Morent: Technical Foundation

System Overview

Morent focuses on enabling a seamless rental experience for users looking for short-term access to vehicles. This system integrates modern technologies and workflows to ensure reliability, scalability, and a user-centric experience. The frontend (built using Next.js) interacts with Sanity CMS for data management and third-party APIs for notifications, payments, and logistics. Together, these components provide a seamless experience for users and administrators.

1. Technical Requirements

Frontend Requirements

- **User Interface (UI):** Intuitive and modern design using React (e.g., Next.js) for fast loading and responsiveness.
- **Pages:**
    - **Home:** Highlight top-rated vehicles, special offers, and user testimonials.
    - **Search & Filters:** Search by location, vehicle type, and features.
    - **Vehicle Details:** Comprehensive information on pricing, features, availability, and environmental ratings.
    - **Booking Flow:** Cart, checkout, and booking confirmation.
    - **User Dashboard:** Manage rentals, track progress, and view history.
- **Mobile-First Design:** Focus on usability across devices.

Backend Requirements

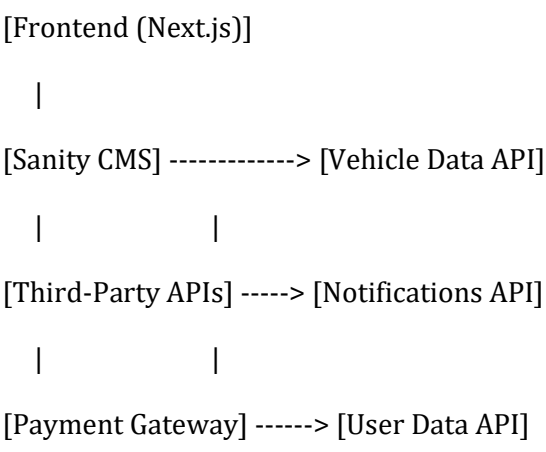
- **Sanity CMS:**
    - Manage vehicle data, customer profiles, orders, and reviews.
    - Flexible schema design for evolving business needs.
- **Third-Party API Integrations:**
    - **Location Services:** Google Maps API for drop-off and pickup locations.
    - **Notifications:** Twilio for SMS/email notifications (booking confirmation and reminders).
    - **Logistics:** APIs for fleet tracking and real-time vehicle location updates.

Key Features

- **Real-Time Availability:** Ensure accurate display of available vehicles.
- **Booking Management:** Allow users to reserve, extend, or cancel bookings.
- **Customer Reviews:** Enable feedback and ratings for vehicles.
- **Environmentally Friendly Features:** Promote electric vehicles and sustainability metrics.
- **Pricing Transparency:** Display clear pricing breakdown, including deposits.

2. System Architecture

High-Level Architecture Diagram

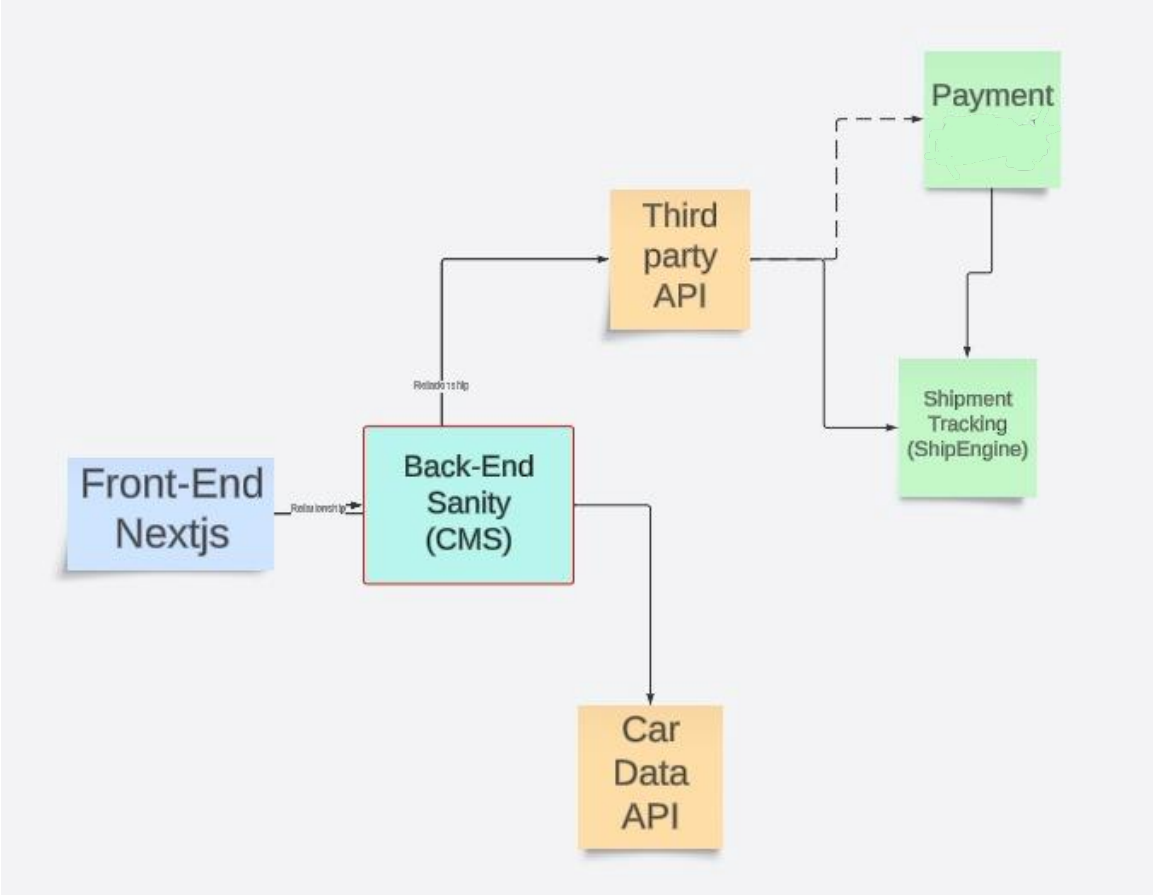


Roles of Components

1. **Frontend:** Handles user interactions, displays vehicle listings, and processes booking forms.
2. **Sanity CMS:** Acts as the central database for managing vehicle details, user profiles, orders, and reviews.
3. **Third-Party APIs:** Supports notifications (e.g., Twilio), location services (e.g., Google Maps), and logistics tracking.
4. **Payment Gateway:** Processes secure payments and records transaction statuses.

Key Workflows

1. **User Registration:**
    - User signs up → Details stored in Sanity CMS → Confirmation email/SMS sent.
2. **Vehicle Search:**
    - User inputs filters → Query to Sanity CMS → Results displayed.
3. **Booking Process:**
    - User selects vehicle → Proceeds to checkout → Booking recorded in CMS.
4. **Real-Time Updates:**
    - Booking status and vehicle location fetched via APIs.



3. API Requirements

Endpoints

Endpoint	Method	Purpose	Response Example
/vehicles	GET	Fetch all available vehicles.	{ "id": 1, "name": "Tesla Model 3", "price": 45.0 }
/vehicles	POST	Add new vehicle data (admin).	{ "status": "success" }
/rentals	POST	Book a vehicle.	{ "status": "success", "bookingId": "B789" }
/rentals/:id	GET	Fetch rental details by ID.	{ "rentalId": "R123", "status": "active" }
/reviews	POST	Submit a review for a vehicle.	{ "reviewId": "RV001", "status": "success" }
/reviews/:vehicleId	GET	Fetch reviews for a vehicle.	[{ "reviewId": "RV001", "score": 5 }]
/notifications	POST	Trigger booking notifications.	{ "status": "sent" }

Example Payloads

1. Booking:

{

"vehicleId": "V123",

"userId": "U456",

"rentalDuration": "3 days",

"totalCost": 120.0

}

Response:

{

```
 "status": "success",
 "bookingId": "B789"
}
```


2. Vehicle Search:


```
{
 "location": "New York",
 "type": "Electric"
}
```


Response:


```
[
 {
  "vehicleId": "V001",
  "name": "Tesla Model 3",
  "pricePerDay": 45.0
 }
]
```


4. Sanity Schema Examples

Vehicle


```
export default {
 name: 'vehicle',
 type: 'document',
 fields: [
  { name: 'name', type: 'string', title: 'Vehicle Name' },
  { name: 'type', type: 'string', title: 'Vehicle Type' },
  { name: 'rentalPrice', type: 'number', title: 'Rental Price per Day' },
  { name: 'availabilityStatus', type: 'boolean', title: 'Available for Rent' },
  { name: 'environmentalRating', type: 'number', title: 'Sustainability Score' },
```

```
  { name: 'features', type: 'array', of: [{ type: 'string' }], title: 'Features' }
 ]
};
```

Rental

```
export default {
 name: 'rental',
 type: 'document',
 fields: [
  { name: 'rentalId', type: 'string', title: 'Rental ID' },
  { name: 'vehicleId', type: 'reference', to: [{ type: 'vehicle' }] },
  { name: 'userId', type: 'reference', to: [{ type: 'user' }] },
  { name: 'duration', type: 'string', title: 'Duration' },
  { name: 'totalCost', type: 'number', title: 'Total Cost' },
  { name: 'status', type: 'string', title: 'Rental Status' }
 ]
};
```

Rental eCommerce-Specific Fields

- **rentalDuration:** Length of the rental period (e.g., "3 days").
- **depositAmount:** Security deposit required for the rental.
- **conditionStatus:** Current condition of the vehicle before/after rental.

5. Documentation

Deliverables

- **System Architecture Document:** Visual representation and descriptions of key components.
- **API Documentation:** Detailed list of endpoints with methods, payloads, and responses.

- **Sanity Schema Files:** Structured schemas for key entities.
- **Workflow Diagrams:** Graphical depiction of user journeys and data flows.

This plan provides a robust technical foundation for Morent, ensuring a smooth transition to implementation while maintaining alignment with business goals.