

Steps to Import Api Data into Sanity & Integrate Sanity with Next.js and Fetch Data

1. Open the Next.js Project

- Open the Next.js folder that was created during the UI/UX Hackathon.

2. Install Sanity

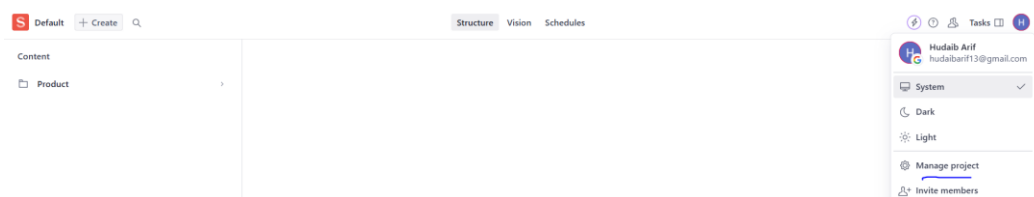
- Install Sanity in your Next.js project by running the following command:
`npm create sanity@latest -- --template clean --create-project "learning-sanity-project" --dataset production`
- Sanity automatically adds the projectId to the .env file during setup.

3. Start Sanity Studio

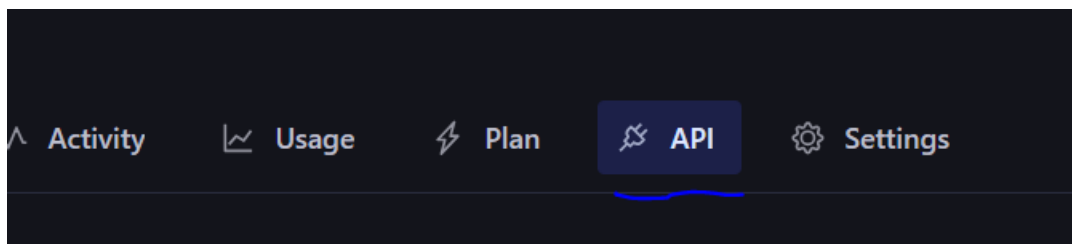
- Run the local server for the Sanity Studio:
- Navigate to the given URL (e.g., <http://localhost:3000/studio>) to access the studio.

4. Manage Project and Generate API Token

- Go to the Sanity dashboard at Manage Project.



- Click on the **API** section.



- Scroll down and click on **Add API Token**.
- Generate a token with either **Developer** or **Editor** permissions.

Name
Examples: "Employee import", "Website preview" or "PDF generator".

E-Commerce-api-token

Permissions
Choose the access privileges for the token.

Contributor

☐ Read and write access to draft content within all datasets, with no access to project settings. (Tokens: read+write drafts)

☐ **Deploy Studio (Token only)**
Access to deploy Sanity Studio and GraphQL APIs to our hosted service.

Developer

☐ Read and write access to all datasets, with access to project settings for developers. (Tokens: read+write)

Editor

☒ Read and write access to all datasets, with limited access to project settings. (Tokens: read+write)

Viewer

☐ Read access to all datasets, with limited access to project settings. (Tokens: read-only)

[Save](#) [Cancel](#)

- Copy the generated API token and paste it into the `.env` file.

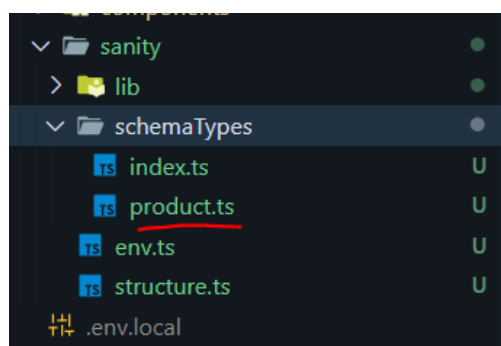
```

1  NEXT_PUBLIC_SANITY_PROJECT_ID="enter project id"
2  NEXT_PUBLIC_SANITY_DATASET="production"
3  SANITY_API_TOKEN="enter you api token"

```

5. Create a Schema in Sanity

- Navigate to the `schemaTypes` folder in the Sanity Studio project.
- Create a new file named `product.ts` inside the `schemaTypes` folder.

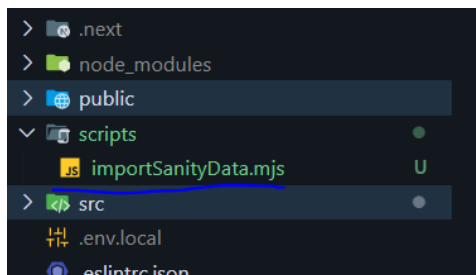


- Define the schema for your product:
- Import the `product.ts` schema into the `index.ts` file in the same folder.

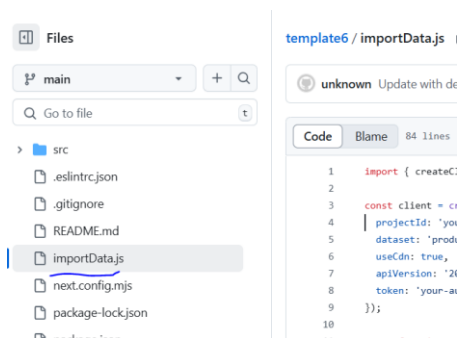
```
src > sanity > schemaTypes > ts index.ts > ...
1  import { type SchemaTypeDefinition } from 'sanity'
2  import { product } from './product'
3
4  export const schema: { types: SchemaTypeDefinition[] } = {
5    types: [product],
6  }
7
```

6. Add Scripts to Import Data

- In the root directory of the Next.js folder, create a new folder named `scripts`.
- Inside the `scripts` folder, create a file named `importSanityData.mjs`.



- Copy the code from the **importData.js** file available in the Template Six GitHub repository and paste it into `importSanityData.mjs`.



- Update the `importSanityData.mjs` file with your Sanity `projectId` and `API token`.

7. Install Axios

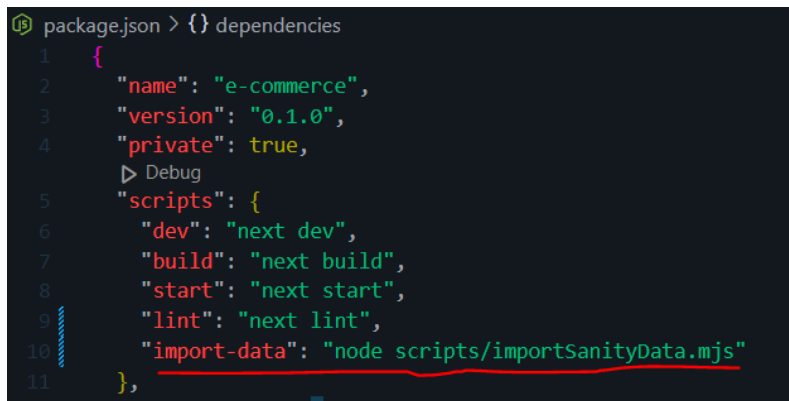
- Install Axios by running the following command



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
C:\Users\Muhammad Hudaib Arif\Desktop\hackathon\practice-hackathon-three>npm install @sanity/client axios dotenv
```

8. Add a Script in package.json

- Add the following script in the package.json file:



```
package.json > {} dependencies
1  {
2    "name": "e-commerce",
3    "version": "0.1.0",
4    "private": true,
5    "scripts": {
6      "dev": "next dev",
7      "build": "next build",
8      "start": "next start",
9      "lint": "next lint",
10     "import-data": "node scripts/importSanityData.mjs"
11   },
```

9. Import Data into Sanity

- Run the following command in the terminal:
`npm run import-data`
- This will send the data to Sanity. Verify the data in the Sanity Studio by navigating to <http://localhost:3000/studio/structure>.

10. Fetch Sanity Data in Next.js

- Write a GROQ query to fetch data from Sanity. Example:

```
*[_type == "product"] {  
  title,  
  price,  
  Description  
}
```

- Add the query in your Next.js project and use it to fetch data.

11. Allow External Image Sources

- Update `next.config.js` to allow external images from `cdn.sanity.io`:

A screenshot of a code editor showing the configuration for Next.js. The file is named 'next.config.ts' and is at the 'default' export. The code imports 'NextConfig' from 'next' and defines a 'nextConfig' object. Inside this object, there is an 'images' property with a 'domains' array containing 'cdn.sanity.io'. The code is as follows:

```
1 import type { NextConfig } from "next";  
2  
3 const nextConfig: NextConfig = {  
4   /* config options here */  
5   images: {  
6     domains: ['cdn.sanity.io'], // Add Sanity's image CDN  
7   },  
8 };  
9  
10 export default nextConfig;
```

12. Display Data in Browser

- Use the `map` function in your component to render the fetched data in the browser: