DAY 2

PLANNING THE TECHNICAL FOUNDATION

Food Restaurant Website

Technical Requirements

1. Framework:

- **Next.js** – Chosen for building the food restaurant website due to its performance and SEO-friendly features, especially beneficial for e-commerce platforms.

2. Frontend:

- **Styling** – Tailwind CSS for responsive and quick UI design.
- **Libraries** – Using ShadCN UI for components like hamburger menus and React Icons for various visual elements.
- **Pages and Components**:
    - Pages: Home Page, Menu Page, Reservation Page, Order Page, Payment Page, Admin Page.
    - Components: Navbar, Hero, Menu Card, Header, Footer, Review Section, Order Summary, and other required web components.

3. Backend:

- **Sanity CMS** – Headless CMS for managing menu data, reservation details, and delivery orders.
- **API Integration:**
    - Stripe for secure payment processing.
    - Custom APIs for tracking deliveries.
    - Clerk for user authentication.

4. Future Plans:

- Integrate AI-powered features to enhance user experience and stand out in the food industry.
    - **AI Chatbot:**
        - Provide instant customer support, answer FAQs, and guide users through menu selection or order tracking.
        - Use AI tools like Dialogflow or OpenAI's GPT models.
    - **Smart Search:**
        - Enable users to search for dishes or offers using natural language.

- Use NLP models like OpenAI's GPT for parsing user queries into actionable filters.

Design System Architecture

Workflow

1. **User Registration**
   - Users create an account by signing up with their name, email, and password.
   - Saves user data in the database (Sanity CMS).
   - A confirmation email is sent to the user for login.
2. **Menu Browsing**
   - Users browse the menu by categories (e.g., Starters, Main Courses, Desserts).
   - Fetches menu data from the database and displays it dynamically on the website.
3. **Order Placement (Delivery):**
   - Users add items to their cart, proceed to checkout, and confirm delivery orders.
   - System saves order details in the database and processes payments via Stripe.
   - Order confirmation is displayed to the user.
4. **Dine-In Reservation:**
   - Users select a table and time for dine-in.
   - Reservation details are saved in Sanity CMS.
   - A confirmation is sent to the user.
5. **Order Tracking:**
   - Users check the status of their delivery (e.g., "Out for Delivery", "Delivered").
   - Status updates are fetched via third-party APIs and displayed in real-time.

Plan API Requirements:

Endpoint	Method	Purpose	Response
`/menu`	GET	Fetch all menu items.	`{"menuId": 101, "name": "Pizza", "category": "Main Course", "price": 12.99, "availability": true}`
`/menu/:id`	GET	Fetch details of a menu item.	`{"menuId": 101, "name": "Pizza", "ingredients": ["Cheese", "Tomato"], "price": 12.99}`
`/reservation`	POST	Add dine-in reservation details	`{"reservationId": 202, "status": "Confirmed"}`
`/order`	POST	Add a delivery order.	`{"orderId": 303, "status": "Created"}`
`/order/:id`	GET	Get order details.	`{"orderId": 303, "status": "Out for Delivery", "deliveryTime": "45 mins"}`
`/payment`	POST	Process payment for orders.	`{"paymentId": 404, "status": "Success"}`
`/tracking/:id`	GET	Fetch delivery	`{"trackingId": 505, "status": "In Transit"}`

Sanity Schema for Food Menu

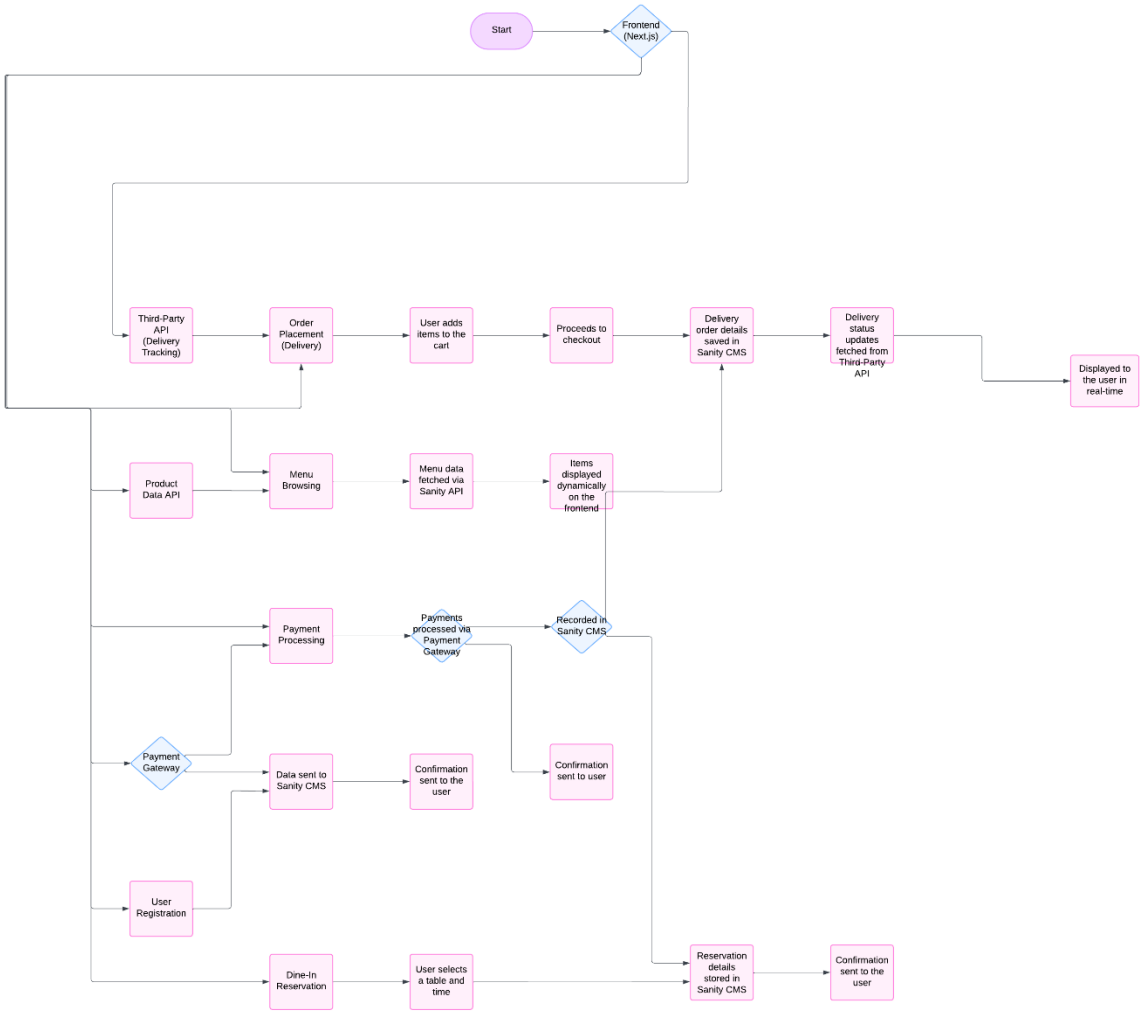
```
javascript
CopyEdit
export default {
  name: 'menuItem',
  type: 'document',
  title: 'Menu Item',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Dish Name',
      validation: Rule => Rule.required(),
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
      options: {
        list: [
          { title: 'Starter', value: 'starter' },
          { title: 'Main Course', value: 'main_course' },
          { title: 'Dessert', value: 'dessert' },
          { title: 'Beverage', value: 'beverage' },
        ],
      },
      validation: Rule => Rule.required(),
    },
```

```
    {
      name: 'price',
      type: 'number',
      title: 'Price',
      validation: Rule => Rule.min(0).required(),
    },
    {
      name: 'availability',
      type: 'boolean',
      title: 'Available',
    },
    {
      name: 'ingredients',
      type: 'array',
      of: [{ type: 'string' }],
      title: 'Ingredients',
    },
    {
      name: 'image',
      type: 'image',
      title: 'Dish Image',
      options: { hotspot: true },
    },
    {
      name: 'description',
      type: 'text',
      title: 'Description',
    },
  ],
};
```

Conclusion

The **Food Restaurant Website** is built with **Next.js** for performance and SEO, styled with **Tailwind CSS**, and includes key features for **menu browsing**, **dine-in reservations**, and **delivery order tracking**. The backend uses **Sanity CMS** for managing data, **Stripe** for payments, and APIs for seamless user experience. Future plans include integrating AI for chatbots and smart search capabilities.

Design System Architecture:


Start
Frontend (Next.js)
Third-Party API (Delivery Tracking)
Order Placement (Delivery)
User adds items to the cart
Proceeds to checkout
Delivery order details saved in Sanity CMS
Delivery status updates fetched from Third-Party API
Displayed to the user in real-time
Product Data API
Menu Browsing
Menu data fetched via Sanity API
Items displayed dynamically on the frontend
Payment Processing
Payments processed via Payment Gateway
Recorded in Sanity CMS
Payment Gateway
Data sent to Sanity CMS
Confirmation sent to the user
Confirmation sent to user
User Registration
Dine-In Reservation
User selects a table and time
Reservation details stored in Sanity CMS
Confirmation sent to the user