Day 2

My E-Commerce Website Plan and Documentation

Overview

I’m building a single-page e-commerce website to sell "Ready-to-Cook" and "Ready-to-Eat" meals. The website will have a landing page showcasing all the products, along with a login/signup option for user accounts. Orders will be processed from the home page, with payments handled via a secure gateway. Here’s the complete breakdown of the steps I’ve planned and implemented.

1. Frontend Requirements

Page Layout

1. **Hero Section**:
    - A welcoming banner with my business logo and tagline: *“Fresh, Fast, Flavorful – Ready-to-Cook and Ready-to-Eat Meals!”*
    - A prominent "Order Now" call-to-action button.
2. **Product Section**:
    - Grid-style display showing all products. Each product will include:
        - Image.
        - Name (e.g., Chicken Biryani).
        - Price.
        - Description.
        - "Order Now" button.
3. **About Us Section**:
    - A brief story about why I started this business and what makes it unique.
    - Optionally, a picture of my brother and me.
4. **Footer**:
    - Contact info (email, phone).
    - Links to social media accounts.

---

2. Backend Setup with Sanity CMS

Why Sanity CMS?

I'm using Sanity CMS to store and manage all my website data, including:

1. **Products**: Details like name, price, image, and description.
2. **Orders**: Customer info, products ordered, and order status.
3. **Users**: Registered users' accounts, with hashed passwords for security.

Sanity Schemas

Here's how I've structured my schemas:

1. **Product Schema**:

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    { name: 'name', type: 'string', title: 'Product Nam
e' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'image', type: 'image', title: 'Image' },
    { name: 'description', type: 'text', title: 'Descrip
tion' }
  ]
};
```

2. **Order Schema**:

```
export default {
  name: 'order',
  type: 'document',
```

```
  title: 'Order',
  fields: [
    { name: 'customer', type: 'reference', to: [{ type:
'user' }], title: 'Customer' },
    { name: 'products', type: 'array', of: [{ type: 'ref
erence', to: [{ type: 'product' }] }], title: 'Products'
},
    { name: 'totalPrice', type: 'number', title: 'Total
Price' },
    { name: 'paymentStatus', type: 'string', title: 'Pay
ment Status', options: { list: ['Pending', 'Paid', 'Fail
ed'] } }
  ]
};
```

3. **User Schema**:

```
export default {
  name: 'user',
  type: 'document',
  title: 'User',
  fields: [
    { name: 'name', type: 'string', title: 'Full Name'
},
    { name: 'email', type: 'string', title: 'Email' },
    { name: 'password', type: 'string', title: 'Passwor
d', options: { isHidden: true } },
    { name: 'address', type: 'text', title: 'Address' },
    { name: 'orders', type: 'array', of: [{ type: 'refer
ence', to: [{ type: 'order' }] }], title: 'Order Histor
y' }
  ]
};
```

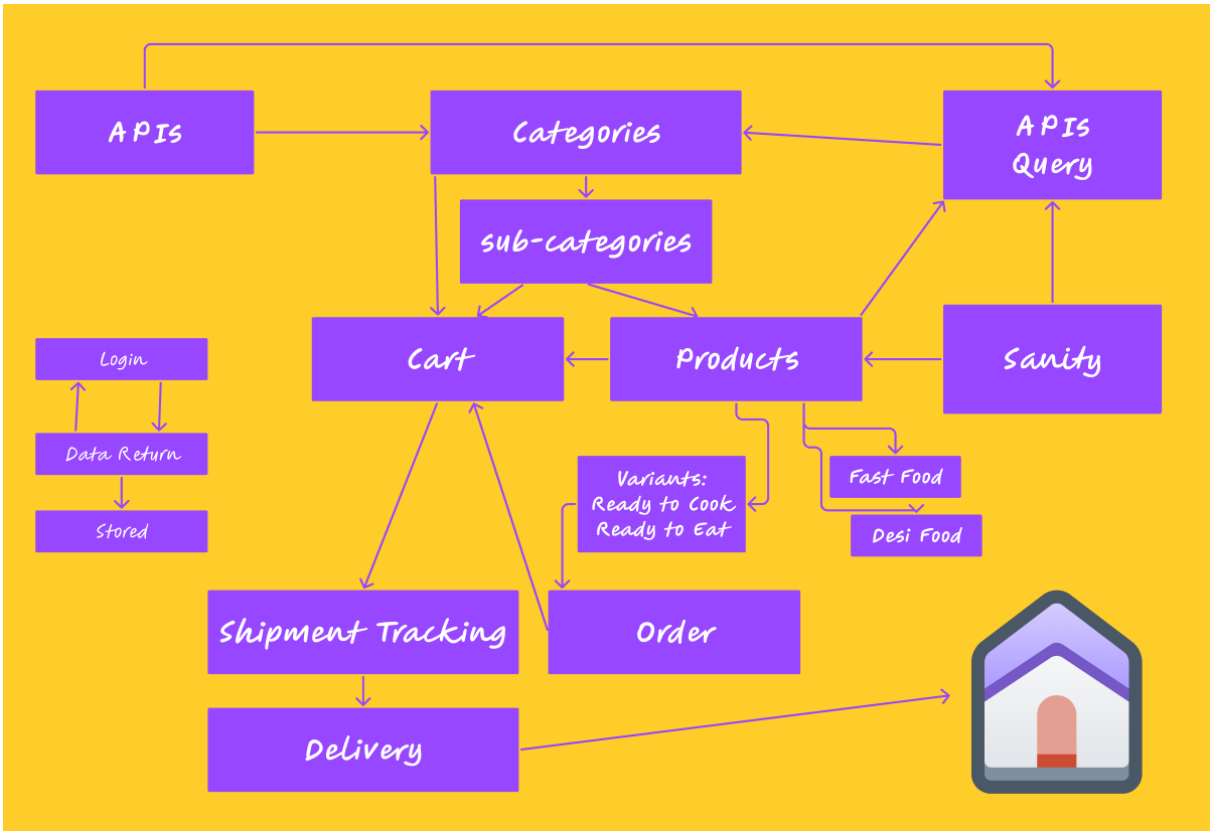
3. System Architecture

Here’s how everything fits together:

1. The **frontend** fetches product data from Sanity CMS using APIs.
2. Users can log in or sign up, and their data is stored in Sanity CMS.
3. Orders are submitted via the frontend, stored in Sanity CMS, and linked to the logged-in user.
4. Payments are processed through a third-party payment gateway.

System Diagram:

```
flowchart TD
  A[Frontend] -->|Fetch Products| B[Sanity CMS]
  A -->|Submit Order| B
  A -->|Authenticate Users| B
  A -->|Process Payment| C[Payment Gateway]
```



4. API Requirements

Endpoints

1. **Fetch All Products**:
   - **Endpoint**: `/products`
   - **Method**: `GET`
   - **Response**:

```
[
  { "id": "1", "name": "Chicken Biryani", "price": 200, "image": "biryani.jpg", "description": "Delicious spicy chicken biryani" }
]
```

2. **Sign Up**:
    - **Endpoint**: `/signup`
    - **Method**: `POST`
    - **Payload**:

```
{ "name": "John Doe", "email": "john@example.com", "p
assword": "secure123", "address": "123 Street" }
```

3. **Login**:
    - **Endpoint**: `/login`
    - **Method**: `POST`
    - **Payload**:

```
{ "email": "john@example.com", "password": "secure12
3" }
```

4. **Place Order**:
    - **Endpoint**: `/orders`
    - **Method**: `POST`
    - **Payload**:

```
{ "customerId": "user123", "products": [{ "id": "1",
"quantity": 2 }], "totalPrice": 400 }
```

---

5. Authentication

I decided to include accounts to let users save their orders and track them. The **Sign-Up** and **Login** pages handle this, using JWT for secure authentication.

1. **Sign-Up**:
    - Users create an account with their name, email, password, and address.
    - Passwords are securely hashed before storage.

2. **Login**:
    - Users log in with their email and password.
    - A JWT is generated for session handling.

3. **Order Tracking**:
    - Logged-in users can view their order history.

---

6. Payment Gateway

I’m integrating **Stripe** as my payment gateway because it’s simple, secure, and widely trusted.

Steps:

1. The user clicks "Pay Now" at checkout.
2. Stripe.js collects their payment details.
3. The backend processes the payment and updates the order status in Sanity CMS.

7. Final User Journey

1. **Sign Up/Login**:
    - Users create an account or log in.
2. **Browse Products**:
    - All products are displayed on the homepage.
3. **Place Order**:
    - Users select a product, choose quantity, and click "Place Order."
4. **Checkout**:
    - Users enter their payment details, and the order is confirmed.
5. **Order Tracking**:
    - Logged-in users can view their order history.