

**Project**

**Overview**

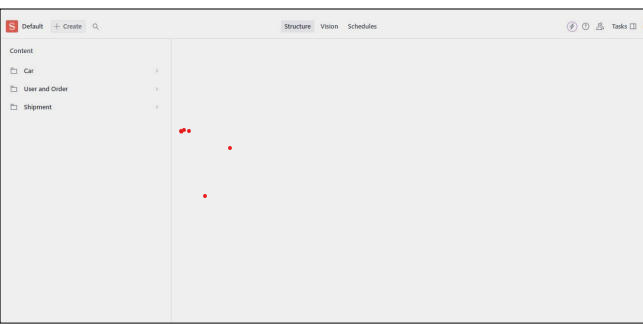
This report documents the API integration process, schema adjustments, and data migration steps performed to populate Sanity CMS with imported data and display it on the front-end.

Screenshots and code snippets are included to provide clarity and demonstrate success.

# Steps Completed

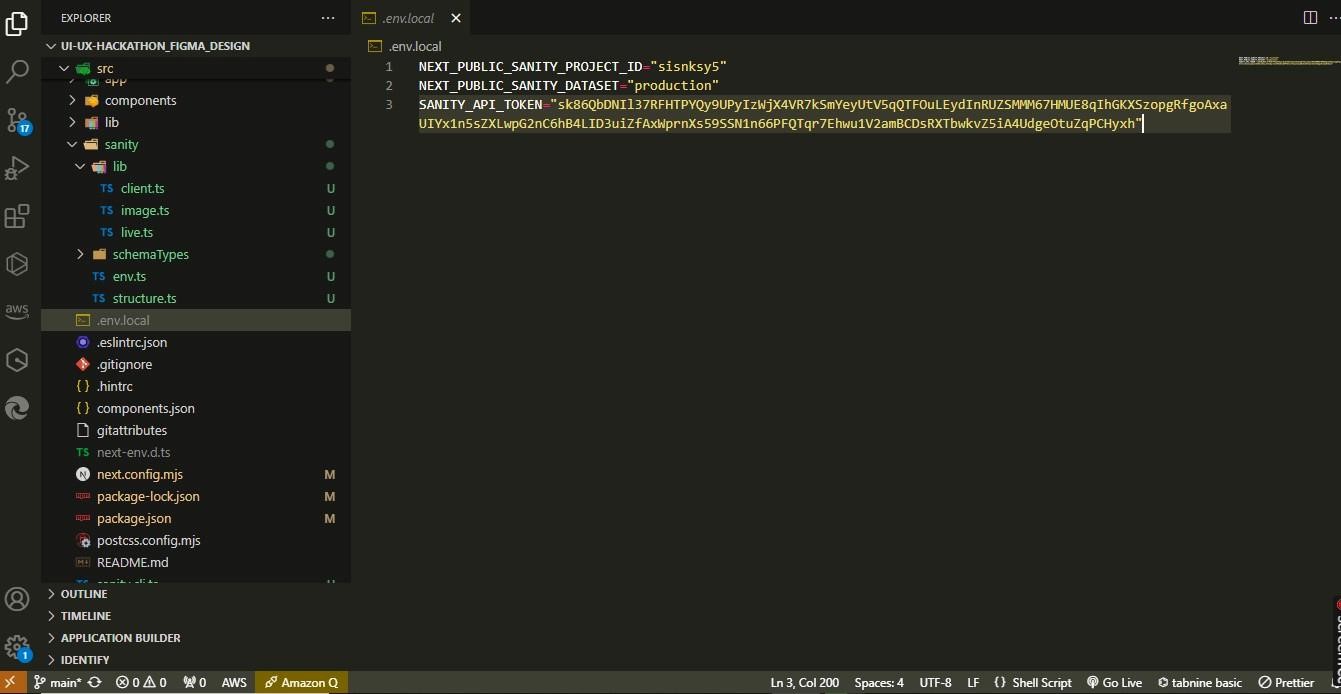
## Step 1: Installing Sanity Studio

1. Installed Sanity Studio into the Next.js project using the command npx sanity init.
2. Verified the installation by accessing the Sanity dashboard.
3. **Outcome**: Successfully set up Sanity Studio in the project structure.



## Step 2: Setting Environment Variables

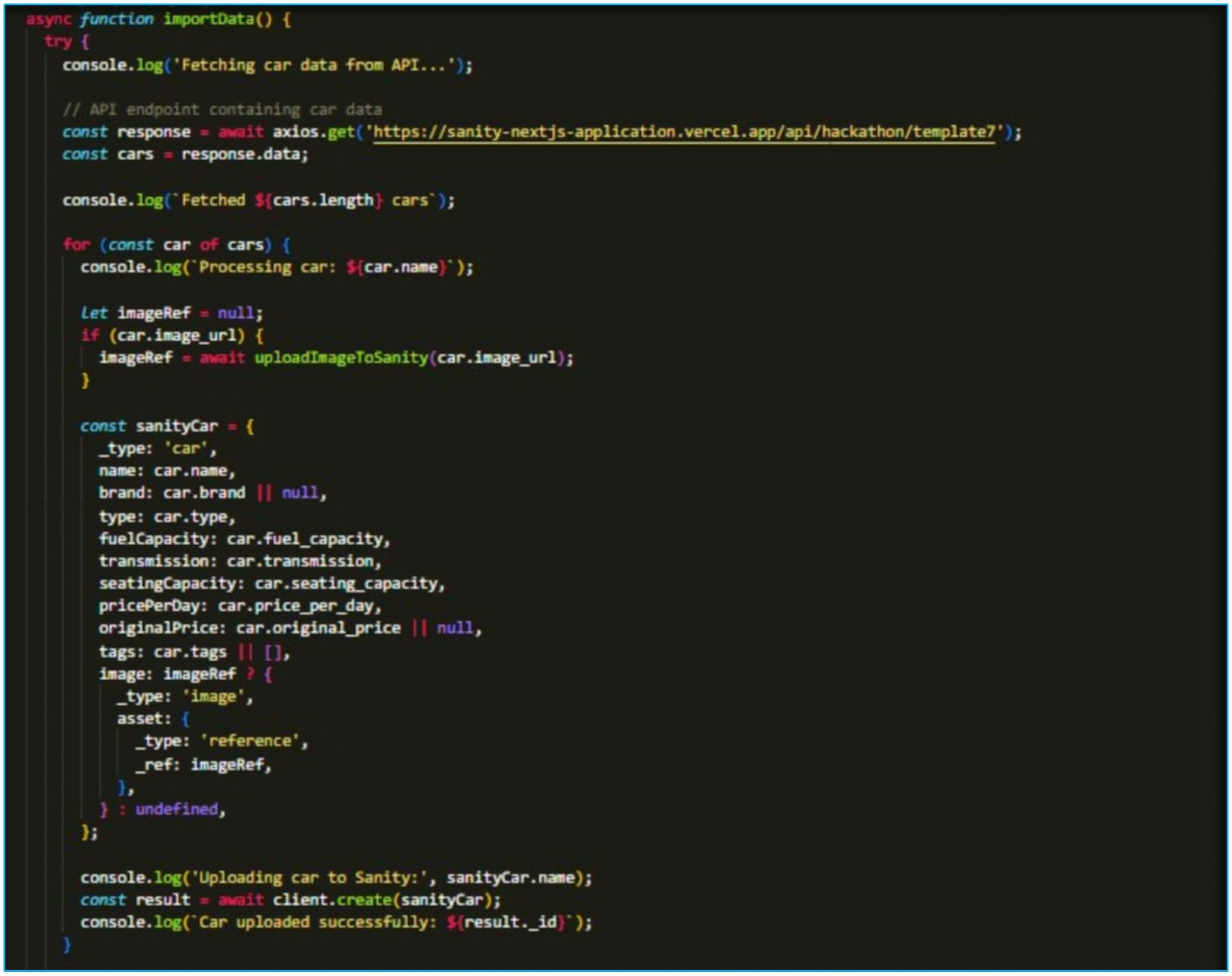
1. Created a .env.local file at the root of the project.
2. Added the following environment variables:
   1. NEXT\_PUBLIC\_SANITY\_PROJECT\_ID: Your Sanity project ID.
   2. NEXT\_PUBLIC\_SANITY\_DATASET: Dataset name (e.g., production).
   3. SANITY\_API\_TOKEN: Token for API authentication.
3. Used the dotenv library to securely load these variables.
4. **Outcome**: Environment variables were successfully configured.



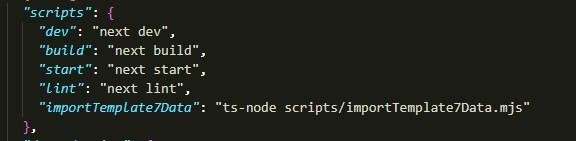
**Note**: Ensure these variables are not exposed in public repositories for security purposes.

## Step 3: Fetching API Data into Sanity Studio

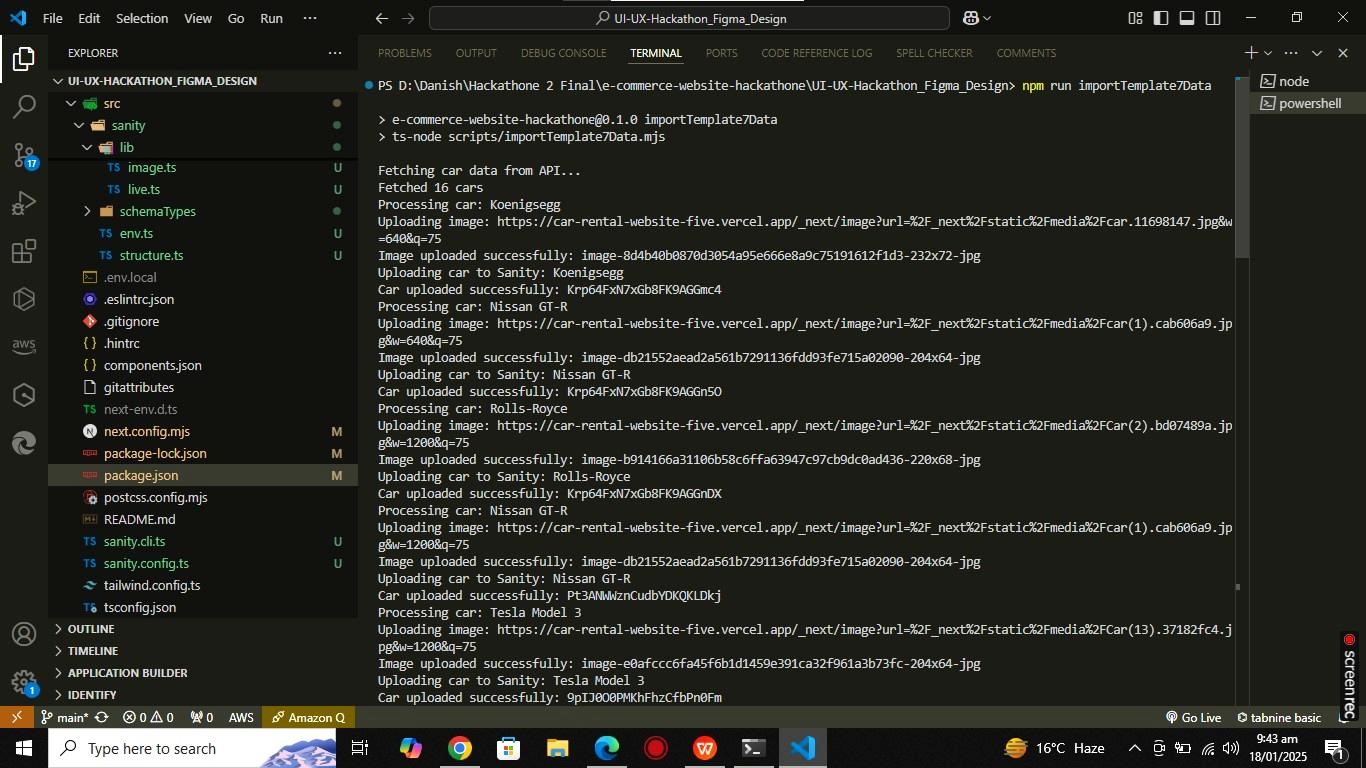
1. **Script Setup:** Created a script importTemplate7Data.ts to handle API data import.
   * 1. Utilized axios to fetch data from the provided external API.
     2. Utilized @sanity/client to create and upload content to Sanity CMS.



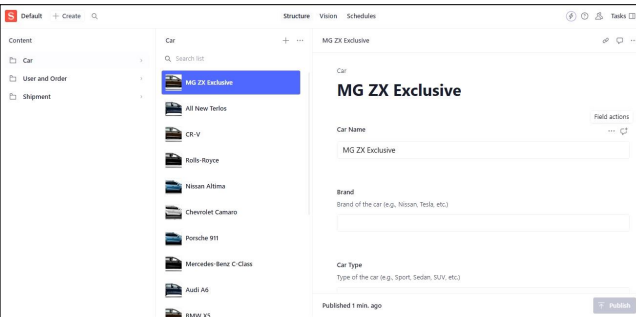
1. **Logic Implementation:**
   * 1. Implemented a function to upload images and map API data to Sanitycompatible schema types.
     2. Used the create Or Replace method to upload each record into Sanity Studio.
2. **Execution:**
   1. Added a custom script in package.json: "importData": "ts-node importTemplate7Data.ts".



* 1. Executed the script with the command npm run importData.

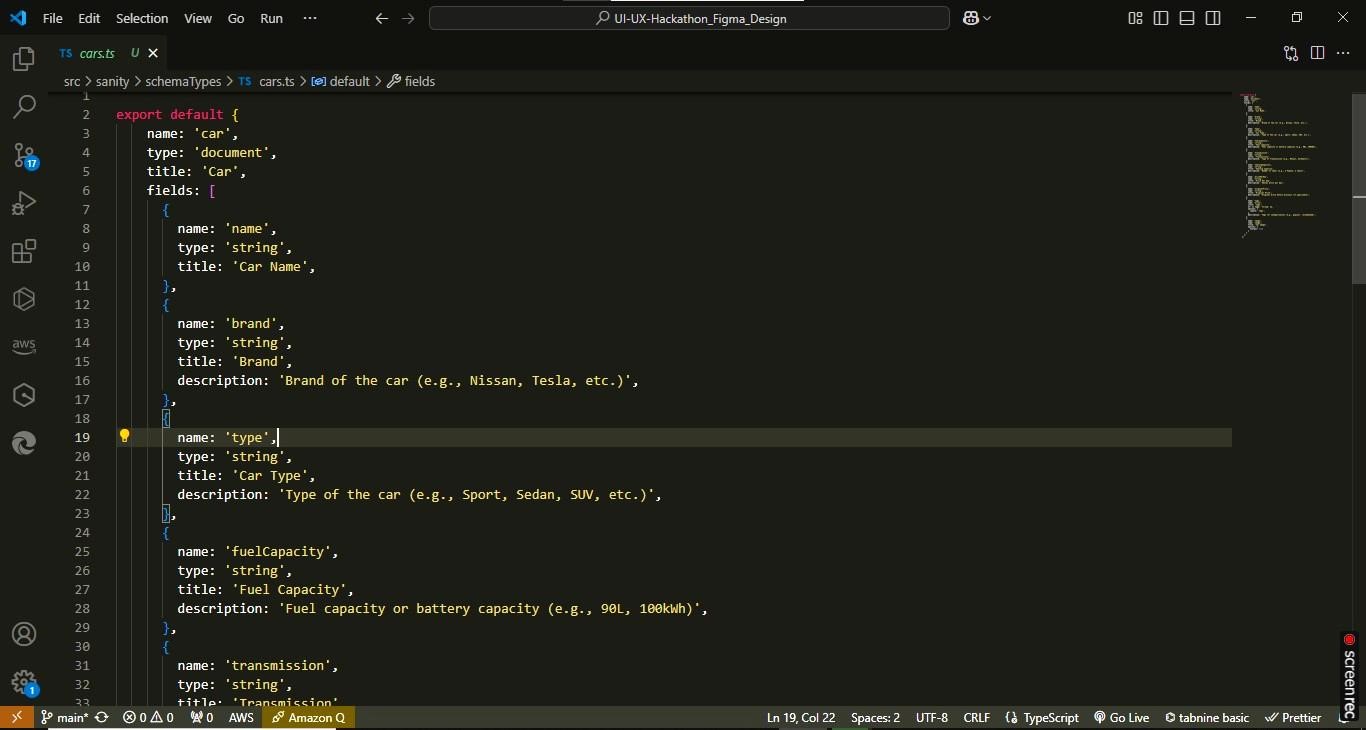


1. **Outcome:** Successfully imported external data into Sanity Studio.

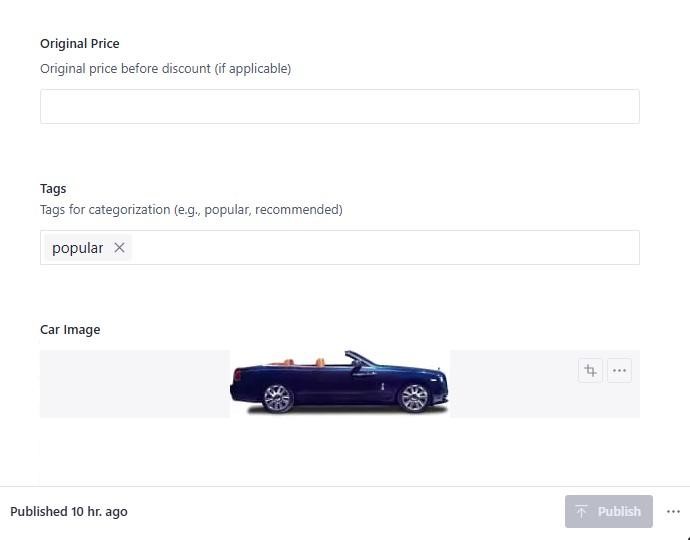
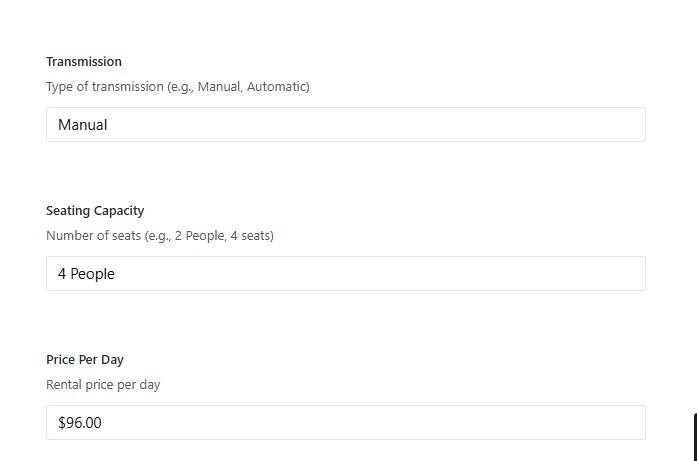
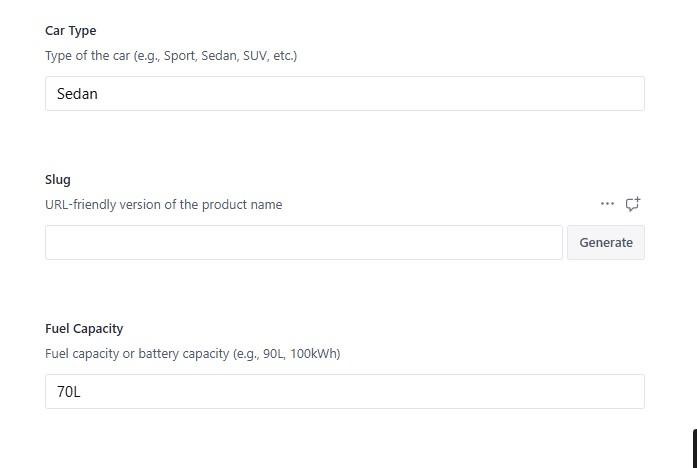


## Step 4: Adjusting Schemas

1. Reviewed and updated schemas to match imported data structure (e.g., car type).
2. Added fields such as title, description, price, and image to ensure compatibility.

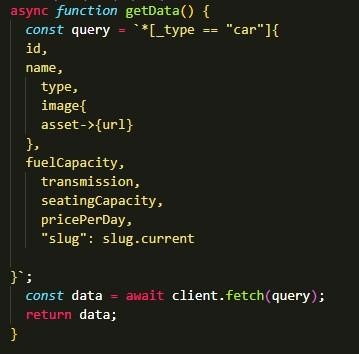


1. Validated schema adjustments using Sanity Studio’s preview feature.
2. **Outcome**: Schema successfully aligned with external API data structure.



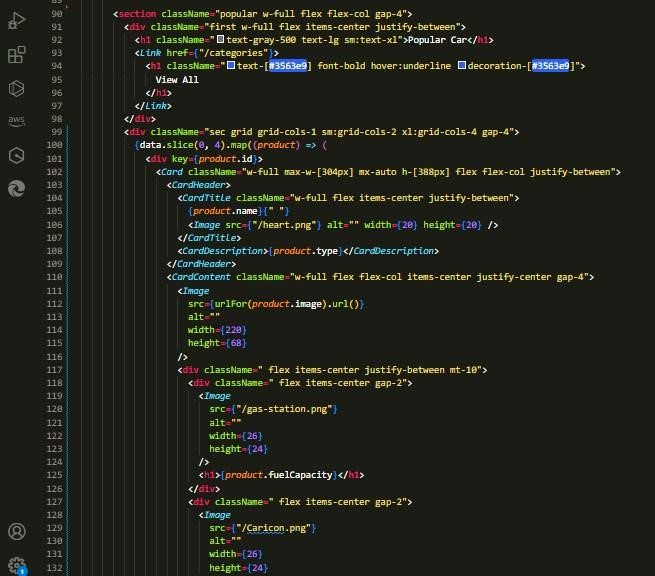
## Step 5: Front-end API Integration

1. **Data Fetching:** Used next-sanity to fetch data from Sanity.
   1. Created a query to retrieve car listings and their details.

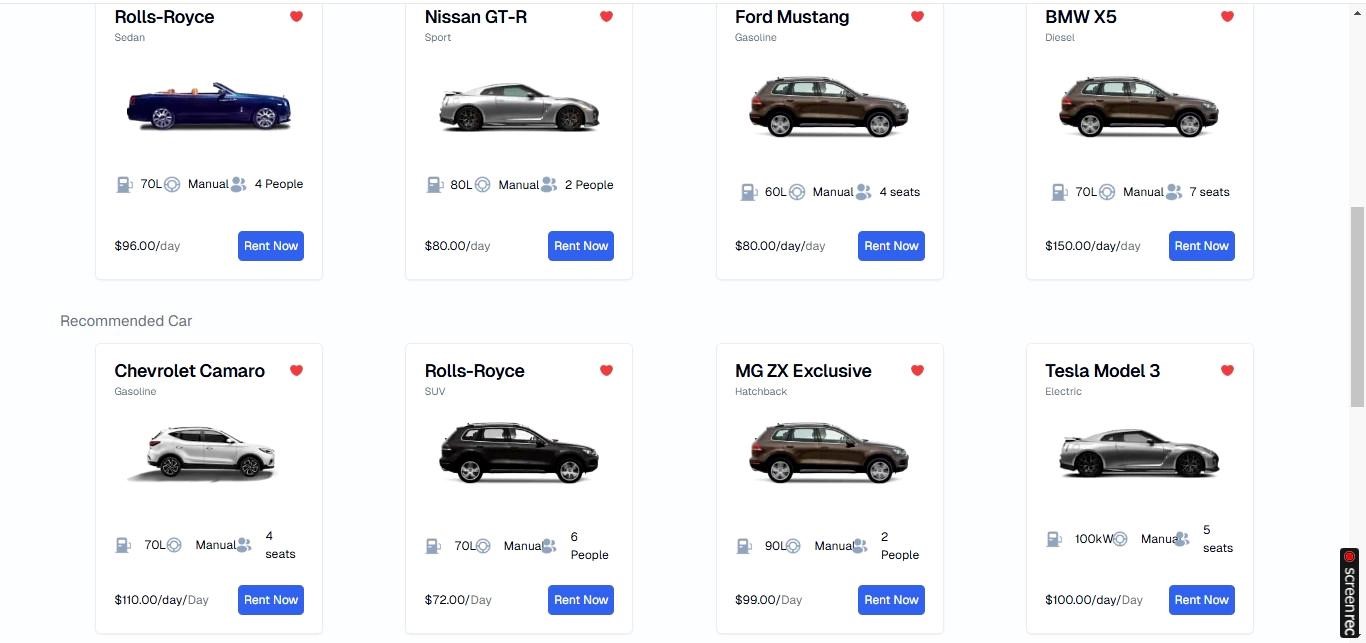


* 1. Verified the API response using the browser console and Postman.

1. **Component Development:**
   1. Designed a grid layout component to display car details (e.g., title, price, image).
   2. Integrated Tailwind CSS for styling.



1. **Testing:** Verified data rendering on the front-end and ensured no layout or API errors.
2. **Outcome**: Successfully displayed fetched data in a grid layout on the front-end.



# Final Outcome

* **Sanity CMS**: Populated with imported data.
* **Front-end**: Data displayed successfully in a user-friendly format.
* **Verification**: All components and workflows tested and validated.

# Conclusion

Through a structured and systematic process, we successfully integrated external API data into Sanity Studio and displayed it seamlessly on the front-end of the Next.js project. The following milestones were achieved:

1. **Sanity Studio Setup:** Installed and configured Sanity Studio as a CMS for managing content, ensuring it integrates effectively with the Next.js project.
2. **Environment Variables:** Securely configured environment variables to manage sensitive information such as project ID, data set, and API token.
3. **Data Import:** Developed and executed a robust script to fetch API data, process it, and populate Sanity Studio with the required data fields.
4. **Frontend Integration:** Fetched data from Sanity Studio and displayed it dynamically in a visually appealing grid layout on the front-end using reusable components and modern design practices.
5. **Verification:** Ensured the accuracy and functionality of the workflow through thorough testing and debugging.

This workflow ensures a scalable, reusable, and efficient approach to integrating and managing external data sources in a modern web application, leveraging the capabilities of Sanity Studio and Next.js. The successful outcome highlights the effectiveness of the implementation and sets a strong foundation for future enhancements.