# HACKATHON DAY 2: PLANNING THE TECHNICAL FOUNDATION OF OUR E-COMMERCE WEBSITE

# 1. Technology Stack

#### Frontend:

- Framework: Next.js (for SSR and SSG, ensuring fast and responsive user experience)
- **Styling:** Tailwind CSS (utility-first approach to styling)
- State Management: Context API (efficient state management)
- **Animations:** Framer Motion (smooth and engaging animations)

#### Backend:

- Framework: Custom API routes in Next.js (server-side logic)
- **Database:** Sanity CMS (managing products, orders, categories)
- Authentication: NextAuth.js (secure login and user sessions)

# **Third-Party APIs:**

- Payment Gateway: Stripe (secure payment processing)
- **Shipping:** Shippo (real-time shipment tracking and rate calculations)
- **Notifications:** SendGrid (order confirmations and other notifications)

# 2. Website Architecture

#### Overview:

The website architecture ensures seamless interaction between:

- User interface (Frontend)
- Backend API routes
- Sanity CMS
- Third-party integrations (Stripe, Shippo)

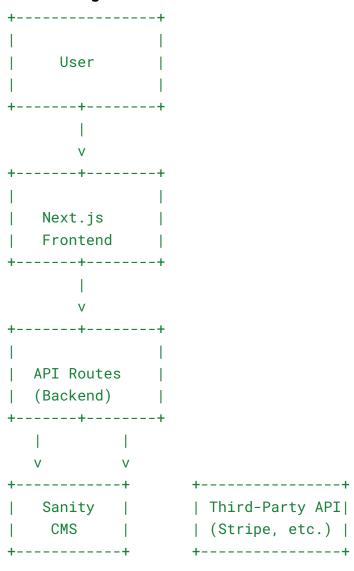
# **Data Flow Diagram:**

```
[User] -> [Next.js Frontend] -> [API Endpoints] -> [Sanity CMS |
Third-Party APIs]
```

#### Interaction Flow:

- 1. Users browse products, manage their cart, and complete purchases.
- 2. The backend handles user authentication and communicates with Sanity CMS and third-party APIs.
- 3. Sanity CMS stores product, category, and order data dynamically.
- 4. Stripe and Shippo handle payments and shipping.

# **Process Diagram:**



# 3. Features Breakdown

# **User Signup/Login:**

- Registration and secure login using NextAuth.js.
- Token-based authentication for session security.

# **Product Listing:**

- Dynamic fetching from Sanity CMS using GROQ queries.
- Server-side rendering (SSR) or incremental static regeneration (ISR) for data rendering.

# **Cart Management:**

- For guest users, cart data is stored in localStorage.
- For logged-in users, cart is synced with the backend to ensure persistence across devices.

# **Checkout:**

- Payments processed securely using Stripe.
- Real-time shipping calculations and delivery times via Shippo API.
- Email notifications via SendGrid upon successful order placement.

# 4. API Requirements

#### **Authentication API:**

• Endpoint: /api/auth/signup

Method: POST

• **Description:** Registers new users.

# **Products API:**

• Endpoint: /api/products

Method: GET

• **Description:** Fetches product details from Sanity CMS.

#### Cart API:

• Endpoint: /api/cart/add

Method: POST

• **Description:** Adds items to the user's cart.

# **Checkout API:**

• Endpoint: /api/checkout

Method: POST

• **Description:** Processes payments and finalizes orders.

# 5. Data Fetching Plan

- **Home Page:** Incremental Static Regeneration (ISR) for featured products.
- Product Details Page: Server-side rendering (SSR) for up-to-date product details.
- User Dashboard: Client-side rendering (CSR) for personalized data like orders and cart items.

# 6. Sanity CMS Schemas

# **Product Schema:**

- Fields:
  - Name (string)
  - Price (number)
  - Description (text)
  - Image (image with hotspot support)
  - Category (reference to Category Schema)

# **Category Schema:**

- Fields:
  - Name (string)
  - Description (text)

# **Order Schema:**

- Fields:
  - Customer Name (string)
  - Products (array of references to the Product Schema)
  - Total Price (number)
  - Status (enum: pending, shipped, completed)

# **Customer Schema:**

- Fields:
  - Name (string)
  - Email (string)
  - Address (object: street, city, state, zip)
  - Orders (array of references to the Order Schema)

# 7. Folder Structure

```
/project
├── /components  # Reusable UI components
├── /pages
├── /api  # API endpoints
├── index.tsx  # Home page
├── product/[id].tsx # Product details page
├── cart.tsx  # Cart page
├── checkout.tsx  # Checkout page
├── /styles  # Tailwind CSS files
├── /utils  # Utility functions
├── /sanity  # Sanity CMS schemas
├── /public  # Static assets
```

# 8. Technical Documentation Summary

#### Frontend:

Developed using Next.js, with SSR, SSG, and dynamic routing.

# CMS:

Sanity CMS is used to manage products, categories, and orders.

# **Third-Party Integrations:**

- Stripe: Secure payment processing.
- Shippo: Shipping rates and tracking.

# **API Endpoints:**

Authentication, product management, cart operations, checkout, and order processing.

# Data Flow:

• Dynamic data fetching using Next.js, ensuring a smooth user experience and efficient management of transactions and updates.