HACKATHON 2025 DAY 2 TASK

TECHNICAL REQUIREMENTS FOR CAR RENTAL E-COMMERCE MARKETPLACE

Technical requirements refer to the specific technical details and specifications that a system, application, or platform must meet in order to fulfill its intended purpose and functions

Sania Sajjad

‘

Technical Requirements for

Car Rental E-Commerce Marketplace

This document outlines the essential tools, technologies, and requirements that I will use to build the **Car Rental E-Commerce Marketplace**, ensuring clarity and alignment with the project's business objectives

1. Frontend Requirements

- **Interface:** The interface will be easy to use, making it simple for anyone to interact with the platform.
- I'll organize the products into clear categories, so users can find what they need easily.
- There will be a search bar to quickly look for specific products.
- Users can filter products according to rental duration, price, and category to easily narrow their choices.
- **Responsive:** The platform will be fully accessible for all devices.
- I will make sure the platform works smoothly on both mobile and desktop.
- I'll use Tailwind CSS to create a responsive design that adjusts well on any device

- **Essential Pages:** The platform will be fully accessible for all devices.
- **Home Page**: Showcasing products and a search bar for easy navigation.
- **Product Listing Page:** Displaying products with sort and filter options.

- **Product Details Page:** Providing detailed info, rental terms, availability, and pricing:
- **Cart Page**: Showing selected items with options to adjust or remove.
- **Checkout Page:** Collecting customer details, payment, and delivery preferences.
- **Order Confirmation Page**: Summarizing the order with tracking info.

2. Backend

Sanity CMS will be used as the backend to manage product data, customer details, and order records. The following will be done:

Product Data Management: I will create schemas in Sanity for storing product information such as name, category, description, images, pricing, rental terms, and stock availability.

Customer Data Management: A schema for customer information including name, email, phone number, and rental history will be implemented.

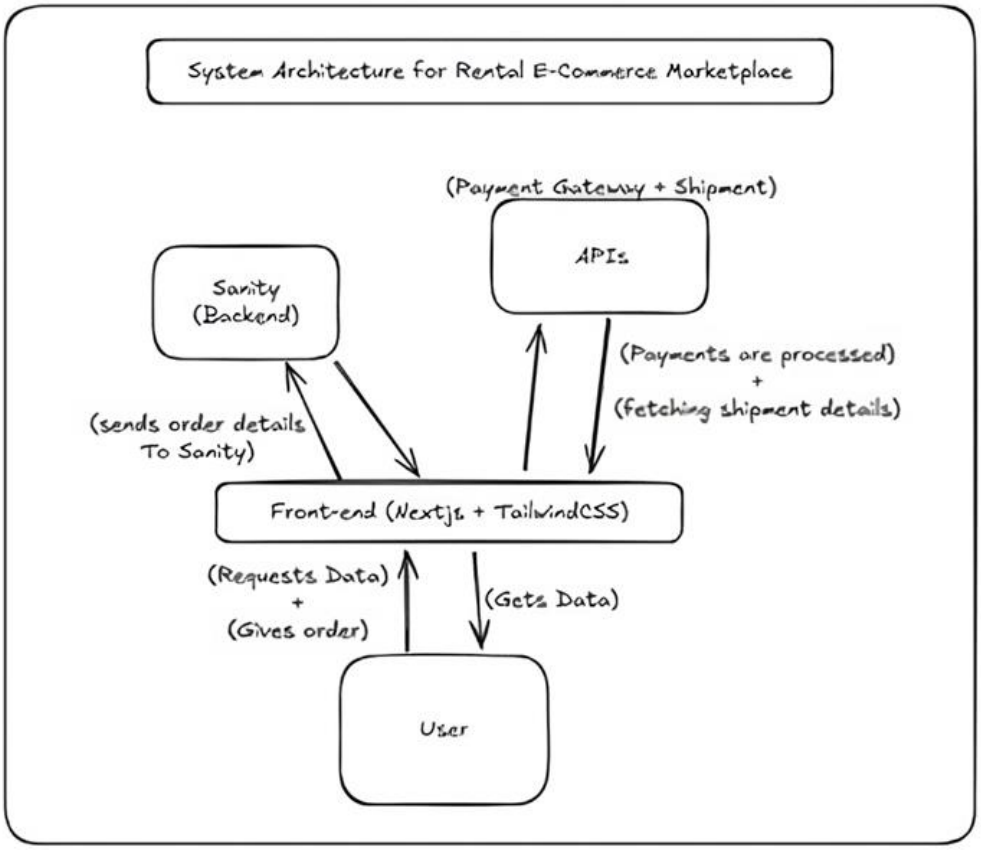
Order Records Management. I will create schemas for tracking orders, including fields for product reference, customer reference, rental duration, total cost delivery/pickup status, and payment status.

3. Third-Party APIs

Several third-party APIs will be integrated to enhance the functionality of the platform

- **Payment Gateway:** I will integrate either **Stripe** or **PayPal** to process payments securely during checkout.
- **Shipment Tracking:** I will use APIs like **Shippo** to provide real-time tracking information for delivered items.

System Architecture:



API Requirements:

1. Endpoint Name:/products

Method: GET

Description: Fetch all available products from Sanity.

Response:

[

]

Id: 123,

"**name**" "Electric Car

"**Price**: 50,

"**stock**": 10.

"**Image**": https://example.com/images/electric-car.log

}
{
“**Id**”: 124,
"**name**": "Hybrid Car
“**price**”: 70
"**stock**" 5
"**Image**": "https://example.com/images/hybrid-car.jpg"
]
[
2. Endpoint Name: /rental-duration
Method: POST
Description: Add rental details for a specific product
Payload:
{
"**productid**": 456,
"**duration**" "7 days",
"**Deposit**: 500,
]
Response:
{
“**confirmation id**”: 789,
"**status**": "Success"
]

3. Endpoint Name:/order
Method: POST
Description: Creates a new order in Sanity.
Payload:
{
"customerinfo": {

"**name**": "John Doe
"**email** "Johndoedexample.com",
"**address**": "123 Main St. City, Country" }
“orderDetails":[
]
"**productid**": 123
"**name**": "Electric Car",
"**rental Duration”:** “7 days”,
“price”: 50 ,
]
Response:
"**orderId”**: 987,
"**status**" "Order Created Successfully”
]
4. Endpoint Name: /shipment
Method: GET
Description: Track order status via third-party API.
Response:
[
"**shipmentid**" "abc123xyz"
"**orderId**": 987
"**status**" "In Transit",
“**expectedDeliveryDate”** : ” 2025-01-20”
]
Workflow Alignment with System Design.
User Actions:
1. Registration
Users sign up and log in using Clerk, which securely manages user authentication and data handling.

2. Exploring Products:
Product data is retrieved via the /products endpoint, with features for filtering by category and sorting by preferences.
3. Adding Items to Cart:
Users interact with the "Cart Page" to view, modify, and manage their selected items.
4. Checkout Process:
Payments are completed securely using Stripe integration through the "Stripe Page
5. Order Creation:
Orders are placed using the /order endpoint, which processes payment details and order information.

Conclusion
This document outlines the key technical requirements and workflows for the Car Rental E-Commerce Marketplace, ensuring a user-friendly, efficient, arid scalable platform. By integrating modem tools and technologies like Next.js, Sanity CMS, and third-party APls, this solution is designed to meet user needs and support seamless operations.
With this foundation, the marketplace is ready to deliver an exceptional experience while providing room for future growth and innovation.

---

1. Executive Summary
This document outlines the technical requirements for the Print My Case project, a print-on-demand marketplace for phone cases. The platform allows users to create custom designs by uploading

features, adding text, changing fonts, and colors. Once finalizing their designs, users can order phone cases for their specific devices. We will implement Stripe for payments, ShipEngine for shipments, and Clerk for user authentication. Sanity CMS will be used for order management, asset management, and database-related tasks.

2. Scope

Print My Case is a POD (Print on Demand) platform for phone covers**.**

2.1 Key Deliverables

- **Design**: A sleek, modern design that is user-friendly and simple to navigate.
- **Frontend:** A responsive frontend built with React.js/Next.js, TailwindCSS, ShadCN UI, Framer Motion, Fabric.js, React Hook Form, TanStack Query, and TypeScript.
- **Backend**: A scalable backend powered by Sanity CMS, Next.js API routes, Stripe for payments, ShipEngine for shipping, and Clerk for authentication. The backend will ensure seamless interaction between the front end, CMS, and external APIs.
- **Features:** Implementation of user authentication (via Clerk), email notifications (using SendGrid), and payment processing with Stripe. Integration of ShipEngine for real- time shipping quotes and order tracking.

3. Functional Requirements

3.1 User Roles

Customer: End-users who browse, customize, and order phone cases.
Seller: Users who supply designs and manage inventory.

3.2 Features

Customer Features:

• Browse/search products by device model or category.
• Customize cases with text, images, and colors via a design canvas.
• Add products to the cart, proceed to checkout, and track orders.
• View order history and receive notifications.
• **Email Resend Feature:**

Customers can request to resend order confirmation or shipping status emails if they haven't received them.

Seller Features:

• Add and manage product templates.
• Track sales and view analytics dashboards.
• Access customer feedback.
• **Email Resend Feature**: Sellers can request to resend order updates to customers.

4. Non-Functional Requirements

- **Performance**: Response time under 2 seconds for major operations.
- **Scalability**: The system supports up to 10,000 concurrent users initially.
- **Security**: Implement secure authentication (OAuth2) and data encryption (HTTPS, AES-256).
- **Compliance**: Adhere to GDPR, PCI- DSS, and local e-commerce laws.
- **Reliability**: Ensure 99.9% uptime with failover mechanisms.

5. System Architecture

5.1 Overview

A high-level architecture diagram includes:

•**Frontend**: React/Next.js application.
•**Backend**: API endpoints powered by Next.js and Sanity CMS.
•**Database**: Sanity CMS will serve as both the content management system and the primary database for managing orders, assets, and other relevant data.
•**Third-Party Services**: Stripe for payments, ShipEngine for shipping, SendGrid for transactional emails, and Clerk for user authentication.

5.2 Component Descriptions

•**Frontend**: Fully responsive with real- time design previews.
•**Backend**: Orchestrates API calls and database interactions with Sanity CMS.
•**APIs**: RESTful endpoints for user authentication, order management, and third-party integrations.
•**CMS**: Sanity for dynamic content and inventory management.

6. API Requirements

6.1 Key Endpoints

Products:
GET /products: Fetch available products.
• GET /products/:id: Fetch product details.
• POST /products: Add a new product template (Seller only).
• **Orders**:
POST/orders: Create a new order.
GET /orders/:id: Fetch order details.

• POST /orders/resend-email: Resend order confirmation email to the customer.

•**Users:**

Users:

POST/auth/register: Register a new user.

• POST /auth/login: Authenticate a user.

• GET /users/me: Fetch current user details.

• **Shipping:**

• GET /shipment: Fetch shipment tracking details via ShipEngine.

• **Email Resend:**

POST/emails/resend: Request to resend an email (e.g., order confirmation, shipping details).

• **Request Payload:** { "userId": "string", "orderId": "string", "emailType": "string" // Values: "orderConfirmation", "shippingUpdate" }

• **Response Example**: { "status": "success", "message": "Email has been resent successfully."}

6.2 Sample Response for API Calls

Order Details: { "orderId": 123, "status": "In Transit", "ETA": "2 days" }

7. Data Schema

Core Entities

• **Users:**

id: Unique identifier.

name: Full name.

email: User email (unique).

password: Hashed password.
role: Customer/Seller.
• **Stripe Fields:**
■ stripe_customer_id: Stripe customer ID for payment processing.
Products:
• id: Unique identifier.
name: Product name.
• price: Base price.
• description: Text description.
image: Default design template.
• created_at, updated_at : Timestamps.
• **ShipEngine Fields:**
■weight: Product weight for shipping calculations.
■ dimensions: Product dimensions (L x W x H).

Orders:
• id: Unique identifier.
user id: Linked user ID.
status: Order status (Pending/Completed).
total_amount: Final cost.
• shipping_address: Shipping address details.
• payment_status: Stripe payment status.
• shipping_status: ShipEngine status.
Custom Designs:
id: Unique identifier.
user_id: Linked user ID.
product_id: Linked product ID.
design_url: URL to the custom design.
• customization_details: JSON object with design specifics.

8. Third-Party Integrations

• **Stripe**: Secure payment processing. Integrate for handling payments and generating invoices.
ShipEngine: Real-time shipping quotes and tracking integration.
Cloud Storage: Store uploaded assets and designs.
Email Notifications (SendGrid): Use SendGrid for sending order confirmations, shipping updates, and account-related emails.
Clerk for Authentication: Handling secure user authentication and registration.

9. Technology Stack

Frontend: React.js, Next.js, TailwindCSS, ShadCN UI, Fabric.js, Framer Motion.
Backend: Node.js, Sanity CMS, Next.js API routes.
Database: Sanity CMS for dynamic content and database management (products, orders, assets, and user data).
APIs: REST-based with JSON responses.

10. Appendices

10.1 Glossary

POD: Print on Demand.
MVP: Minimum Viable Product.
CMS: Content Management System.
API: Application Programming Interface.
Stripe: Payment gateway service.
ShipEngine: Shipping API service.

10.2 References

Stripe API Documentation: Stripe API Docs
ShipEngine API Documentation: ShipEngine API Docs
Sanity CMS: Sanity CMS Docs

Clerk Authentication: Clerk API Docs