Hackathon Day 2: Planning the Technical Foundation

Marketplace Technical Foundation – TrueBuy

1. Define Technical Requirements

Frontend Requirements

For the frontend, I will use:

1. **Next.js**: To create a fast, SEO-friendly, and scalable interface for the marketplace.
2. **Tailwind CSS**: For designing a clean, responsive, and user-friendly layout that works seamlessly on both mobile and desktop devices.

Key Pages:

- Home Page
- Product Listing Page
- Product Details Page
- Cart Page
- Checkout Page
- Order Confirmation Page

Backend and CMS

I will use **Sanity CMS** as the backend to manage all critical data, including:

- Product data (name, price, stock, and images).
- Customer details (name, contact, and addresses).
- Order records (order history, status, and tracking).

I'll design schemas in Sanity CMS that align with the business goals defined on Day 1, ensuring efficient data management and easy integration with the frontend.

Third-Party Tools and APIs

1. **Cart and Checkout**:
   - I plan to use **Snipcart** for a quick and efficient cart and checkout solution.
   - Alternatively, if I build it manually, I will implement cart functionality using **local storage** to manage items in the cart.
2. **Shipment Tracking**:
   - I'll integrate **ShipEngine** to provide real-time shipment tracking.
3. **Payment Gateway**:
   - I'll use **Stripe** for secure and reliable payment processing.

4. **Authentication**:
   - If required, I'll use **Clerk** for user authentication to manage customer accounts seamlessly.
5. **Live Chat**:
   - Depending on time and project requirements, I'll add **GetStream** to implement a live chat feature for enhanced customer support.

3. Design System Architecture

$$\begin{tikzcd}
& \text{{[Frontend (Next.js)]}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{{[Sanity CMS]}} \arrow[d] & \text{{[Third-Party APIs]}} \arrow[d] & \text{{[Payment Gateway]}} \arrow[d] \\
\text{{[Product Data API]}} & \text{{[Shipment Tracking API]}} & \text{{[Stripe (Payment)]}}
\end{tikzcd}$$

Detailed Workflows

1. **User Registration** (if authentication is implemented):
   - **Step 1**: User signs up via the frontend.
   - **Step 2**: User data is sent to **Clerk** for authentication and stored.
   - **Step 3**: Sanity CMS updates customer details, if needed.
   - **Outcome**: User receives a confirmation email.
2. **Product Browsing**:
   - **Step 1**: User visits the website and navigates to the product listing page.
   - **Step 2**: The frontend fetches product data from **Sanity CMS** using the Product Data API.
   - **Step 3**: Products are dynamically displayed with filtering and sorting options.
3. **Order Placement**:
   - **Step 1**: User adds products to the cart, either via **Snipcart** or a custom local storage implementation.
   - **Step 2**: At checkout, the order details are sent to **Sanity CMS**, where they are recorded.
   - **Step 3**: User proceeds to payment via **Stripe**.
4. **Shipment Tracking**:
   - **Step 1**: After an order is placed, shipping details are passed to **ShipEngine**.
   - **Step 2**: Real-time shipment updates are fetched from ShipEngine's API and displayed to the user on the Order Tracking page.
5. **Payment Processing**:
   - **Step 1**: User enters payment details on the checkout page.

    - **Step 2**: Stripe processes the payment securely and sends a confirmation response.
    - **Step 3**: Payment confirmation is logged in Sanity CMS and displayed to the user.

3. Plan API Requirements

1. Fetch All Products

- **Endpoint Name**: /products
- **Method**: GET
- **Description**: Retrieve all available products from Sanity CMS.
- **Response Example**:

```
[{
   "id": 1,
   "name": "Wireless Headphones",
   "price": 120,
   "stock": 25,
   "image": "url-to-image"
 }]
```

2. Create a New Order

- **Endpoint Name**: /orders
- **Method**: POST
- **Description**: Save a new order in Sanity CMS.
- **Payload Example:**

```
{
    "customerInfo": {
      "name": "John Doe",
      "email": "john.doe@example.com",
      "address": "123 Main St, City, Country"
    },
    "orderDetails": [
      { "productId": 1, "quantity": 2 },
      { "productId": 2, "quantity": 1 }
    ],
    "paymentStatus": "Paid"
  }
```

Response Example:

```
{ "orderId": "ORD12345", "status": "Order Created" }
```

3. Track Shipment

- **Endpoint Name**: /shipment

- **Method**: GET
- **Description**: Fetch order shipment status via ShipEngine API.
- **Request Query Parameters**:
    - orderId: The ID of the order being tracked.
- **Response Example**:

```
{
  "shipmentId": "SHIP67890",
  "orderId": "ORD12345",
  "status": "In Transit",
  "expectedDeliveryDate": "2025-01-20"
}
```