Car Rental Website

Executive Summary

This document outlines the technical requirements for the **Rental Car** project. The platform allows users to get a luxury cars on rent for specific event, corporate event, wedding etc. 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.

Day 2 Activities: Transitioning to Technical Planning

1) Define Technical Requirements

This step translates our business goal into clear technical requirement.

a) *Frontend Requirements:*

- We'll use Nextjs15 to make frontend of our car rental website which include:

A) Account-Based System:

- Users can create accounts to:
  - Track rental history.
  - Save preferences for faster future bookings.

B) Car Rental Options:

- Users can select between:
  - **Self-Drive:** Cars rented without a driver.
  - **Chauffeur-Driven:** Cars rented with a driver for added convenience.

C) Booking Flow:

- Customers must enter:
  - **Pickup and Drop-Off Locations:** To calculate delivery and return logistics.
  - **Rental Duration:** Options include hourly, daily, and weekly rates.
  - A map integration displays locations for pickup and drop-off points.

D) Additional Services:

- Display add-ons such as child seats, chauffeur services, and event-specific rentals (e.g., weddings, corporate).

E) Responsive Design:

- Fully responsive across desktop, tablet, and mobile devices.

F) Essential Pages:

- **Home Page:** Highlight featured, popular and recommended cars.
- **Car Listing Page:** Display luxury cars with filters like type, price tier, seating capacity, and fuel type.
- **Car Details Page:** Detailed car profiles with images, features, availability, and pricing tiers.
- **Cart Page:** Allow users to review selections, specify rental duration, and calculate total cost.

- **Checkout Page:** Collect customer details, delivery preferences, and payment information.
- **Order Confirmation Page:** Show booking details, including order ID, pickup/drop-off times, and payment status.

G) Push Notifications:

- Notify app users about:
  - Booking confirmations.
  - Reminders for car return.
  - Exclusive discounts or offers.

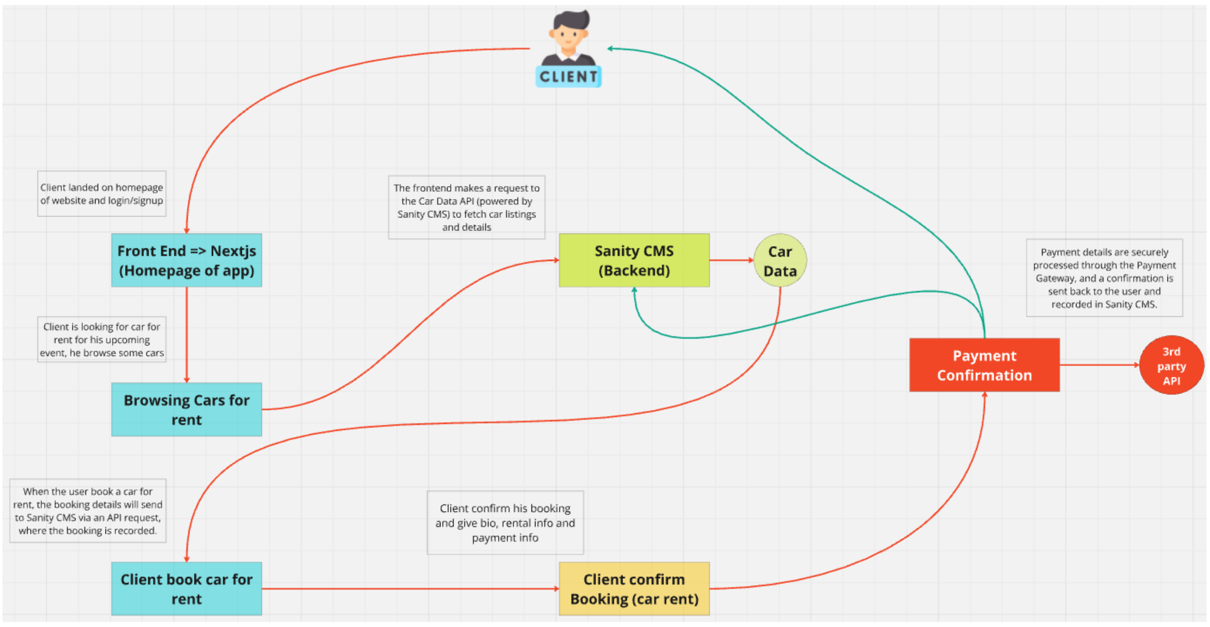
b) *Backend*

- We'll use **Sanity CMS** as our backend to manage cars data, customer details and order records
- We'll use '**Clerk**' for authentication

c) *Third-Party APIs:*

- **Stripe API:** For secure and seamless payment process

2) Design System Architecture



Client landed on homepage of website and login/signup

Client is looking for car for rent for his upcoming event, he browse some cars

When the user book a car for rent, the booking details will send to Sanity CMS via an API request, where the booking is recorded.

The frontend makes a request to the Car Data API (powered by Sanity CMS) to fetch car listings and details

Client confirm his booking and give bio, rental info and payment info

Payment details are securely processed through the Payment Gateway, and a confirmation is sent back to the user and recorded in Sanity CMS.

Key Workflows to Include:

In this architecture, a typical work flow could look like this:

1) User Registration
- Client landed on homepage of website and login/signup

2) Product Browsing:
- Client is looking for car for rent for his upcoming event, he browse some cars
- The frontend makes a request to the Car Data API (powered by Sanity CMS) to fetch cars listings and details

3) Order Placement:
- When the user book a car for rent, the booking details will send to Sanity CMS via an API request, where the booking is recorded.

4) Payment gateway

- Client confirm his booking and give bio, rental and payment info
- Payment details are securely processed through the Payment Gateway, and a confirmation is sent back to the user and recorded in Sanity CMS.

3) API Specification Document:

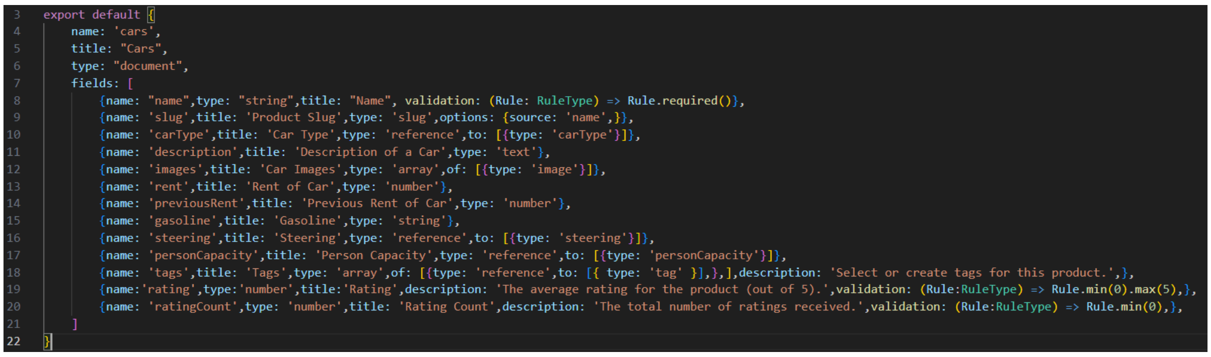
Here are the main API endpoints we'll be working with:

API Endpoints		
Endpoint	**Method**	**What it does**
/api/cars	GET	Fetches all available cars with optional filters (e.g., car type, price range).
/api/cars/[carId]	GET	Fetches details of a specific car by ID.
/api/create-booking	POST	Creates a new rental booking with car, user, and rental details.
/api/booking	GET	Fetches all booking (for admin purposes).
/api/booking/[bookingId]	GET	Fetches details of a specific booking by ID.
/api/Cancelbooking	DELETE	Cancel booking
/api/payment/status	GET	Retrieves the payment status for a specific order.
/api/reviews/[carId]	POST	Adds a review for a specific car.
/api/reviews/[carId]	GET	Fetches all reviews for a specific car.

4) Sanity Schema

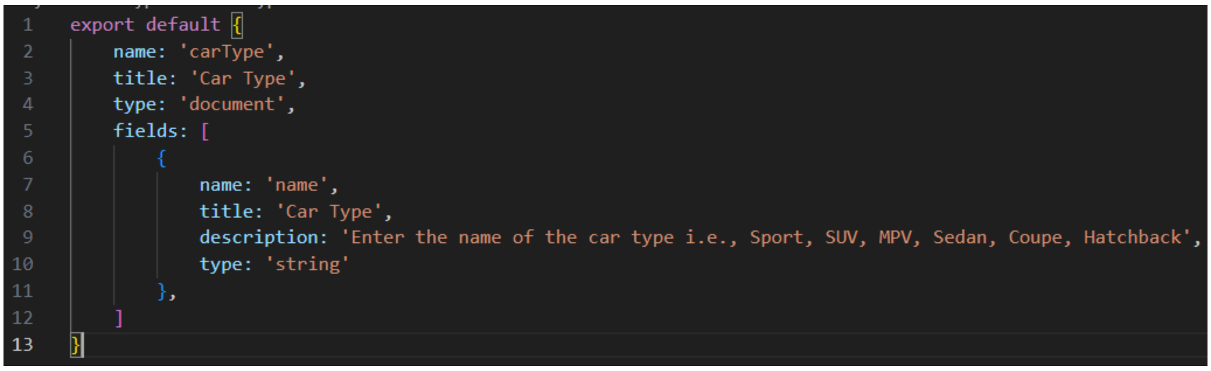
Fundamental entities are:

a) Car Schema



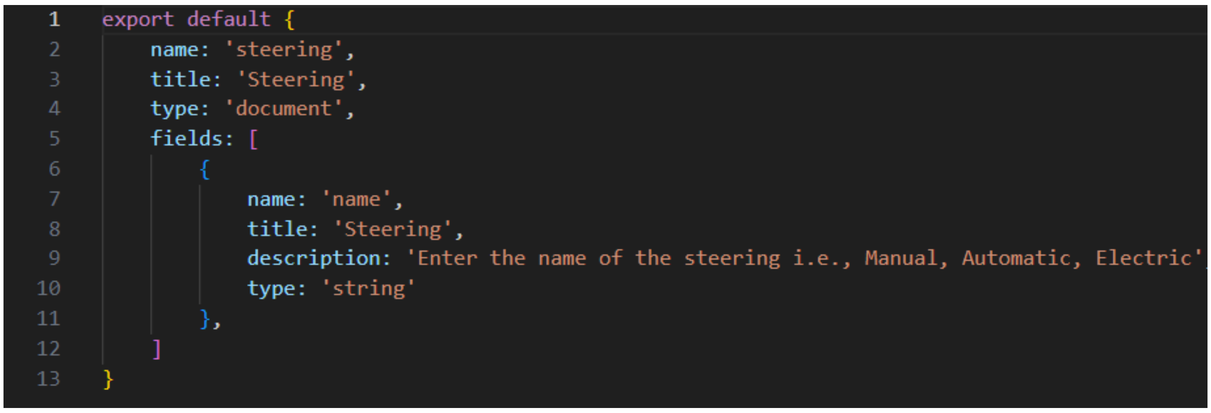
```
export default {
    name: 'cars',
    title: "Cars",
    type: "document",
    fields: [
        {name: "name",type: "string",title: "Name", validation: (Rule: RuleType) => Rule.required()},
        {name: 'slug',title: 'Product Slug',type: 'slug',options: {source: 'name',}},
        {name: 'carType',title: 'Car Type',type: 'reference',to: [{type: 'carType'}]},
        {name: 'description',title: 'Description of a Car',type: 'text'},
        {name: 'images',title: 'Car Images',type: 'array',of: [{type: 'image'}]},
        {name: 'rent',title: 'Rent of Car',type: 'number'},
        {name: 'previousRent',title: 'Previous Rent of Car',type: 'number'},
        {name: 'gasoline',title: 'Gasoline',type: 'string'},
        {name: 'steering',title: 'Steering',type: 'reference',to: [{type: 'steering'}]},
        {name: 'personCapacity',title: 'Person Capacity',type: 'reference',to: [{type: 'personCapacity'}]},
        {name: 'tags',title: 'Tags',type: 'array',of: [{type: 'reference',to: [{ type: 'tag' }],},],description: 'Select or create tags for this product.',},
        {name:'rating',type:'number',title:'Rating',description: 'The average rating for the product (out of 5).',validation: (Rule:RuleType) => Rule.min(0).max(5),},
        {name: 'ratingCount',type: 'number',title: 'Rating Count',description: 'The total number of ratings received.',validation: (Rule:RuleType) => Rule.min(0),},
    ]
}
```

b) Car Type Schema



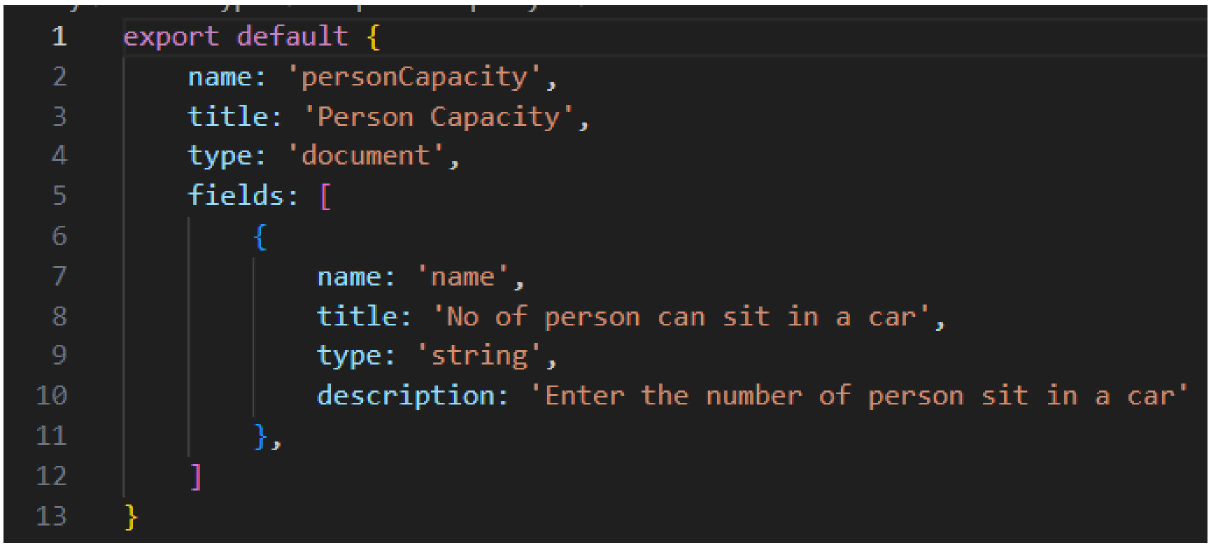
```
export default {
    name: 'carType',
    title: 'Car Type',
    type: 'document',
    fields: [
        {
            name: 'name',
            title: 'Car Type',
            description: 'Enter the name of the car type i.e., Sport, SUV, MPV, Sedan, Coupe, Hatchback',
            type: 'string'
        },
    ]
}
```

c) Steering Schema (Automatic, Manual, Electric)



```
export default {
    name: 'steering',
    title: 'Steering',
    type: 'document',
    fields: [
        {
            name: 'name',
            title: 'Steering',
            description: 'Enter the name of the steering i.e., Manual, Automatic, Electric'
            type: 'string'
        },
    ]
}
```

d) Person Capacity Schema



```
export default {
    name: 'personCapacity',
    title: 'Person Capacity',
    type: 'document',
    fields: [
        {
            name: 'name',
            title: 'No of person can sit in a car',
            type: 'string',
            description: 'Enter the number of person sit in a car'
        },
    ]
}
```