

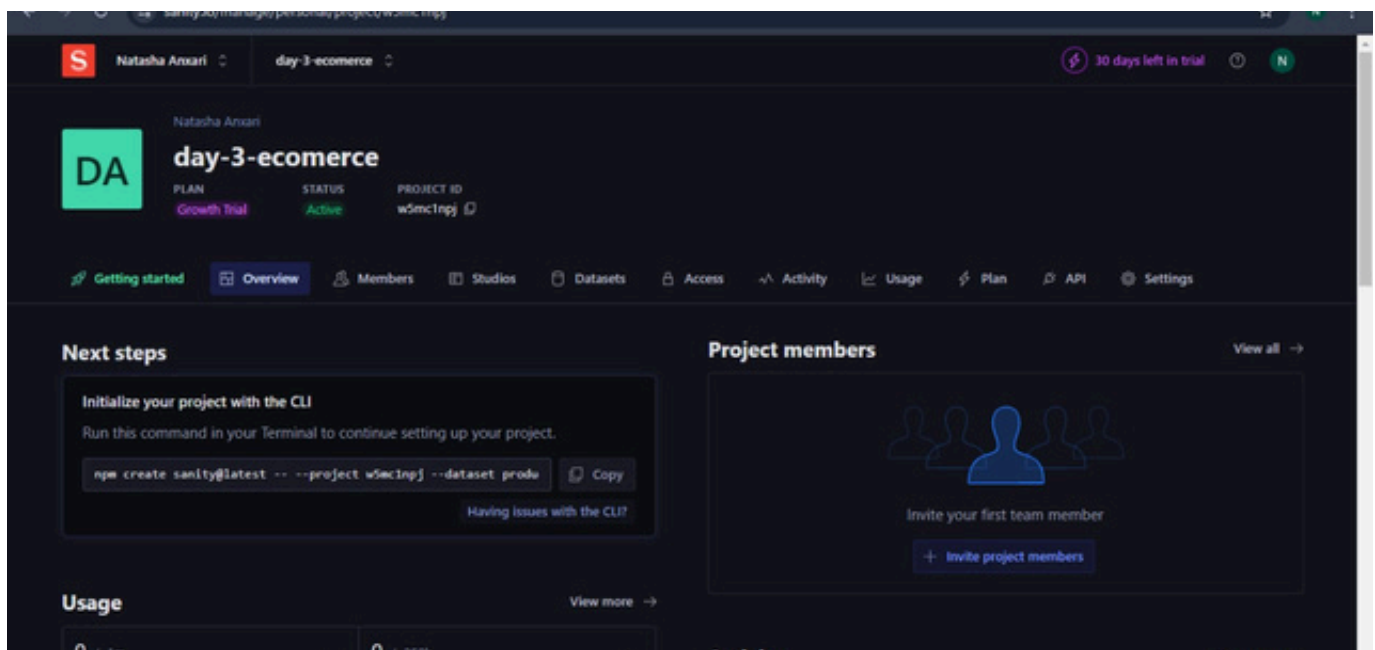
Day 3: API Integration and Data Migration

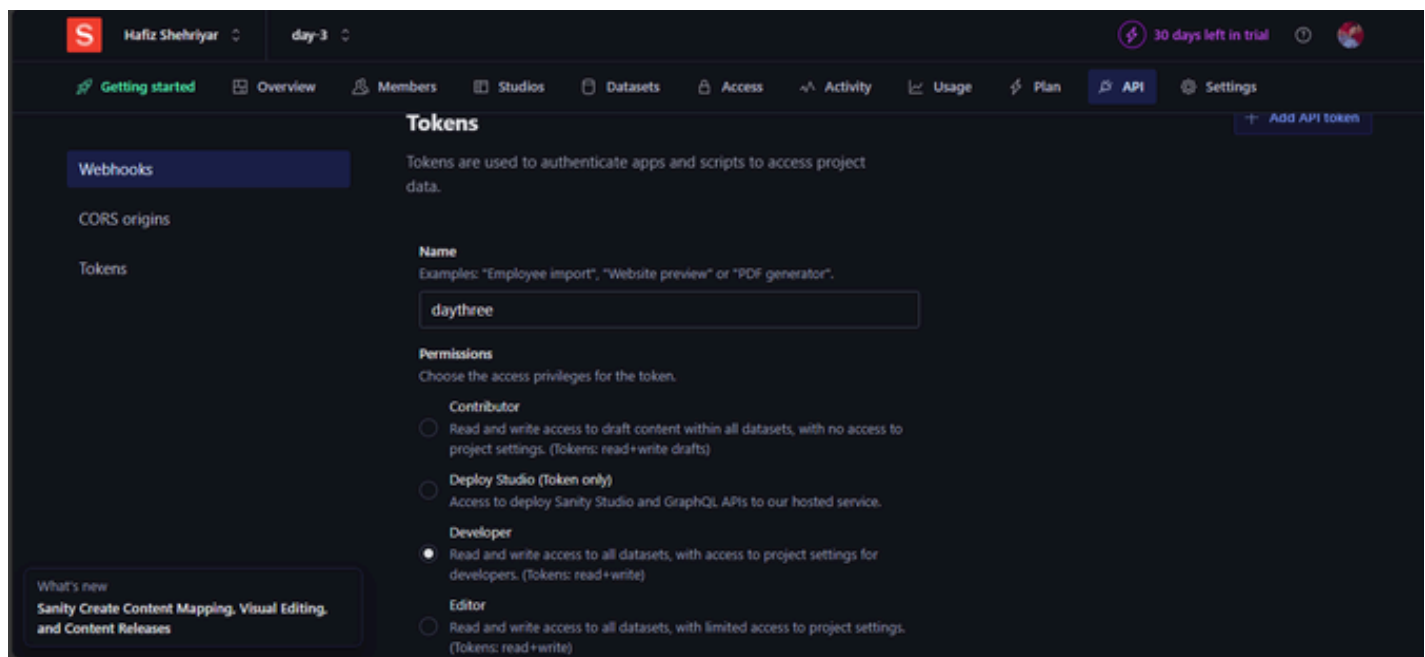
1. Created a project in Sanity:

A new project was created in the Sanity platform.

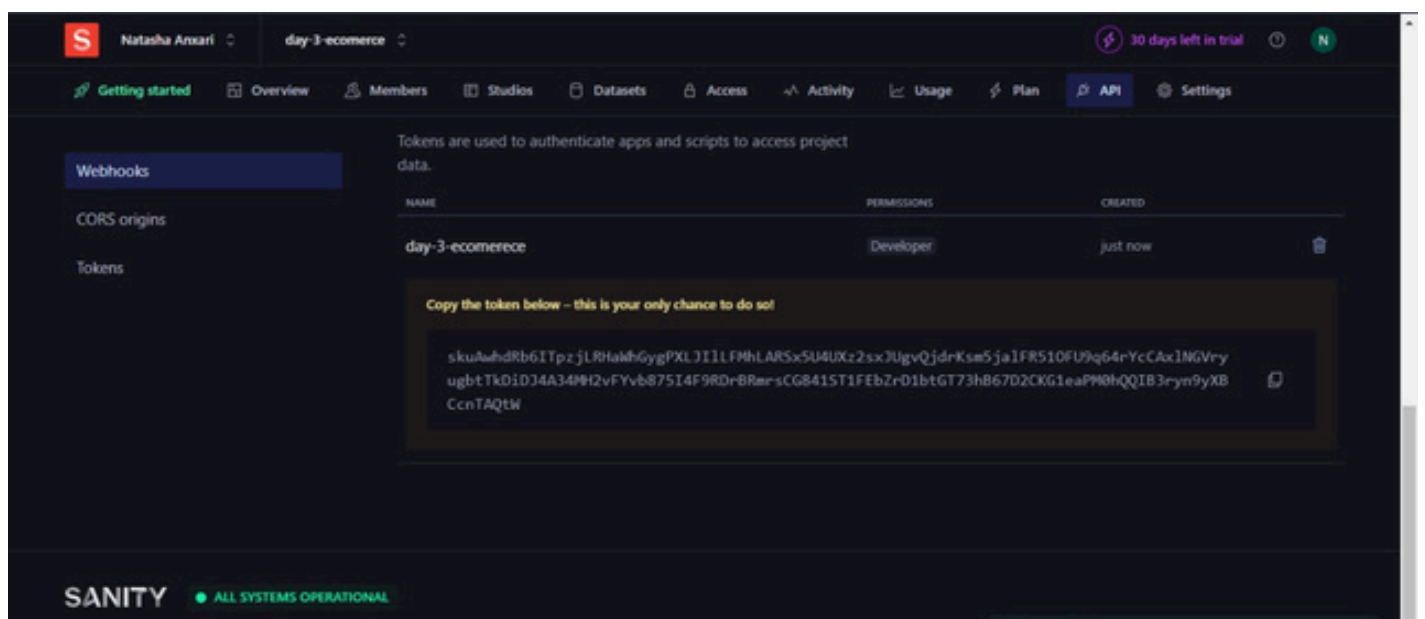
2) Created a project named 'Day3' in Sanity

A separate project, titled 'Day3', was created for better organization.



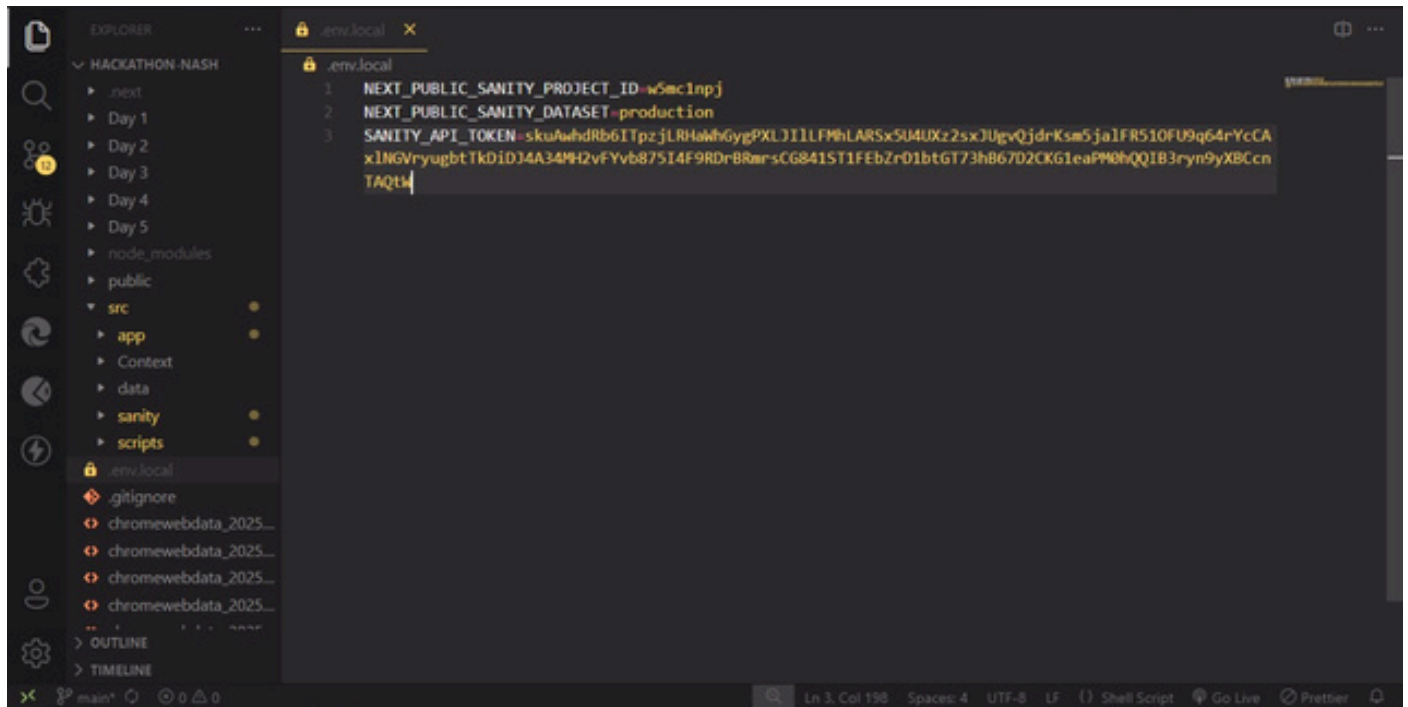


1. Generated API Token and stored it in Developer Options
2. An API token was generated and securely placed in the developer section.



1. Added environment file for all three steps

An environment file was added to manage configurations for all tasks.



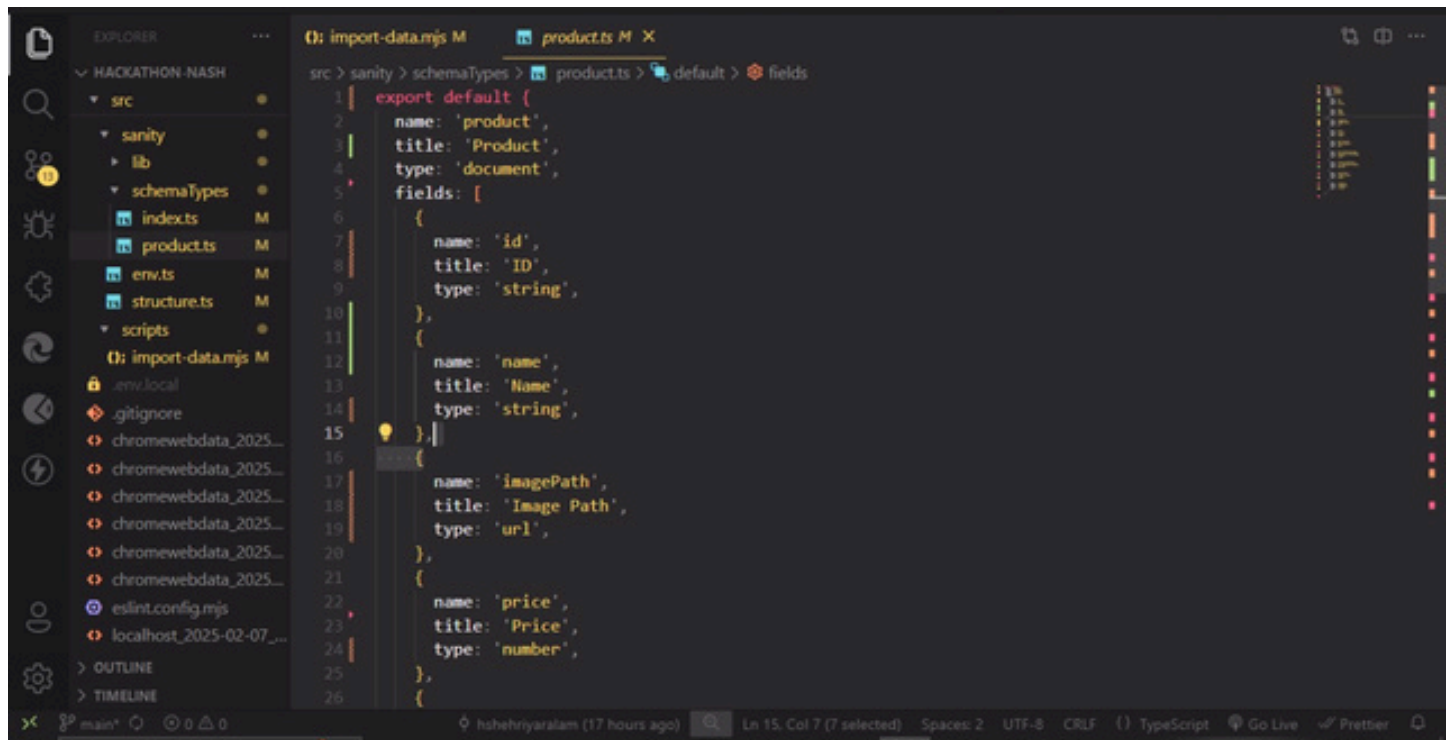
The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'HACKATHON-NASH'. The file structure includes folders for 'next', 'Day 1' through 'Day 5', 'node_modules', 'public', 'src' (containing 'app', 'Context', 'data', 'sanity', and 'scripts'), and files like '.env.local', '.gitignore', and several 'chromewebdata_2025...' files. The main editor window is open to the '.env.local' file, which contains the following configuration:

```
1 NEXT_PUBLIC_SANITY_PROJECT_ID=w5mc1npj
2 NEXT_PUBLIC_SANITY_DATASET=production
3 SANITY_API_TOKEN=skuAwhdRb6ITpzjLRHawhGygPXLJIIlFMhLARSx5U4UXz2sx0UgvQjdrKsm5ja1FR510FU9q64rYcCA
  x1NGVryugbtTkDiD34A34H42vFYvb875I4F9RDr8RmrCG841ST1FEbZr01btGT73hB6702CKG1eaPM0hQQI83ryn9yXBCcn
  TAQtlw
```

The status bar at the bottom indicates the current file is 'main', the cursor is at 'Ln 3, Col 198', and the file is encoded in 'UTF-8' with 'LF' line endings.

1. Created Product schema file and added schema

A schema file for products was created and the schema structure was added.

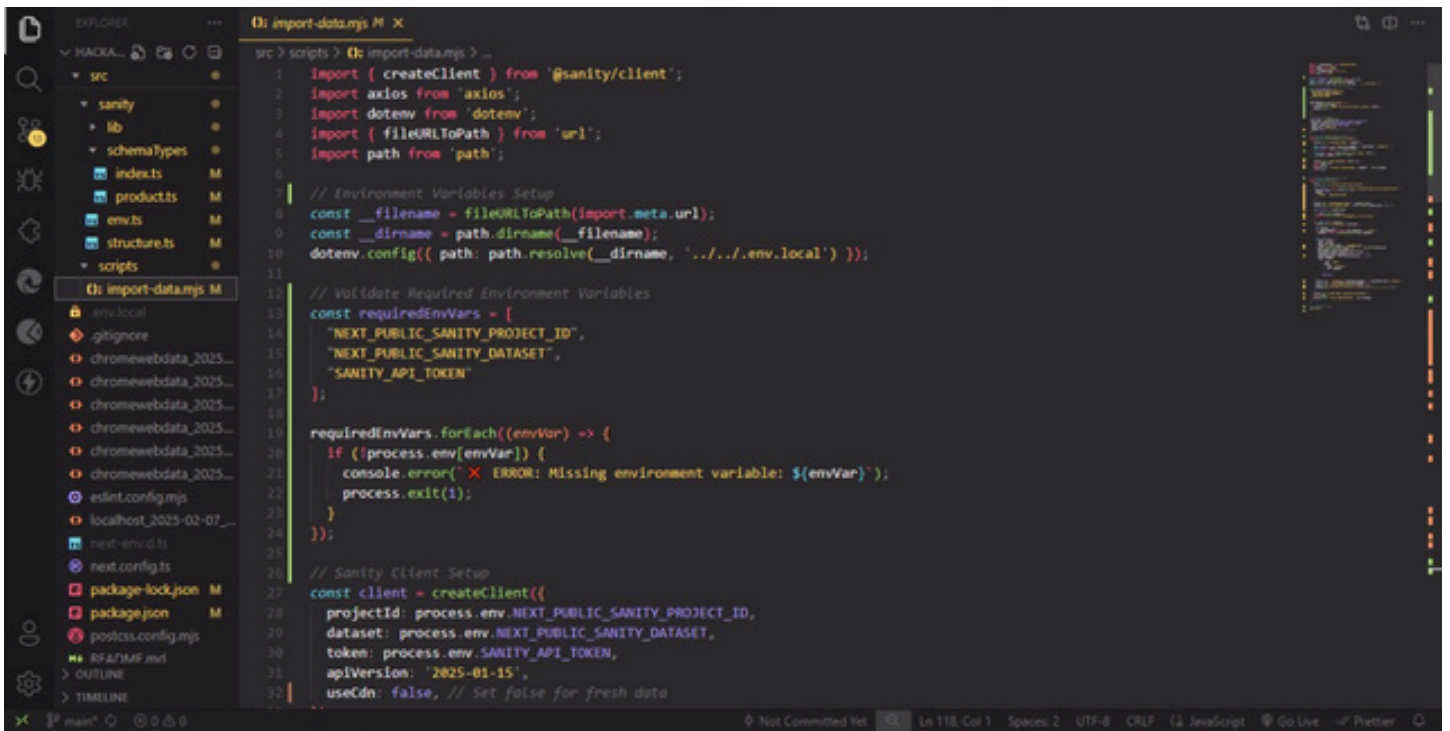


The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left displays a file tree for a project named 'HACKATHON-NASH'. The tree includes a 'src' folder containing 'sanity', 'lib', 'schemaTypes', 'indexes', 'product.ts', 'env.ts', 'structure.ts', and 'scripts'. The 'product.ts' file is selected and open in the main editor. The breadcrumb navigation at the top of the editor reads: 'src > sanity > schemaTypes > product.ts > default > fields'. The code in the editor is a TypeScript file defining a Sanity schema. It starts with 'export default {' followed by 'name: 'product'', 'title: 'Product'', and 'type: 'document''. The 'fields' array contains three objects: 1. 'id' field: 'name: 'id'', 'title: 'ID'', 'type: 'string''. 2. 'name' field: 'name: 'name'', 'title: 'Name'', 'type: 'string''. 3. 'imagePath' field: 'name: 'imagePath'', 'title: 'Image Path'', 'type: 'url''. The code is partially visible, showing lines 1 through 26. The status bar at the bottom indicates 'Ln 15, Col 7 (7 selected)', 'Spaces: 2', 'UTF-8', 'CRLF', 'TypeScript', and 'Go Live' and 'Prettier' icons.

```
1 export default {
2   name: 'product',
3   title: 'Product',
4   type: 'document',
5   fields: [
6     {
7       name: 'id',
8       title: 'ID',
9       type: 'string',
10    },
11    {
12      name: 'name',
13      title: 'Name',
14      type: 'string',
15    },
16    {
17      name: 'imagePath',
18      title: 'Image Path',
19      type: 'url',
20    },
21    {
22      name: 'price',
23      title: 'Price',
24      type: 'number',
25    },
26  ],
27 }
```

1. Created a scripts folder and added import-data.mjs file

A folder for scripts was created, and an import-data.mjs file was added for data import functionality.

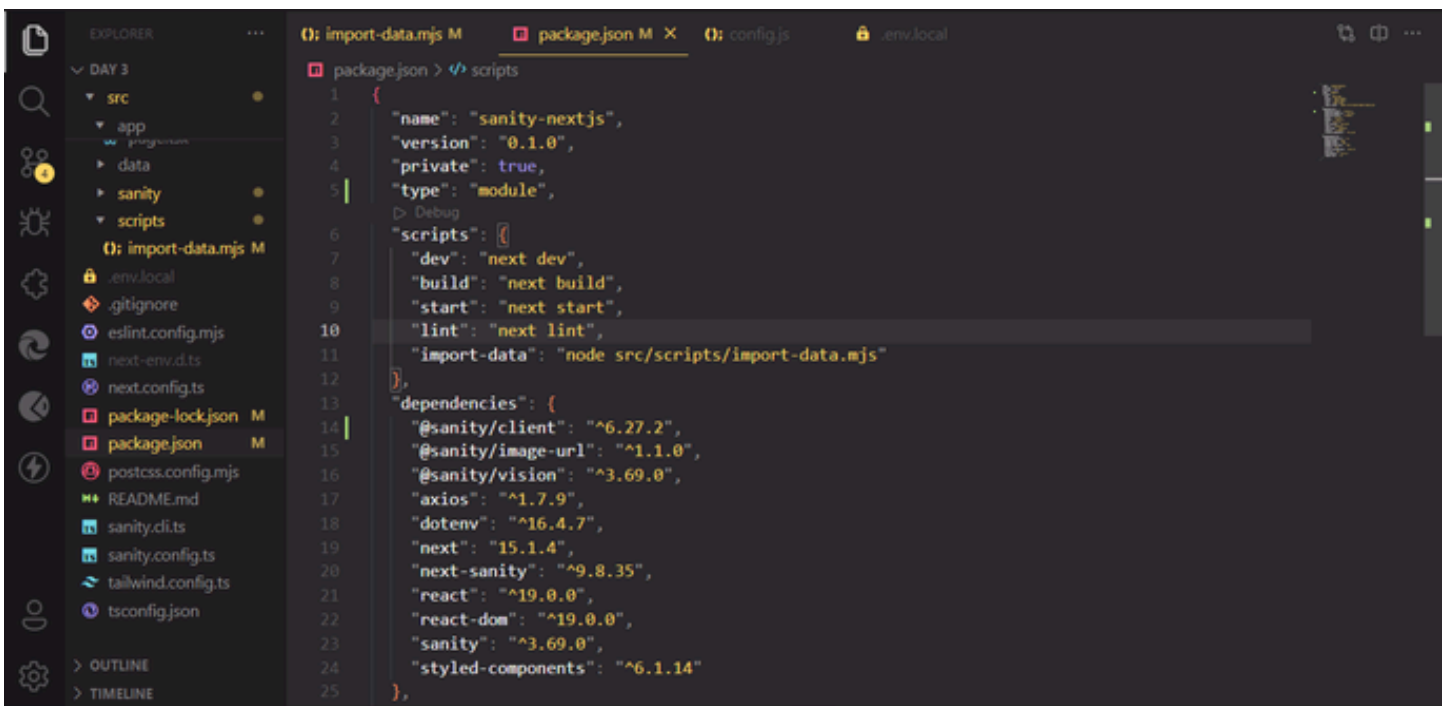


The screenshot shows a VS Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with a 'src' directory containing 'sanity', 'lib', 'schemasTypes', 'index.ts', 'products.ts', 'env.ts', 'structure.ts', and 'scripts'. The 'scripts' directory is expanded, showing 'import-data.mjs'. The code editor displays the content of 'import-data.mjs', which is a JavaScript script for setting up a Sanity client and validating environment variables. The script includes imports for 'createClient' from '@sanity/client', 'axios' from 'axios', 'dotenv' from 'dotenv', 'fileURLToPath' from 'url', and 'path' from 'path'. It sets up environment variables, validates required environment variables (NEXT_PUBLIC_SANITY_PROJECT_ID, NEXT_PUBLIC_SANITY_DATASET, SANITY_API_TOKEN), and creates a Sanity client with the specified project ID, dataset, and token. The script also sets the API version to '2025-01-15' and sets 'useCdn' to false.

```
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Environment Variables Setup
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Validate Required Environment Variables
13 const requiredEnvVars = [
14   'NEXT_PUBLIC_SANITY_PROJECT_ID',
15   'NEXT_PUBLIC_SANITY_DATASET',
16   'SANITY_API_TOKEN'
17 ];
18
19 requiredEnvVars.forEach((envVar) => {
20   if (!process.env[envVar]) {
21     console.error(` ❌ ERROR: Missing environment variable: ${envVar}`);
22     process.exit(1);
23   }
24 });
25
26 // Sanity Client Setup
27 const client = createClient({
28   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
29   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
30   token: process.env.SANITY_API_TOKEN,
31   apiVersion: '2025-01-15',
32   useCdn: false, // Set false for fresh data
```

1. Defined type module and added import-data script in package-lock.json

The package-lock.json file was updated to define the type module and include the import-data script.

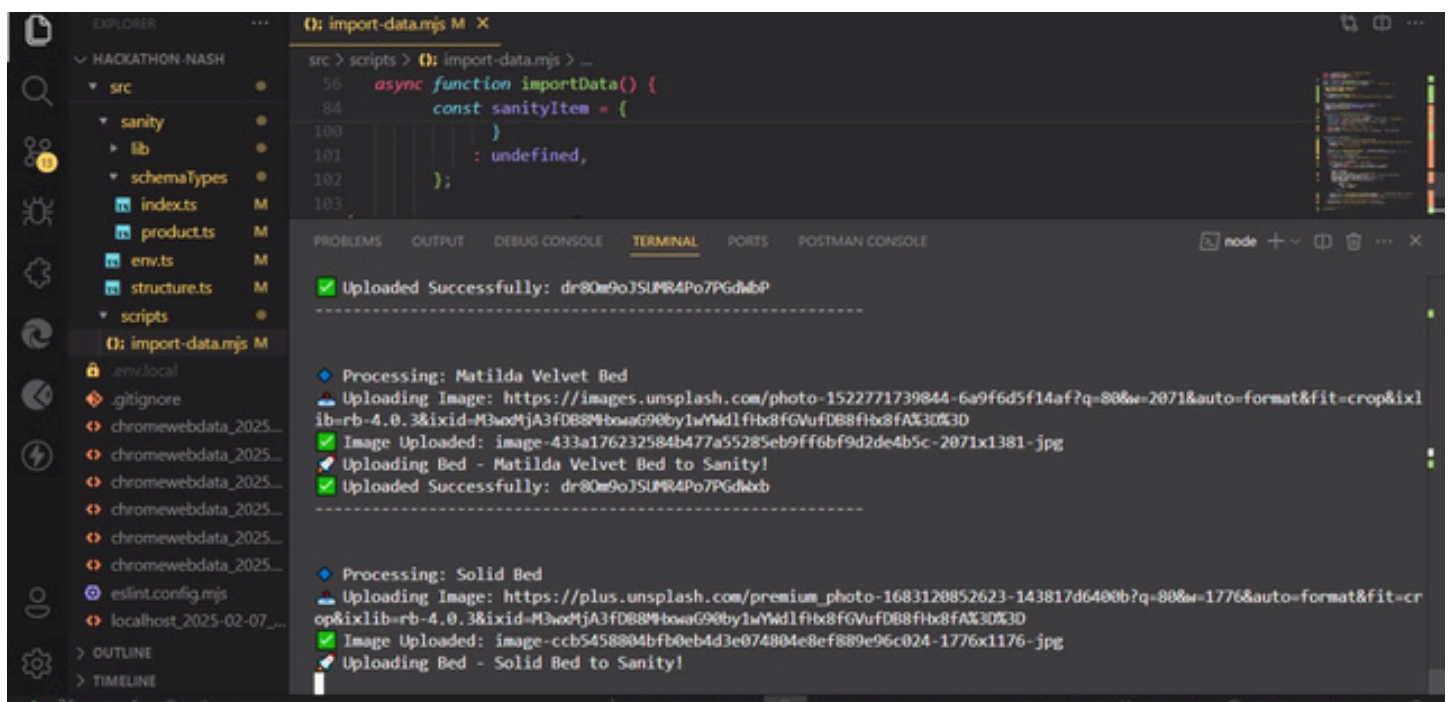


The screenshot shows a VS Code editor with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with a 'src' directory containing 'app', 'data', 'sanity', and 'scripts'. The 'scripts' directory is expanded, showing 'import-data.mjs'. The code editor displays the content of 'package.json', which is a JSON file defining the project's metadata and dependencies. The 'scripts' section includes 'dev', 'build', 'start', 'lint', and 'import-data'. The 'dependencies' section lists various dependencies, including '@sanity/client', '@sanity/image-url', '@sanity/vision', 'axios', 'dotenv', 'next', 'next-sanity', 'react', 'react-dom', 'sanity', and 'styled-components'.

```
1 {
2   "name": "sanity-nextjs",
3   "version": "0.1.0",
4   "private": true,
5   "type": "module",
6   "scripts": {
7     "dev": "next dev",
8     "build": "next build",
9     "start": "next start",
10    "lint": "next lint",
11    "import-data": "node src/scripts/import-data.mjs"
12  },
13  "dependencies": {
14    "@sanity/client": "^6.27.2",
15    "@sanity/image-url": "^1.1.0",
16    "@sanity/vision": "^3.69.0",
17    "axios": "^1.7.9",
18    "dotenv": "^16.4.7",
19    "next": "15.1.4",
20    "next-sanity": "^9.8.35",
21    "react": "^19.0.0",
22    "react-dom": "^19.0.0",
23    "sanity": "^3.69.0",
24    "styled-components": "^6.1.14"
25  },
```

1. Executed the import data command, and all data fetched successfully in terminal

The import data command was executed, and all the data was fetched and displayed in the terminal.



The screenshot shows a VS Code editor with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project named 'HACKATHON-NASH' with a 'src' directory containing 'sanity', 'lib', 'schemaTypes', 'index.ts', 'product.ts', 'env.ts', 'structure.ts', 'scripts', and 'import-data.mjs'. The terminal window shows the output of the 'import-data.mjs' script. It starts with 'Processing: Matilda Velvet Bed', followed by 'Uploading Image: https://images.unsplash.com/photo-1522771739844-6a9f6d5f14af?q=80&w=2071&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wodMjA3fD88MbowG90by1wYWdlfHx8fGVufD88fHx8fAX30X3D'. It then shows 'Image Uploaded: image-433a176232584b477a55285eb9ff6bf9d2de4b5c-2071x1381-jpg', 'Uploading Bed - Matilda Velvet Bed to Sanity!', and 'Uploaded Successfully: dr80m9oJSLMR4Po7PGdwbP'. This is followed by 'Processing: Solid Bed', 'Uploading Image: https://plus.unsplash.com/premium_photo-1683120852623-143817d6400b?q=80&w=1776&auto=format&fit=cr...', 'Image Uploaded: image-ccb5458804bfb0eb4d3e074804e8ef889e96c024-1776x1176-jpg', and 'Uploading Bed - Solid Bed to Sanity!'.

```
src > scripts > node import-data.mjs > ...
56   async function importData() {
84     const sanityItem = {
100       // ...
101       // ...
102       // ...
103     };
104   }
105 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

node + - x

✓ Uploaded Successfully: dr80m9oJSLMR4Po7PGdwbP

Processing: Matilda Velvet Bed

Uploading Image: https://images.unsplash.com/photo-1522771739844-6a9f6d5f14af?q=80&w=2071&auto=format&fit=crop&ixlib=rb-4.0.3&ixid=M3wodMjA3fD88MbowG90by1wYWdlfHx8fGVufD88fHx8fAX30X3D

✓ Image Uploaded: image-433a176232584b477a55285eb9ff6bf9d2de4b5c-2071x1381-jpg

✓ Uploading Bed - Matilda Velvet Bed to Sanity!

✓ Uploaded Successfully: dr80m9oJSLMR4Po7PGdwbP

Processing: Solid Bed

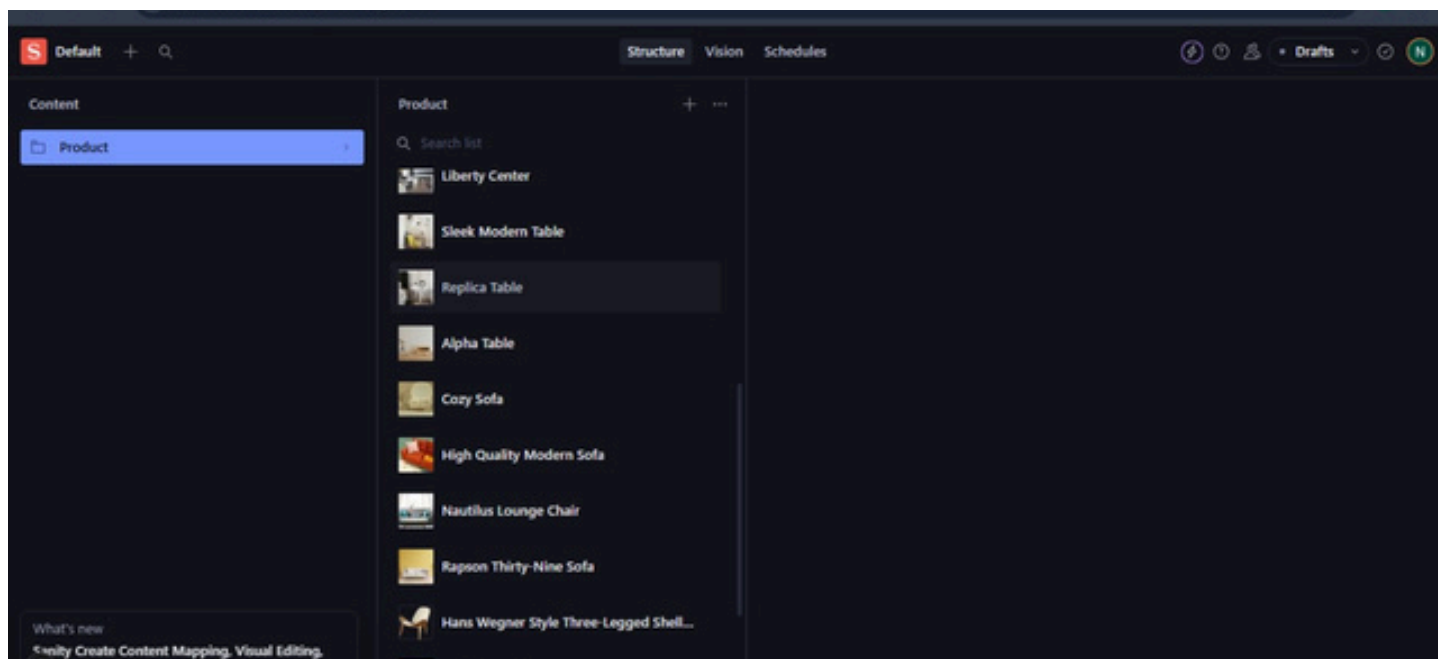
Uploading Image: https://plus.unsplash.com/premium_photo-1683120852623-143817d6400b?q=80&w=1776&auto=format&fit=cr...

✓ Image Uploaded: image-ccb5458804bfb0eb4d3e074804e8ef889e96c024-1776x1176-jpg

✓ Uploading Bed - Solid Bed to Sanity!

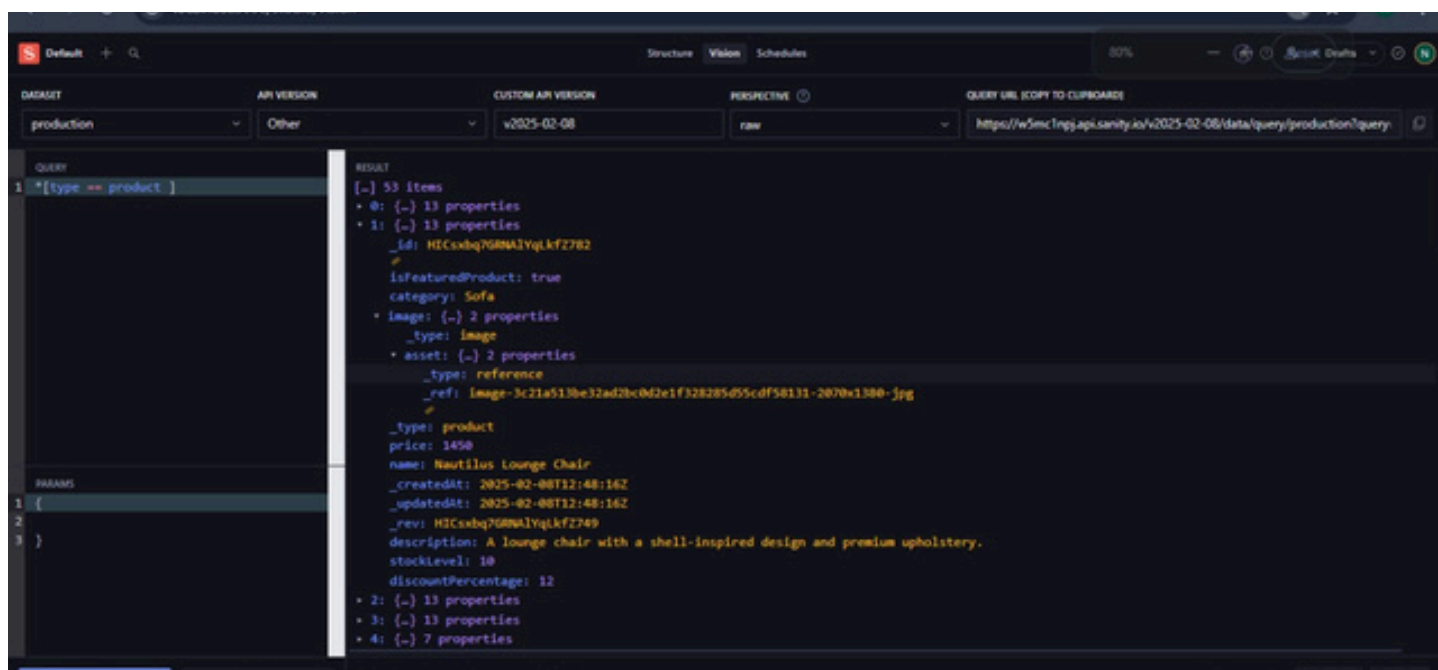
1. Verified collection in Sanity.io, confirming data presence

The data was successfully verified in the collection section of Sanity.io.



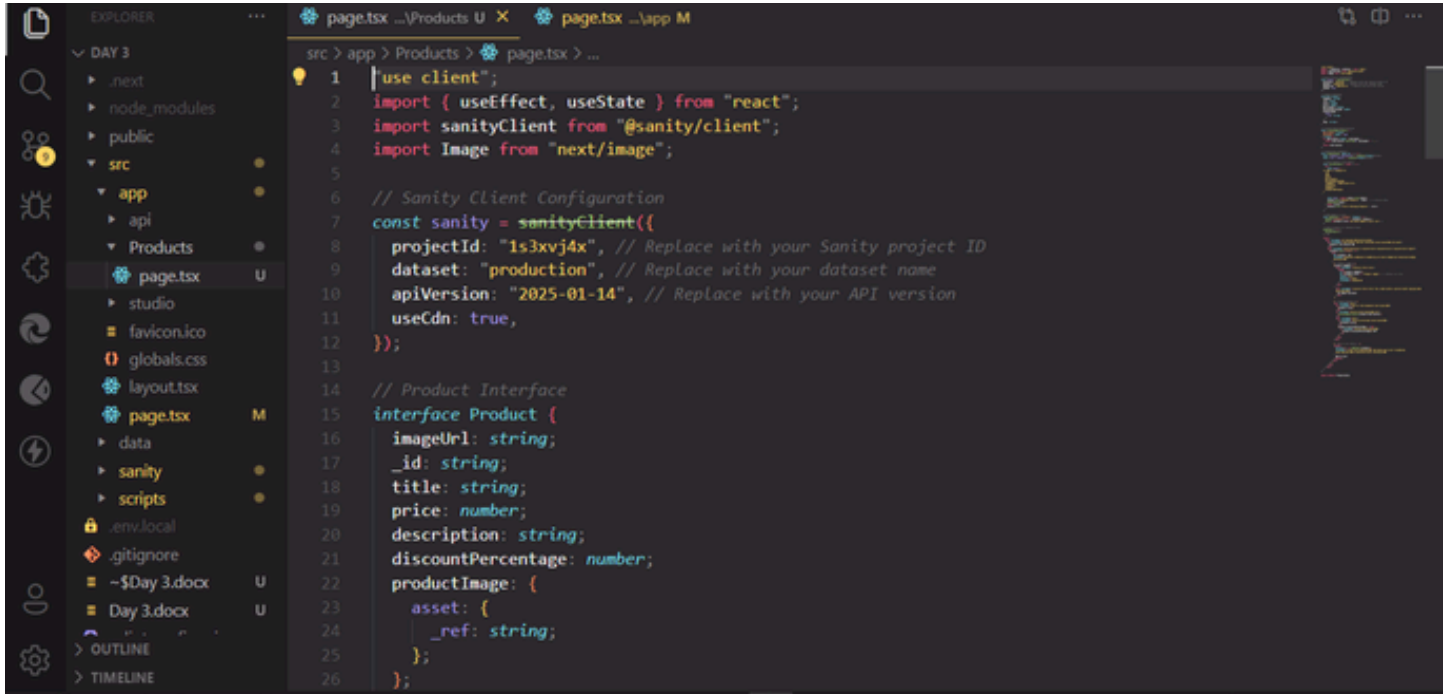
1. Created and executed a query in Sanity's Vision tool to fetch data

A query was created in Sanity's Vision tool to retrieve the data.



1. Created 'Product' folder and rendered data in the corresponding file

A 'Product' folder was created, and a page to render product data was built.

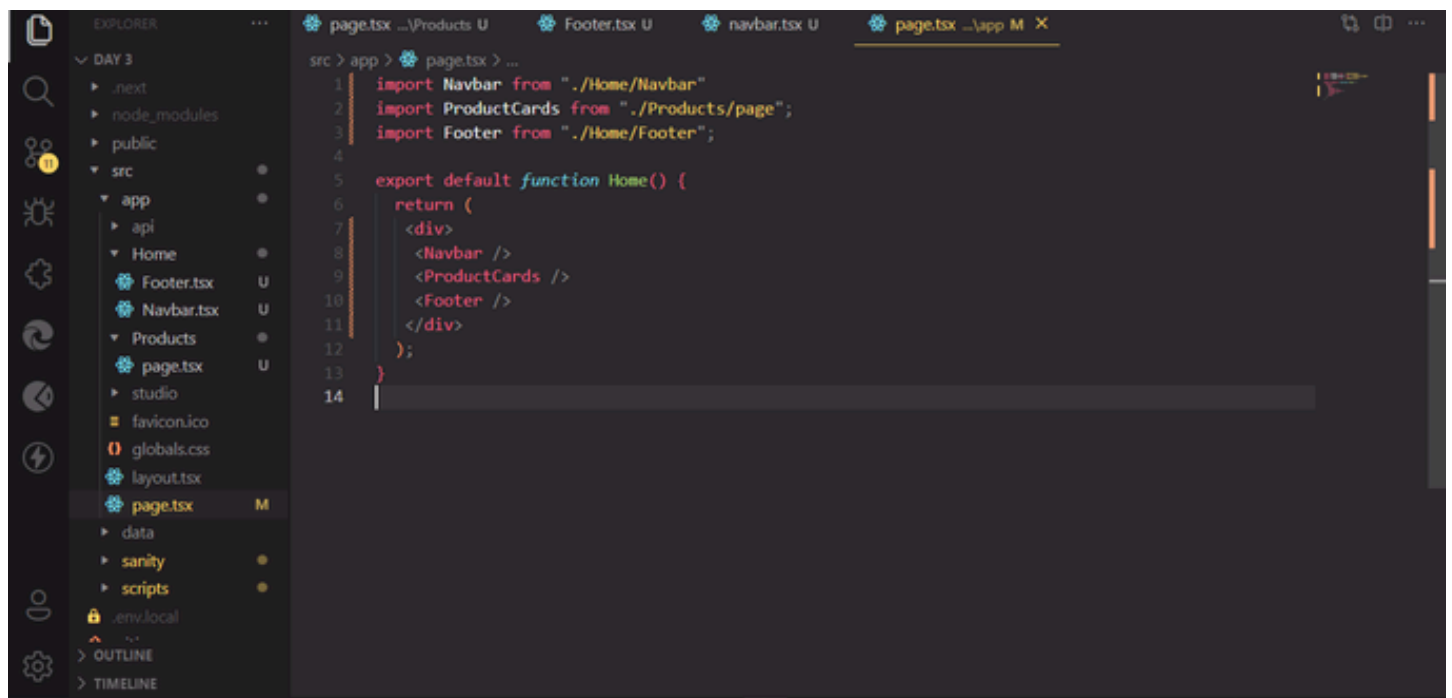


The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for a project named 'DAY 3'. The tree includes folders like '.next', 'node_modules', 'public', and 'src'. Under 'src', there is an 'app' folder containing 'api', 'Products', 'page.tsx' (marked with a 'U' icon), 'studio', 'favicon.ico', 'globals.css', 'layout.tsx', and another 'page.tsx' (marked with an 'M' icon). Below these are files like '.env.local', '.gitignore', and some Word documents. The main editor area shows the content of the selected 'page.tsx' file. The code is in TypeScript and includes imports for 'use client', 'useEffect', 'useState' from 'react', 'sanityClient' from '@sanity/client', and 'Image' from 'next/image'. It also shows the Sanity client configuration and a 'Product' interface with fields like 'imageUrl', '_id', 'title', 'price', 'description', 'discountPercentage', and 'productImage'.

```
1  |use client";
2  import { useEffect, useState } from "react";
3  import sanityClient from "@sanity/client";
4  import Image from "next/image";
5
6  // Sanity Client Configuration
7  const sanity = sanityClient({
8    projectId: "1s3xvj4x", // Replace with your Sanity project ID
9    dataset: "production", // Replace with your dataset name
10   apiVersion: "2025-01-14", // Replace with your API version
11   useCdn: true,
12 });
13
14 // Product Interface
15 interface Product {
16   imageUrl: string;
17   _id: string;
18   title: string;
19   price: number;
20   description: string;
21   discountPercentage: number;
22   productImage: {
23     asset: {
24       _ref: string;
25     };
26   };
27 }
```

1. Imported and added components to the Home page

The required components were imported and added to the Home page for display.



```
src > app > page.tsx > ...
1 import Navbar from "../Home/Navbar"
2 import ProductCards from "../Products/page";
3 import Footer from "../Home/Footer";
4
5 export default function Home() {
6   return (
7     <div>
8       <Navbar />
9       <ProductCards />
10      <Footer />
11    </div>
12  );
13 }
14
```

1. Finalized and rendered data on the browser

Finally, the data was rendered successfully on the browser.

