Hackhathon Day 02 Challange

1. Define Technical Requirements

To create a marketplace system, begin by turning business goals into technical requirements:

- **Frontend Requirements**:
    - The interface must be user-friendly for product browsing.
    - Ensure the design is responsive for both mobile and desktop users.
    - Develop essential pages: Home, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.
- **Backend (Sanity CMS)**:
    - Use Sanity CMS to manage key data such as products, customers, and orders.
    - Create schemas in Sanity that align with the marketplace's goals.
- **Third-Party APIs**:
    - Integrate APIs for shipment tracking, payment gateways, and backend services.
    - Ensure APIs provide the data needed for the frontend.

2. Design System Architecture

Develop a clear structure for how components interact:

- Use tools like Lucidchart, Figma, or Excalidraw to draft a diagram.
- Example architecture:
    - **Frontend (Next.js)** interacts with:
        - **Sanity CMS** to fetch product data through the Product Data API.
        - **Third-Party APIs** for shipment tracking and payment processing.

3. Data Flow Example

Illustrate how data flows in the system:

1. A user visits the marketplace frontend to browse products.
2. The frontend requests product information from the Product Data API.
3. Sanity CMS provides the requested data dynamically.
4. APIs manage external functions like shipment tracking and payment processing.

4. Tools & Technologies

- **Frontend Framework**: Next.js
- **Backend System**: Sanity CMS
- **Third-Party APIs**: Payment gateways, shipment tracking
- **Design Tools**: Figma, Lucidchart

1. Workflow for Order Management

When a user places an order, the process involves several key steps to ensure seamless interaction between components:

1. **Order Placement**:
   a. The order details are sent to Sanity CMS via an API request.
   b. Sanity CMS records the order information.
2. **Shipment Tracking**:
   a. Shipment tracking information is fetched in real-time through a Third-Party API.
   b. This data is displayed to the user on the frontend for updates on order delivery.
3. **Payment Processing**:
   a. Payment details are securely processed through a Payment Gateway.
   b. A confirmation is sent back to the user and recorded in Sanity CMS.

This workflow highlights how components like Sanity CMS, APIs, and payment gateways work together in real-world scenarios, ensuring smooth data flow and clear integration points.

2. Example System Architecture

A sample system architecture can include the following components:

- **Frontend (Next.js)**: Handles user interactions and displays data.
- **Sanity CMS**: Acts as the backend for storing and managing product, user, and order data.
- **Third-Party APIs**: Manage external services like shipment tracking and payment gateways.

Diagram Representation:

```
css
CopyEdit
[Frontend (Next.js)]
   |      |       |
[Sanity CMS]  [Third-Party APIs]
```

3. Key Workflows to Illustrate

1. **User Registration**:
    a. User signs up.
    b. Data is stored in Sanity CMS.
    c. A confirmation email is sent to the user.
2. **Product Browsing**:
    a. User views product categories.
    b. Sanity API fetches product data.
    c. Products are displayed dynamically on the frontend.
3. **Order Placement**:
    a. User adds items to the cart and proceeds to checkout.
    b. Order details are saved in Sanity CMS.
4. **Shipment Tracking**:
    a. Order status updates are fetched via a Third-Party API.
    b. Updates are displayed to the user in real-time.

4. Plan API Requirements

Based on the defined data schema, the API endpoints can be designed as follows:

- **Example Endpoint for Q-Commerce**:
    - **Endpoint Name**: `/express-delivery-status`
    - **Method**: GET
    - **Functionality**: Fetches the real-time delivery status of an order.

1. API Requirements for Specific Use Cases

Q-Commerce Example:

- **Endpoint Name**: `/express-delivery-status`
- **Method**: GET
- **Description**: Fetch real-time delivery updates for perishable items.
- **Response Example**:

```
json
CopyEdit
{
  "orderId": 123,
  "status": "In Transit",
  "ETA": "15 mins"
}
```

Rental eCommerce Example:

- **Endpoint Name**: `/rental-duration`
- **Method**: POST
- **Description**: Add rental details for a specific product.
- **Payload**:

```
json
CopyEdit
{
  "productId": 456,
  "duration": "7 days",
  "deposit": 500
}
```

- **Response Example**:

```
json
CopyEdit
{
  "confirmationId": 789,
  "status": "Success"
}
```

General eCommerce Example:

- **Endpoint Name**: `/products`
- **Method**: GET
- **Description**: Fetch all product details.
- **Response Example**:

```
json
CopyEdit
{
  "id": 1,
  "name": "Product A",
  "price": 100
}
```

2. API Endpoints Aligned with Marketplace Workflows

Endpoint: /products

- **Method**: GET
- **Description**: Fetch all available products from Sanity CMS.
- **Response Example**:

```
json
CopyEdit
{
  "id": 1,
  "name": "Product A",
  "price": 100,
  "stock": 20,
  "image": "image_url"
}
```

Endpoint: /orders

- **Method**: POST
- **Description**: Create a new order in Sanity CMS.
- **Payload**:

```
json
CopyEdit
{
  "customerInfo": {
    "name": "John Doe",
    "email": "john.doe@example.com"
  },
  "productDetails": [
    {
      "id": 1,
      "quantity": 2
    }
  ],
  "paymentStatus": "Paid"
}
```

- **Response Example**:

```
json
CopyEdit
{
  "orderId": 456,
  "status": "Order Created"
}
```

Endpoint: /shipment

- **Method**: GET
- **Description**: Track order status via a third-party API.
- **Response Example**:

```
json
CopyEdit
{
  "shipmentId": "SH123",
  "orderId": 456,
  "status": "Out for Delivery",
  "expectedDeliveryDate": "2025-01-20"
}
```

Marketplace Technical Foundation - [Your Marketplace Name]

1. System Architecture Overview

Architecture Diagram

A high-level system architecture showing the interaction between components:

```
css
CopyEdit
[Frontend (Next.js)]
     ↕
[Sanity CMS] ↔ [Third-Party APIs]
```

Component Descriptions

- **Frontend (Next.js)**: Handles user interactions, product browsing, cart management, and order placement.
- **Sanity CMS**: Acts as the backend for managing product data, customer details, order records, and additional workflows like rentals or express delivery.
- **Third-Party APIs**: Integrates functionalities for shipment tracking, payment processing, and other backend services.

2. Key Workflows

General eCommerce Workflows

1. **Product Browsing**:
    a. User navigates the marketplace.
    b. Frontend requests product data from Sanity CMS using `/products` endpoint.
    c. Data is displayed dynamically.
2. **Order Placement**:
    a. User adds items to the cart.

   b. Order details (user info, product list) are sent to Sanity CMS via `/orders` endpoint.
   c. Payment Gateway processes the transaction and confirms payment.
3. **Shipment Tracking**:
   a. Order status is fetched through `/shipment` endpoint from a Third-Party API.
   b. Status updates are displayed to the user in real-time.

Q-Commerce Workflows

1. **Express Delivery**:
   a. User places an order for perishable items.
   b. Real-time delivery updates are fetched via `/express-delivery-status` endpoint.
2. **Inventory Management**:
   a. Sanity CMS updates inventory in real-time to reflect availability.
   b. API ensures synchronization between CMS and frontend.

Rental eCommerce Workflows

1. **Rental Order Placement**:
   a. User selects a product and rental duration.
   b. Rental details (duration, deposit) are sent to Sanity CMS using `/rental-duration` endpoint.
   c. Sanity CMS calculates rental cost and stores details.
2. **Return Management**:
   a. Upon return, Sanity CMS updates `conditionStatus` and clears deposit refunds.

3. API Specification Document

General Endpoints

1. **/products**
   a. **Method**: GET
   b. **Description**: Fetch all product listings.
   c. **Response**:

```
json
```

```
CopyEdit
{
  "id": 1,
  "name": "Product A",
  "price": 100,
  "stock": 20,
  "image": "image_url"
}
```

2. **/orders**
    a. **Method**: POST
    b. **Description**: Create a new order.
    c. **Payload**:

```
json
CopyEdit
{
  "customerInfo": {
    "name": "John Doe",
    "email": "john.doe@example.com"
  },
  "productDetails": [
    {
      "id": 1,
      "quantity": 2
    }
  ],
  "paymentStatus": "Paid"
}
```

3. **/shipment**
    a. **Method**: GET
    b. **Description**: Fetch shipment status.
    c. **Response**:

```
json
CopyEdit
{
  "shipmentId": "SH123",
  "orderId": 456,
  "status": "Out for Delivery",
  "expectedDeliveryDate": "2025-01-20"
}
```

Category-Specific Endpoints

1. **Q-Commerce**:
    a. **/express-delivery-status**
        i. **Method**: GET
        ii. **Description**: Fetch real-time tracking details for perishable items.
        iii. **Response**:

```
json
CopyEdit
{
  "orderId": 123,
  "status": "In Transit",
  "ETA": "15 mins"
}
```

2. **Rental eCommerce**:
    a. **/rental-duration**
        i. **Method**: POST
        ii. **Description**: Save rental duration details for a product.
        iii. **Payload**:

```
json
CopyEdit
{
  "productId": 456,
  "duration": "7 days",
  "deposit": 500
}
```

4. Data Schema Design

Sanity CMS Schemas

1. **Products Schema**:
    a. Fields: `id`, `name`, `price`, `description`, `stock`, `image`.
2. **Orders Schema**:
    a. Fields: `orderId`, `customerInfo`, `productDetails`, `paymentStatus`, `orderStatus`.
3. **Rentals Schema** (Rental-Specific):
    a. Fields: `productId`, `rentalDuration`, `depositAmount`, `conditionStatus`.

5. Technical Roadmap

1. **Phase 1**:
    a. Finalize data schema in Sanity CMS.
    b. Build frontend structure in Next.js.
2. **Phase 2**:
    a. Integrate API endpoints for products, orders, and shipments.
    b. Test real-time workflows (e.g., express delivery, rental processing).
3. **Phase 3**:
    a. Deploy the system on a cloud platform.
    b. Monitor API and CMS interactions for performance.

4. API Endpoints

Endpoint	Method	Purpose	Response Example
`/products`	GET	Fetches all product details	`{ "id": 1, "name": "Product A", "price": 100 }`
`/orders`	POST	Creates a new order	`{ "orderId": 123, "status": "Success", "details": {...} }`
`/shipment`	GET	Tracks order shipment	`{ "shipmentId": "SH123", "status": "In Transit" }`
`/rental-duration`	POST	Adds rental details for a product	`{ "confirmationId": 789, "status": "Success" }`
`/express-delivery-status`	GET	Fetch real-time delivery updates	`{ "orderId": 123, "status": "In Transit", "ETA": "15 mins" }`

5. Sanity Schema Example

Product Schema

The following schema can be used in Sanity CMS to manage product data:

```
javascript
CopyEdit
export default {
  name: 'product',
  type: 'document',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Product Name'
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price'
    },
    {
      name: 'stock',
      type: 'number',
      title: 'Stock Level'
    },
    {
      name: 'image',
      type: 'image',
      title: 'Product Image',
      options: { hotspot: true }
    },
    {
      name: 'description',
      type: 'text',
      title: 'Product Description'
    }
  ]
};
```

6. Collaboration and Refinement

1. Group Discussions

- **Organize Brainstorming Sessions**:
- Use tools like **Slack**, **Discord**, or **Google Meet** to discuss ideas and solve challenges.
- Focus topics:
    - Innovative system architecture designs.
    - Efficient API integrations.
- **Exchange Ideas**:

Share your understanding of workflows and technical approaches with your peers.

2. Peer Review

- **Constructive Feedback**:

Share your technical plans with teammates or mentors for insights and suggestions.

- **Review Focus Areas**:
    - Data schemas: Ensure all fields are comprehensive and necessary.
    - API designs: Check endpoint clarity, payloads, and responses.
    - Documentation: Ensure readability and proper structuring.

3. Version Control

- **Use GitHub or Similar Platforms**:
  - Utilize platforms like **GitHub**, **GitLab**, or **Bitbucket** to track changes and collaborate effectively.
  - **Commit Regularly**: Ensure consistent updates to track progress.
  - **Write Clear Commit Messages**: Use descriptive commit messages to maintain transparency (e.g., "Added API endpoint schema for orders").

4. Divide and Conquer

- **Group Brainstorming**:
  - Collaborate on general ideas, system designs, and solutions for common challenges.
  - Use shared tools like Google Docs or whiteboarding platforms (e.g., Miro).
- **Individual Work**:
  - Each team member creates their own document for submission, reflecting their unique understanding and solutions.
  - Encourage creativity while adhering to the shared framework and goals.

5. Submission Requirements

- **Individual Submissions**:
  - Ensure each submission represents your own analysis and planning.
  - Collaboration should inform your work but not duplicate it.

Collaboration & Refinement

- **Discuss and Review**:
  - **Key Topics**:
    - Innovative solutions to technical challenges.
    - System architecture improvements.
    - API design enhancements for better scalability or performance.
- **Refinement Areas**:

- Focus on **scalability**, **performance**, and **clarity** of workflows and documentation.

Key Outcomes of Day 2

1. **Technical Plan Aligned with Business Goals**:
   a. A detailed technical plan tailored to the marketplace type (Q-Commerce, Rental eCommerce, or General eCommerce).
   b. Plans should address all relevant business needs effectively.
2. **System Architecture Visualized**:
   a. A diagram illustrating interactions between the **frontend**, **Sanity CMS**, and **third-party APIs**.
   b. Include specific workflows:
      i. **Q-Commerce**: Real-time inventory updates, delivery tracking, express workflows.
      ii. **Rental eCommerce**: Rental duration tracking, return management, condition reporting.
      iii. **General eCommerce**: Product browsing, cart management, and checkout processes.
3. **Clear Documentation**:
   a. Final document with headings, diagrams, and workflows to reflect a professional technical foundation.

Key Outcomes of Day 2

1. **Marketplace-Specific Workflows Documented**:
    a. **Q-Commerce**: Real-time inventory updates, SLA tracking, and delivery workflow optimization.
    b. **Rental eCommerce**: Rental duration management, deposit handling, and item condition tracking.
    c. **General eCommerce**: Standard processes like product browsing, cart management, and order placement.
2. **System Architecture & Workflows Visualized**:
    a. Clear diagrams showing how **frontend**, **Sanity CMS**, and **third-party APIs** interact.
    b. Marketplace-specific workflows incorporated for practical application.
3. **Detailed API Requirements Documented**:
    a. Comprehensive list of endpoints, methods, payloads, and responses, including marketplace-specific examples:
        i. **Q-Commerce**: `/express-delivery-status` for real-time delivery updates.
        ii. **Rental eCommerce**: `/rental-duration` for handling rental timelines and deposits.
        iii. **General eCommerce**: `/products` for fetching product details.
4. **Sanity Schemas Drafted**:
    a. Schemas designed for core entities such as **products**, **orders**, and **customers**, tailored to marketplace needs.
    b. Example for Products:

```
javascript
CopyEdit
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Product Name' },
    { name: 'price', type: 'number', title: 'Price' },
    { name: 'stock', type: 'number', title: 'Stock Level' }
  ]
};
```

5. **Collaborative Feedback Incorporated**:
    a. Refinements based on peer and mentor input to improve clarity, scalability, and alignment with goals.
6. **Portfolio-Ready Submission**:
    a. A polished, professional document showcasing technical skills and marketplace understanding, suitable for job interviews and client presentations.

Industry Best Practices

1. **Plan Before You Code**:
    a. Develop a clear roadmap to streamline implementation and reduce rework.
2. **Leverage the Right Tools**:
    a. **Sanity CMS**:
        i. Customizable schemas to manage data effectively.
        ii. Built-in APIs for seamless backend integration.
        iii. Real-time collaboration for efficient team workflows.
    b. **Third-Party APIs**: Simplify complex tasks like shipment tracking and payment processing.
3. **Focus on Frontend Innovation**:
    a. Reduce backend complexity using CMS and APIs, allowing more time for creating dynamic, user-friendly interfaces.

Setting Up for Day 3

- **Goal**: Start implementing the technical plan.
- **Focus Areas**:
    - Establish key workflows and endpoints.
    - Begin frontend-backend integration using Sanity CMS and third-party APIs.
    - Ensure alignment with business goals and marketplace requirements.