# Below is a technical plan for your website, aligned with the requirements and structure provided for Hackathon Day 2:

**Marketplace Technical Foundation - [Your Marketplace Name]**

## Day 2 Goal:

The objective of this plan is to define the technical foundation for building a scalable and user-friendly marketplace. This includes system architecture, API requirements, workflows, and documentation to transition smoothly to the implementation phase.

## Recap of Day 1: Business Focus

**Key Achievements:**

**Business Goals Defined:**

Solving the problem of accessible and reliable [specific niche].

Target audience: [Demographic details, e.g., tech enthusiasts, small business owners].

Unique Value Proposition (UVP): [Unique features of your marketplace].

## Preliminary Data Schema:

Key entities include:

Products: ID, Name, Price, Stock, Image URL, Description.

Users: ID, Name, Email, Password (hashed), Address.

Orders: ID, User ID, Product Details, Total Price, Order Status.

## Single Focus:

Ensured alignment of technical and business objectives.

## 1. Define Technical Requirements

**Frontend Requirements:**

User-friendly interface using Next.js and Tailwind CSS.

Responsive design optimized for mobile and desktop users.

### Key Pages:

**Home Page:** Display featured categories and products.

**Product Listing Page:** Filter and search products.

**Product Details Page:** Detailed view of selected product.

**Cart:** Display selected products and total cost.

**Checkout:** User information, payment, and confirmation.

**Order Confirmation Page:** Show success message and order details.

Sanity CMS as Backend

### Schemas:

Products: Name, Price, Description, Stock, Category.

Users: Name, Email, Orders.

Orders: Product IDs, Quantities, User Details.

Leverage Sanity Studio for real-time updates and ease of management.

### Third-Party APIs

Payment Gateway Integration (e.g., Stripe, PayPal): Ensure secure payment processing.

Shipment Tracking API: Real-time order tracking.

Email Notifications: Trigger transactional emails for order confirmations.

## 2. Design System Architecture

Architecture Diagram

```
 [Frontend (Next.js)] <---> [Sanity CMS (Backend)]
       |                       |
[Payment Gateway API]     [Shipment Tracking API]
```

Workflows

**User Registration:**

User signs up → Data saved in Sanity CMS → Email confirmation sent.

**Product Browsing:**

Frontend requests product data → Sanity API fetches → Displays on UI.

**Order Placement:**

User adds items to the cart → Proceeds to checkout → Order saved in Sanity CMS.

**Payment Processing:**

Payment Gateway processes the transaction → Updates order status.

**Shipment Tracking:**

Fetch status from API → Updates user in real-time.

## 3. Plan API Requirements

Endpoints

| Endpoint Name | Method | Description | Payload/Response Example |
|---|---|---|---|
| /products | GET | Fetch all product data | Response:<br><br>{ "id": 1,<br>"name": "Product A",<br>"price": 100,<br>"stock": 50 } |
| /products/: id | GET | Fetch specific product details | Response:<br><br>{ "id": 1,<br> "name": "Product A",<br>"price": 100,<br>"description":<br>"Details" } |
| /orders | POST | Create a new order | Payload:<br>{ "userId": 123,<br>"products": [...],<br>"totalPrice": 200 } |

/orders/: id/status GET    Fetch order status  Response:

{ "orderId": 123,

"status": "Shipped", "ETA": "2 days" }

/payment     POST  Process payment    Payload:

{ "amount": 200,

"paymentMethod": "card" },

Response: { "status": "Paid" }

## 4. Write Technical Documentation

System Architecture Document

**Components:**

Frontend (Next.js): Dynamic rendering, user interactions.

Backend (Sanity CMS): Data storage, API management.

APIs: Payment Gateway, Shipment Tracking.

**Key Interactions:**

Product data fetched from Sanity.

Orders processed and stored in Sanity.

APIs handle payments and tracking.

API Specification Document

## 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: 'description', type: 'text', title: 'Description' },
  ],
};
```

Example Payloads/Responses provided above.

## 5. Collaborate and Refine

Use GitHub for version control and collaboration.

Schedule peer reviews to refine workflows and API designs.

Incorporate feedback into technical documents.

**Key Outcome of Day 2**

Comprehensive Technical Plan aligned with business goals.

System Architecture Visualized with dependencies and workflows.

API Documentation ready for implementation.

Sanity CMS Schema drafted for seamless data management.

Prepared for Day 3 implementation.

This plan ensures a robust technical foundation for your marketplace project, setting it up for success. Let me know if you'd like to expand on any section or need further details!