Hackathon Day:2

PLANNING THE TECHNICAL FOUNDATION

WEBSITE NAME:
Comforty - E-Commerce website

OBJECTIVE:
A sophisticated, responsive, and highly functional e-commerce website leveraging cutting-edge technologies like Next.js, Tailwind CSS, Sanity, and Figma. Integrate third-party APIs to ensure seamless product management, secure user authentication, and efficient shipping processes.

GOALS:

- **Seamless User Experience:** Design an intuitive and visually engaging interface that enhances product browsing and purchasing.
- **Cross-Device Compatibility:** Implement a mobile-first design to ensure seamless functionality on all devices.
- **Effortless Content Management:** Leverage Sanity CMS for easy updates to products and content.
- **Dynamic API Integration:** Incorporate third-party APIs for real-time management of products, orders, and shipping.
- **Optimized Performance:** Harness Next.js features like Static Site Generation (SSG) and Server-Side Rendering (SSR) for lightning-fast load times.

SYSTEM ARCHITECTURE

FRONTEND (CLIENT-SIDE):

- Built with **Next.js** and styled using **Tailwind CSS**.
- Features:
  - Interactive UI/UX.
  - Dynamic page routing (e.g., product, cart, checkout).
  - Data fetching and rendering (SSG, SSR, CSR).
  - API integration for data display and user actions.

BACKEND:

1. **Sanity CMS:**
    - Manages product data, categories, and promotions.
    - Fetch data using GROQ queries or the client SDK (e.g., product details for product pages).
2. **Custom API (Next.js API Routes):**
    - `/api/cart`: Handle cart operations (add, update, remove).
    - `/api/orders`: Create and manage orders.
    - `/api/users`: Manage user profiles and order history.
3. **Third-Party API Orchestration:**
    - **Payment Gateway** (e.g., Easypaisa, JazzCash, HBL):
        - Secure payment processing and tokenization.
        - Validates payments before creating orders.
    - **Shipping API** (e.g., ShipEngine):
        - Real-time rates, label generation, and tracking.

IN-DEPTH SYSTEM FRAMEWORK

FRONTEND:

1. **Components:**
    - Reusable UI elements: Navbar, ProductCard, Footer.
    - Pages: Home, Product Listing, About, Contact, FAQs, Login/Signup, Account, Admin, Product Details, Cart, and Checkout.
2. **State Management:**
    - Manage client-side state for cart and user sessions using **Context API** or **Zustand**.
3. **Data Fetching:**
    - **SSG (Static Site Generation):** Pre-render static pages like the product catalog for better performance.
    - **SSR (Server-Side Rendering):** Fetch dynamic data like personalized recommendations.
    - **CSR (Client-Side Rendering):** Fetch data for user actions like adding items to the cart.
4. **API Integration:**
    - Fetch data from **Sanity CMS** and custom APIs using `getStaticProps`, `getServerSideProps`, or `useSWR`.

BACKEND

1. Sanity CMS:

- **Role:** Manage product data, categories, and promotional content.
- **Integration:** Use GROQ queries or client SDK to fetch data (e.g., product details for rendering).

2. Custom API (Next.js API Routes):

- **Endpoints:**
  - `/api/cart` (GET): Handle cart operations (add, update, remove).
  - `/api/orders` (POST): Create and manage orders.
  - `/api/users` (GET): Manage user profiles and order history.

3. Third-Party API Orchestration:

- **Payment API (e.g., Easypaisa, JazzCash):**
  - Frontend collects secure payment details via a form.
  - Sends data to the custom API, which interacts with the payment gateway for processing.
  - Returns success/failure response to the frontend.
- **Shipping API (e.g., ShipEngine):**
  - Frontend sends shipping details (e.g., address, cart weight) to the custom API.
  - Custom API fetches rates and labels from the shipping service.
  - Returns shipping options and tracking information to the frontend.

SECURITY BEST PRACTICES:

1. **API Authentication:**
   - Implement secure API keys for accessing Sanity, payment gateways, and shipping services.
   - Store sensitive credentials safely, such as using environment variables or a secure vault.
2. **Frontend Security:**
   - Ensure all communications are encrypted using HTTPS and enforce strict Content Security Policies (CSP).
   - Never expose sensitive information, such as API keys, in client-side code or front-end files.
3. **Payment Security:**
   - Integrate PCI-compliant payment gateways like Stripe to handle transactions securely.
   - Avoid storing any sensitive payment data on your servers to reduce the risk of data breaches.

USER WORKFLOW:

1. **Login/Signup**
   - **Frontend:** User accesses the login/signup page, entering email/password or using social login (e.g., Google).
   - **Backend:** Sends credentials for authentication (e.g., Firebase Auth), returns a session token (JWT).
   - **Post-login:** Token is stored securely (e.g., HTTP-only cookie), and user is redirected to the homepage/dashboard.
2. **Browse Products**
   - **Frontend:** User navigates to the product listing page, filters and sorts products (e.g., by category, price, rating).
   - **Backend:** Fetches and returns filtered product data from Sanity CMS.
   - **Frontend:** Displays the product grid.
3. **View Product Details**
   - **Frontend:** User clicks on a product to see more details.
   - **Backend:** Fetches product details (description, price, reviews, availability) from Sanity CMS.
   - **Frontend:** Displays product information and "Add to Cart" option.
4. **Add to Cart**
   - **Frontend:** User selects quantity and clicks "Add to Cart"; cart updates locally or through an API call.
   - **Backend:** If using server-side cart, updates the cart in the database and returns updated data.
   - **Frontend:** Updates cart UI.
5. **Checkout**
   - **Frontend:** User enters shipping details and selects a shipping method.
   - **Backend:** Calls shipping API (e.g., ShipEngine) to calculate shipping costs and returns options.
   - **Frontend:** Displays the total cost including shipping.
6. **Payment**
   - **Frontend:** User enters payment details or selects a saved method and sends data to the API.
   - **Backend:** API processes payment via a payment gateway (e.g., Easypaisa) and returns confirmation or error.
   - **Frontend:** Displays success message and order confirmation.
7. **Order Tracking**
   - **Frontend:** User navigates to "Order History" and selects an order to track.
   - **Backend:** Fetches order status and tracking details from the database and shipping API.
   - **Frontend:** Displays order status and tracking link.

ADMIN WORKFLOW:

1. **Admin Login**
   - **Frontend:** Admin accesses the login page and inputs credentials.
   - **Backend:** Authenticates credentials using Firebase Auth or a custom system, returning an admin token.
   - **Post-login:** Admin is redirected to the dashboard.
2. **Manage Products**
   - **Frontend:** Admin navigates to the "Products" section, adds, edits, or deletes products.
   - **Backend:** Fetches product data from Sanity CMS, handles CRUD operations.
   - **Frontend:** Updates product list.
3. **View and Manage Orders**
   - **Frontend:** Admin navigates to the "Orders" section, views orders with status, and updates them.
   - **Backend:** Fetches order data and updates status (e.g., "Shipped").
   - **Frontend:** Updates order status in the admin panel.
4. **Manage Users**
   - **Frontend:** Admin navigates to the "Users" section, views all users, and can block/unblock or reset passwords.
   - **Backend:** Fetches and updates user data.
   - **Frontend:** Reflects changes in the user list.
5. **Monitor Analytics**
   - **Frontend:** Admin accesses the "Analytics" section to view key metrics.
   - **Backend:** Fetches data from the database or analytics service.
   - **Frontend:** Displays charts and insights dynamically.
