Day : 3

API Integration Report

Marketplace Name: [Rental-Ecommerce]
Prepared by: [Sowaiba Naz]
Roll No: [00490877]

---

1. Introduction

The objective of Day 3 was to integrate APIs and migrate data into Sanity CMS to build a functional marketplace backend. This exercise aimed to equip practical skills in API integration, data migration, schema validation, and frontend integration using Next.js. The process involved understanding provided APIs, adjusting schemas, migrating data, and ensuring seamless integration with the frontend.

---

2. API Integration Process

2.1 Understanding the Provided API

- Reviewed the API documentation for the assigned template. (https://sanity-nextjs-application.vercel.app/api/hackathon/template7)
- The API documentation provided clear guidance on schema validation and data migration.

2.2 Schema Validation and Adjustments

- Compared the existing Sanity CMS schema with the API data structure. (https://github.com/AsharibAli/sanity-nextjs/blob/main/sanity/schemaTypes/cars.ts)

- Adjusted field names, types, and relationships to ensure compatibility. For example:
    - API Field: `product_title` → Schema Field: `name`
- Mapped fields during migration to align with the schema requirements.

2.3 Migration Steps and Tools Used

Method were used for data migration:

- **Using the Provided API**:
    - Wrote scripts to fetch and transform data from the API.
    - Referenced migration scripts from the assigned template repository. (https://github.com/AsharibAli/sanity-nextjs/blob/main/scripts/importTemplate7Data.mjs).
    - Mapped fields to the Sanity schema and followed a script-based migration process.

---

3. API Integration in Next.js

3.1 Utility Functions

- Created utility functions to fetch data from the provided API or Sanity CMS.
- Ensured robust error handling and modular coding for scalability.

3.2 Rendering Data in Components

- Integrated data into next js components to display:
    - Home
    - Details
    - Payments
    - etc

3.3 API Integration in Next.js

- Implemented fallback data or skeleton loaders for a better user experience.

---

4. Screenshots

1. Setup API in Next.js

- Installed “axios” and “dotenv” for making APIs calls.
- Created a file scripts/importTemplate7Data.mjs

```
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}`);

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

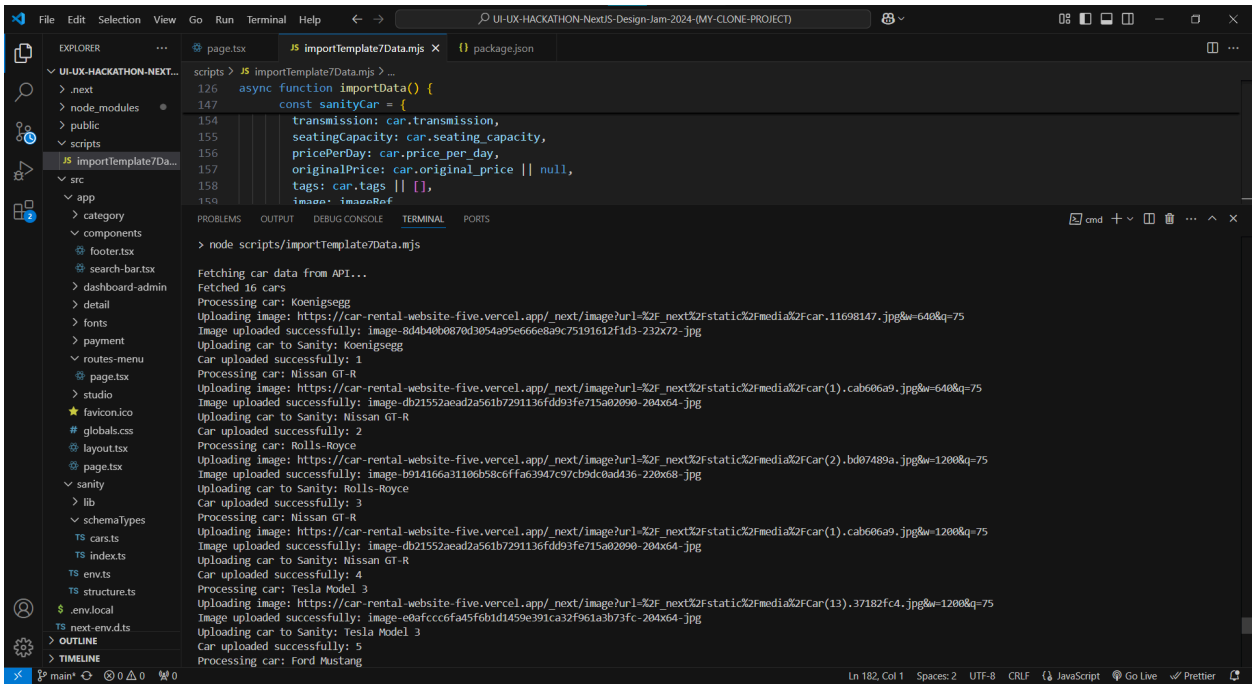
2. Adjustment made to Schemas

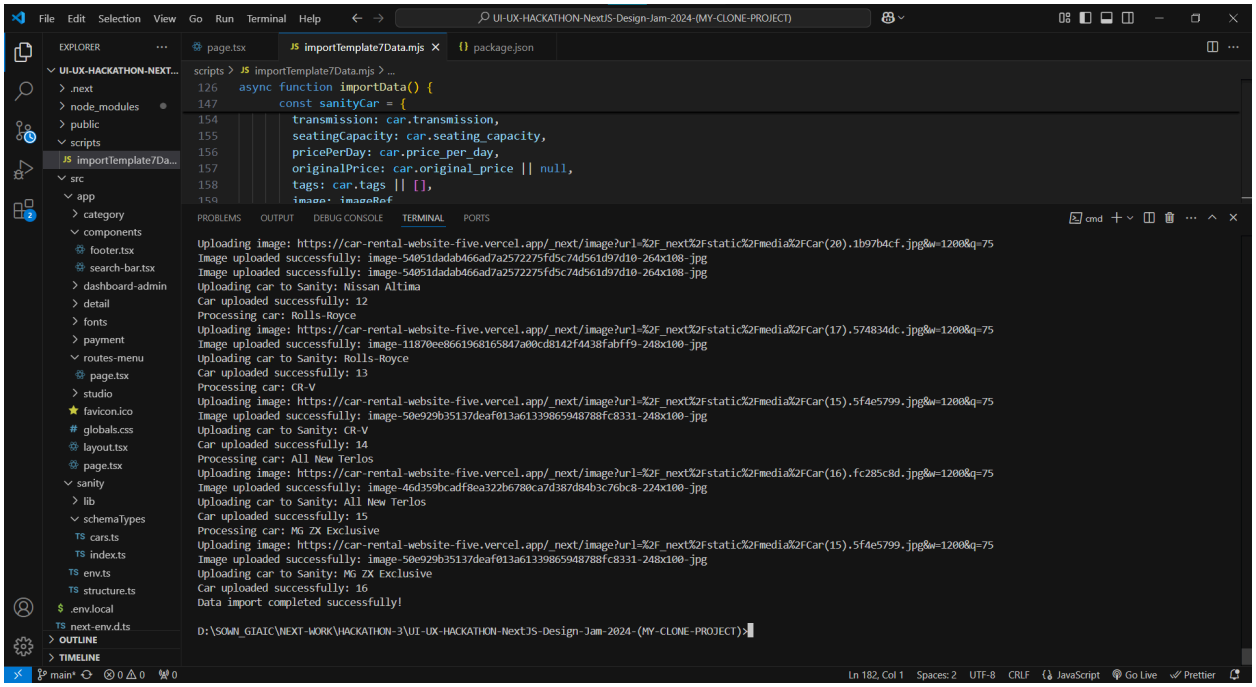
- Do changes in provided schemas to match my frontend design.
- Modified Field Validations.
- Adjusted Schema in Sanity studio.

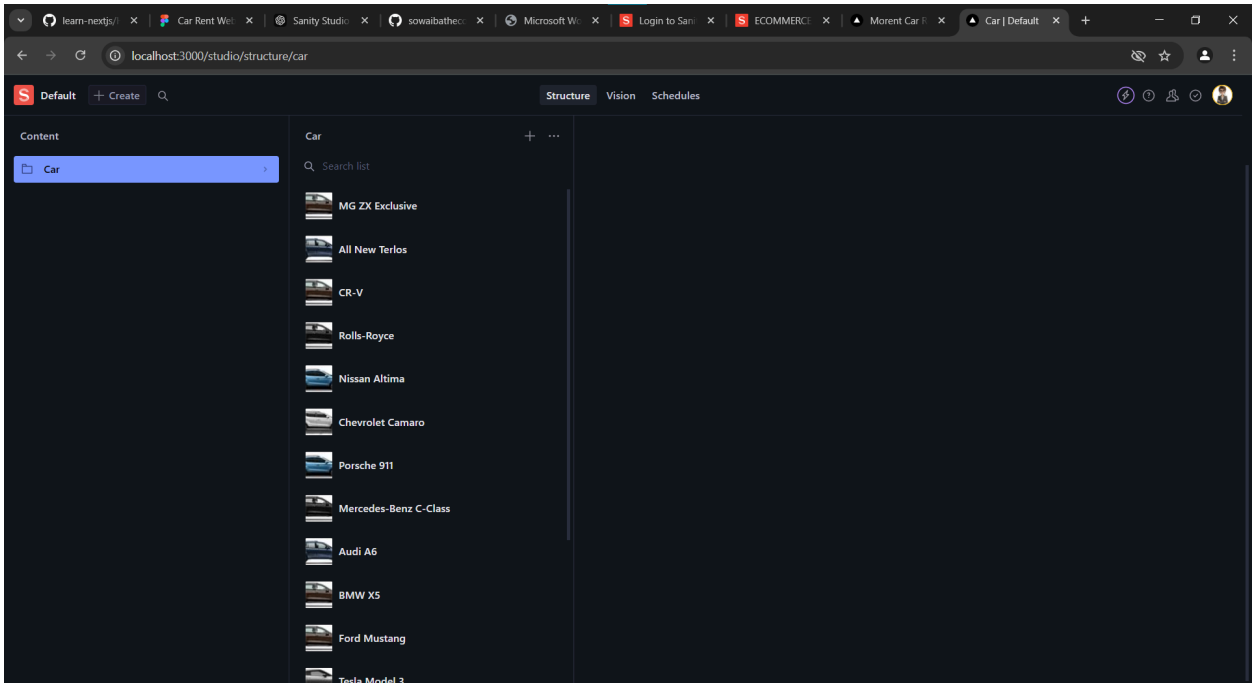
```
export const carsSchema = {
  name: 'car',
  type: 'document',
  title: 'Car',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Car Name',
    },
    {
      name: 'type',
      type: 'string',
      title: 'Car Type',
      description: 'Type of the car (e.g., Sport, Sedan, 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: 'string',
      title: 'Price Per Day',
      description: 'Rental price per day',
    },
    {
      name: 'originalPrice',
      type: 'string',
      title: 'Original Price',
      description: 'Original price before discount (if applicable)',
    },
    {
      name: 'image',
      type: 'image',
      title: 'Car Image',
      options: {
        hotspot: true
      }
    }
  ],
};
```

3. Migration Steps and Tools Used

- Used the provided migration script
- Imported data into sanity
- Node.js for running migration scripts
- Sanity Studio for schema and data management

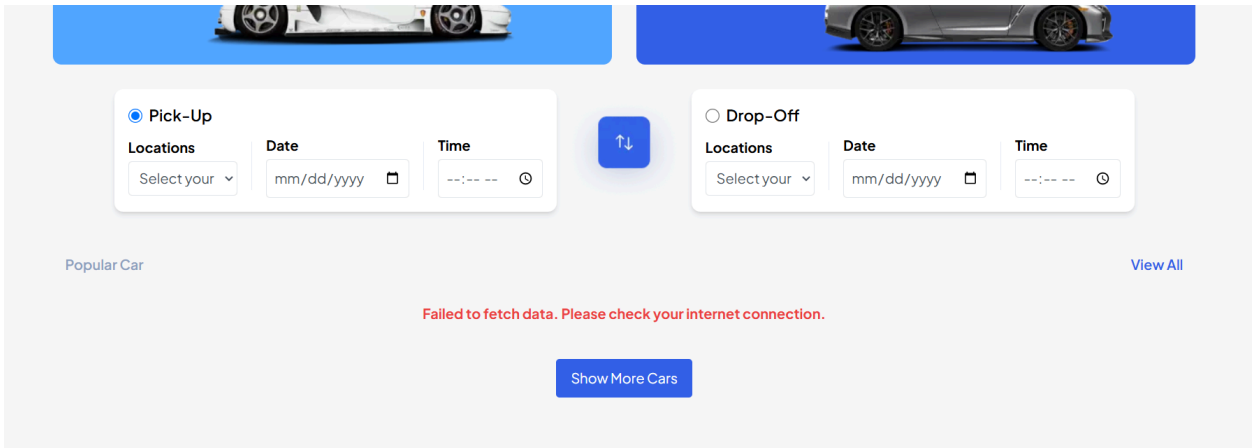






4. Offline Handling

- Implemented error handling using `try...catch` to manage scenarios where the internet connection is unavailable.



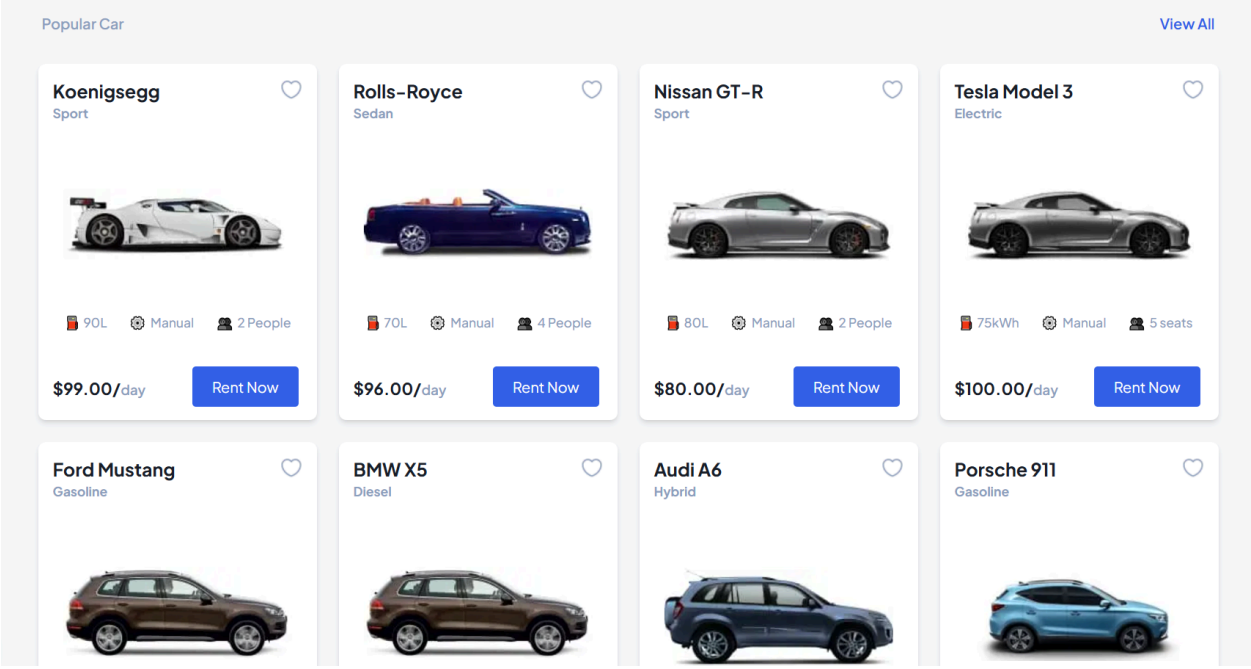
```tsx
export default function CarCards({ visibleCar }: CarCardsProps) {
  const [carData, setCarData] = useState<any[]>([]);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const getCars = async () => {
      try {
        const data = await fetchCars(); // Fetching data
        setCarData(data);
        setError(null);
      } catch (err: any) {
        setError(err.message); // Set error message
      }
    };

    getCars();
  }, []);
```

5. Final Result

- Data displayed on frontend.
- Successfully fetched data from sanity and displayed it using next.js components.
- Added styles to match the frontend design.



---

5. Best Practices Followed

✔ Stored sensitive data in `.env.local` file and added it to `.gitignore`.
✔ Implemented data validation and error handling.
✔ Properly documented schema mapping and migration.
✔ Followed clean code practices.

6. Submission Checklist

✅ Documentation of API integration and data migration.
✅ Screenshots of API calls, frontend, and Sanity CMS.
✅ Code snippets of API integration and migration scripts.
✅ Completed self-validation checklist.

---

7. Conclusion

The Day 3 exercise successfully achieved its objectives by enabling students to:

- Integrate APIs into their Next.js projects.
- Migrate data into Sanity CMS.
- Validate and adjust schemas for compatibility.
- Build scalable marketplaces using headless CMS systems like Sanity.