Hackathon Day 2 "Marketplace Technical Foundation [Rental E-Commerce]"

1. Introduction

- **Goal:**
  To create a technical plan that aligns with the business goals defined on Day 1.
- **Focus Areas:**
  - System Architecture
  - Workflows
  - API Integration

---

2. Business Focus

- **Business Goals Identified:**
  - Problem-solving for customers.
  - Target audience and unique value proposition (e.g., affordability, user-friendly design).
- **Data Schema:**
  - Core entities: **Products**, **Orders**, **Customers**, etc.
  - Relationships between these entities.
- **Single Focus:**
  - A solid foundation to transition into technical planning.

---

3. Technical Preparation

- **System Architecture:**
  - Diagram showing the connection between:
    - Frontend (e.g., React/Next.js).
    - Backend (Sanity CMS for data management).
    - Third-party APIs (e.g., payment gateway, shipment tracking).
- **Workflows:**
  - User Actions:
    1. Product Search → Add to Cart → Checkout.
    2. Payment Processing → Order Confirmation → Shipment Tracking.
- **API Integration:**
  - Example APIs:
    - **Payment Gateway** for secure transactions.
    - **Shipment Tracking** for delivery updates.

---

4. Visual Aids

- Add diagrams:
    - **System Architecture**: Show how components interact.
    - **Workflow Diagram**: Outline user journey step-by-step.

---

5. Conclusion

- Technical plan ensures:
    - Alignment with business goals.
    - A scalable, efficient marketplace.

Frontend Requirements

1. **User-Friendly Interface for Browsing Products**
    - **Tasks:**
        - Design an intuitive navigation bar for easy product browsing.
        - Add search functionality and filters (e.g., price, category, availability).
        - Ensure product cards display key details: name, price, image, and availability.
    - **Tools:**
        - Use **React/Next.js** for dynamic pages.
        - Apply **Tailwind CSS** or **Bootstrap** for styling.
2. **Responsive Design**
    - **Tasks:**
        - Use a mobile-first design approach for scalability.
        - Test on multiple devices for responsiveness.
    - **Tools:**
        - Utilize CSS Grid/Flexbox for layout structure.
        - Leverage responsive breakpoints in Tailwind CSS or custom media queries.
3. **Essential Pages**
    - **Pages to Build:**
        - **Home:** Featured products, categories, and promotions.
        - **Product Listing:** Grid view of all available products with filters and search.
        - **Product Details:** Detailed view with pricing, description, ratings, and availability.
        - **Cart:** List of selected items with the total price and "Proceed to Checkout" button.
        - **Checkout:** Form for user details, payment options, and confirmation.

        - **Order Confirmation:** Summary of the completed order with tracking details.

Sanity CMS as Backend

1. **Manage Product Data**
    - Define a **Product Schema** with fields like:
        - Name, Description, Price, Rental Duration, Availability, and Image.
2. **Customer Details**
    - Create a **Customer Schema** for:
        - Name, Contact Info, and Order History.
3. **Order Records**
    - Build an **Order Schema** to track:
        - Ordered Products, Customer Info, Payment Status, and Delivery Status.
4. **Setup and Integration**
    - Install Sanity CMS:
    - `npm install -g @sanity/cli`
    - `sanity init`
    - Connect Sanity CMS with the frontend using the **Sanity client**:
    - `npm install @sanity/client`

Third-Party APIs

1. **APIs to Integrate**
    - **Payment Gateway:** Use **Stripe** or **PayPal** for secure transactions.
    - **Shipment Tracking:** Integrate APIs like **AfterShip** or **Shippo**.
2. **Steps for Integration**
    - Register and obtain API keys from the selected service.
    - Add API calls to the respective functionality:
        - Payment processing during checkout.
        - Shipment tracking in the order confirmation page.
3. **Testing**
    - Use **Postman** to test API endpoints.
    - Verify API responses align with the frontend's data needs.

Sanity Schema.js

```
export default {
  // Define the document type and its name
  name: 'rentalProduct',
  type: 'document',
```

```
title: 'Rental Product', // The title displayed in the CMS for this document
fields: [
  {
    // Field for the product's name
    name: 'name',
    type: 'string',
    title: 'Product Name',
    validation: (Rule) =>
      Rule.required()
        .max(100)
        .error('Product name is required and cannot exceed 100 characters.'),
  },
  {
    // Slug field for generating a URL-friendly identifier
    name: 'slug',
    type: 'slug',
    title: 'Slug',
    description: 'URL-friendly identifier for the product.',
    options: {
      source: 'name',
      maxLength: 200,
    },
    validation: (Rule) =>
      Rule.required().error('Slug is required for product identification.'),
  },
  {
    // Field for a detailed description of the product
    name: 'description',
    type: 'text',
    title: 'Description',
    description: 'Detailed description of the product.',
    validation: (Rule) =>
      Rule.required()
        .min(20)
        .max(500)
        .error('Description must be between 20 and 500 characters.'),
  },
  {
    // Field for the rental price
    name: 'rentalPrice',
    type: 'number',
    title: 'Rental Price',
    validation: (Rule) =>
      Rule.required()
        .min(0)
        .error('Rental price must be a positive value.'),
```

```
},
{
  // Field for availability status
  name: 'availability',
  type: 'boolean',
  title: 'Availability',
  description: 'Is this product currently available for rent?',
},
{
  // Field for rental duration options
  name: 'rentalDuration',
  type: 'array',
  title: 'Rental Duration Options',
  of: [{ type: 'string' }],
  description: 'Available rental durations (e.g., daily, weekly, monthly).',
  options: {
    layout: 'tags',
  },
},
{
  // Field for the product image
  name: 'image',
  type: 'image',
  title: 'Product Image',
  description: 'High-quality image of the product.',
  options: {
    hotspot: true,
  },
  validation: (Rule) => Rule.required().error('Product image is required.'),
},
{
  // Field for the product category
  name: 'category',
  type: 'string',
  title: 'Category',
  description: 'Category of the rental product (e.g., SUV, Sedan, Luxury).',
  validation: (Rule) => Rule.required().error('Category is required.'),
},
{
  // Field for the product features
  name: 'features',
  type: 'array',
  title: 'Features',
  of: [{ type: 'string' }],
  description: 'Key features of the rental product (e.g., GPS, Air Conditioning).',
  options: {
```

```
      layout: 'tags',
    },
  },
  {
    // Field for the SEO-friendly title
    name: 'seoTitle',
    type: 'string',
    title: 'SEO Title',
    description: 'Title for SEO optimization (max 60 characters).',
    validation: (Rule) => Rule.max(60).error('SEO title cannot exceed 60 characters.'),
  },
  {
    // Field for the SEO-friendly description
    name: 'seoDescription',
    type: 'text',
    title: 'SEO Description',
    description: 'Meta description for SEO optimization (max 160 characters).',
    validation: (Rule) => Rule.max(160).error('SEO description cannot exceed 160 characters.'),
  },
 ],
};
```