Hackathon 3

Day 2

Planning the Technical foundation

Hackathon Day 1 Recap

On Day 1, we laid the foundation for our e-commerce marketplace. We defined the marketplace type (E-commerce), identified the primary purpose (offering stylish, customer-demanded furniture), and outlined business goals. We also designed a data schema for key entities like Products, Orders, Customers, Delivery Zone, Shipment, and Payments, and visualized their relationships.

Hackathon Day 2: Building the Technical Foundation

1. Frontend Framework and Styling

- **Framework**: The project uses **Next.js** for building a user-friendly, SEO-optimized, and high-performance web application.
- **Styling**: **Tailwind CSS** is used for styling to create a modern, responsive design. It ensures the site adapts seamlessly across all devices (desktop, tablet, mobile).

2. Essential Pages and User Interaction

1. **Home Page**: Displays trending and featured products with categories like sofas, chairs, and tables.
2. **About Page**: Provides information about the business vision and values.

3. **Contact Page**: Allows users to send queries or feedback.
4. **Shop Page**: Lists all products with filtering options (categories, price range).
5. **My Account Page**: Enables users to view and manage their profile and order history.
6. **Add to Cart Page**: Allows users to review selected items and proceed to checkout.
7. **Checkout Page**: Facilitates payment and shipping details.
8. **Place Order Confirmation Page**: Shows the summary and status of the order.

Backend: Sanity CMS

- **Installation**:

```
bash
CopyEdit
npm install -g @sanity/cli
sanity init
```

- **Schema Design**: Create schemas for `Products`, `Orders`, `Customers`, etc. Each schema includes fields like `title`, `price`, `category`, and `stock`.
- **Fetching Data**: Use Sanity GROQ queries:

```
js
CopyEdit
const query = `*[_type == "product"]`;
const products = await sanityClient.fetch(query);
```

- **Preview and Publish**:

Run the studio locally to preview:

```
bash
CopyEdit
sanity start
```

Deploy to the cloud for publishing:

```
bash
CopyEdit
sanity deploy
```

Third-Party API Integration

- **Shipment Tracking API**: Integrate a third-party API to enable real-time shipment tracking. Example: Fetch delivery status by order ID and display updates on the "Order Tracking" page.

5. System Architecture Components

- **Frontend (Next.js)**: Handles user interactions like browsing products, viewing details, and managing the cart.
- **Backend (Sanity CMS)**: Stores and manages content like product details, customer orders, and shipment information.
- **Third-Party API**: Provides functionality like shipment tracking and payment processing.

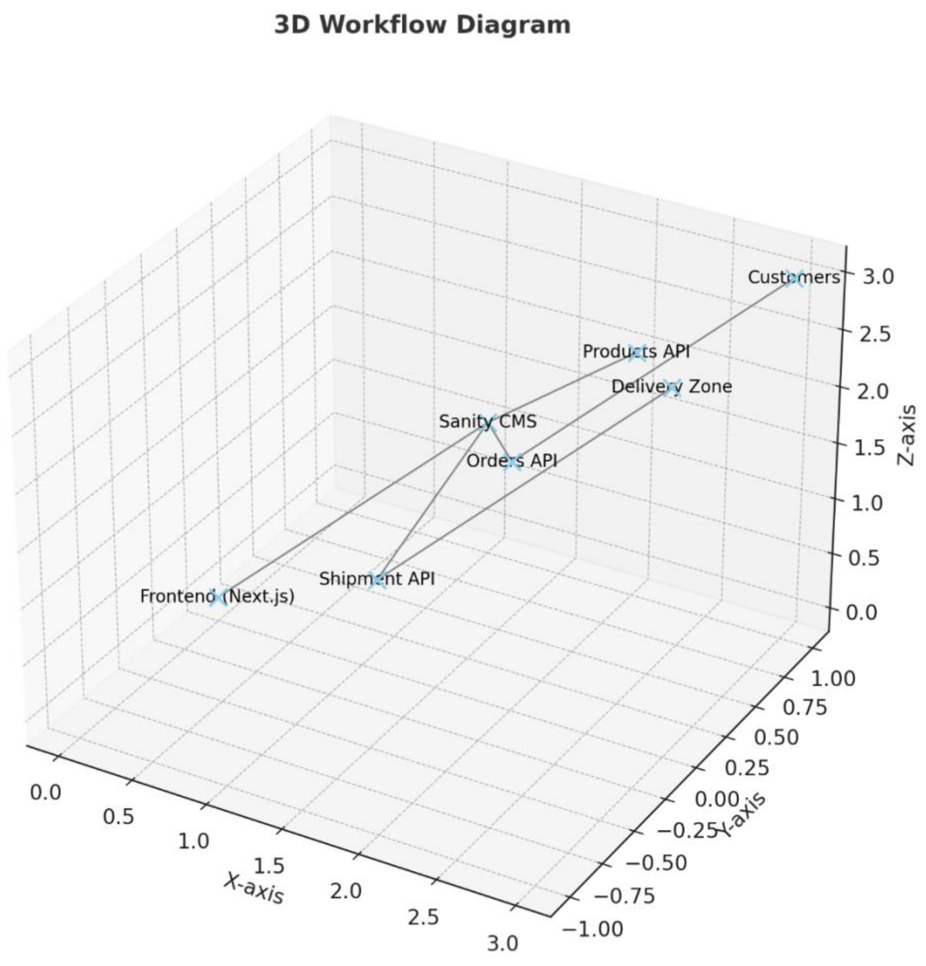
6. Data Workflow

1. **Browsing Products**: Users view all products categorized by type.
2. **Product Details Page**: Displays information like price, stock, and description.
3. **Adding to Cart**: Users add products to their cart and review items.
4. **Placing an Order**: Users provide shipping information and confirm payment.
5. **Order Tracking**: Users check the status of their shipment.

7. Planning API Requirements

1. **Products**:
   a. **Endpoint Name**: `/products`
   b. **Method**: GET
   c. **Description**: Fetch all available products.
   d. **Request**: No parameters.
2. **Product Details**:
   a. **Endpoint Name**: `/products/:id`
   b. **Method**: GET
   c. **Description**: Fetch details of a specific product.
   d. **Request**: Product ID.
3. **Orders**:

    a. **Endpoint Name**: `/orders`
    b. **Method**: POST
    c. **Description**: Create a new order.
    d. **Request**: Customer info, product details, payment status.
4. **Shipment Tracking**:
    a. **Endpoint Name**: `/shipment/:orderId`
    b. **Method**: GET
    c. **Description**: Get shipment status for a specific order.
    d. **Request**: Order ID.


3D Workflow Diagram
Customers
Products API
Delivery Zone
Sanity CMS
Orders API
Shipment API
Frontend (Next.js)
X-axis
Y-axis
Z-axis