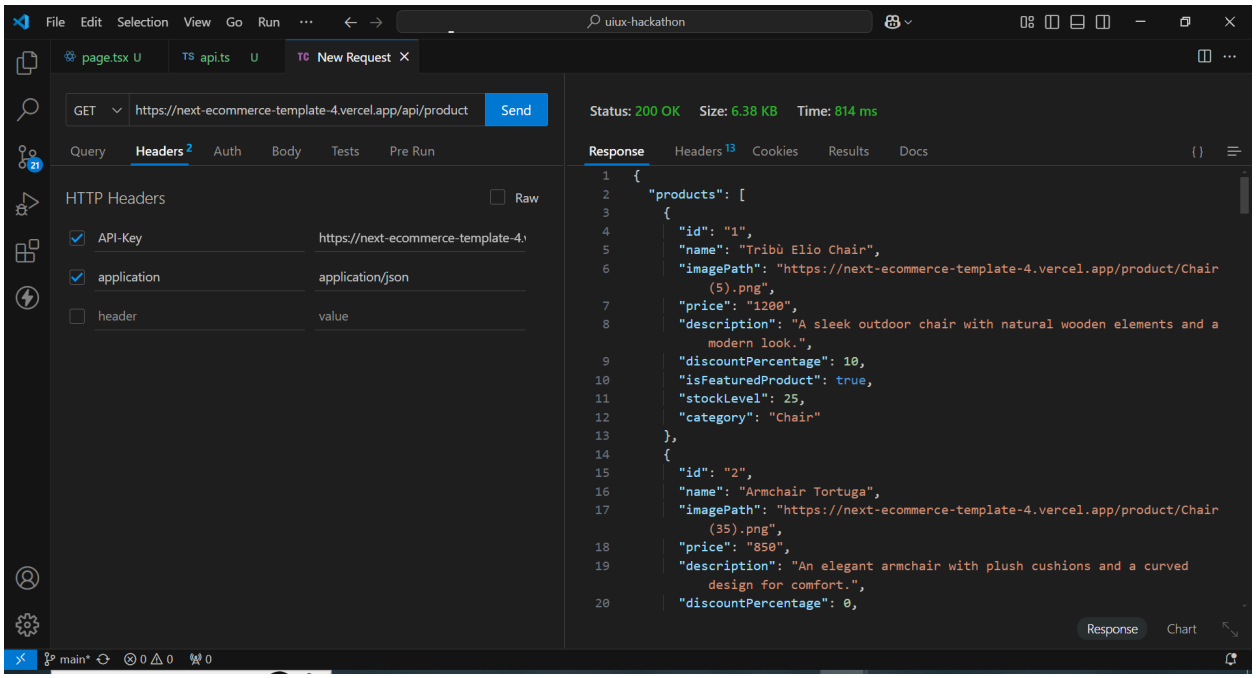
Day 3- API - Integration- Report Ecommerce Website (Hekto)

This documentation outlines the work I have completed on Day 3 of the HEKTO my ecommerce marketplace hackathon.it covers migration, data Integration from Sanity, Creating Schema, and displaying the data using GROQ queries in a next.js application.

Api Integration Process

To successfully migrate data into Sanity and integrate it into a Next.js project, I utilized the provided API. The first step involved testing the API to ensure it was functioning as expected. I used Thunder Client, a powerful extension for VS Code, to send GET requests to the API endpoints and verify the data retrieval process. The GET method was chosen to fetch data from the API and check if the responses matched the expected results. Below, I've included a screenshot that demonstrates how I tested the API endpoint using Thunder Client.



Adjustments made in Old Schema:-

In order to accommodate the new API structure and ensure better compatibility with the Next.js application, the following adjustments were made to the existing schemas.

- **Added Fields**:

  `isFeaturedProduct` (Boolean): To identify featured products.

  `discountPercentage` (Number): To represent the discount offered on a product.

- **Modified Fields**:

  Category: modified the field according to the given api

- **Removed Fields:**

  Ratings (Number): Deprecated unused fields like `ratings and ratingcout`

Migration Steps:-

These are the step-by-step instructions to set up a project and import data into Sanity for the Products models.

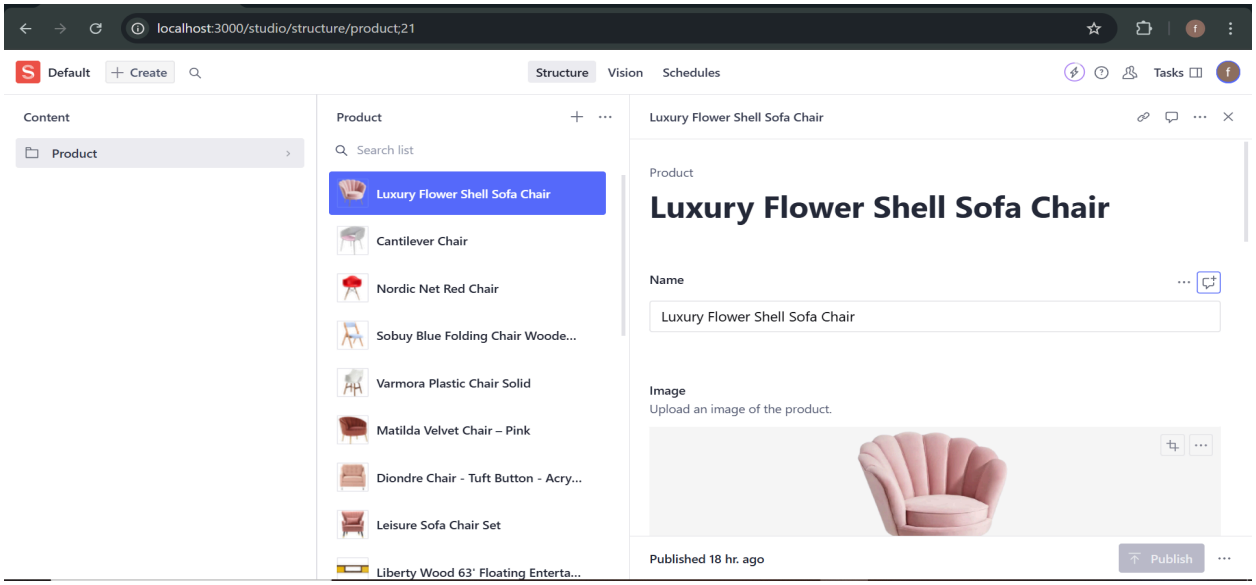
1. ### Sanity Setup:-

   Initialize a new sanity projection next.js application, create an API token in your Sanity dashboard and store the project ID, token, and dataset name in your project's `.env` file for secure access. This allows you to interact with your Sanity CMS and manage content within your Next.js application.

2. **Fetch Data from the Given API:-**

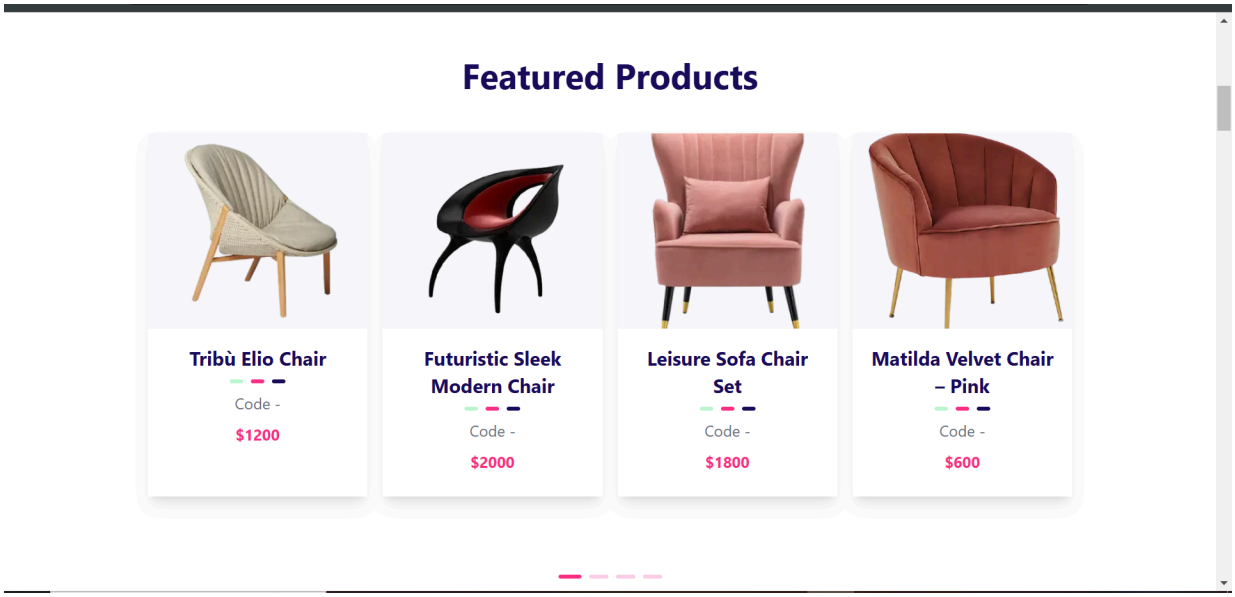
   Use a tool like ThunderClient to test the REST API endpoint and fetch data.

   In my Next.js project, I have created a script (**migrate.ts**) that fetches the data from the provided API .This script will fetch data from the external API and prepare it for import into Sanity.



3. **Displaying Data in Next.js Using a Custom GROQ Query:-**

   Created a GROQ query in code to fetch data from Sanity. The query filters for products where `_type` is `"product"`. It also selects specific fields such as `name`, `description`, `price`, and more, and resolves image paths using `image.asset->url`.



Conclusion:-

On Day 3, I successfully focused on refining and improving the integration of Sanity CMS with my Next.js application. I set up Sanity in my project, configured the environment variables for secure API communication, and migrated data from an external API to the Sanity database.

After ensuring the data was imported correctly, I utilized GROQ queries to fetch and display the data dynamically in my Next.js application. Additionally, I made adjustments to old schemas to better align with the updated data structure, enhancing compatibility and efficiency. This progress has streamlined the workflow and set a strong foundation for the upcoming steps in the project.