Step-by-Step Day 2 Plan

1. Define Technical Requirements
Based on Day 1 brainstorming:

Frontend Requirements:

Pages Needed:
Home: Displays product categories like jackets and shoes.
Product Listing: Shows all available products with filters (price, size, etc.).
Product Details: Displays detailed information about a specific item.
Cart: Lets users view selected items before checkout.
Checkout: Processes orders and collects payment/shipping info.
Order Confirmation: Shows order details and status.

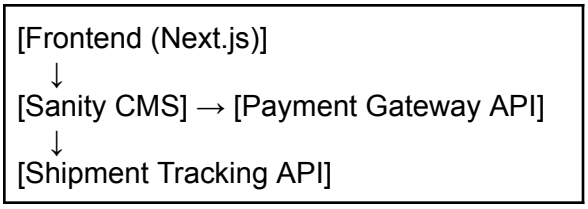
Backend Requirements:

Use Sanity CMS to:
Store products with fields like name, price, condition, and seller info.
Save orders with buyer details, product IDs, and order status.
Third-Party APIs:

Payment gateway (e.g., JazzCash, EasyPaisa).
Shipment tracking (integrate with a logistics API in Pakistan).
2. Design System Architecture
Visualize How the System Works:

Frontend: Built with Next.js to provide a user-friendly interface.
Sanity CMS: Serves as the database for storing product and order details.
Third-Party APIs: Handles payment and shipment tracking.
Workflow Example:

User visits the website and browses products.
Product data is fetched from Sanity CMS and displayed.
User places an order → Order saved in Sanity CMS.
Payment processed via API.
Shipment tracked via API and displayed to the user.
Diagram:



3. Plan API Requirements

Design APIs based on the workflows:

Fetch Products:
Endpoint: /products
Method: GET
Purpose: Retrieve all product details.
Response Example: { id: 1, name: "Leather Jacket", price: 2000 }

Create Order:
Endpoint: /orders
Method: POST
Purpose: Save a new order.
Payload Example: { customerInfo, items, paymentStatus }

Track Shipment:
Endpoint: /shipment
Method: GET
Purpose: Fetch shipment status.
Response Example: { orderId: 1, status: "In Transit" }

4. Create Sanity CMS Schema
Define schemas for storing data in Sanity CMS:

Product Schema:

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title:
'Product Name' },
    { name: 'price', type: 'number', title:
'Price' },
    { name: 'condition', type: 'string', title:
'Condition (New/Used)' },
    { name: 'category', type: 'string', title:
'Category (Jackets, Shoes)' },
    { name: 'image', type: 'image', title:
'Product Image' },
```

```
  ],
};
```

Order Schema:

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer', type: 'string', title:
'Customer Name' },
    { name: 'items', type: 'array', of: [{
type: 'reference', to: [{ type: 'product' }]
}], title: 'Ordered Items' },
    { name: 'status', type: 'string', title:
'Order Status (Pending, Shipped,
Delivered)' },
  ],
};
```

5. Document Everything

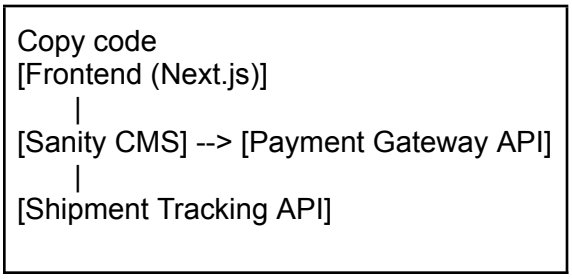
1.Introduction:
Marketplace Technical Foundation - LundaOnline
This document outlines the technical foundation for **LundaOnline**, an online marketplace designed to provide affordable second-hand items like jackets, shoes, and other essentials imported from Western countries.
Our goal is to create a user-friendly platform that allows middle-class individuals in Pakistan to access high-quality products conveniently, mirroring the experience of Lunda Markets in an online setting.

2.System Architecture:

System Architecture Overview
The system is composed of three main components:

1.Frontend (Next.js): The user interface where customers browse products and place orders.
2.**Sanity CMS**: Backend for managing product details, customer data, and order records.
3.
Third-Party APIs: For processing payments and tracking shipments.
Workflow:

The customer visits the website and browses the available products.
Product data is fetched from Sanity CMS and displayed on the frontend.
When an order is placed, the details are stored in Sanity CMS.
Payments are processed via a payment gateway API.
Shipment details are retrieved from a logistics API and shared with the customer.
Diagram: (You can draw this on paper or use a tool like Lucidchart or Figma.)



3.API List:
API Endpoints

1. **Fetch Products**
    - **Endpoint:** `/products`
    - **Method:** GET
    - **Purpose:** Retrieve all product details.

Response Example:

```
{
  "id": 1,
  "name": "Leather Jacket",
  "price": 2000,
  "category": "Jackets"
}
```

    -
2. **Create Order**
    - **Endpoint:** `/orders`
    - **Method:** POST
    - **Purpose:** Save a new order.

Payload Example:

```
{
  "customerInfo": {
    "name": "Ali",
    "address": "Karachi"
  },
  "items": [
    { "id": 1, "quantity": 2 }
  ],
  "paymentStatus": "Paid"
}
```

3. **Track Shipment**
    - **Endpoint:** `/shipment`
    - **Method:** GET
    - **Purpose:** Fetch shipment status.

Response Example:

```
{
  "orderId": 123,
  "status": "In Transit",
  "deliveryDate": "2025-01-20"
}
```

---

4. Sanity Schema

Product Schema:

```
export default {
  name: 'product',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title:
```

```
'Product Name' },
    { name: 'price', type: 'number', title:
'Price' },
    { name: 'condition', type: 'string',
title: 'Condition (New/Used)' },
    { name: 'category', type: 'string', title:
'Category (e.g., Jackets, Shoes)' },
    { name: 'image', type: 'image', title:
'Product Image' }
  ]
};
```

Order Schema:

```
export default {
  name: 'order',
  type: 'document',
  fields: [
    { name: 'customer', type: 'string',
title: 'Customer Name' },
    { name: 'items', type: 'array', of: [{
type: 'reference', to: [{ type: 'product'
}] }], title: 'Ordered Items' },
    { name: 'status', type: 'string', title:
'Order Status (Pending, Shipped,
Delivered)' }
  ]
};
```
