DAY 3 - API INTEGRATION REPORT FOR MORENT

1. Understanding the Provided API

1.1 API Documentation Review

- The provided API contains car rental data, including fields such as:
  - `id`: Unique identifier for each car.
  - `name`: Name of the car (e.g., Koenigsegg, Nissan GT-R).
  - `type`: Type of car (e.g., Sport).
  - `fuel_capacity`: Fuel capacity of the car (e.g., 90L).
  - `transmission`: Transmission type (e.g., Manual).
  - `seating_capacity`: Number of seats (e.g., 2 People).
  - `price_per_day`: Rental price per day (e.g., $99.00).
  - `image_url`: URL of the car image.
  - `tags`: Tags for categorization (e.g., "popular").

1.2 Key Endpoints

- **Product Listings**: `/api/hackathon/template7` (to fetch all cars).
- **Categories**: Not explicitly provided, but car types (e.g., Sport, SUV) can be derived from the `type` field.
- **Order History**: Not applicable for this API.

Endpoints overview has been provided in the documentation of api specifications.

---

2. Validating and Adjusting the Schema

2.1 Comparison of API Data and Sanity Schema

API Fields:

```
{
  "id": 1,
  "name": "Koenigsegg",
  "type": "Sport",
  "fuel_capacity": "90L",
  "transmission": "Manual",
  "seating_capacity": "2 People",
  "price_per_day": "$99.00",
  "image_url": "https://example.com/car.jpg",
  "tags": ["popular"]
   }
```

Sanity Schema Fields:

```
export default {
  name: 'car',
  type: 'document',
  fields: [
    { name: 'name', type: 'string', title: 'Name' },
    { name: 'type', type: 'string', title: 'Type' },
    { name: 'fuelCapacity', type: 'string', title: 'Fuel Capacity' },
    { name: 'transmission', type: 'string', title: 'Transmission' },
    { name: 'seatingCapacity', type: 'string', title: 'Seating Capacity'
},
    { name: 'pricePerDay', type: 'string', title: 'Price per Day' },
    { name: 'originalPrice', type: 'string', title: 'Original Price' },
    { name: 'tags', type: 'array', of: [{ type: 'string' }], title: 'Tags'
},
    { name: 'image', type: 'image', title: 'Car Image' },
  ],
      };
```

2.2 Schema Adjustments

- Adjusted the schema to match the API data:
  - `name`: Mapped directly.
  - `type`: Mapped directly.
  - `fuel_capacity`: Mapped to `fuelCapacity`.
  - `transmission`: Mapped directly.
  - `seating_capacity`: Mapped to `seatingCapacity`.
  - `price_per_day`: Mapped to `pricePerDay`.
  - `image_url`: Mapped to `image` after uploading to Sanity.
  - `tags`: Mapped directly.

---

3. Data Migration

3.1 Migration Script

- Used the provided migration script to fetch data from the API and upload it to Sanity CMS.

3.2 Migration Steps

1. **Set Up Environment Variables**:
   - Created a .env.local file with the following variables:

```
NEXT_PUBLIC_SANITY_PROJECT_ID=your-project-id
```

```
NEXT_PUBLIC_SANITY_DATASET=your-dataset
SANITY_API_TOKEN=your-api-token
```

2. **Run the Migration Script**:
   - Compiled the TypeScript file using:
     ```
     tsc
     ```
   - Executed the script using:
     ```
     node importData.js
     ```
3. **Verify Data in Sanity CMS**:
   - Checked the Sanity Studio to ensure all car data was imported correctly.
   - Verified that images were uploaded and linked to the respective car entries.

3.3 Tools Used

- **Sanity CLI**: For data import.
- **Axios**: For fetching data from the API.
- **Node.js**: For running the migration script.
-

---

4. API Integration in Next.js

4.1 Utility Functions

- Created utility functions to fetch data from the API and Sanity CMS.

4.2 Rendering Data

- Displayed data in components such as product listings and car details.

4.3 Testing

- Tested API endpoints using Postman and browser developer tools.

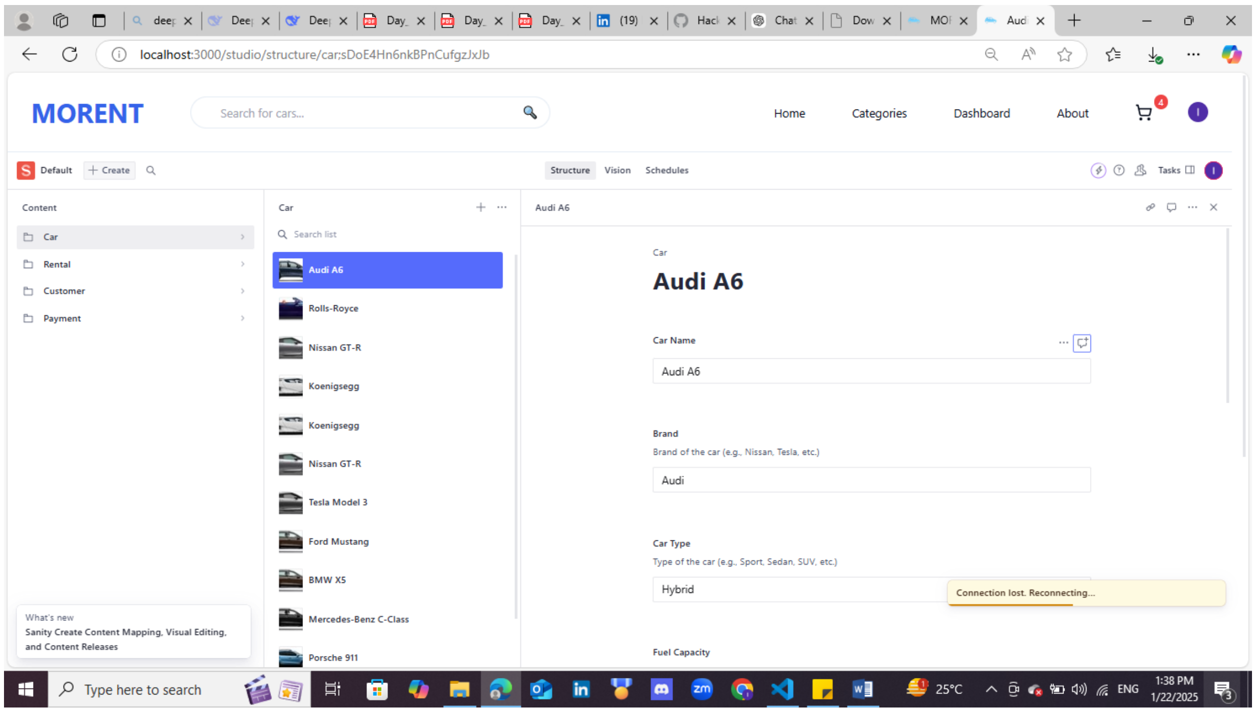
---

5. Expected Output

- **Sanity CMS**: Successfully populated with car data.
- **Next.js Frontend**: Functional integration displaying product listings with accurate data.

Submission Attachments

Screenshot of populated schema:



localhost:3000/studio/structure/car;sDoE4Hn6nkBPnCufgzJxJb

MORENT

Content: Car, Rental, Customer, Payment

Car: Audi A6, Rolls-Royce, Nissan GT-R, Koenigsegg, Koenigsegg, Nissan GT-R, Tesla Model 3, Ford Mustang, BMW X5, Mercedes-Benz C-Class, Porsche 911

Car
Audi A6

Car Name: Audi A6

Brand — Brand of the car (e.g., Nissan, Tesla, etc.): Audi

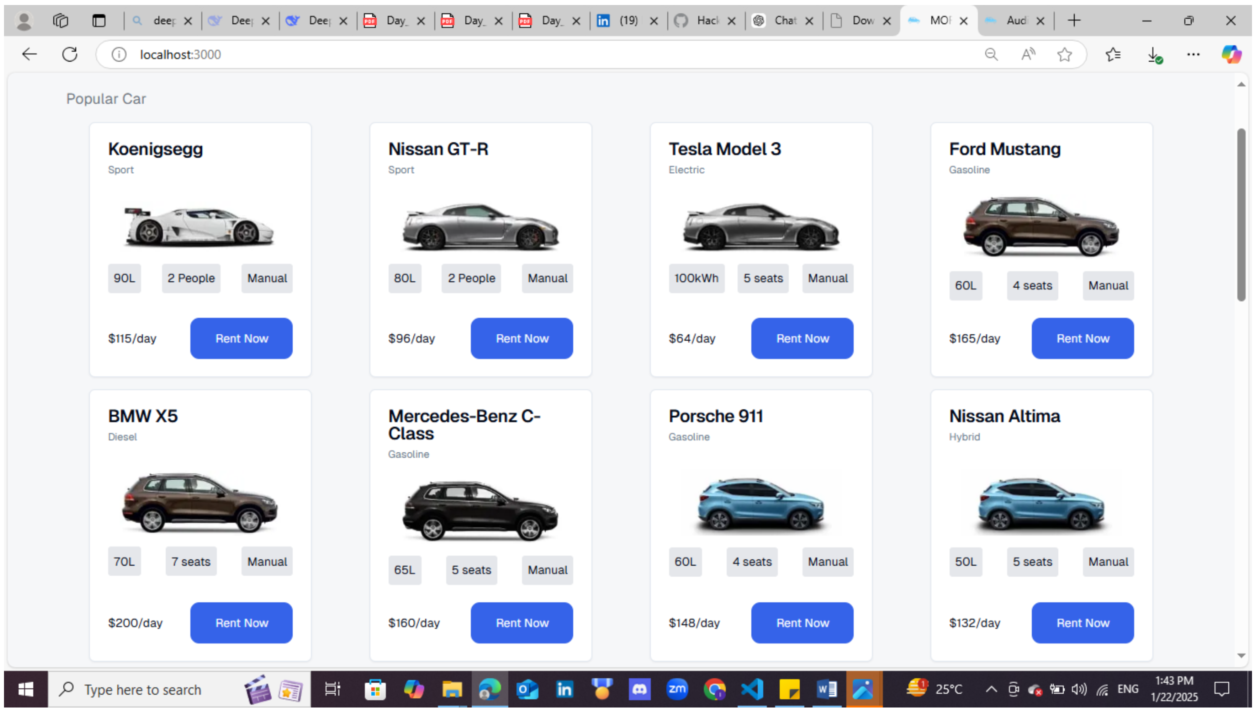
Car Type — Type of the car (e.g., Sport, Sedan, SUV, etc.): Hybrid

Connection lost. Reconnecting...

Fuel Capacity

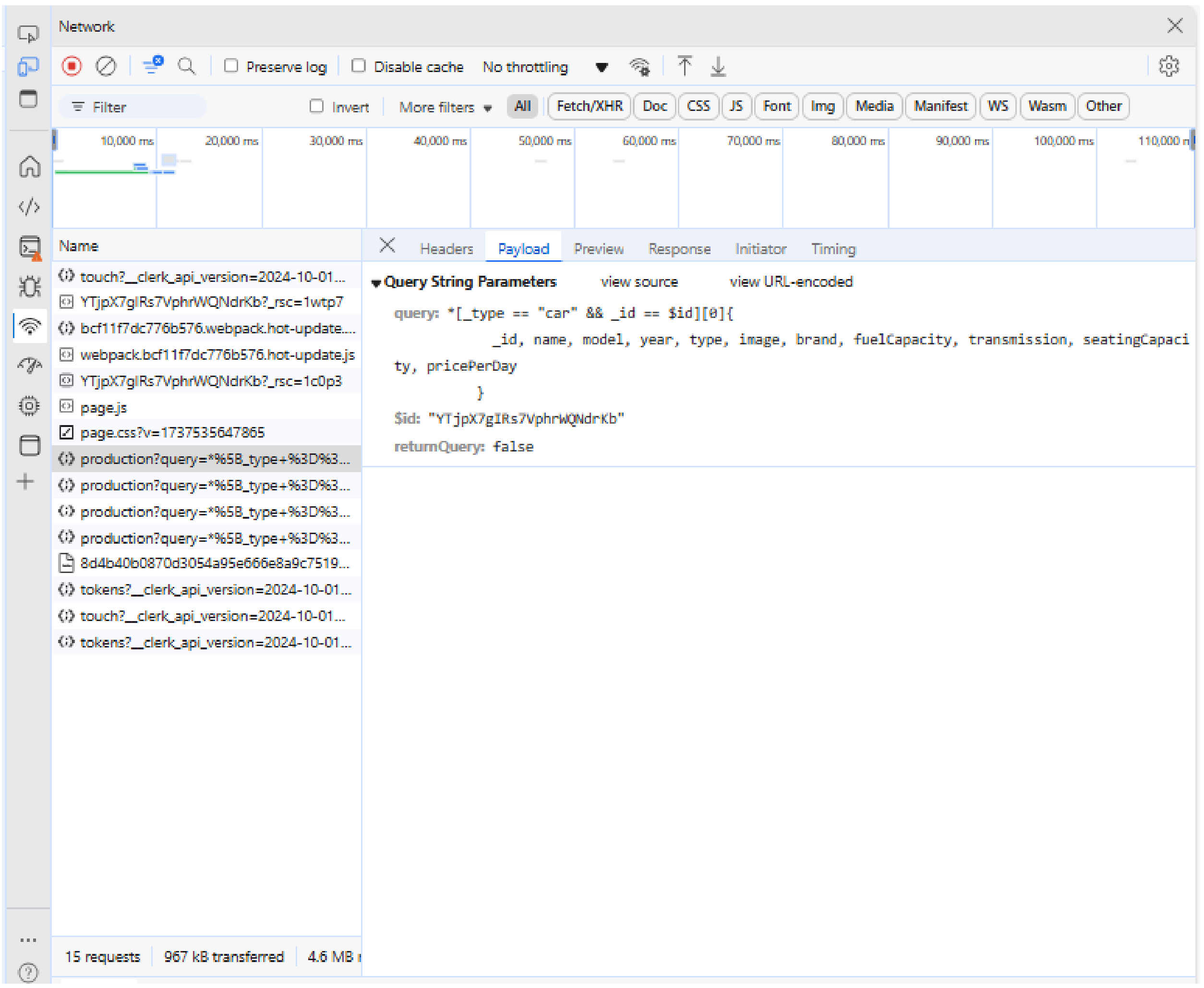
What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

Screenshot of data successfully displayed in frontend:


Popular Car

Koenigsegg Sport 90L · 2 People · Manual $115/day · Rent Now	Nissan GT-R Sport 80L · 2 People · Manual $96/day · Rent Now	Tesla Model 3 Electric 100kWh · 5 seats · Manual $64/day · Rent Now	Ford Mustang Gasoline 60L · 4 seats · Manual $165/day · Rent Now
BMW X5 Diesel 70L · 7 seats · Manual $200/day · Rent Now	Mercedes-Benz C-Class Gasoline 65L · 5 seats · Manual $160/day · Rent Now	Porsche 911 Gasoline 60L · 4 seats · Manual $148/day · Rent Now	Nissan Altima Hybrid 50L · 5 seats · Manual $132/day · Rent Now

Screenshot of API calls:



Name

```
touch?__clerk_api_version=2024-10-01...
YTjpX7gIRs7VphrWQNdrKb?_rsc=1wtp7
bcf11f7dc776b576.webpack.hot-update....
webpack.bcf11f7dc776b576.hot-update.js
YTjpX7gIRs7VphrWQNdrKb?_rsc=1c0p3
page.js
page.css?v=1737535647865
production?query=*%5B_type+%3D%3...
production?query=*%5B_type+%3D%3...
production?query=*%5B_type+%3D%3...
production?query=*%5B_type+%3D%3...
8d4b40b0870d3054a95e666e8a9c7519...
tokens?__clerk_api_version=2024-10-01...
touch?__clerk_api_version=2024-10-01...
tokens?__clerk_api_version=2024-10-01...
```

15 requests | 967 kB transferred | 4.6 MB

Query String Parameters view source view URL-encoded

```
query: *[_type == "car" && _id == $id][0]{
          _id, name, model, year, type, image, brand, fuelCapacity, transmission, seatingCapaci
ty, pricePerDay
        }
$id: "YTjpX7gIRs7VphrWQNdrKb"
returnQuery: false
```

Code snippets for API integration:

```
export default function Home() {
  const [cars, setCars] = useState<Car[]>([]);
  const searchParams = useSearchParams();
  const searchQuery = searchParams.get('search') || '';

  useEffect(() => {
    const fetchCars = async () => {
      let query = '*[_type == "car"]{_id, name, type, image, pricePerDay,
fuelCapacity, transmission, seatingCapacity}';
      const data = await client.fetch(query);
      setCars(data);
    };

    fetchCars();
  }, []);

   const filteredCars = cars.filter((car) =>
    car.name.toLowerCase().includes(searchQuery.toLowerCase())
  );
```

```
import { NextResponse } from 'next/server';
import { client } from '../../../sanity/lib/client';
import { groq } from 'next-sanity';

const query = groq`*[_type == "car"]{
  _id,
  name,
  type,
  image,
  pricePerDay,
  fuelCapacity,
  transmission,
  seatingCapacity
}`;

export async function GET() {
  try {
    const cars = await client.fetch(query);
    return NextResponse.json(cars);
  } catch (error) {
    return NextResponse.json({ message: 'Error fetching data' }, { status: 500
});
  }
}
```

Migration Scripts:

```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop()
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log('Fetching car data from API...');

    // API endpoint containing car data
    const response = await axios.get('https://sanity-nextjs-
application.vercel.app/api/hackathon/template7');
    const cars = response.data;

    console.log(`Fetched ${cars.length} cars`);

    for (const car of cars) {
      console.log(`Processing car: ${car.name}`);
```

```javascript
      let imageRef = null;
      if (car.image_url) {
        imageRef = await uploadImageToSanity(car.image_url);
      }

      const sanityCar = {
        _type: 'car',
        name: car.name,
        brand: car.brand || null,
        type: car.type,
        fuelCapacity: car.fuel_capacity,
        transmission: car.transmission,
        seatingCapacity: car.seating_capacity,
        pricePerDay: car.price_per_day,
        originalPrice: car.original_price || null,
        tags: car.tags || [],
        image: imageRef ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: imageRef,
          },
        } : undefined,
      };

      console.log('Uploading car to Sanity:', sanityCar.name);
      const result = await client.create(sanityCar);
      console.log(`Car uploaded successfully: ${result._id}`);
    }

    console.log('Data import completed successfully!');
  } catch (error) {
    console.error('Error importing data:', error);
  }
}

importData();
```