time, allowing her to continue working with peace of mind.

7. As promised, the delivery arrives promptly at her doorstep, and Lisa eagerly digs into her piping hot meal, grateful for the convenience and comfort provided by the Food Ordering App during her late-night study session.

This scenario illustrates how a Food Ordering App caters to users' needs, even during unconventional hours, by offering a seamless and convenient solution for satisfying late-night cravings without compromising on quality or convenience.

**TECHNICAL ARCHITECTURE:**

****

In this architecture diagram:

•        The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart, Products, Profile, Admin dashboard, etc.,

•        The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc., It also includes Admin Authentication and an Admin Dashboard.

•        The Database section represents the database that stores collections for Users, Admin, Cart, Orders, and products.

**ABSTRACT:**

This project presents a comprehensive **SB FOOD APPLICATION** developed using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js), designed to enhance the booking experience . The application leverages the Model-View-Controller (MVC) architecture to establish a clear separation between data management, user interface, and core business logic, ensuring a scalable, maintainable, and high-performing solution.

In today’s digital era, user-friendly, responsive, and scalable web applications are essential in sectors like travel and tourism. This project presents a robust food application developed using the MERN stack, with a strong focus on providing a seamless experience for service providers. Through the Model-View-Controller (MVC) architecture, our application separates data management, user interaction, and core business logic, enhancing maintainability and scalability.

Our project is dedicated to addressing the significant challenges faced by common people ordering late night, service providers by creating a comprehensive, integrated platform using the MERN stack (MongoDB, Express.js, React.js, Node.js). This platform is designed to facilitate seamless interactions between customers and various owners within the cooking industry, aiming to enhance the overall food experience and streamline service delivery.

Our focus on a responsive user interface extends across devices, ensuring that the platform is accessible on both desktop and mobile, so customers can conveniently taste their desired food even at midnight from anywhere. The platform design emphasizes ease of use, providing clear navigation, robust search features, and easy access to essential information. Through careful attention to design and user experience, we enable customers to efficiently find services, make informed decisions, and customize their food. The platform also offers service providers tools to manage listings, interact with customers, and analyses feedback to improve their offerings.

**GANTT CHART:**

A Gantt chart is a visual project management tool that provides a timeline for project tasks, showing the start and end dates of each task, the sequence of tasks, and how they relate to one another. It is widely used to plan, schedule, and track project progress. The chart typically consists of horizontal bars that represent the duration of tasks or activities in a project, displayed along a time scale.

Creating a **Gantt chart** for the development of a **Food App** involves breaking down the project into specific phases and tasks, assigning timelines, and showing dependencies between tasks. Below is an example of a Gantt chart that might be used for the development of a food delivery or restaurant app. This chart covers common stages such as requirement gathering, design, development, testing, and deployment.

**Gantt Chart for Food App Development**

| **Phase/Task** | **Duration** | **Dependencies** | **Start Date** | **End Date** |
| --- | --- | --- | --- | --- |
| **1. Requirement Gathering** | 2 weeks | None | 01/01/2024 | 14/01/2024 |
| - User research | 5 days | Requirement Gathering | 01/01/2024 | 05/01/2024 |
| - Feature list & specifications | 5 days | Requirement Gathering | 06/01/2024 | 10/01/2024 |
| - Technology stack selection | 4 days | Requirement Gathering | 11/01/2024 | 14/01/2024 |
| **2. UI/UX Design** | 3 weeks | Requirement Gathering | 15/01/2024 | 04/02/2024 |
| - Wireframing & Mockups | 1 week | UI/UX Design | 15/01/2024 | 21/01/2024 |
| - User Flow Design | 1 week | Wireframing & Mockups | 22/01/2024 | 28/01/2024 |
| - High-fidelity UI Design | 1 week | User Flow Design | 29/01/2024 | 04/02/2024 |
| **3. Database & Backend Development** | 6 weeks | Requirement Gathering | 05/02/2024 | 15/03/2024 |
| - Database Design | 1 week | Requirement Gathering | 05/02/2024 | 11/02/2024 |
| - API Development | 4 weeks | Database Design | 12/02/2024 | 11/03/2024 |
| - User Authentication (Login, SignUp) | 1 week | API Development | 12/03/2024 | 18/03/2024 |
| **4. Frontend Development** | 6 weeks | UI/UX Design | 05/02/2024 | 15/03/2024 |
| - Home Screen & Navigation | 2 weeks | UI/UX Design | 05/02/2024 | 18/02/2024 |
| - Product/Restaurant Listings | 2 weeks | Home Screen & Navigation | 19/02/2024 | 03/03/2024 |
| - Order Placement & Cart | 1 week | Product Listings | 04/03/2024 | 10/03/2024 |
| - Checkout & Payment Integration | 1 week | Order Placement & Cart | 11/03/2024 | 17/03/2024 |
| **5. Integration** | 2 weeks | Backend Development, Frontend Development | 16/03/2024 | 29/03/2024 |
| - Backend & Frontend Integration | 2 weeks | Backend Development, Frontend Development | 16/03/2024 | 29/03/2024 |
| **6. Testing & QA** | 3 weeks | Integration | 30/03/2024 | 19/04/2024 |
| - Unit Testing | 1 week | Integration | 30/03/2024 | 05/04/2024 |
| - Functional Testing | 1 week | Unit Testing | 06/04/2024 | 12/04/2024 |
| - User Acceptance Testing (UAT) | 1 week | Functional Testing | 13/04/2024 | 19/04/2024 |
| **7. Deployment & Go Live** | 1 week | Testing & QA | 20/04/2024 | 26/04/2024 |
| - Final Deployment to Production | 3 days | Testing & QA | 20/04/2024 | 22/04/2024 |
| - App Store/Play Store Submission | 3 days | Final Deployment | 23/04/2024 | 25/04/2024 |
| - Go Live & Monitoring | 1 day | App Store/Play Store Submission | 26/04/2024 | 26/04/2024 |

**In Summary:**

A Gantt chart for the SB Food application development would help in organizing, scheduling, and tracking progress at every stage. It serves as both a planning and analytical tool, allowing you to assess how the project unfolds over time, identify delays or issues, and make adjustments for smoother implementation. By visually breaking down the tasks and timelines, you can ensure efficient management of resources, timelines, and dependencies, ultimately leading to the successful development and deployment of the application.

How a Gantt Chart Focuses on Analyzing the Overall Implementation for Developing the SB Food Application

In the context of developing an SB Food application, the Gantt chart can be highly effective for both planning and analyzing the overall implementation process. Here’s how it can help:

**1. Task Breakdown and Scheduling**

A Gantt chart helps break down the entire development process of the SB Food application into manageable tasks or milestones. For instance:

* Requirement Analysis
* Design Phase (UI/UX)
* Database Development
* Backend Development
* Frontend Development
* Integration
* Testing & Debugging
* Deployment

Each of these tasks will be assigned a timeline and resources, with dependencies between tasks clearly shown. For example, the "Backend Development" task might depend on the completion of "Database Development," and "Testing" might depend on the completion of both "Frontend" and "Backend" development.

**2.** **Monitoring Progress**

By visualizing the start and end dates of each task, the Gantt chart enables project managers and developers to track progress. If a particular task is delayed, it will immediately show how that affects the overall timeline. This helps in identifying potential bottlenecks or areas where the project is lagging behind and requires attention.

**3. Resource Allocation**

For the SB Food application, multiple teams or individuals may be working on different parts of the project (e.g., front-end developers, back-end developers, database designers, quality assurance teams). The Gantt chart helps ensure that resources are allocated efficiently and that no team is overburdened or underutilized. It can also highlight resource dependencies and the need for collaboration between different teams at various stages of the project.

**4. Risk and Dependency Management**

The Gantt chart identifies task dependencies, so project managers can anticipate how delays in one task might affect others. For example, if the "Design" phase takes longer than expected, it may delay the start of "Frontend Development" or "User Testing." By highlighting these dependencies, project managers can take proactive measures, such as adjusting timelines or reassigning resources, to mitigate risks.

**5. Timeline Visualization**

Since a Gantt chart provides a visual timeline, it helps stakeholders, including project managers, developers, and clients, to quickly understand when key milestones in the development of the SB Food app will be achieved. This can be particularly useful in meetings to give updates on progress, and also for adjusting expectations if there are delays or changes to the timeline.

**6. Post-Implementation Analysis**

After the implementation phase is completed, the Gantt chart can serve as a retrospective tool for analyzing the entire development process of the SB Food application. It allows the team to review whether the project stayed on schedule, whether resources were efficiently utilized, and where improvements can be made for future projects. This analysis helps identify areas where processes can be streamlined and potential pitfalls to avoid.

**WHAT IS MVC AND WHY ITS IMPORTANT IN THE APPLICATION WE ARE GONNA DEVELOP:**

**Model-View-Controller (MVC)** is a design pattern widely used in software development, particularly in web applications. It divides an application into three interconnected components, each responsible for a distinct aspect of the application’s functionality. This separation helps manage the complexity of the application, improve maintainability, and allow developers to work more efficiently by focusing on one aspect of the application at a time.

**1. Model**

* **Definition**: The **Model** is responsible for managing the data and business logic of the application. It defines how data is stored, retrieved, and manipulated, and it represents the "real-world" entities in the application.
* **Role**: It communicates with the database, applies business rules, and manages the data structure.

**2. View**

* **Definition**: The **View** is the presentation layer of the application. It is responsible for displaying the data to the user and defining the structure and layout of the UI.
* **Role**: The View interacts with the user and reflects the data provided by the Model.

**3. Controller**

* **Definition**: The **Controller** acts as the intermediary between the Model and View. It processes user inputs, communicates with the Model to fetch or update data, and determines which View to display.
* **Role**: The Controller interprets user inputs from the View, updates the Model, and then selects the appropriate View for the user.

**HOW THE MVC ENHANCES THE OVERALL FUNCTIONING OF OUR APPLICATION?**

Since our application is built using React for the frontend and Node.js for the backend, it follows the Model-View-Controller (MVC) design pattern to organize the code effectively and improve functionality.

**Model:** We use MongoDB as our database to store and manage information such as flight data, user profiles, and booking details. MongoDB’s flexible, schema-less structure allows us to store data in a variety of formats, making it adaptable and scalable. The Model layer handles all database interactions, providing a seamless and optimized data experience for the application.

**View:** Our React frontend serves as the View layer, creating an interactive and responsive user interface for users to explore foods, exploring restaurants, and view details. React' s component-based design enables a modular approach, ensuring that the UI is engaging and easy to update. This layer solely focuses on the presentation and does not handle data directly, making it easier to maintain and scale the UI independently.

**Controller:** The Node.js and Express backend functions as the Controller, connecting the user interface with the database. It processes user requests, communicates with the Model to retrieve or update data as needed, and sends responses to the View to update the display accordingly. This layer handles the business logic, ensuring data flows smoothly between the frontend and the backend.

**By following the MVC pattern, we’ve created a structure that:**

1. Separates concerns between data management, UI presentation, and business logic:

By isolating the different concerns (data, UI, and user input), MVC makes your codebase more organized and easier to maintain. Changes to one part of the system (e.g., updating the UI) can be made without affecting the business logic or data processing.

1. Enhances the app’s scalability and maintainability, making it easy to introduce new features:

The modularity of MVC allows the application to grow over time by adding new features or supporting more complex interactions without significant refactoring. Each part can be worked on independently.

1. Provides a responsive and efficient user experience, with a backend that ensures data consistency and security:

MVC allows developers to maintain a clear structure where each component is responsible for a specific part of the functionality, making it easier to fix bugs, implement changes, or add new features.

**WHAT IS SB FOOD APP? REASON FOR CHOOSING IT?**

Choosing to develop a SB Food Ordering App (or any food ordering app) for a project can be motivated by several factors, ranging from addressing market needs to leveraging emerging technologies. Below are some compelling reasons why a food ordering app could be a great choice for a project:

1. High Market Demand and Popularity

Food delivery and ordering apps have become a major part of the global market due to the growing demand for convenience. With busy lifestyles, people are increasingly opting for online food ordering rather than cooking or dining out. Major players like UberEats, DoorDash, and Grubhub have demonstrated the viability of this business model, and a smaller or niche app like SB Food can cater to specific needs, such as local restaurants, healthier food options, or even particular cuisines.

* Trend Analysis: According to various market research reports, the food delivery market has seen consistent growth, with projections indicating continued growth in the coming years.
* Consumer Behavior: People prefer the convenience of ordering food from home, with real-time tracking, contactless payments, and wide variety.

2. Opportunity for Innovation

The food delivery industry is competitive, but there are still areas for innovation:

* Personalized Recommendations: Integrating AI-based recommendations based on user preferences, order history, or even dietary restrictions (e.g., vegan, gluten-free).
* Local Restaurant Focus: The app could focus on smaller or independent restaurants, providing a platform for local businesses to compete with larger chains.
* Sustainability Features: Offering options for sustainable packaging, eco-friendly delivery methods, or promoting healthier eating habits could set the app apart.
* Multi-Platform Integration: You could integrate SB Food with other services (e.g., smart home assistants or IoT devices) for a seamless food ordering experience.

3. Hands-On Learning Experience

Developing a food ordering app involves a diverse set of skills, making it an ideal project for learning and applying multiple concepts:

* Mobile App Development: Building both the front-end and back-end of the app for iOS and Android platforms.
* Database Design: Creating a robust database to manage user data, orders, payments, and inventory.
* UI/UX Design: Designing an intuitive user interface for a smooth ordering experience.
* Payment Gateway Integration: Implementing secure payment solutions for transactions.
* Real-Time Features: Integrating GPS for delivery tracking or notifications for order status updates.

4. User-Centric and Community Engagement

A food ordering app like SB Food can foster a sense of community by allowing users to discover new local restaurants, explore new food options, and engage with food reviews, ratings, and recommendations. Additionally, apps can provide a personalized experience that builds customer loyalty through rewards, promo codes, and subscription-based models.

* User Engagement: Features like ratings, reviews, and personalized offers can keep customers coming back.
* Customer Retention: Loyalty programs and special deals (e.g., free delivery, discounts, etc.) can improve retention rates.

5. Monetization Potential

A food ordering app can generate revenue in multiple ways:

* Commissions: The app can charge a percentage of each order from the restaurants.
* Delivery Fees: Charging users a fee for delivery services.
* Premium Features: Offering premium services for users (e.g., express delivery, access to exclusive restaurants).
* Advertising: Local businesses or restaurants can advertise on the platform to reach a larger audience.

These monetization strategies provide flexibility in business models and growth opportunities.

6. Real-World Relevance

Food is a universal need, and the app can address real-world problems such as:

* Convenience: Users can save time by ordering food with just a few taps.
* Accessibility: Bringing the best local food to users' doorsteps, particularly in areas where delivery options are limited.
* Inclusivity: Serving various dietary needs (e.g., vegan, vegetarian, low-carb) with custom options for a wide range of users.

7. Scalability and Flexibility

Once you’ve built the initial version of the app, SB Food can easily scale to serve larger markets, expand to multiple cities, or incorporate additional services, such as:

* Subscription Plans: Offering subscription models for frequent users (e.g., unlimited free deliveries, member-only discounts).
* International Expansion: If the app succeeds locally, you could scale the app to other countries, adapting to local food preferences and market conditions.
* Partnerships: Partnering with grocery stores, meal kits, or even fitness programs to expand the range of food-related services offered.

8. Technical Skill Development

Building an app like SB Food allows you to work with several important technologies:

* Backend Development: Managing servers, databases, user authentication, order processing, etc.
* Frontend Development: Ensuring the user interface is engaging, intuitive, and functional across mobile devices.
* Cloud Services: Utilizing cloud services for scalability (e.g., AWS, Google Cloud, etc.) to store and manage large amounts of data, ensuring the app can handle many users and orders simultaneously.
* Geolocation Services: Implementing GPS features for real-time delivery tracking.

9. Flexibility in Features

You can decide which features to include in the app based on your objectives. Some options are:

* Real-Time Order Tracking: Let users see the progress of their order, from preparation to delivery.
* Customizable Orders: Allow users to modify their orders (e.g., add or remove ingredients, change the size).
* Push Notifications: Notify users about special offers, order status, or promotional discounts.
* Payment Integration: Support for multiple payment options such as credit/debit cards, digital wallets, and cash on delivery.

Summary: Why Choose SB Food Ordering App for a Project?

* Relevance and Market Demand: The growing popularity of food delivery services makes it a relevant and in-demand project.
* Skill Development: You will gain experience in mobile app development, UI/UX design, backend/server-side development, and integrating third-party services (e.g., payment gateways, GPS).
* Opportunity for Innovation: There’s potential to introduce unique features that differentiate SB Food from competitors.
* Monetization and Business Viability: Multiple revenue streams such as commission-based models, delivery fees, and advertising provide strong business potential.
* Real-World Impact: Addressing the everyday needs of consumers and local businesses in a user-friendly, scalable app.

Overall, a food ordering app like SB Food offers both technical challenges and real-world relevance, making it a rewarding and practical project for students, developers, or entrepreneurs interested in app development and the food tech industry.

To get started with the **frontend development** of your **SB Food Ordering App**, you'll need to set up the development environment and install the required dependencies. The installation process will depend on the framework or tools you decide to use for the frontend. Below, I'll provide installation commands for a few popular frontend technologies that are commonly used in food ordering apps.

**Frontend Technologies & Installation Commands**

**1. React (ReactJS) – For building a modern, responsive web app.**

React is a popular JavaScript library for building user interfaces, especially for single-page applications (SPAs).

**Steps to install React:**

1. **Install Node.js**  
   Make sure you have **Node.js** installed. If not, download it from the official site: <https://nodejs.org/>.
2. **Create a New React App**  
   You can use **Create React App**, a tool that sets up everything for you (webpack, Babel, etc.), so you can focus on coding.
   * Open your terminal or command prompt and run:

bash

Copy code

npx create-react-app sb-food-app

This will create a new React app called sb-food-app in a directory with the same name.

1. **Navigate to your project folder:**

bash

Copy code

cd sb-food-app

1. **Start the development server:**

bash

Copy code

npm start

Your app should now be running at http://localhost:3000/.

**2. Vue.js – Lightweight and flexible framework for building frontend applications.**

Vue.js is a progressive JavaScript framework, designed for building user interfaces and single-page applications.

**Steps to install Vue.js:**

1. **Install Vue CLI**  
   First, you need to install Vue CLI globally. If you haven’t installed Vue CLI yet, use the following command:

bash

Copy code

npm install -g @vue/cli

1. **Create a New Vue Project**  
   Run the following command to create a new project named sb-food-app:

bash

Copy code

vue create sb-food-app

During the setup, you'll be asked to pick preset options. You can go with the default ones or manually configure Vue Router and Vuex if needed.

1. **Navigate to your project folder:**

bash

Copy code

cd sb-food-app

1. **Start the development server:**

bash

Copy code

npm run serve

Your app should be running at http://localhost:8080/.

**3. Angular – A full-featured framework for building robust web applications.**

Angular is a comprehensive frontend framework that provides tools for building large-scale applications with routing, state management, and more.

**Steps to install Angular:**

1. **Install Angular CLI**  
   If you don’t have Angular CLI installed, use the following command:

bash

Copy code

npm install -g @angular/cli

1. **Create a New Angular Project**  
   Use the CLI to generate a new project named sb-food-app:

bash

Copy code

ng new sb-food-app

This will prompt you to choose options like adding Angular routing and stylesheets (CSS, SCSS, etc.).

1. **Navigate to your project folder:**

bash

Copy code

cd sb-food-app

1. **Start the development server:**

bash

Copy code

ng serve

Your app should now be running at http://localhost:4200/.

**4. React Native – For building a mobile app (iOS and Android).**

If you're building a **mobile app** for SB Food, React Native is a great choice because it allows you to write once and deploy to both **iOS** and **Android**.

**Steps to install React Native:**

1. **Install Node.js and Watchman**
   * Make sure you have **Node.js** installed.
   * **Watchman** is required by React Native to watch file changes. Install it using Homebrew on macOS:

bash

Copy code

brew install watchman

1. **Install React Native CLI**  
   Use npm to install the React Native CLI:

bash

Copy code

npm install -g react-native-cli

1. **Create a New React Native Project**  
   Run the following command to create a new React Native project:

bash

Copy code

npx react-native init SbFoodApp

1. **Navigate to your project folder:**

bash

Copy code

cd SbFoodApp

1. **Start the Metro Bundler and run the app:** For **iOS**:

bash

Copy code

npx react-native run-ios

For **Android**:

bash

Copy code

npx react-native run-android

Your app should open in the emulator or on your connected device.

**Dependencies and Libraries**

Once you’ve set up your frontend project, you might want to install additional **dependencies** (e.g., for routing, state management, UI libraries, etc.):

1. **React Router (for React):**

bash

Copy code

npm install react-router-dom

1. **Axios (for making API requests):**

bash

Copy code

npm install axios

1. **Redux (for state management in React or React Native):**

bash

Copy code

npm install redux react-redux

1. **Vuetify (for Material Design UI in Vue.js):**

bash

Copy code

vue add vuetify

1. **Angular Material (for Angular UI components):**

bash

Copy code

ng add @angular/material

1. **Styled Components (for styled components in React):**

Bash

A **workflow diagram** for the **SB Food Ordering App** outlines the sequential flow of processes and interactions between different components of the system, such as users, servers, databases, and external services. The workflow diagram typically represents the **steps** involved in activities like order placement, payment, delivery tracking, and user management.

**Key Workflow Processes for SB Food Ordering App**

The primary workflow processes in a food ordering app like SB Food include:

1. **User Registration/Login**
2. **Browse Menu and Add Items to Cart**
3. **Checkout and Payment**
4. **Order Confirmation**
5. **Order Processing (Restaurant)**
6. **Order Delivery**
7. **Feedback and Rating**

**Example of the Workflow Diagram:**

**1. User Registration/Login Process:**

This flow allows the user to sign up or log in.

* **User** → *Registration/Login Page*
  + Enter credentials (username/password or social login)
* *App Server* → Validate credentials with database.
  + If valid → Redirect to the home screen (Browse Menu).
  + If invalid → Show error message.

**2. Browse Menu and Add Items to Cart:**

Users browse the menu and add items to the cart.

* **User** → *App Interface*
  + Browse menu categories (e.g., appetizers, main dishes, drinks).
  + Add items to cart.
  + Optionally, modify the quantity or remove items.
* *App Server* → Store items in the user's cart (session or database).

**3. Checkout and Payment:**

Once the user is ready to place an order, they proceed to checkout.

* **User** → *Checkout Page*
  + Review items in the cart.
  + Provide delivery address.
  + Select payment method (credit card, PayPal, etc.).
  + Apply promo codes if any.
* *App Server* → Process payment via third-party payment gateway (Stripe, PayPal).
  + If successful → Proceed to confirmation page.
  + If failed → Show error message and allow retry.

**4. Order Confirmation:**

After successful payment, the order is confirmed.

* **User** → View order summary and confirmation.
* *App Server* → Generate an order ID.
  + Save order details in the database (user, items, price, delivery address).
* **Restaurant** → Receive order notification (via mobile app or web interface).
  + Start preparing the order.

**5. Order Processing (Restaurant):**

The restaurant prepares the food.

* **Restaurant Staff** → View the order in the restaurant management system.
  + Start preparing the order.
* *App Server* → Update order status in real-time (e.g., "Preparing", "Out for delivery").

**6. Order Delivery:**

Once the food is ready, the order is assigned to a delivery driver.

* **App Server** → Notify the delivery driver.
  + Assign driver via GPS integration or app notification.
* **Delivery Driver** → Pick up the order from the restaurant and deliver it to the user's address.
* **User** → Track the delivery status in real-time (via app).

**7. Feedback and Rating:**

After receiving the order, users can rate the restaurant and delivery service.

* **User** → Provide feedback/rating for the restaurant and delivery experience.
* ***App Server*** → Store ratings and feedback in the database.
  + Optionally, send feedback to the restaurant and delivery drive.

Here’s an overview of an ER (Entity-Relationship) Diagram for an SB Food App. This app allows users to browse food items, place orders, make payments, and receive deliveries. Each entity and relationship is described below.

**Entities and Relationships**

**1. User**

* **Attributes**:
  + UserID (Primary Key)
  + Name
  + Email
  + Phone
  + Address
  + Password
* **Relationships**:
  + Places **Orders**
  + Views **Food Items**
  + Makes **Payments**

**2. Food Item**

* **Attributes**:
  + FoodItemID (Primary Key)
  + Name
  + Description
  + Price
  + Image
* **Relationships**:
  + Belongs to a **Category**
  + Can be part of multiple **Order Items**

**3. Category**

* **Attributes**:
  + CategoryID (Primary Key)
  + Name
* **Relationships**:
  + Contains multiple **Food Items**

**4. Order**

* **Attributes**:
  + OrderID (Primary Key)
  + OrderDate
  + Status (e.g., Pending, Completed, Canceled)
  + TotalAmount
* **Relationships**:
  + Placed by a **User**
  + Contains multiple **Order Items**

**5. Order Item**

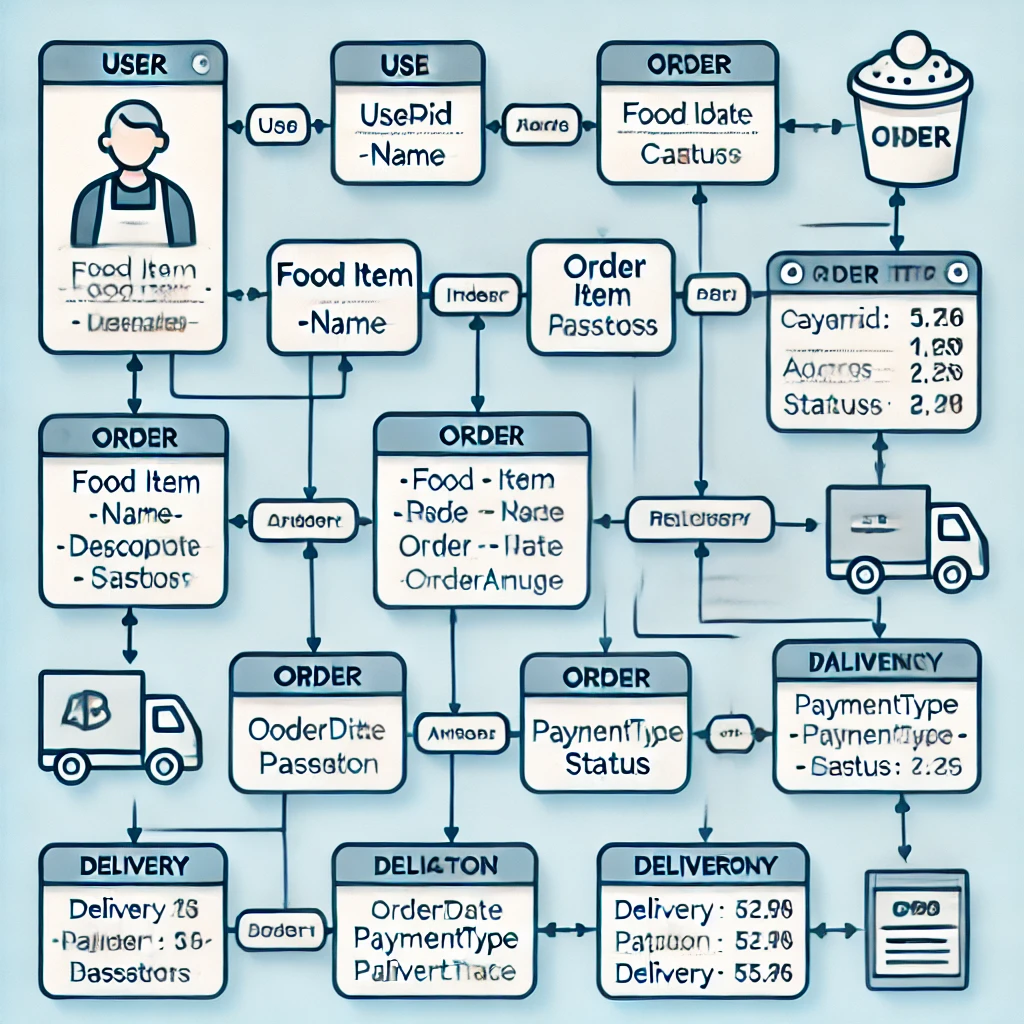
* **Attributes**:
  + OrderItemID (Primary Key)
  + Quantity
* **Relationships**:
  + Associated with a single **Order**
  + Refers to a specific **Food Item**

**6. Payment**

* **Attributes**:
  + PaymentID (Primary Key)
  + Amount
  + PaymentDate
  + PaymentType (e.g., Credit Card, PayPal)
  + Status (e.g., Paid, Pending, Failed)
* **Relationships**:
  + Associated with a single **Order**
  + Made by a **User**

**7. Delivery**

* **Attributes**:
  + DeliveryID (Primary Key)
  + DeliveryDate
  + Status (e.g., In Progress, Delivered, Canceled)
* **Relationships**:
  + Linked to an **Order**



**APPLICATION FLOW GIVEN BELOW**:



Here is the application flow image for the SB Food App, which shows a streamlined user journey through each stage of the app experience.

**Application Flow Outline:**

1. **User Login or Sign-up**
   * Users log in if they already have an account or create one to access the app.
2. **Browsing Food Items**
   * Users view different food items displayed, categorized by type (e.g., snacks, meals, desserts).
3. **Selecting a Food Item from Categories**
   * After browsing, users can select food items of interest within specific categories.
4. **Adding Items to Cart**
   * Selected items are added to the shopping cart for review before checkout.
5. **Placing an Order**
   * Users review the items in the cart and place an order.
6. **Making a Payment**
   * Users complete payment using available payment methods (credit card, PayPal, etc.).
7. **Receiving Delivery Status**
   * The app shows updates on the delivery status, from confirmation to final delivery.

Each stage is represented with a clear box and directional arrows showing flow from one action to the next.

**ARCHITECTURE OF OUR APPLICATION:**

For an SB Food App, the architecture generally includes layers that handle user interactions, business logic, data management, and external integrations. Here’s a basic outline of the architecture:

1. Front-End (Client Layer)

* User Interface: The main interface for user interactions, including screens for browsing food items, placing orders, making payments, and checking order status.
* Technologies: React Native or Flutter for cross-platform mobile apps, or React.js/Vue.js for a web interface.
* Components:
  + Authentication: Handles login, registration, and secure access.
  + Food Menu Display: Shows categories and detailed views of food items.
  + Order Cart: Allows users to add or remove items and view the final list.
  + Order Status & Tracking: Provides real-time updates on order and delivery status.

2. Back-End (Server Layer)

* RESTful API / GraphQL: Provides an interface for the front end to communicate with the server.
* Authentication Service: Manages user sessions, login, sign-up, and password security (e.g., JWT, OAuth).
* Order Management Service: Handles the logic of creating, updating, and managing orders, including item availability checks.
* Payment Gateway Integration: Connects to third-party payment services (e.g., Stripe, PayPal) to process payments securely.
* Notification Service: Sends real-time notifications (push notifications, emails, or SMS) for order updates and delivery statuses.
* Delivery Coordination: Manages delivery partners or integrates with a delivery service API for order tracking.

3. Database Layer

* Database: Stores all app data, including user information, food items, categories, orders, and payment histories.
  + User Data: Stores login credentials, user details, and preferences.
  + Food Menu Data: Stores food item details, prices, and availability.
  + Order Data: Contains order details, timestamps, statuses, and tracking information.
* Database Types:
  + Relational (SQL): Useful for structured data (e.g., PostgreSQL, MySQL) for easy querying.
  + NoSQL (e.g., MongoDB): Good for flexible and scalable data, especially if the menu or order structure varies.

4. External Services and Integrations

* Payment Processing: Integrate with secure payment providers (e.g., Stripe, PayPal) to handle transactions.
* Delivery Service API: Use an API like Google Maps for real-time delivery tracking or integrate with a delivery partner.
* Analytics and Monitoring: Tools like Google Analytics or Firebase for tracking user behavior, app performance, and error monitoring.
* Push Notifications: Integrate with Firebase Cloud Messaging (FCM) or Apple Push Notification Service (APNS) to keep users updated on order and delivery status.

5. Deployment and Scalability

* Containerization: Docker containers for easy scalability and portability across different environments.
* Orchestration: Kubernetes for managing containerized applications and ensuring scalability, load balancing, and resilience.
* Cloud Services: Use cloud providers like AWS, Google Cloud, or Azure to host servers, databases, and API services.
* Load Balancing: Distribute incoming requests across multiple servers to handle high traffic efficiently.
* Caching: Use Redis or Memcached to cache frequently accessed data like menu items or user preferences for quick retrieval.

6. Security

* Data Encryption: Use SSL/TLS encryption for all communications between clients and servers.
* Authentication: Secure login with JWT or OAuth and strong password policies.
* Data Access Control: Ensure only authorized users and services can access sensitive data, using role-based access control (RBAC).

**ARCHITECTURE SUMMARY:**

**Architecture of our application:**

**1. Model (MongoDB Schema):**

**The Model represents the data structure and the business logic of the application. In MongoDB, models are represented using schemas with Mongoose.**

**2. View (Frontend - React):**

**The view will be responsible for rendering the UI. For the FOOD APP system, you will have various components such as food listings, booking dining , and resort names.**

**3.Controller (Backend - Express):**

**The controller handles requests, interacts with models, and returns the response.**

****