

Name: MUHAMMAD ABUBAKAR

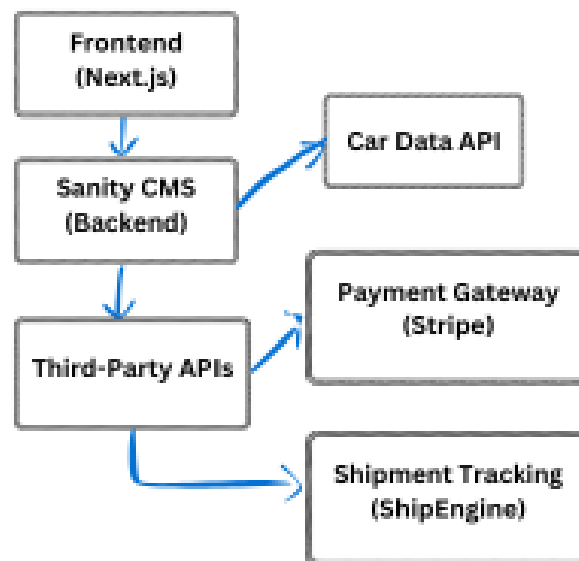
Roll no: 420397

Slot: 7 to 10 Friday

## Technical Documentation for Car Rental Website

---

### 1. System Architecture Overview



### Description of Components

- Frontend (Next.js):

- The user interface where customers browse cars, book rentals, and make payments.
- Built with Next.js for fast, responsive, and dynamic performance.
- **Sanity CMS:**
  - The backend system that stores all car details, customer information, and booking records.
  - Acts as the central database for the website.
- **Third-Party APIs:**
  - **Payment Gateway:** Handles secure payment processing.
  - **Shipment Tracking API:** Provides real-time updates on car deliveries.
  - **Location API:** Fetches pickup and drop-off points.

## 2. Key Workflows

### 1. User Sign-Up

- **Steps:**
  1. User fills out the registration form.
  2. Data is sent to Sanity CMS and stored securely.
  3. User receives a confirmation email.

### 2. Car Browsing

- **Steps:**
  1. User visits the website and searches for cars.
  2. Frontend fetches car data from Sanity CMS via the **Car Data API**.
  3. Cars are displayed with filters (e.g., price, type, availability).

### 3. Booking a Car

- **Steps:**
  1. User selects a car and chooses rental dates.
  2. Booking details are sent to Sanity CMS via the **Booking API**.
  3. Booking is confirmed, and details are saved in Sanity.

### 4. Payment Processing

- **Steps:**
  1. User enters payment details.
  2. Frontend sends data to the **Payment Gateway API**.
  3. Payment is processed, and a confirmation is sent to the user.

## 5. Shipment Tracking

- **Steps:**
  1. If the car is being delivered, the frontend fetches tracking details from the **Shipment Tracking API**.
  2. Real-time updates (e.g., status, ETA) are displayed to the user.

## 3. API Endpoints

API Name	Method	Purpose Example	Request/Response
/cars	GET	Fetch all available cars.	<b>Request:</b> GET /cars <b>Response:</b> [{ "id": 1, "name": "Toyota Corolla", ... }]
/bookings	POST	Create a new booking.	<b>Request:</b> POST /bookings <b>Response:</b> { "bookingId": 123, "status": "Confirmed" }
/payments	POST	Process payment for a booking.	<b>Request:</b> POST /payments <b>Response:</b> { "paymentId": 789, "status": "Paid" }

/shipment-tracking	GET	Track the status of a booking.	<b>Request:</b> GET /shipment-tracking?bookingId=123 <b>Response:</b> { "status": "In Transit", "ETA": "30 mins" }
--------------------	-----	--------------------------------	---

## 4. Sanity Schema Examples

- **Car Schema**

```
export default {
  name: 'car',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Car Name' },
    { name: 'type', type: 'string', title: 'Car Type' },
    { name: 'price', type: 'number', title: 'Price per Day' },
    { name: 'availability', type: 'boolean', title: 'Availability' },
    { name: 'image', type: 'image', title: 'Car Image' }
  ]
};
```

- **Booking Schema**

```
export default {
  name: 'booking',
  type: 'document',
  fields: [
    { name: 'car', type: 'reference', to: [{ type: 'car' }], title: 'Car' },
    { name: 'customer', type: 'reference', to: [{ type: 'customer' }], title: 'Customer' },
    { name: 'startDate', type: 'date', title: 'Start Date' },
    { name: 'endDate', type: 'date', title: 'End Date' },
    { name: 'status', type: 'string', title: 'Booking Status' }
  ]
};
```

## 5. Technical Roadmap

### Milestones

1. **Week 1-2:**

- Set up Sanity CMS and design schemas for cars, customers, and bookings.
- Develop the frontend using Next.js.

2. **Week 3-4:**

- Integrate third-party APIs (payment gateway, shipment tracking).

- Implement user registration and car browsing features.
- 3. **Week 5-6:**
  - Add booking and payment functionality.
  - Test the entire system and fix any issues.
- 4. **Week 7:**
  - Launch the website and gather user feedback.

## Why This Documentation Works for My Website

- **Clear and Organized:** Every component, workflow, and API is explained in detail.
- **Scalable:** The architecture and workflows can grow with the website.
- **Professional:** Follows industry standards for technical documentation.