

DATA MIGRATION AND API INTEGRATION

Clone the migration file provide in Day 3 Hackathon Document according to your template

Migration file Link

<https://github.com/developer-hammad-rehman/template1.git>

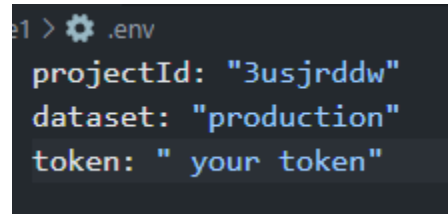
clone the repository? (run bellow command)

git clone <https://github.com/developer-hammad-rehman/template1.git>

command: npm install

add .env file in your migration file

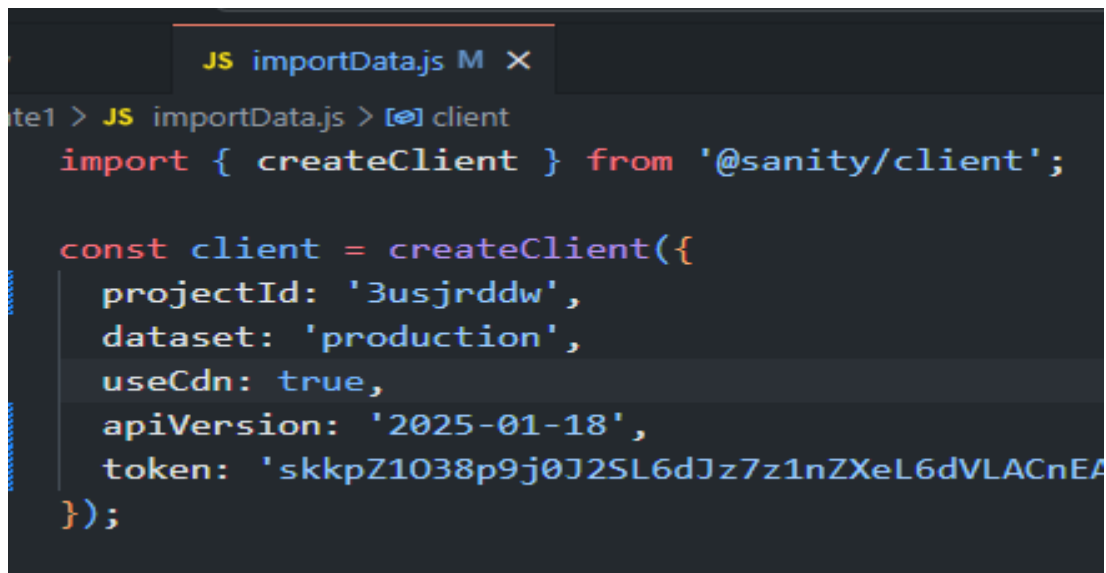
inside .env file add projectId, dataset, and token.



```
.env
projectId: "3usjrddw"
dataset: "production"
token: "your token"
```

You have to generate token from the project of sanity which you have used in your website project.

after that, change the project id and token in importData.js file.



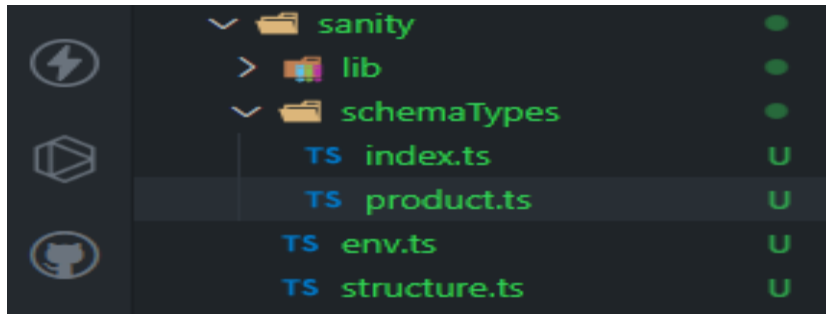
```
JS importData.js M X
te1 > JS importData.js > client
import { createClient } from '@sanity/client';

const client = createClient({
  projectId: '3usjrddw',
  dataset: 'production',
  useCdn: true,
  apiVersion: '2025-01-18',
  token: 'skkpZ1038p9j0J2SL6dJz7z1nZXeL6dVLACnEA',
});
```

Now your migration api is connected with your main project file let's move to the main project.

Open sanity schemaTypes create file for your schema

Example: product.ts

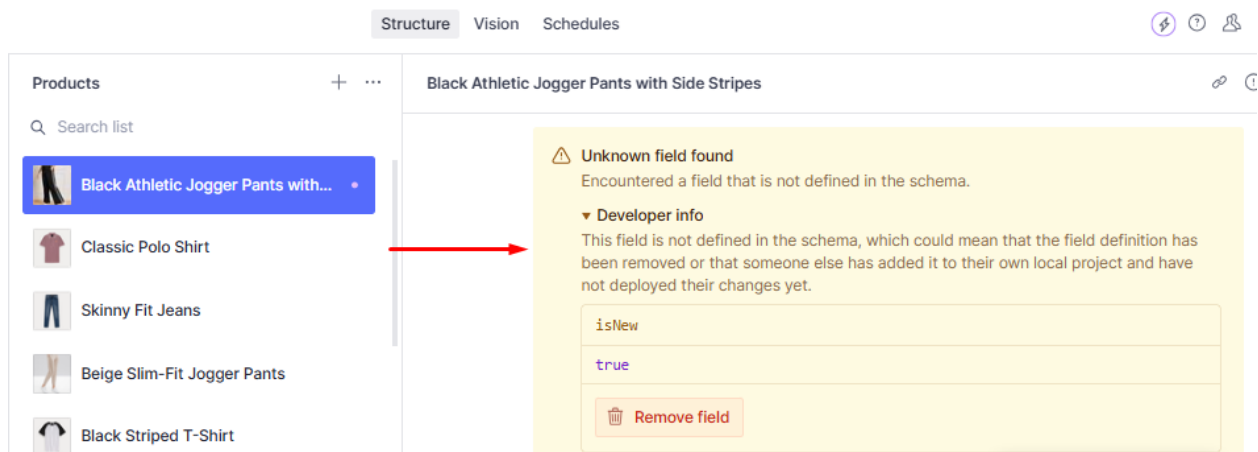


Create or paste provided schema in product.ts file.

I have added the validation in schema.



I have got one error in the sanity studio



That I have solved by adding this code in schema.

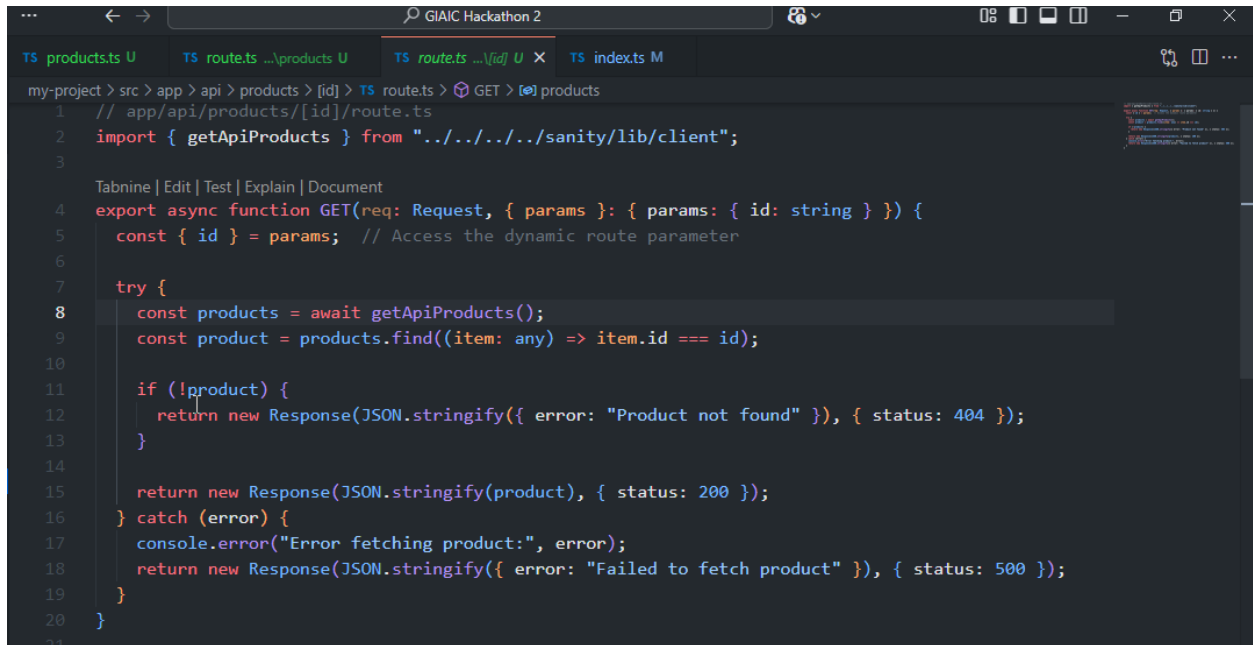
```
{
  name: 'isNew',
  title: 'Is New',
  type: 'boolean',
  validation: (Rule: ValidationRule) =>
    Rule.required().error('Please specify whether the product is new.'),
},
```

After solving error it shows like this.

Product Page api route

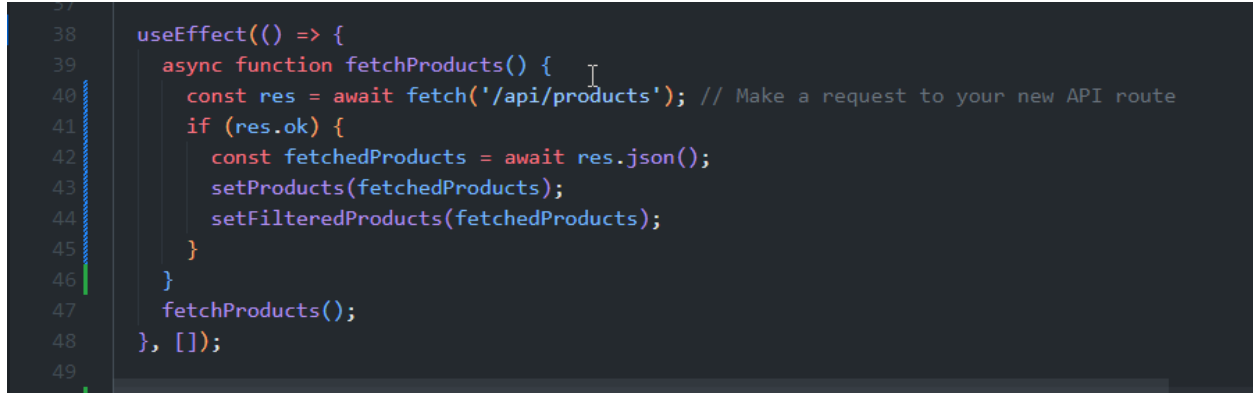
```
...  ← →  GIAIC Hackathon 2  [Icons]  -  X
TS products.ts U  TS routes.ts U X  TS index.ts M  [Icons]  ...
my-project > src > app > api > products > TS routes.ts > ...
1  import { NextResponse } from 'next/server';
2  import { getApiProducts } from '../sanity/lib/client'; // Import function to fetch products
3
Tabnine | Edit | Test | Explain | Document
4  export async function GET() {
5    try {
6      const productsDetail = await getApiProducts(); // Fetch the products from sanity client
7      return NextResponse.json(productsDetail);
8    } catch (error) {
9      return NextResponse.error(); // Handle error if any
10   }
11 }
12
```

DYNAMIC API ROUTE PRODUCT DETAIL PAGE



```
1 // app/api/products/[id]/route.ts
2 import { getApiProducts } from "../../../../../sanity/lib/client";
3
4 export async function GET(req: Request, { params }: { params: { id: string } }) {
5   const { id } = params; // Access the dynamic route parameter
6
7   try {
8     const products = await getApiProducts();
9     const product = products.find((item: any) => item.id === id);
10
11     if (!product) {
12       return new Response(JSON.stringify({ error: "Product not found" }), { status: 404 });
13     }
14
15     return new Response(JSON.stringify(product), { status: 200 });
16   } catch (error) {
17     console.error("Error fetching product:", error);
18     return new Response(JSON.stringify({ error: "Failed to fetch product" }), { status: 500 });
19   }
20 }
```

Fetch api to show product on frontend



```
38 useEffect(() => {
39   async function fetchProducts() {
40     const res = await fetch('/api/products'); // Make a request to your new API route
41     if (res.ok) {
42       const fetchedProducts = await res.json();
43       setProducts(fetchedProducts);
44       setFilteredProducts(fetchedProducts);
45     }
46   }
47   fetchProducts();
48 }, []);
49
```

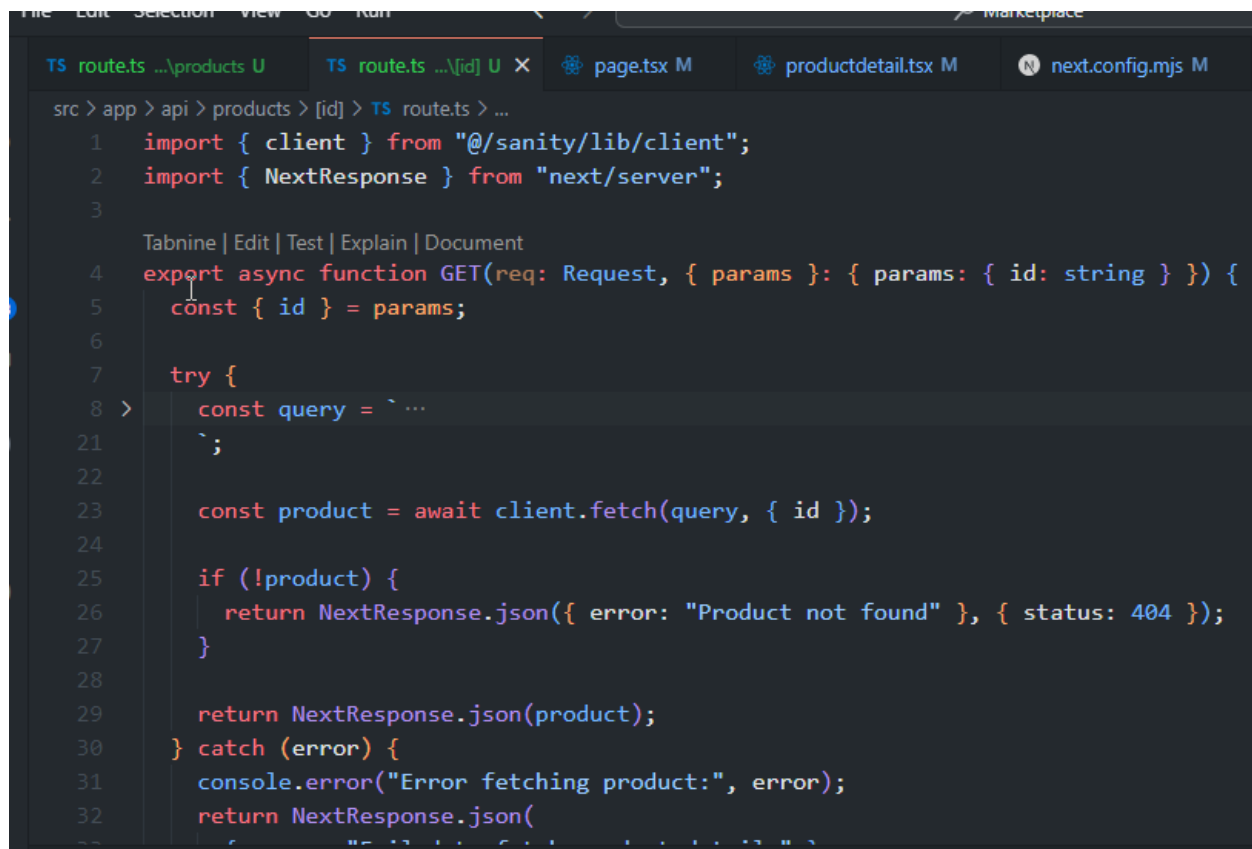
Product page.

```
// Fetch data from API
useEffect(() => {
  const fetchProducts = async () => {
    try {
      const response = await fetch("/api/products"); // Adjust the
API endpoint as per configuration
      const data = await response.json();
      setProducts(data);
      setFilteredProducts(data);
      // Set price range dynamically based on the fetched products
      const prices = data.map((product: any) => product.price);
      const minPrice = Math.min(...prices);
      const maxPrice = Math.max(...prices);
      setPriceRange([minPrice, maxPrice]);
      setSelectedPrice(maxPrice);
    } catch (error) {
      console.error("Error fetching products:", error);
    }
  };

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

Then, create a product/[id]/ route.ts to get data into api from sanity through GROQ Query and then pass the data into frontend product detail page.

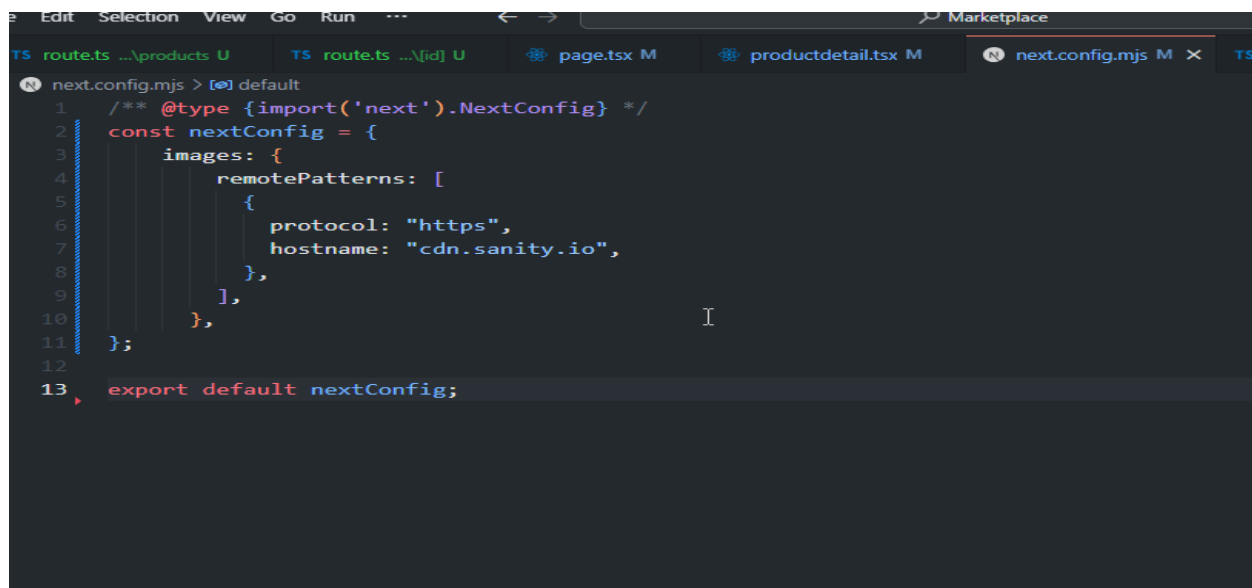
Product detail page api route.



The screenshot shows a VS Code editor window with the file explorer on the left and the editor on the right. The editor is open to the file `route.ts` in the `src > app > api > products > [id]` directory. The code defines an asynchronous GET function that takes a request and parameters, fetches a product from Sanity, and returns a NextResponse. If the product is not found, it returns a 404 status. If there is an error, it logs the error and returns a 500 status.

```
1 import { client } from "@sanity/lib/client";
2 import { NextResponse } from "next/server";
3
4 export async function GET(req: Request, { params }: { params: { id: string } }) {
5   const { id } = params;
6
7   try {
8     const query = `
21   `;
22
23     const product = await client.fetch(query, { id });
24
25     if (!product) {
26       return NextResponse.json({ error: "Product not found" }, { status: 404 });
27     }
28
29     return NextResponse.json(product);
30   } catch (error) {
31     console.error("Error fetching product:", error);
32     return NextResponse.json(
33       { error: "Error fetching product" }, { status: 500 }
34     );
35   }
36 }
```

Changes in `next.config.mjs` file for Image error



The screenshot shows a VS Code editor window with the file explorer on the left and the editor on the right. The editor is open to the file `next.config.mjs`. The code defines a `nextConfig` object with an `images` property. The `remotePatterns` array is updated to include a new pattern for the Sanity CDN.

```
1 /** @type {import('next').NextConfig} */
2 const nextConfig = {
3   images: {
4     remotePatterns: [
5       {
6         protocol: "https",
7         hostname: "cdn.sanity.io",
8       },
9     ],
10  },
11 };
12
13 export default nextConfig;
```