Day 2 - Marketplace Technical Foundation

1. System Architecture

System architecture outlines the components of the e-commerce platform and their interactions, ensuring seamless functionality.

Components

1. **Frontend (Next.js):**
    - Interactive user interface for browsing products, managing the cart, and completing checkout.
2. **Backend (API):**
    - Processes user requests, handles logic, and communicates with the database and external services.
3. **Database (Sanity CMS):**
    - Stores data such as products, customer details, and orders. Enables content management and updates.
4. **Third-Party APIs:**
    - Handles payments (e.g., Stripe) and shipment tracking.

How It Works (Interactions)

1. **Frontend to Backend:** Sends API requests for data (e.g., product details).
2. **Backend to Database:** Manages data storage and retrieval in Sanity CMS.
3. **Backend to Third-Party APIs:** Processes payments, fetches shipping details, and updates order status.

2. Key Workflows

Frontend (Next.js):

- User signs up → Data stored in Sanity CMS.
- User browses products → Product data fetched from Sanity CMS.
- User places an order → Order details sent to Sanity CMS.

Sanity CMS:

- Stores user, product, and order data.
- Serves as the database for frontend requests and backend processes.

Third-Party APIs:

- Fetch shipment updates and payment statuses.
- Provide real-time order and delivery information.

Flow Example:

1. User signs up → Data saved in Sanity CMS.
2. User browses products → Products displayed dynamically via fetched data.
3. Order placement:
   - Order saved to Sanity CMS.
   - Payment processed via third-party API.
   - Shipping info fetched and updated via third-party API.
4. User receives confirmation of signup, payment, and order updates.

3. Category-Specific Instructions

General E-commerce Workflows:

1. **Product Browsing:**
   - Filter products by categories (e.g., electronics, furniture).
2. **Cart Management:**
   - Add, update, or remove items.
3. **Order Placement:**
   - Includes checkout, payment, and confirmation processes.

4. API Endpoints

Endpoint	Method	Purpose	Response Example
`/products`	GET	Fetch all product details.	`{ "id": 1, "name": "Chair", "price": 250 }`
`/orders`	POST	Save order details.	`{ "orderId": 123, "status": "Pending" }`
`/customers`	POST	Save customer details.	`{ "customerId": 456, "name": "John Doe" }`
`/payment`	POST	Process payment.	`{ "orderId": 123, "status": "Success" }`
`/shipment-status`	GET	Fetch shipment updates.	`{ "orderId": 123, "status": "In Transit" }`

5. Sanity Schema

Product Schema:

```
const ProductSchema = {
  name: "product",
  type: "document",
  fields: [
    { name: "name", type: "string", title: "Product Name" },
    { name: "price", type: "number", title: "Price" },
    { name: "image", type: "image", title: "Image" },
    { name: "stock", type: "number", title: "Stock Level" },
    { name: "description", type: "text", title: "Description" }
  ]
};
export default ProductSchema;
```

Order Schema:

```
const OrderSchema = {
  name: "order",
  type: "document",
  fields: [
    { name: "orderId", type: "string", title: "Order ID" },
    { name: "customer", type: "reference", to: [{ type: "customer" }], title:
"Customer" },
    { name: "products", type: "array", of: [{ type: "reference", to: [{ type:
"product" }] }], title: "Products" },
    { name: "totalAmount", type: "number", title: "Total Amount" },
    { name: "status", type: "string", title: "Order Status" }
  ]
};
export default OrderSchema;
```