Marketplace Technical Foundation - Luxury Car Rentals

This document outlines the **technical foundation** for the **Luxury Car Rental Marketplace**, covering **system architecture, workflows, API specifications, data schema, and technical roadmap**.

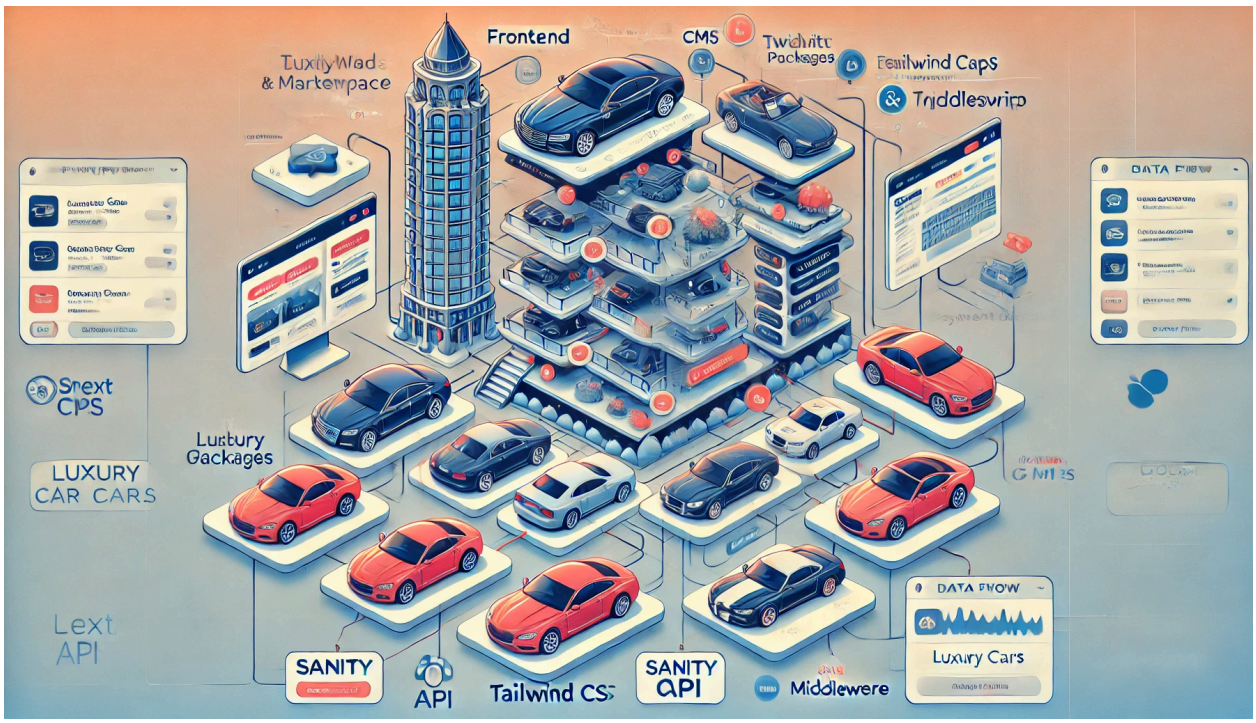
---

1. System Architecture Overview

📌 High-Level System Architecture

📌 System Components & Responsibilities

System architecture diagram



Component	Role & Function
Frontend (Next.js, Tailwind CSS, TypeScript)	Provides a responsive UI for browsing luxury cars, managing bookings, and processing payments.
Sanity CMS	Stores car listings, rental packages, user profiles, booking history, and rental transactions.
Next.js API (Backend & Middleware)	Manages business logic, communicates between frontend, CMS, and third-party services.
Payment Gateway (Stripe/PayPal)	Securely processes transactions for rental payments.
Third-Party APIs (Google Maps, Twilio, Cloudinary)	Provides real-time location tracking, SMS notifications, and image storage for luxury car listings.

📌 Data Flow Overview

1️⃣ User browses available luxury cars on the **Next.js frontend**.
2️⃣ The **frontend requests car data from Sanity CMS** via the `/cars` API.
3️⃣ User selects a rental package (1-day, weekly, monthly) and proceeds to checkout.
4️⃣ The rental booking is processed via the `/bookings` API, storing details in **Sanity CMS**.
5️⃣ Payment is securely processed via **Stripe/PayPal**, and upon success, a **confirmation SMS is sent via Twilio**.
6️⃣ Admins can track rental status, update car availability, and manage orders via **Sanity CMS**.

2. Key Workflows

1. User Registration & Authentication

- Users **sign up** via the frontend.
- Data is stored in **Sanity CMS** under `user` schema.
- A **confirmation SMS/email** is sent via Twilio.

2. Luxury Car Browsing & Selection

- Users view available **luxury cars** (brands, features, rental rates).
- Car data is retrieved via the `/cars` API from **Sanity CMS**.
- Users select a **rental package (daily, weekly, monthly)** and proceed to checkout.

3. Rental Booking & Payment Processing

- Users add a rental package to the **cart** and proceed to checkout.
- The booking request is sent to the `/bookings` API and stored in **Sanity CMS**.
- Payment is processed via **Stripe/PayPal** and a confirmation receipt is generated.
- Upon successful payment, an **SMS notification** is sent to the user.

4. Rental Order Management & Tracking

- Admins update car availability via **Sanity CMS**.
- Users track their rental status via the `/rental-status` API.
- Google Maps API provides **real-time location tracking** for deliveries (if applicable).

5. Rental Return & Condition Verification

- Users return rented cars based on the selected rental period.
- Condition reports are updated in **Sanity CMS** (`conditionStatus` field).
- Admins process **security deposit refunds** (if applicable).

3. API Endpoints & Specifications

🔹 Rental Car API Endpoints

Endpoint	Method	Purpose	Request Payload	Response
`/cars`	GET	Fetch all available cars	-	`{ cars: [...] }`
`/car/:id`	GET	Fetch specific car details	-	`{ id: 1, name: "BMW X7", price: 500 }`
`/bookings`	POST	Create a new rental booking	`{ userId, carId, rentalPackage }`	`{ confirmationId: 789, status: "Success" }`
`/rental-status/:id`	GET	Check rental status	-	`{ orderId: 123, status: "Active" }`

`/payments`	POST	Process payment via Stripe/PayPal	`{ userId, amount, method }`	`{ paymentId: 456, status: "Paid" }`

Third-Party API Integrations

API Name	Purpose	Service Used
Stripe/PayPal	Secure Payment Processing	Stripe, PayPal
Google Maps API	Location & Rental Tracking	Google Maps
Twilio API	SMS Notifications	Twilio
Cloudinary API	Car Image Storage	Cloudinary

4. Sanity CMS Schema Design

Car Schema (`car.ts`)

```
export default defineType({
  name: 'car',
  title: 'Luxury Car',
  type: 'document',
  fields: [
    { name: 'name', title: 'Car Name', type: 'string' },
    { name: 'brand', title: 'Brand', type: 'string' },
    { name: 'price', title: 'Rental Price (per day)', type: 'number'
},
    { name: 'rentalPackage', title: 'Rental Package', type: 'string'
},
    { name: 'availability', title: 'Availability', type: 'boolean' },
    { name: 'image', title: 'Car Image', type: 'image' }
  ]
});
```

Booking Schema (`booking.ts`)

```
export default defineType({
  name: 'booking',
  title: 'Car Booking',
  type: 'document',
  fields: [
    { name: 'userId', title: 'User ID', type: 'reference', to: [{
type: 'user' }] },
    { name: 'carId', title: 'Car ID', type: 'reference', to: [{ type:
'car' }] },
    { name: 'rentalPackage', title: 'Rental Package', type: 'string'
},
    { name: 'totalPrice', title: 'Total Price', type: 'number' },
    { name: 'paymentStatus', title: 'Payment Status', type: 'string' }
  ]
});
```

5. Technical Roadmap & Milestones

Milestone	Description	Deadline
Phase 1: System Planning	Define system architecture, APIs, and CMS setup	✅ Completed
Phase 2: Sanity CMS Schema	Implement schema for cars, bookings, users	⏳ In Progress
Phase 3: Frontend Development	Build UI using Next.js & Tailwind CSS	🔜 Upcoming
Phase 4: API Development	Develop Next.js API routes for bookings & payments	🔜 Upcoming
Phase 5: Payment Integration	Implement Stripe/PayPal for secure transactions	🔜 Upcoming

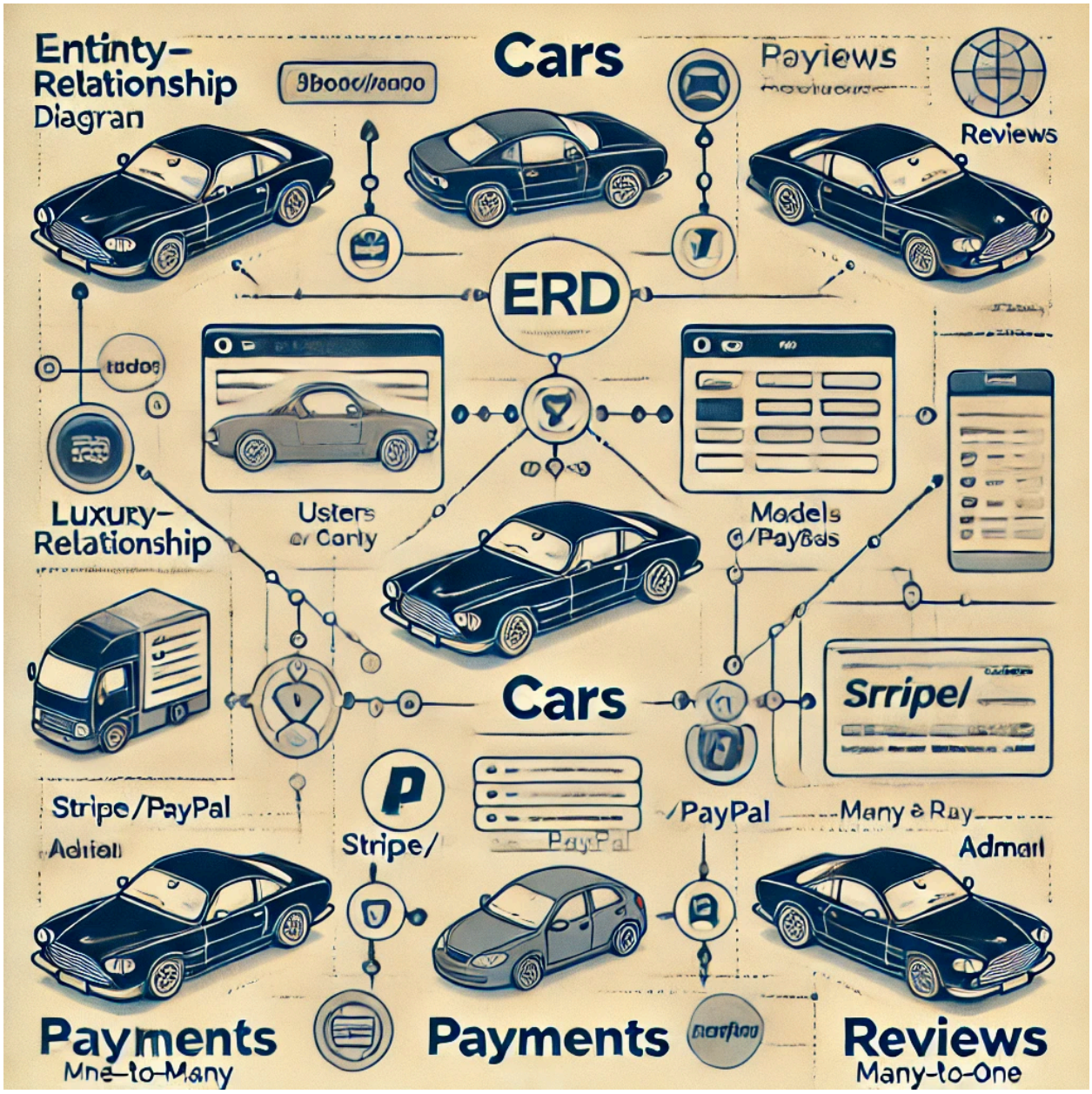
Phase 6: Testing & Deployment	Perform QA, bug fixes, and deploy to Vercel	➡️ Upcoming

Final Summary & Next Steps

This document provides a **well-structured technical foundation** for the **Luxury Car Rental Marketplace**, covering:
✅ **System Architecture & Data Flow**
✅ **Workflows for User Interaction**
✅ **API Endpoints & Specifications**
✅ **Sanity CMS Schema Design**
✅ **Technical Roadmap & Milestones**

📌 Next Steps:

- 🔹 **Complete API development & frontend integration**
- 🔹 **Set up Sanity CMS with final schema**
- 🔹 **Test Stripe/PayPal payment flows**
- 🔹 **Finalize UI components & deploy on Vercel**


Entinty–
Relationship
Diagran
Cars
Reviews
ERD
Luxury–
Relationship
Cars
Stripe/PayPal
Payments
Mne–to–Many
Payments
Reviews
Many–to–One