Technical Documentation for Morent: Car Rental Marketplace

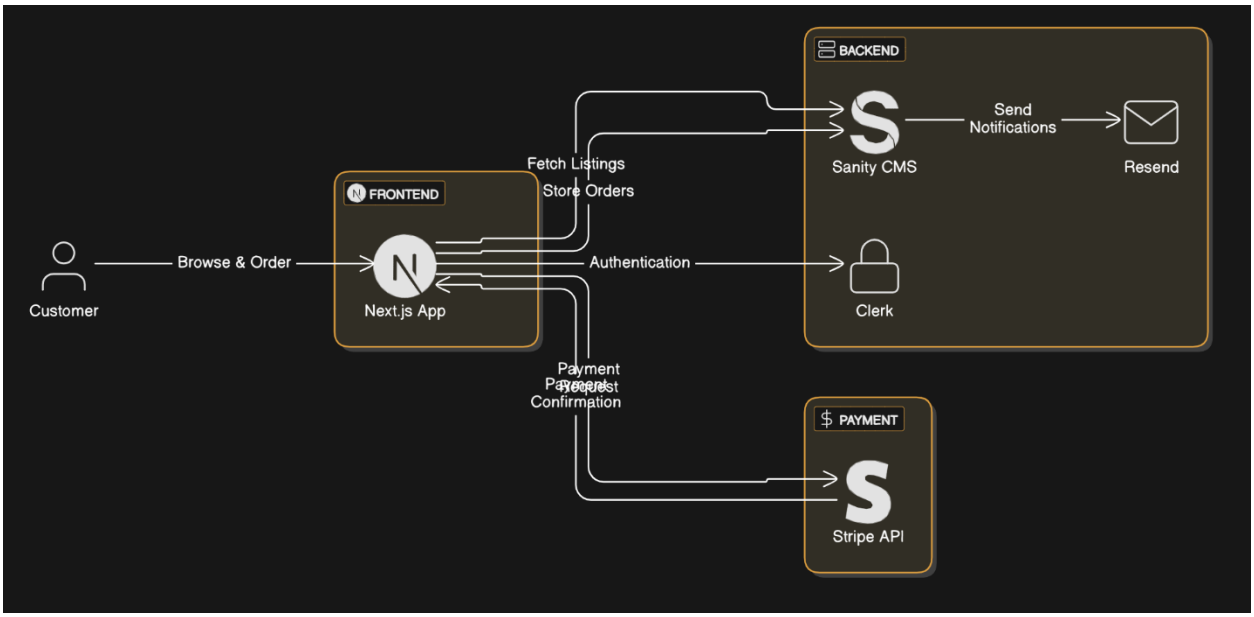
1. Introduction

Morent is a car rental marketplace designed to connect car owners with renters. This document provides an in-depth technical overview of the system architecture, workflows, API specifications, data schema design, and a detailed technical roadmap. The goal is to build a scalable and user-centric platform that aligns with business objectives while ensuring high performance and reliability.

---

2. System Architecture Document

System Architecture Diagram



Component Descriptions

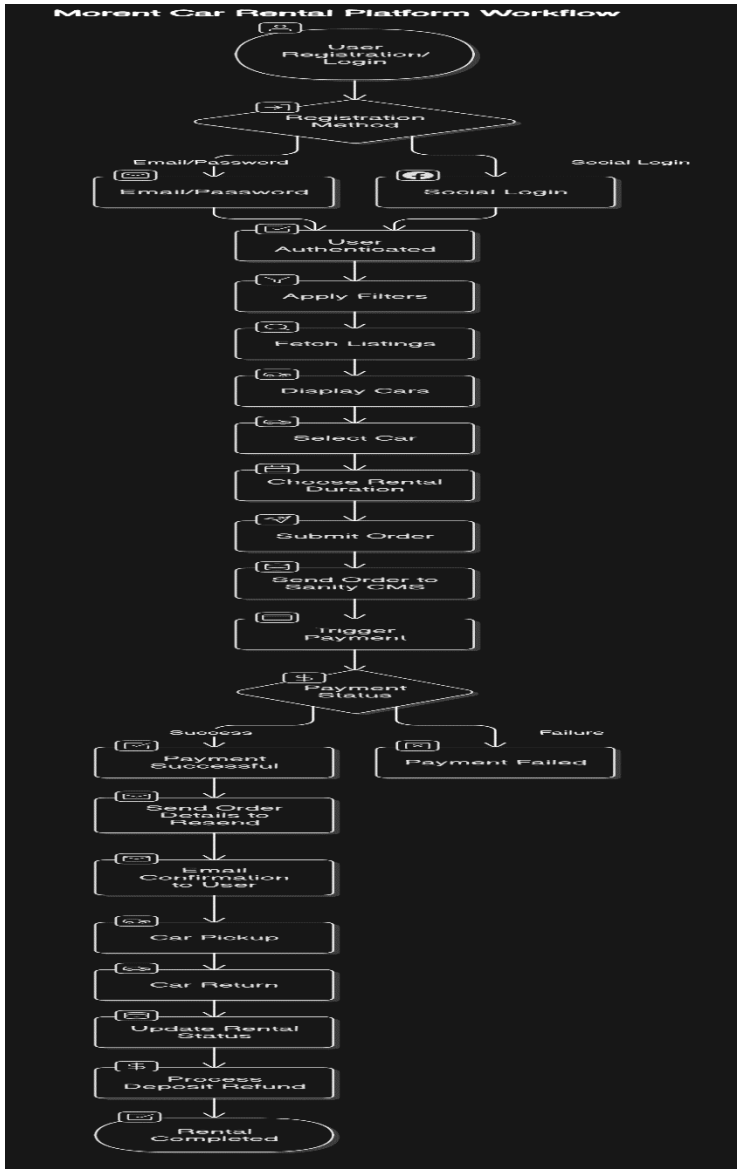
1. **Frontend (Next.js with TypeScript):**
   - User interface for car browsing, filtering, order placement, and payment processing.
   - Communicates with backend services and third-party APIs via RESTful endpoints.
2. **Sanity CMS:**
   - Backend for managing car listings, orders, and user profiles.
   - Provides a real-time content management system with a flexible schema.
3. **Stripe API:**
   - Secure payment processing with support for various payment methods.

- Sends payment confirmation data to the frontend and updates order status.

4. **Clerk:**
   - User authentication and session management for registration, login, and profile updates.
   - Provides secure access tokens for API requests.
5. **Resend:**
   - Sends transactional emails for order confirmations, payment receipts, and rental reminders.

---

3. Workflow Diagra



Detailed Workflow

1. **User Registration and Authentication:**

- User signs up using email/password or social login (Clerk).
- Clerk verifies user details and creates a session token.

2. **Car Browsing and Searching:**
   - User applies filters (location, price, type) on the homepage.
   - Frontend fetches car listings from Sanity CMS and displays results.
3. **Order Placement and Payment Processing:**
   - User selects a car, chooses rental duration, and submits an order.
   - Order details are sent to Sanity CMS; payment is processed via Stripe.
   - Upon successful payment, an order confirmation is emailed (Resend).
4. **Rental Completion:**
   - User picks up the car and completes the rental.
   - Condition reports are submitted, and deposits are processed/refunded.

---

4. API Specification Document

Endpoints Overview

Endpoint	Method	Description	Payload	Response
`/api/cars`	GET	Fetch all available cars.	None	`{ id: 1, name: "Tesla Model 3", brand: "Tesla", pricePerDay: 70 }`
`/api/cars/{id}`	GET	Fetch details of a specific car.	None	`{ id: 1, name: "Tesla Model 3", fuelCapacity: "75 kWh", availability: true }`
`/api/orders`	POST	Create a new order.	`{ carId: 1, userId: "abc123", duration: 3, deposit: 100 }`	`{ orderId: 456, status: "Success" }`
`/api/payments`	POST	Process payment via Stripe.	`{ orderId: 456, amount: 210, paymentMethod: "card" }`	`{ paymentId: 789, status: "Paid" }`
`/api/rentals/{id}`	GET	Fetch rental details (e.g., duration, pickup/drop-off locations).	None	`{ rentalId: 123, carId: 1, duration: 3, status: "Ongoing" }`

Endpoint	Method	Description	Payload	Response
/api/notifications	POST	Send email notifications (e.g., order confirmation).	{ email: "user@example.com", message: "Your order has been confirmed." }	{ status: "Sent" }

---

5. Data Schema Design

Sanity CMS Schemas

Car Schema:

```
export default {
    name: 'car',
    type: 'document',
    title: 'Car',
    fields: [
        { name: 'name', type: 'string', title: 'Car Name' },
        { name: 'brand', type: 'string', title: 'Brand', description: 'Brand
of the car (e.g., Nissan, Tesla, etc.)' },
        { name: 'type', type: 'string', title: 'Car Type', description: 'Type
of the car (e.g., Sport, Sedan, SUV, etc.)' },
        { name: 'fuelCapacity', type: 'string', title: 'Fuel Capacity',
description: 'Fuel capacity or battery capacity (e.g., 90L, 100kWh)' },
        { name: 'transmission', type: 'string', title: 'Transmission',
description: 'Type of transmission (e.g., Manual, Automatic)' },
        { name: 'seatingCapacity', type: 'string', title: 'Seating Capacity',
description: 'Number of seats (e.g., 2 People, 4 seats)' },
        { name: 'pricePerDay', type: 'number', title: 'Price Per Day' },
        { name: 'originalPrice', type: 'string', title: 'Original Price',
description: 'Original price before discount (if applicable)' },
        { name: 'tags', type: 'array', title: 'Tags', of: [{ type: 'string'
}], options: { layout: 'tags' }, description: 'Tags for categorization (e.g.,
popular, recommended)' },
        { name: 'image', type: 'image', title: 'Car Image', options: {
hotspot: true } }
    ],
};
```

Rental Schema:

```
export default {
    name: "rental",
    type: "document",
    title: "Rental",
    fields: [
        { name: "car", type: "reference", to: [{ type: "car" }], title:
"Car", description: "The car being rented" },
```

```
        { name: "customer", type: "reference", to: [{ type: "customer" }],
title: "Customer", description: "The customer renting the car" },
        { name: 'userId', type: 'string', title: 'User ID', description: 'ID
of the authenticated user' },
        { name: "startDate", type: "datetime", title: "Start Date" },
        { name: "endDate", type: "datetime", title: "End Date" },
        { name: "duration", type: "number", title: "Duration (in days)",
description: "The total number of rental days" },
        { name: "deposit", type: "number", title: "Deposit", description:
"The refundable deposit for the rental" },
        { name: "totalPrice", type: "number", title: "Total Price",
description: "Total cost for the rental period" },
        { name: "conditionReport", type: "object", title: "Condition Report",
fields: [
            { name: "beforeRental", type: "text", title: "Before Rental",
description: "Condition of the car before rental" },
            { name: "afterRental", type: "text", title: "After Rental",
description: "Condition of the car after rental" },
            { name: "beforePhotos", type: "array", title: "Before Rental
Photos", of: [{ type: "image" }], description: "Photos of the car before
rental" },
            { name: "afterPhotos", type: "array", title: "After Rental
Photos", of: [{ type: "image" }], description: "Photos of the car after
rental" }
        ] },
        { name: "status", type: "string", title: "Rental Status", options: {
list: ["Pending", "Ongoing", "Completed", "Cancelled"] }, initialValue:
"Pending" },
        { name: "paymentStatus", type: "string", title: "Payment Status",
options: { list: ["Paid", "Unpaid", "Partially Paid"] }, initialValue:
"Unpaid" }
    ]
}
```

---

6. Technical Roadmap

Milestone	Description
Frontend Setup	Set up Next.js with TypeScript and integrate Clerk for authentication.
Sanity CMS Configuration	Define and implement schemas for Cars, Customers, and Rentals.
API Integration	Integrate Stripe for payments and Resend for email notifications.
Workflow Implementation	Implement key workflows (e.g., car browsing, order placement, rental management).
Testing & Debugging	Test the entire system for bugs and performance issues.
Deployment	Deploy the platform to a hosting service (e.g., Vercel).

---

7. Conclusion

In this documentation, have provided the foundation for building **Morent**, a scalable and user-centric car rental marketplace. By adhering to the outlined system architecture, workflows, and technical roadmap, the platform is set to deliver seamless experiences for both car owners and renters.

My goal is to not only meet the requirements of a fully functional marketplace- but to go above and beyond.