Muhammad Anas
Friday 9-12
Roll : 55344

Hekto Technical Foundation

Day 2 : Transitioning to Technical Planning

1. Defining Technical Requirements

To transform business goals into a structured technical framework, we have identified key requirements for the frontend, backend (Sanity CMS), and third-party API integrations.

Frontend Requirements:

- A user-friendly and visually engaging interface for product browsing.
- Responsive design ensuring a seamless experience on mobile and desktop.
- Core pages:
  - Home
  - Product Listing
  - Product Details
  - Cart
  - Checkout
  - Order Confirmation

Backend (Sanity CMS) Requirements:

- Sanity CMS will act as the backend to store product data, customer details, and order records.
- Custom schemas will be designed to align with business objectives outlined on Day 1.
- Real-time content updates via Sanity's built-in API.

Third-Party API Integrations:

- **Shipment Tracking API:** To provide real-time updates on order delivery.
- **Payment Gateway API:** Secure and seamless transaction processing.
- **Inventory Management API:** To update stock levels dynamically.

2. System Architecture Design

A high-level system architecture is designed to define interactions between components.

[Frontend (Next.js)]

|

|--> [Sanity CMS] ------> [Product Data API]

| |

| |--> [Third-Party APIs]

| |--> [Shipment Tracking API]

| |--> [Payment Gateway]

Typical Workflow:

1. Users browse the marketplace frontend, fetching product details from the Product Data API.
2. When a user places an order, order details are sent to Sanity CMS via API.
3. Shipment status is retrieved from the third-party API and displayed to the user.
4. Secure payment transactions are processed through the Payment Gateway.

3. API Requirements & Endpoints

Based on the data schema, API endpoints are structured as follows:

Endpoint	Method	Purpose	Request Payload	Response Example
/products	GET	Fetch all product details	N/A	{ "id": 1, "name": "Sofa", "price": 500 }
/product/:id	GET	Fetch single product details	Product ID	{ "id": 1, "name": "Sofa", "price": 500, "stock": 10 }
/orders	POST	Create a new order	Customer info, product details	{ "orderId": 123, "status": "Processing" }
/shipment	GET	Track order shipment status	Order ID	{ "orderId": 123, "status": "In Transit", "ETA": "15 mins" }

4. Sanity Schema Example

Sanity CMS schemas are designed to manage critical data entities efficiently. Below is an example of the product schema:

```
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: 'category', type: 'string', title: 'Category' },
    { name: 'image', type: 'image', title: 'Product Image' }
  ]
};
```

5. API Usage Flexibility

- **Read-Only API Access:** The provided API will be read-only, allowing flexibility for developers to design their own schemas in Sanity CMS.
- **Custom API Implementation:**
  - **Q-Commerce:** Manage perishable goods and SLA tracking.
  - **Rental eCommerce:** Incorporate rental-specific fields (e.g., rental duration, deposit, and condition reports).
  - **General eCommerce:** Implement core workflows such as inventory management, order placement, and payment processing.

The choice is yours—leverage the provided API schema or innovate with your own to meet your marketplace's unique needs.

6. Collaboration and Refinement

To ensure quality and maintain best practices, collaborative methods include:

- **Brainstorming Sessions:** Discussions via Slack, Discord, or Google Meet.
- **Peer Reviews:** Sharing technical documents with team members for feedback.
- **Version Control:** Using GitHub for tracking changes and maintaining transparency.
- **Task Allocation:** Dividing tasks while maintaining a consistent project vision.

7. Submission Guidelines

Document Structure:

- System Architecture Overview (with a visual diagram)
- Key Workflows (e.g., Product Browsing, Order Placement, Shipment Tracking)
- API Endpoints (detailed table format)
- Sanity Schema Design
- Collaboration Notes (summary of discussions and feedback)

File Naming Conventions:

- SystemArchitecture_Hekto.pdf
- APIEndpoints_Hekto.xlsx
- SanitySchema_Hekto.js

8. Key Outcomes of Day 2

By the end of Day 2, the following milestones should be achieved:

1. **Technical Plan Aligned with Business Goals:** A structured plan ensuring the marketplace meets real-world needs.
2. **System Architecture Visualized:** Clear representation of interactions between the frontend, Sanity CMS, and third-party APIs.
3. **Detailed API Documentation:** Well-defined endpoints for smooth development.
4. **Sanity Schemas Drafted:** Tailored schemas managing product, order, and customer data.
5. **Collaborative Feedback Incorporated:** Refinements based on peer and mentor reviews.
6. **Portfolio-Ready Documentation:** A professional-level submission demonstrating expertise in technical planning.

9. Industry Best Practices

1. **Plan Before You Code:** Create a roadmap to avoid rework.
2. **Leverage the Right Tools:** Use Sanity CMS for backend efficiency and third-party APIs for key integrations.
3. **Emphasize Collaboration:** Incorporate feedback from team members and mentors.
4. **Enhance User Experience:** Ensure technical decisions align with seamless customer interactions.