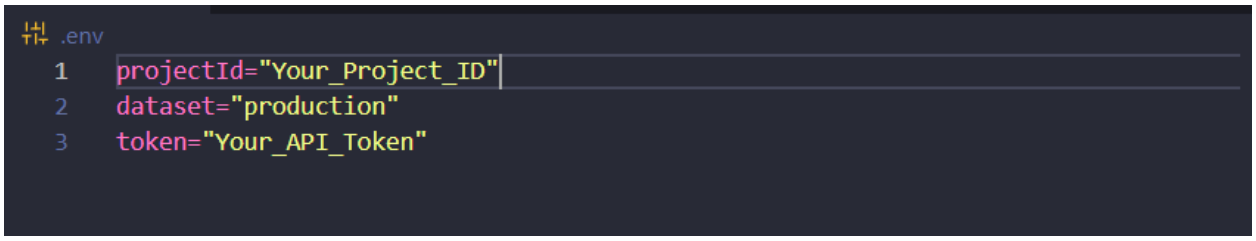
Day3-API Integration Report-Trend Nest

API Integration process:

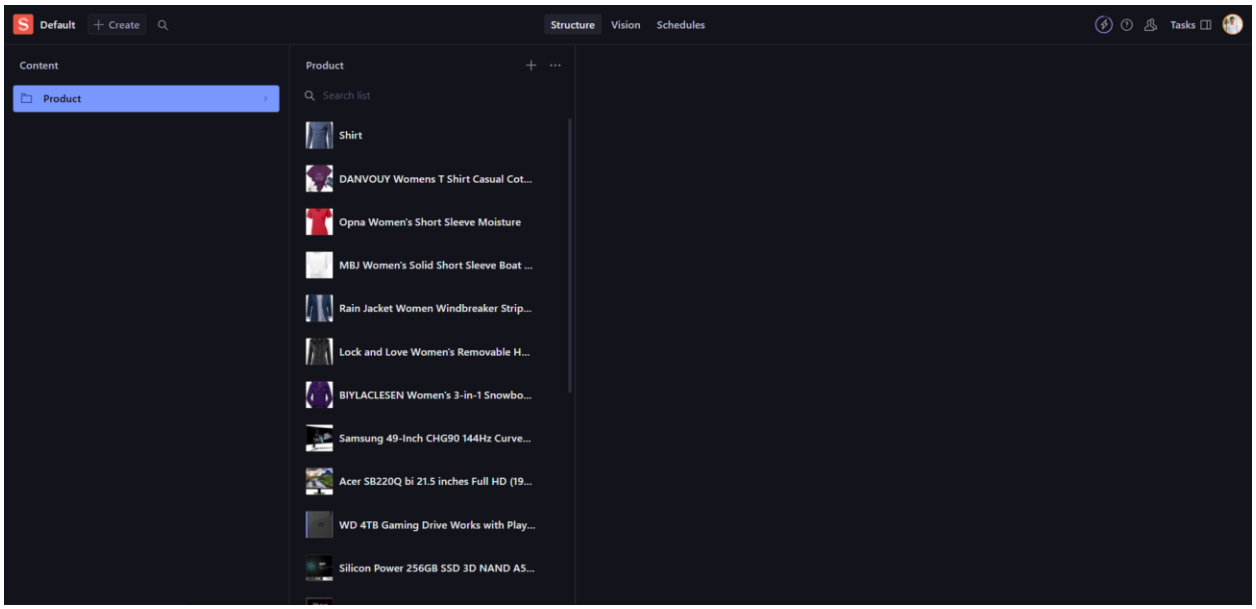
In this process of integrating API in to migrate to Sanity.io following steps were followed:

Migrating Data to Sanity.io

1. Firstly, created a clone of Sir Ali Jawad's sanity migration repo.
   GitHub - jawwad-ali/sanity-migration
2. Now run the command (npm i) to install needed node-modules.
3. Now create a project in Sanity.io.
4. Generate an API token.
5. Create a ".env" file in the root directory and set environment variables like shown in the picture below.



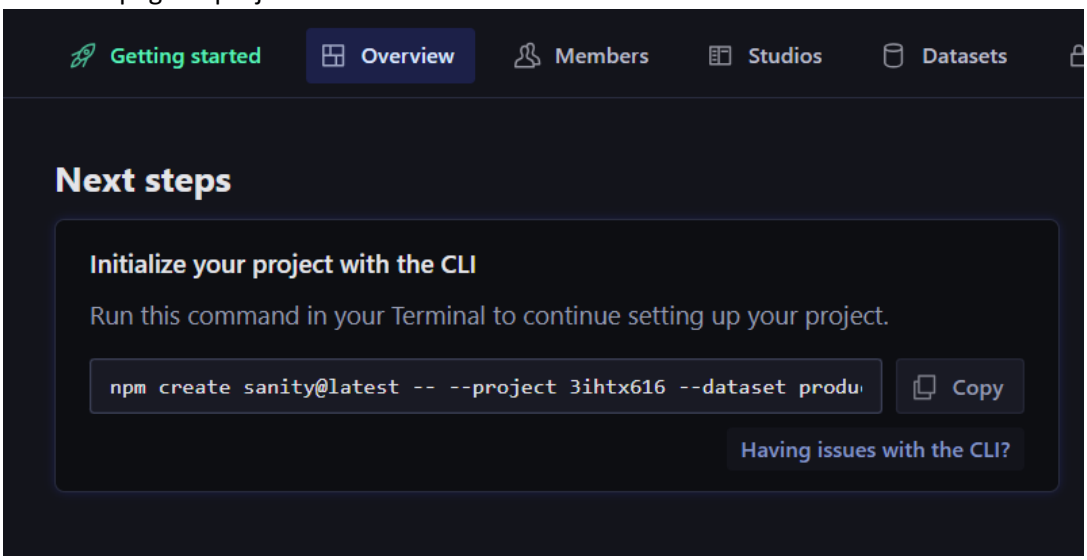
6. Now run the command **tsc && node importData.js** to migrate data from API to Sanity.
7. Now the data should be in Sanity.io project.



Adjusting Schemas

Note: I am using a previous UI from the GIAIC UI/UX Hackathon so that more time can be spent to build functionality. How-ever I have made some changes in the UI according to project goals.

To make and adjust schemas we link our sanity to our Next.js UI by running the command on overview page of project.



Now make a file of product.ts in the schema folder and make schemas according to your data.

src > sanity > schemaTypes > product.ts > product > fields

```ts
import { Rule } from "sanity";


const product = {
  name: "product",
  title: "Product",
  type: "document",
  fields: [
    {
      name: "name",
      title: "Name",
      validation: (rule: Rule) => rule.required(),
      type: "string",
    },
    {
      name: "slug",
      title: "Slug",
      validation: (rule: Rule) => rule.required(),
      type: "slug",
      options: {
        source: "name",
        maxLength: 96,
      },
    },
    {
      name: "price",
      title: "Price",
      validation: (rule: Rule) => rule.required(),
      type: "number",
    },
    {
      name: "image",
      title: "Image",
      validation: (rule: Rule) => rule.required(),
      type: "image",
      options: {
        hotspot: true,
      },
    },
    {
      name: "description",
      title: "Description",
      validation: (rule: Rule) => rule.required(),
      type: "text",
    },
    {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }],
          },
    {
      name: "isNew",
      type: "boolean",
      title: "New Badge",
    },
  ],
};

export default product;

```

Product

New Product

Name

Slug

Generate

Price

Image

Drag or paste image here

Upload

Select

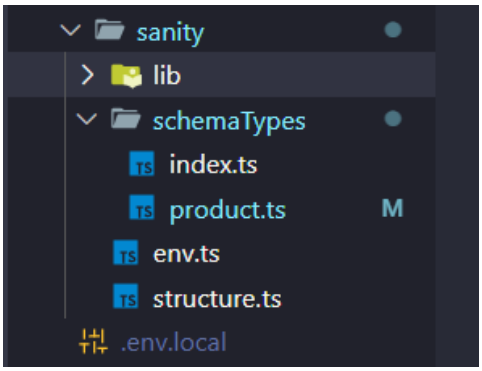
Description

Tags

No items

Add item

New Badge



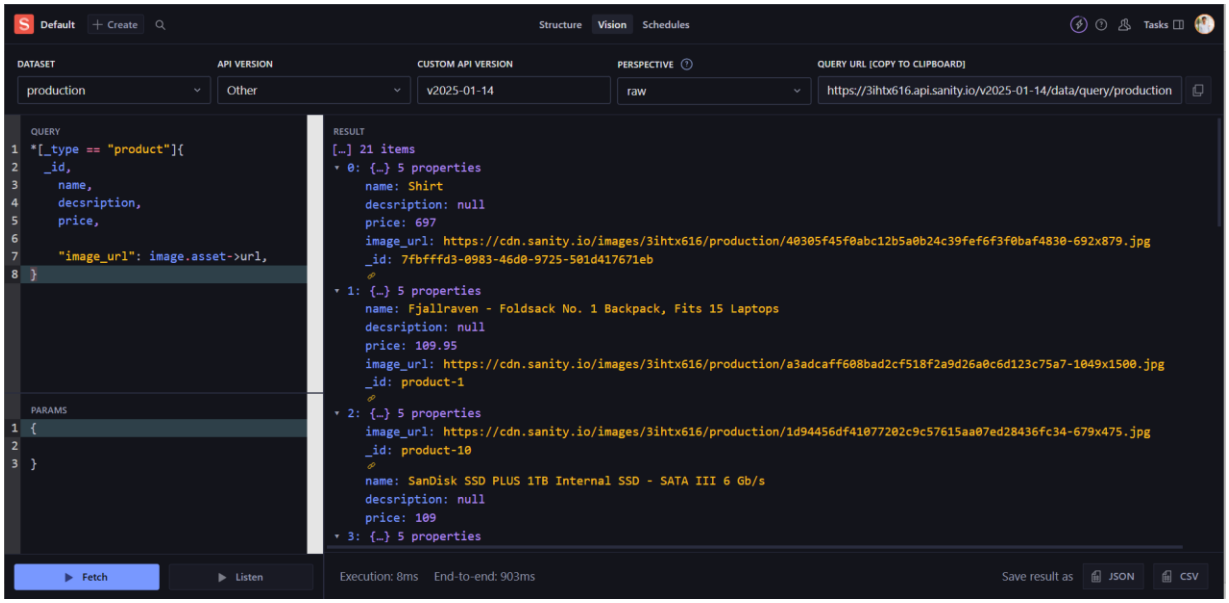
After adjusting the schemas according to requirements we are ready to move ahead and display the data on frontend.

Displaying Content to Frontend

I am using Sanity.io's GROQ to fetch data to my frontend.

Steps:

1. Navigate to vision in sanity.
2. According to my listing requirement I am using this GROQ query:



3. In my code I have used async function to fetch data from sanity.

```
const getFtProducts = async () => {
  const product = await client.fetch(`
    *[_type == "product"]{
  _id,
    name,
    decsription,
    price,

    "image_url": image.asset->url,
}
    `);
  return product;
};
```

4. Now using useEffect and useState to handle and update the UI with the API fetched data as shown in the code below.

```
const [products, setProducts] = useState([]);


 useEffect(() => {
   const fetchProducts = async () => {
     const productsData = await getFtProducts();
     setProducts(productsData);
   };

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

5. I have used array map method for displaying data on the UI.

```
{products.map(
          (product: {
            id: string;
            name: string;
            price: number;
            image_url: string;
          }) => (
            <div className="w-[241px] h-[639px] " key={product.id}>
              <Image
                className="h-80 "
                alt={product.name}
                src={product.image_url}
                width={239}
                height={420}
              />
              <div className="w-[239px] h-[188px] flex flex-col pt-1 items-center gap-2 ">
                <h1 className="font-bold text-[16px] text-[#252B42] leading-[24px]">
                  {truncateText(product.name, 20)}
                </h1>
                <p className="text-[14px] leading-[24px] font-bold text-[#737373]">
                  English Department
                </p>
                <div className="flex gap-1 items-center">
                  <p className="font-bold text-[16px] text-[#23856D] ">
                    {product.price}
                  </p>
                </div>
                <div className="py-3 flex gap-2">
                  <div className="rounded-full w-[16px] h-[16px] bg-[#23A6F0] "></div>
                  <div className="rounded-full w-[16px] h-[16px] bg-[#23856D]"></div>
                  <div className="rounded-full w-[16px] h-[16px] bg-[#E77C40]"></div>
                  <div className="rounded-full w-[16px] h-[16px] bg-[#252B42]"></div>
                </div>
              </div>
            </div>
          )
        )}
```

Conclusion:

The conclusion of this process turns out to be that our UI now updates according to the backend (Sanity.io). The products are now displayed dynamically and will update when the backend (Sanity Project) updates.

Shirt

English Department

697

Fjallraven - Foldsac...

English Department

109.95

SanDisk SSD PLUS 1TB...

English Department

109

Silicon Power 256GB ...

English Department

109

WD 4TB Gaming Drive ...

English Department

Acer SB220Q bi 21.5 ...

English Department

Samsung 49-Inch CHG9...

English Department

BIYLACLESEN Women's ...

English Department