Hackathon Day 2:

"Marketplace Technical Foundation[SHOP.CO]

Technical Foundation Successfully Completed!

Another exciting and productive day at the Hackathon as I completed Day 2, focusing on the Technical Planning & System Architecture for our e-commerce platform. Today was all about laying the groundwork for a scalable, efficient, and high-performing system! 🎯

Key Achievements of Day 2:

1️⃣Defining Technical Requirements 🛠

The foundation of any great product lies in a well-defined technical structure. Today, I outlined:

Frontend Requirements:

Built with Next.js for a seamless and fast UI experience.
A fully responsive design to ensure compatibility across devices.
Core pages like Homepage, Product Listing, Product Details, Cart, Checkout, and Order Confirmation.

Backend & Database (Sanity CMS):

Sanity CMS serves as the backend database, handling product data, customer information, and orders efficiently.
Defined schemas in Sanity to align with the business goals.
Structured content management for easy updates and dynamic product handling.

Third-Party API Integrations:

Payment Gateway Integration (Stripe, PayPal, etc.) for smooth transaction processing.
Shipment Tracking API to provide real-time order tracking for customers.

Designing the System Architecture 🔍

📌 Frontend (Next.js) – User-facing application handling product browsing, cart management, and checkout.
📌 Sanity CMS – Serves as the database for product and order management.
📌 Third-Party APIs – Handles external services like shipment tracking and payments.

Data Flow:

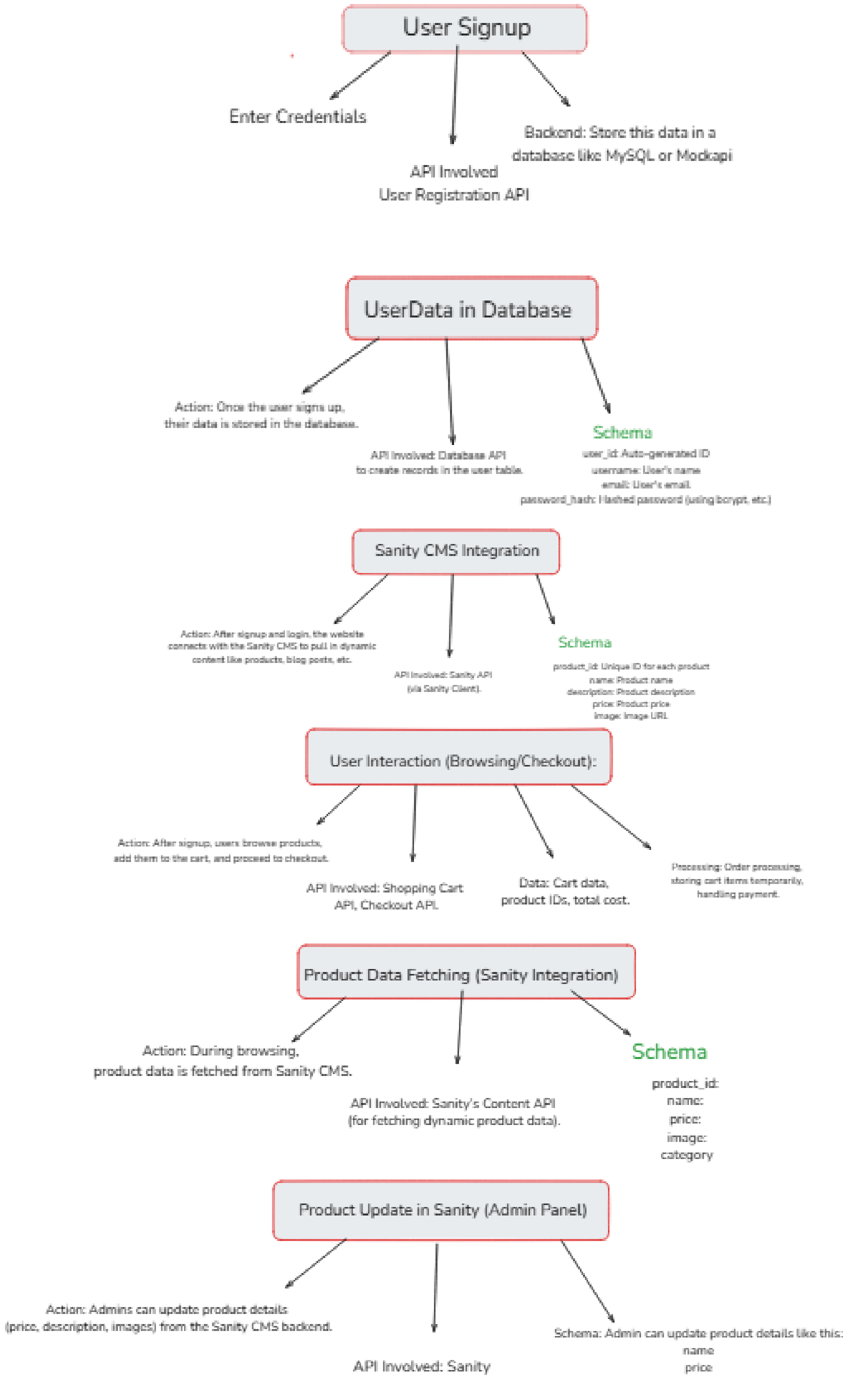
1️⃣User browses the marketplace on Next.js frontend.
2️⃣The frontend fetches product data from Sanity CMS via API requests.
3️⃣When an order is placed, the details are stored in Sanity CMS.
4️⃣Shipment tracking API fetches real-time order status updates.
5️⃣Payment gateway API processes secure transactions.
6️⃣The system updates and syncs data between frontend, backend, and external APIs.
A strong architecture ensures scalability, efficiency, and seamless user experience! 🔥

Planning API Requirements 🌐

A well-structured API is critical for smooth communication between different components. I outlined key RESTful APIs:

📌 **GET /products** → Fetch all available products from Sanity CMS.
📌 **POST /orders** → Create a new order with customer details, product data, and payment status.
📌 **GET /shipment** → Retrieve order tracking details from the shipment tracking API.
Each API is designed to optimize performance, data accuracy, and security, ensuring a seamless shopping experience.

Each step is carefully planned to enhance user experience and system efficiency! 💡



User Signup

Enter Credentials

API Involved
User Registration API

Backend: Store this data in a
database like MySQL or Mockapi

UserData in Database

Action: Once the user signs up,
their data is stored in the database.

API Involved: Database API
to create records in the user table.

Schema
user_id: Auto-generated ID
username: User's name
email: User's email
password_hash: Hashed password (using bcrypt, etc.)

Sanity CMS Integration

Action: After signup and login, the website
connects with the Sanity CMS to pull in dynamic
content like products, blog posts, etc.

API Involved: Sanity API
(via Sanity Client).

Schema
product_id: Unique ID for each product
name: Product name
description: Product description
price: Product price
image: Image URL

User Interaction (Browsing/Checkout):

Action: After signup, users browse products,
add them to the cart, and proceed to checkout.

API Involved: Shopping Cart
API, Checkout API.

Data: Cart data,
product IDs, total cost.

Processing: Order processing,
storing cart items temporarily,
handling payment.

Product Data Fetching (Sanity Integration)

Action: During browsing,
product data is fetched from Sanity CMS.

API Involved: Sanity's Content API
(for fetching dynamic product data).

Schema
product_id:
name:
price:
image:
category

Product Update in Sanity (Admin Panel)

Action: Admins can update product details
(price, description, images) from the Sanity CMS backend.

API Involved: Sanity

Schema: Admin can update product details like this:
name
price