# Hackathon Report

Presented by
**Shagufta Zakir**

Day 3

## API INTEGRATION AND DATA MIGRATION

## 1. API Integration steps:

**Step 1**: **Install Sanity in the Next.js Project**

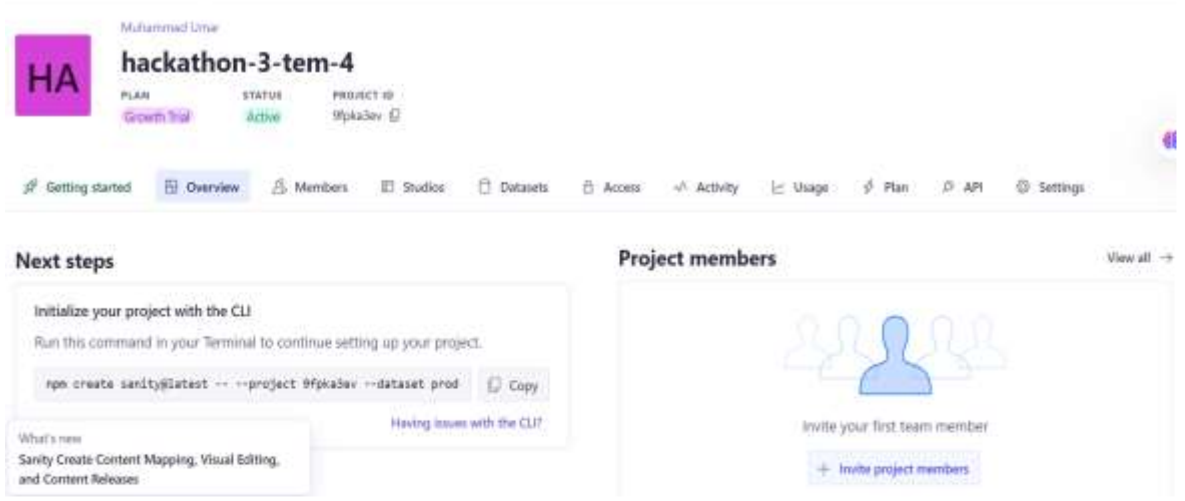
- Begin by setting up Sanity in your Next.js project.
- Create a new project in Sanity and retrieve the project ID and token. it will be utilized in the Next.js application for API integration

**Step 2: Define Schemas in the Sanity Folder**

- Navigate to the sanity/schemas folder in your Sanity project directory.
- Create a file named Products.ts and define the schema for products.

**Step 3: Setup Scripts for Data Migration**

- At the project root, create a folder named scripts.
- Inside this folder, create a file named importSanityData.mjs.
-  This file will be used to import the provided data into Sanity.



Initialize your project with the CLI

Run this command in your Terminal to continue setting up your project.

```
npm create sanity@latest -- --project 9fpka3ev --dataset prod
```

What's new

Sanity Create Content Mapping, Visual Editing, and Content Releases

**Step 4: Install Required Dependencies**

Run the following command in the terminal to install the necessary packages

```
npm install @sanity/client axios dotenv
```

**Step 5: Update package.json**

Add the following script to the package.json file.

```
"import-data": "node scripts/importSanityData.js"
```

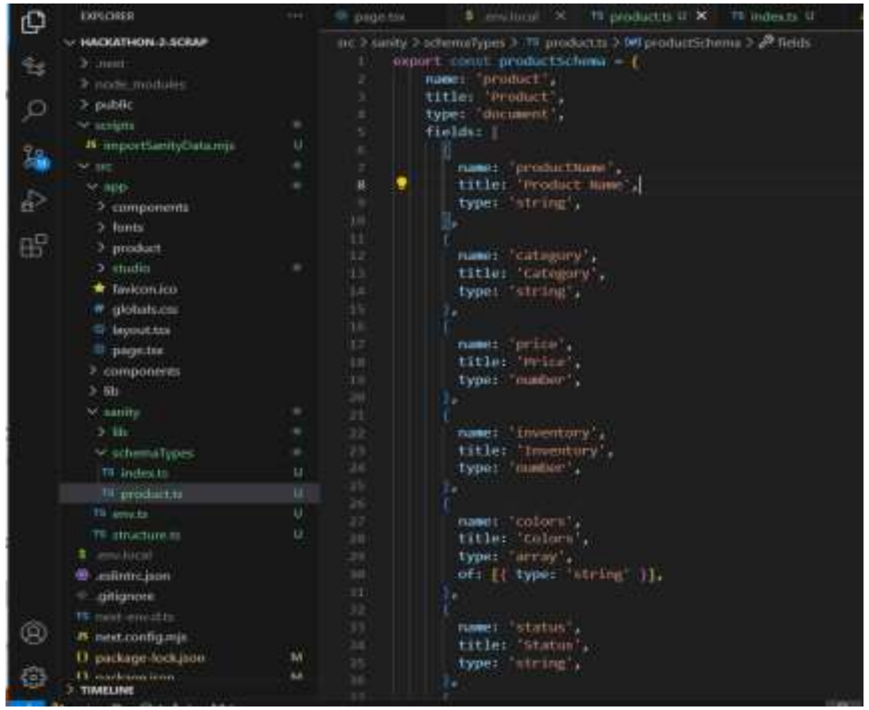
**Step 6: Import Data into Sanity**

To execute the data import, run the following command in the terminal:

```
npm run import-data
```

## 2. Adjustments Made to Schemas:

The Product schema defines how product data is structured and organized within the Sanity content platform. Its purpose is to store detailed information about each product in a way that ensures flexibility and supports various use cases, such as e-commerce platforms or product catalogs.

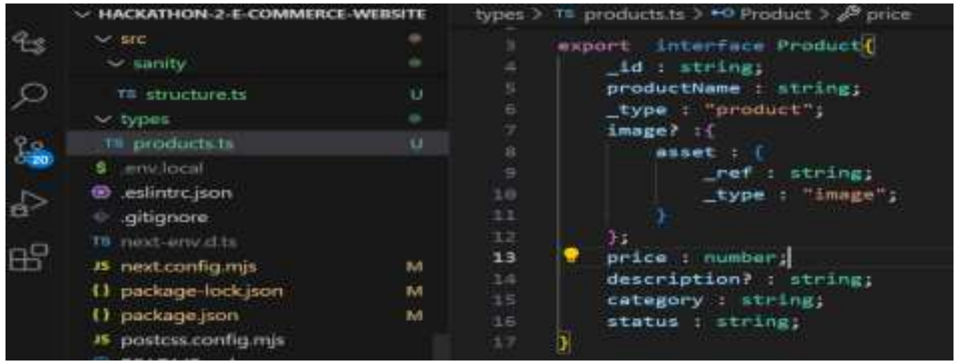


```
export const productSchema = {
    name: 'product',
    title: 'Product',
    type: 'document',
    fields: [
      {
        name: 'productName',
        title: 'Product Name',
        type: 'string',
      },
      {
        name: 'category',
        title: 'Category',
        type: 'string',
      },
      {
        name: 'price',
        title: 'Price',
        type: 'number',
      },
      {
        name: 'inventory',
        title: 'Inventory',
        type: 'number',
      },
      {
        name: 'colors',
        title: 'Colors',
        type: 'array',
        of: [{ type: 'string' }],
      },
      {
        name: 'status',
        title: 'Status',
        type: 'string',
      },
```

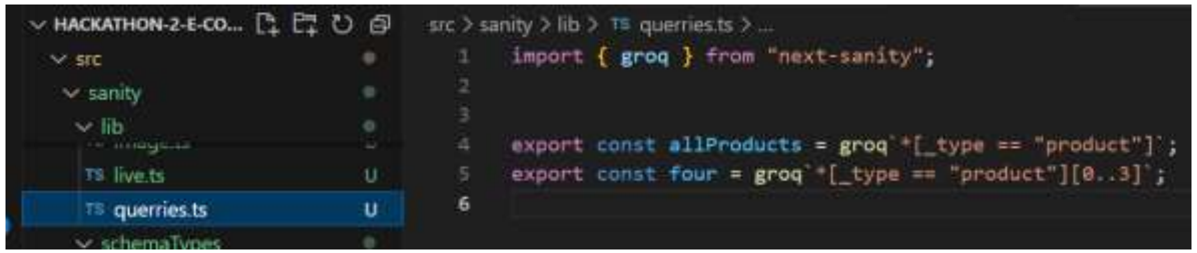
## 3. Migration Steps and Tools Used:

### 1.Query Setup:

A query is defined to fetch complete and well-structured product data from the Sanity backend.



```
export  interface Product{
    _id : string;
    productName : string;
    _type : "product";
    image? :{
        asset : {
            _ref : string;
            _type : "image";
        }
    };
    price : number;
    description? : string;
    category : string;
    status : string;
}
```



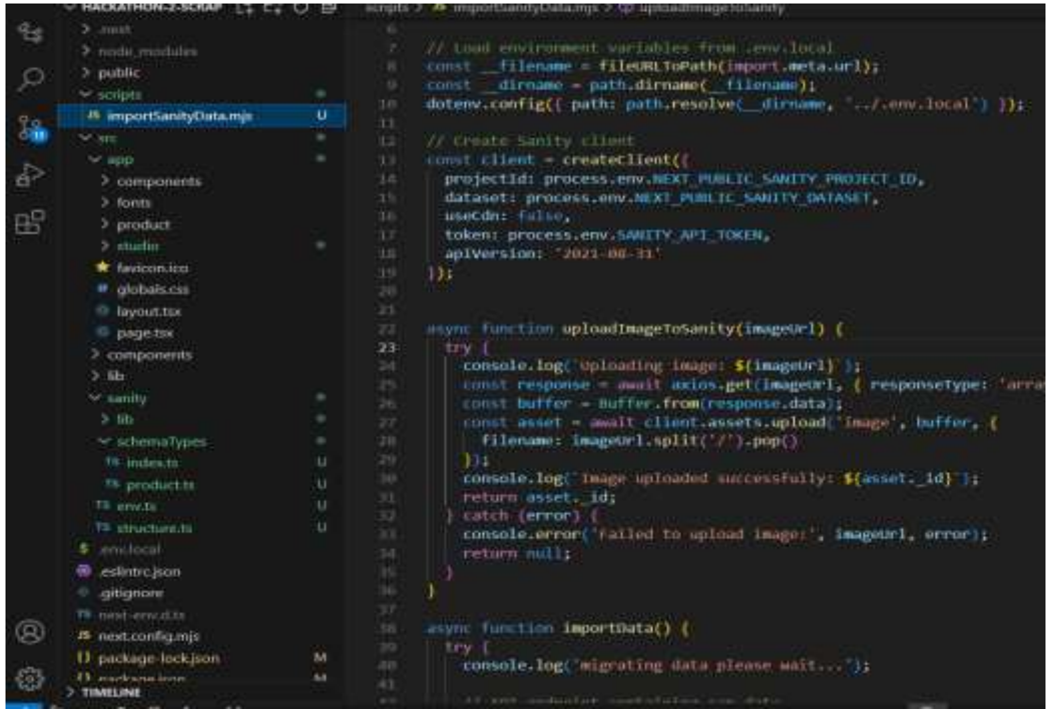
```
import { groq } from "next-sanity";


export const allProducts = groq`*[_type == "product"]`;
export const four = groq`*[_type == "product"][0..3]`;
```

### 2. Code Overview:

- The code initializes a Sanity client using the createClient function provided by the next-sanity package.
- This client connects the application to the Sanity CMS project by specifying the projectId, dataset, and apiVersion.
- A reusable sanityFetch function is created for querying data dynamically. This function accepts a query string and optional params to make data retrieval flexible and efficient.
- By using this setup, the application retrieves content dynamically and simplifies data management.

## Importing Data into sanity

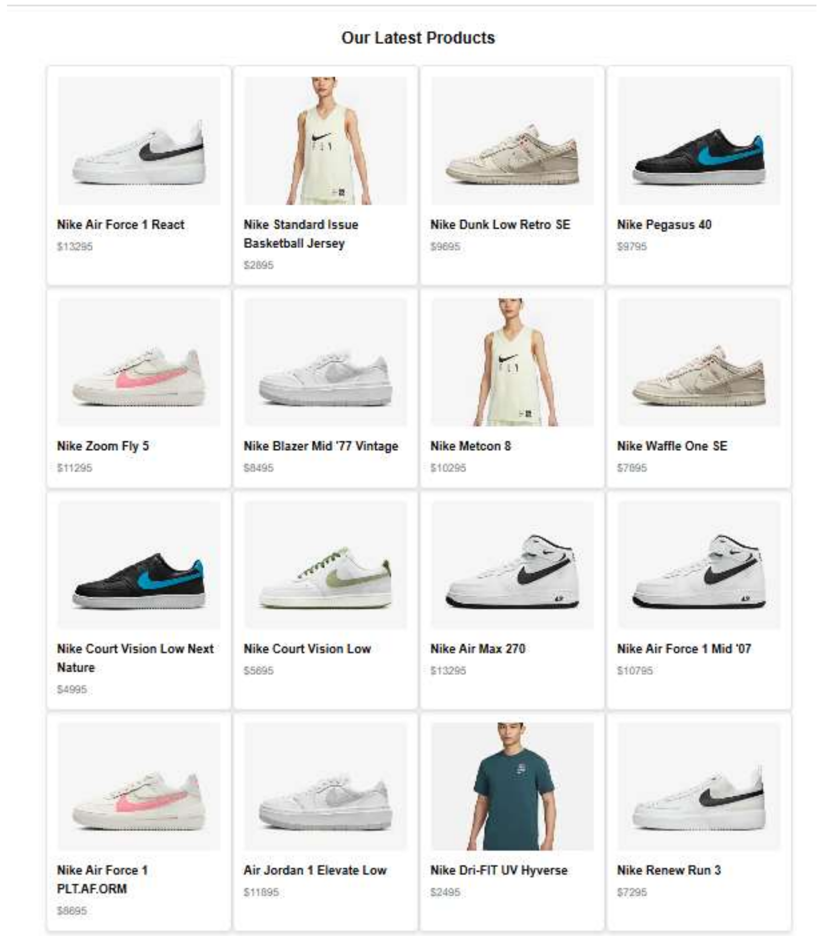
The process involves setting up the Sanity CMS and importing data to create a dynamic and structured database. This allows seamless management and retrieval of product data. Once the data is successfully imported, it can be used for building dynamic applications



```
HACKATHON-2-SCRAP
> .next
> node_modules
> public
v scripts
  JS importSanityData.mjs      U
v src
  v app
    > components
    > fonts
    > product
    > studio
    favicon.ico
    globals.css
    layout.tsx
    page.tsx
  > components
  > lib
  v sanity
    > lib
    v schemaTypes
      TS index.ts              U
      TS product.ts            U
    TS env.ts                  U
    TS structure.ts            U
$ .env.local
.eslintrc.json
.gitignore
TS next-env.d.ts
JS next.config.mjs
{} package-lock.json           M
{} package.json                M
> TIMELINE
```

```
scripts > JS importSanityData.mjs > uploadImageToSanity

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});


async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arra
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      filename: imageUrl.split('/').pop()
    });
    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function importData() {
  try {
    console.log('migrating data please wait...');
```

## DATA SUCCESSFULLY DISPLAY IN FRONTEND



Our Latest Products

Nike Air Force 1 React
$13295

Nike Standard Issue Basketball Jersey
$2895

Nike Dunk Low Retro SE
$9695

Nike Pegasus 40
$9795

Nike Zoom Fly 5
$11295

Nike Blazer Mid '77 Vintage
$8495

Nike Metcon 8
$10295

Nike Waffle One SE
$7895

Nike Court Vision Low Next Nature
$4995

Nike Court Vision Low
$5695

Nike Air Max 270
$13295

Nike Air Force 1 Mid '07
$10795

Nike Air Force 1 PLT.AF.ORM
$8695

Air Jordan 1 Elevate Low
$11895

Nike Dri-FIT UV Hyverse
$2495

Nike Renew Run 3
$7295

## Day 3 Checklist:

### Self-validation checklist

- API Understanding: ✓
- Schema Validation: ✓
- Data Migration: ✓
- API integration in next.js: ✓
- Submission preparation: ✓