hackathon task day 2 marketplace

PROJECT KHALIL LUXURY RENTALS

**TYPE: RENTAL E - COMMERCE**

## **Table of Contents**

### 1. System Architecture Overview

***Component Descriptions***

* Frontend (Next.js) …………………………………………………………………………………………………………………… 02
* Sanity CMS………………………………………………………………………………………………………………………………. 02
* Third-Party APIs………………………………………………………………………………………………………………………. 02
* Payment Gateway…………………………………………………………………………………………………………………... 02

### 2. Key Workflows

* User Registration …………………………………………………………………………………………………………………… 02
* Vehicle Browsing ……………………………………………………………………………………………………………………. 02
* Rental Booking ………………………………………………………………………………………………………………………. 02
* Shipment and Tracking …………………………………………………………………………………………………………… 02

### 3. API Requirements

**API Endpoints**

* /vehicles (GET) ……………………………………………………………………………………………………………………….. 03
* /bookings (POST) ……………………………………………………………………………………………………………………. 03
* /users (GET) ……………………………………………………………………………………………………………………………. 03
* /tracking (GET) …………………………………………………………………………………………………………………….... .03

### 4. Sanity Schema Examples

* **Vehicle Schema ………………………………………………………………………………………………………………………. 03**
* **Booking Schema ……………………………………………………………………………………………………………………… 03**

### 5. Explanation of Schemas

* Vehicle Schema ………………………………………………………………………………………………………………………. 04
* Booking Schema ……………………………………………………………………………………………………………………… 04

### 6. Code Implementations

* **7.1 Frontend (Next.js)**
  + Vehicle Listing Page ……………………………………………………………………………………………………. 05
* **7.2 Backend (API Routes)**
  + API Handler for Vehicle Data ……………………………………………………………………………………….05
* **7.3 Sanity Client Configuration**
  + Sanity Client Setup …………………………………………………………………………………………………….. 05

# **Technical Foundation for Khalil Luxury Rentals**

## **1. System Architecture Overview**

### Component Descriptions:

* **Frontend (Next.js):** Provides a sophisticated, user-friendly interface for customers to browse and rent vehicles.
* **Sanity CMS:** Backend to manage vehicle listings, customer details, and rental records.
* **Third-Party APIs:** Used for real-time shipment tracking and backend services like identity verification.
* **Payment Gateway:** Handles secure and seamless payment processing.

## **2. Key Workflows**

### User Registration

1. User signs up on the platform.
2. Data is securely stored in Sanity CMS.
3. A confirmation email is sent to the user.

### Vehicle Browsing

1. User views available vehicles categorized by type (SUV, Sedan, Hatchback).
2. Sanity API fetches vehicle data, including availability and pricing.
3. Vehicles are dynamically displayed on the frontend with AI-powered recommendations.

### Rental Booking

1. User selects a vehicle and rental duration.
2. Proceeds to checkout with options for add-ons (GPS, insurance).
3. Booking details are saved in Sanity CMS.
4. Payment is processed through the Payment Gateway.
5. Booking confirmation and details are sent to the user.

### Shipment and Tracking

1. User opts for vehicle delivery.
2. Real-time tracking information is fetched via a third-party API.
3. Live updates are provided on the user's dashboard.

## **3. API Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| **API Endpoints** | **Method** | **Description** | **Payload/Response** |
| **/vehicles** | GET | Fetch all available vehicles from Sanity CMS. | **Response:** { "id": 1, "makeModel": "SUV A", "price": 10,000, "availability": "In Stock" } |
| **/bookings** | POST | Create a new booking in Sanity CMS. | **Payload:** { "customerId": 123, "vehicleId": 456, "rentalDuration": "2 days", "addons": ["GPS",”Insurance”], "paymentStatus": "Pending" } |
| **Response:** { "bookingId": 789, "status": "Confirmed" } |
| **/users** | GET | Fetch user details for profile management. | **Response:** { "id": 123, "name": " Khalil ur Rehman", "email": "Khalil.rehman@example.com", "phone": "+9234567890" } |
| **/tracking** | GET | Track rental vehicle delivery status via a third-party API. | **Response:** { "trackingId": 1011, "bookingId": 789, "status": "In Transit", "expectedDelivery": "3 days" } |

## **4. Sanity Schema Examples**

export default {

    name: 'vehicle',

    type: 'document',

    fields: [

      { name: 'makeModel', type: 'string', title: 'Make/Model' },

      { name: 'price', type: 'number', title: 'Daily Rate' },

      { name: 'availability', type: 'string', title: 'Availability' },

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

      { name: 'maintenanceSchedule', type: 'string', title: 'Maintenance Schedule' }

    ]

  };

  Booking Schema

  export default {

    name: 'booking',

    type: 'document',

    fields: [

      { name: 'customerId', type: 'string', title: 'Customer ID' },

      { name: 'vehicleId', type: 'string', title: 'Vehicle ID' },

      { name: 'rentalDuration', type: 'string', title: 'Rental Duration' },

      { name: 'addons', type: 'array', of: [{ type: 'string' }], title: 'Add-Ons' },

      { name: 'paymentStatus', type: 'string', title: 'Payment Status' },

      { name: 'trackingStatus', type: 'string', title: 'Tracking Status' }

    ]

  };

## **Explanation**

## **Vehicle Schema**

The **Vehicle Schema** defines the structure for managing vehicle data within the Sanity CMS. It includes fields such as makeModel, a string representing the vehicle's make and model, and price, a number indicating the daily rental rate. The availability field specifies the vehicle’s status (e.g., "In Stock"), while the features field lists additional attributes like "GPS" or "Air Conditioning." Additionally, the maintenanceSchedule field outlines the vehicle’s maintenance requirements.

## **Booking Schema**

The **Booking Schema** specifies the structure for storing booking data. It contains fields like customerId and vehicleId, both strings that link to the customer and vehicle involved in the booking. The rentalDuration field describes the length of the rental period, while the addons array lists any extra services or products, such as "GPS" or "Insurance." The paymentStatus field tracks the payment progress, such as "Pending" or "Paid," and the trackingStatus field provides information on the delivery status of the rental vehicle, such as "In Transit."

## **7. Some Code Implementations**

### 7.1 Frontend (Next.js)

#### **Vehicle Listing Page**

import React, { useEffect, useState } from 'react';

import axios from 'axios';

type Vehicle = {

id: number;

makeModel: string;

price: number;

};

const VehicleList: React.FC = () => {

const [vehicles, setVehicles] = useState<Vehicle[]>([]);

useEffect(() => {

axios.get('/api/vehicles')

.then(response => setVehicles(response.data))

.catch(error => console.error('Error fetching vehicles:', error));

}, []);

return (

<div>

<h1>Available Vehicles</h1>

<ul>

{vehicles.map(vehicle => (

<li key={vehicle.id}>

{vehicle.makeModel} - PKR{vehicle.price}/day

</li>

))}

</ul>

</div>

);

};

export default VehicleList;

### 7.2 Backend (API Routes)

#### **NextApi Request and NextApi Response from next**

import { NextApiRequest, NextApiResponse } from 'next';

import { sanityClient } from '../../sanity';

export default async function handler(req: NextApiRequest, res: NextApiResponse) {

  if (req.method === 'GET') {

    try {

      const query = `\*[\_type == "vehicle"]`;

      const vehicles = await sanityClient.fetch(query);

      res.status(200).json(vehicles);

    } catch (error) {

      res.status(500).json({ message: 'Error fetching vehicles', error });

    }

  } else {

    res.setHeader('Allow', ['GET']);

    res.status(405).end(`Method {req.method} Not Allowed`);

  }

}

}

### 7.3 Sanity Client Configuration

#### **sanity.ts**

import { createClient } from 'next-sanity';

export const sanityClient = createClient({

  projectId: 'yourProjectId',

  dataset: 'production',

  useCdn: false,

  apiVersion: '2023-01-01'

});

/