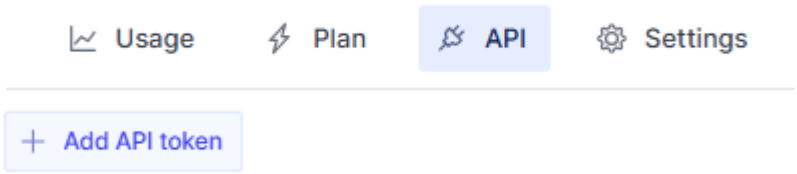
1: Set up a new Next.js project

1. Create a new folder for your project designed for a UI/UX .
2. Initialize a Next.js project inside this folder using the command:

2: Install Sanity and configure it

1. Install Sanity in the project by running:

2. Generate an **API Token** from the Sanity dashboard.
    a. Go to the Sanity dashboard and create a project.
    b. Generate a new API token with appropriate permissions.

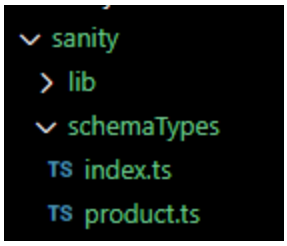


3: Add the API Token to the `.env` file

1. Create a `.env` file in your Next.js project root directory.
2. Add your API Token as an environment variable in the `.env` file:

4: Create the schema for your Sanity project

1. In the Sanity project folder, locate or create a folder named `schemaTypes`.
2. Inside `schemaTypes`, create a file named `product.ts` and define your schema for products.
3. Import this schema into the `index.ts` file in the same folder to register it with Sanity:



5: Create a script for importing data into Sanity

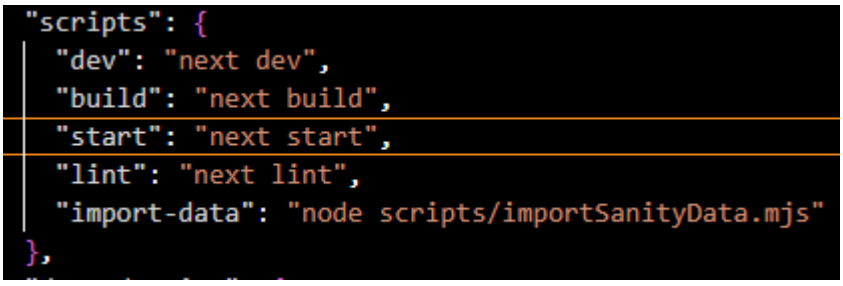
1. In your Next.js project, create a folder named `scripts` under the `routes` directory.
2. Inside `scripts`, create a file named `importSanityData.mjs`.
3. Add the script code provided by your instructor, replacing placeholders with:
    a. Your Sanity Project ID
    b. Your Sanity API Token

4. Install `axios` (if not already installed) to handle API requests:

    npm install @sanity/client axios dotenv

6: Run the script to import data into Sanity

1. Add a script in your `package.json` file to execute the import script:



2. Run the script to import data:

    npm run import-data

Step 7: Query data from Sanity in Next.js

1. Write a query to fetch the data you imported into Sanity.
2. Use Sanity's client in your Next.js project to fetch the data.
3. Install the Sanity client if necessary:

4. Create a function in `next.config.ts` or a utility file to fetch data from Sanity.

```
import type  { NextConfig } from "next";

const nextConfig: NextConfig = {
    images: {
        domains: ['cdn.sanity.io']
    }
}
export default nextConfig;
```

Step 8: Map and display data in the browser

1. Use the `map()` function to iterate over the fetched data and format it as needed.
2. Pass the formatted data to a component.
3. Display the data in your browser by rendering it inside the component.