

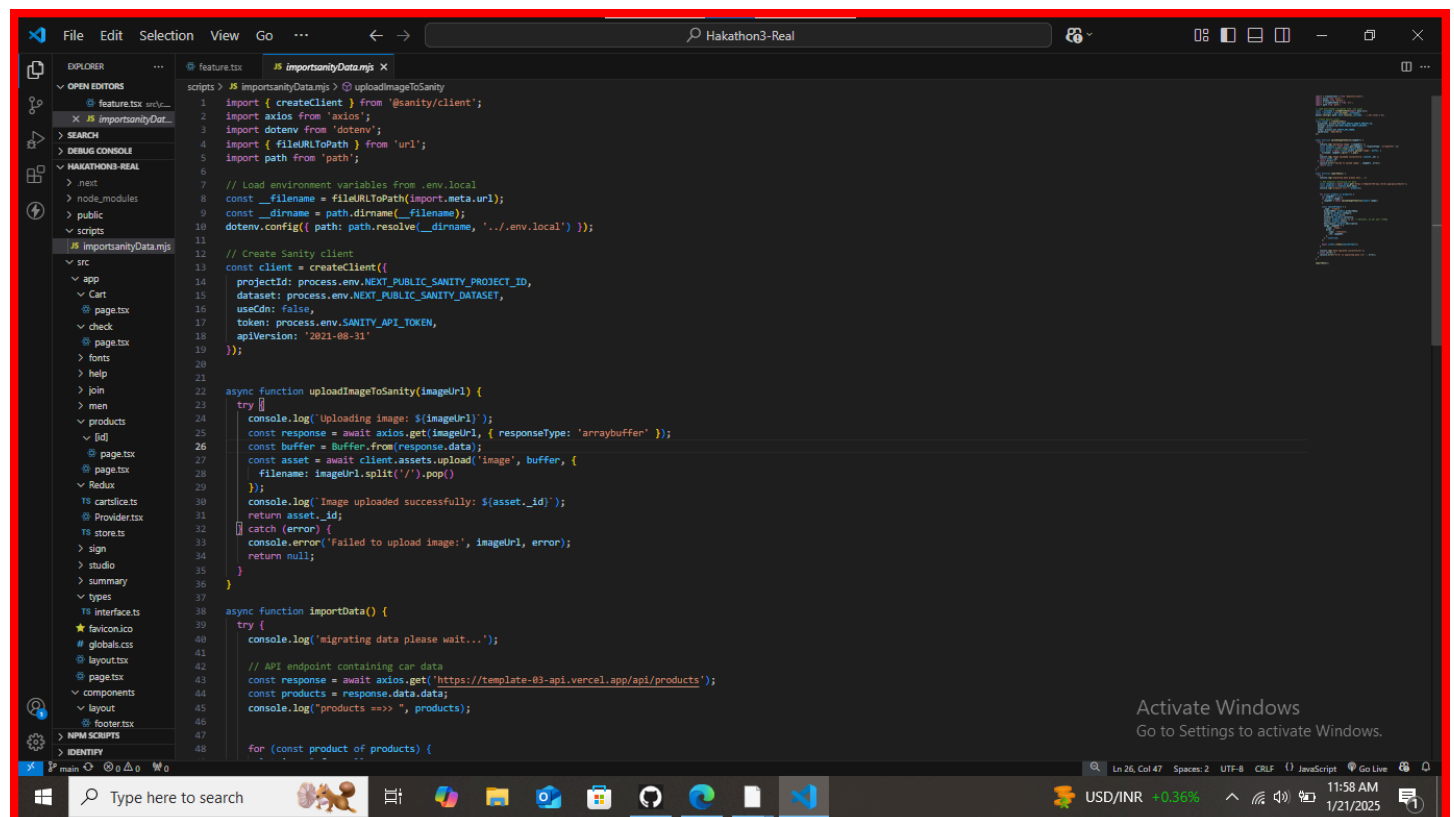
# Day 3 - API Integration Report – General – Ecommerce API Integration Process .

## 1. Followed API Documentation

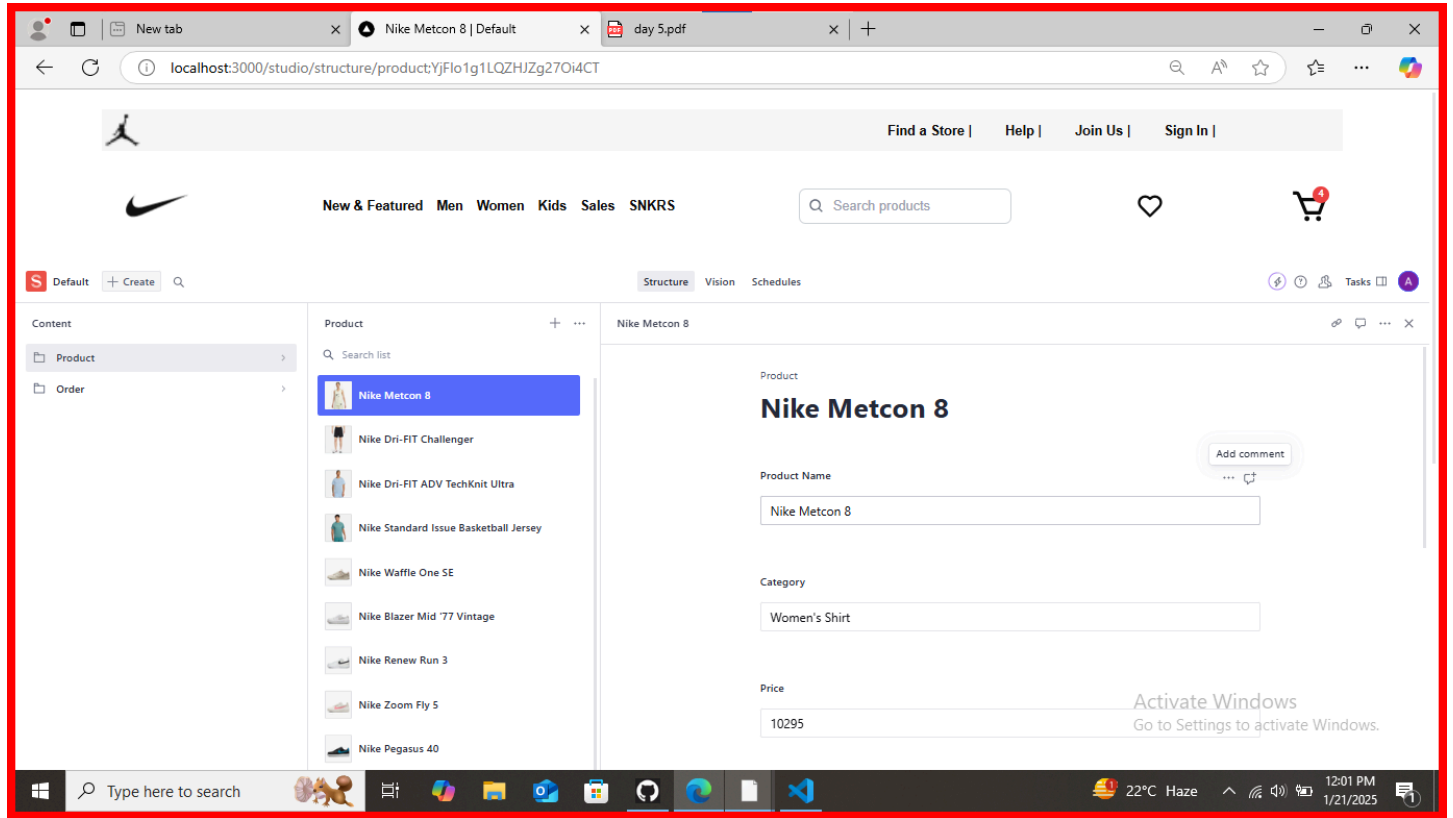
- . Reviewed the provided API documentation of the template to understand its structure and endpoints.
- o Identified required API endpoints for fetching and managing data.

## 2. Data Migration Script Execution

- Executed a custom script designed to migrate API data into Sanity CMS.
- o Verified successful migration by checking the populated Sanity CMS fields

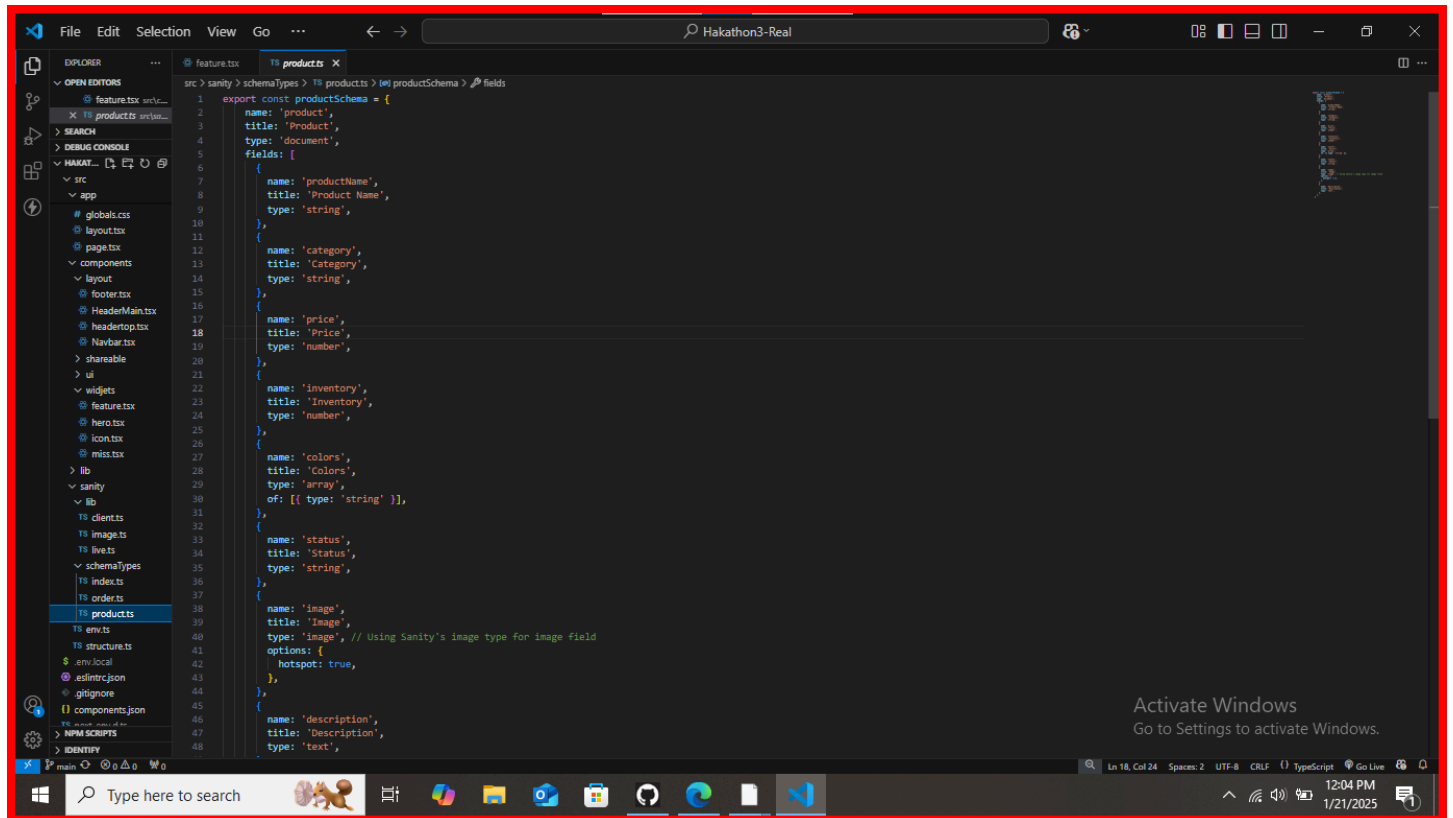


```
1 import { createClient } from '@sanity/client';
2 import axios from 'axios';
3 import dotenv from 'dotenv';
4 import { fileURLToPath } from 'url';
5 import path from 'path';
6
7 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ', imageUrl);
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     });
29     console.log('Image uploaded successfully: ', asset._id);
30     return asset._id;
31   } catch (error) {
32     console.error('Failed to upload image: ', imageUrl, error);
33     return null;
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('migrating data please wait...');
40
41     // API endpoint containing car data
42     const response = await axios.get('https://template-03-api.vercel.app/api/products');
43     const products = response.data.data;
44     console.log('products ==> ', products);
45
46     for (const product of products) {
47
48
```



# Adjustments Made in Product Schemas

- Modifications were made to the default schemas to accommodate the migrated data.
- Ensured compatibility with the structure of the API data



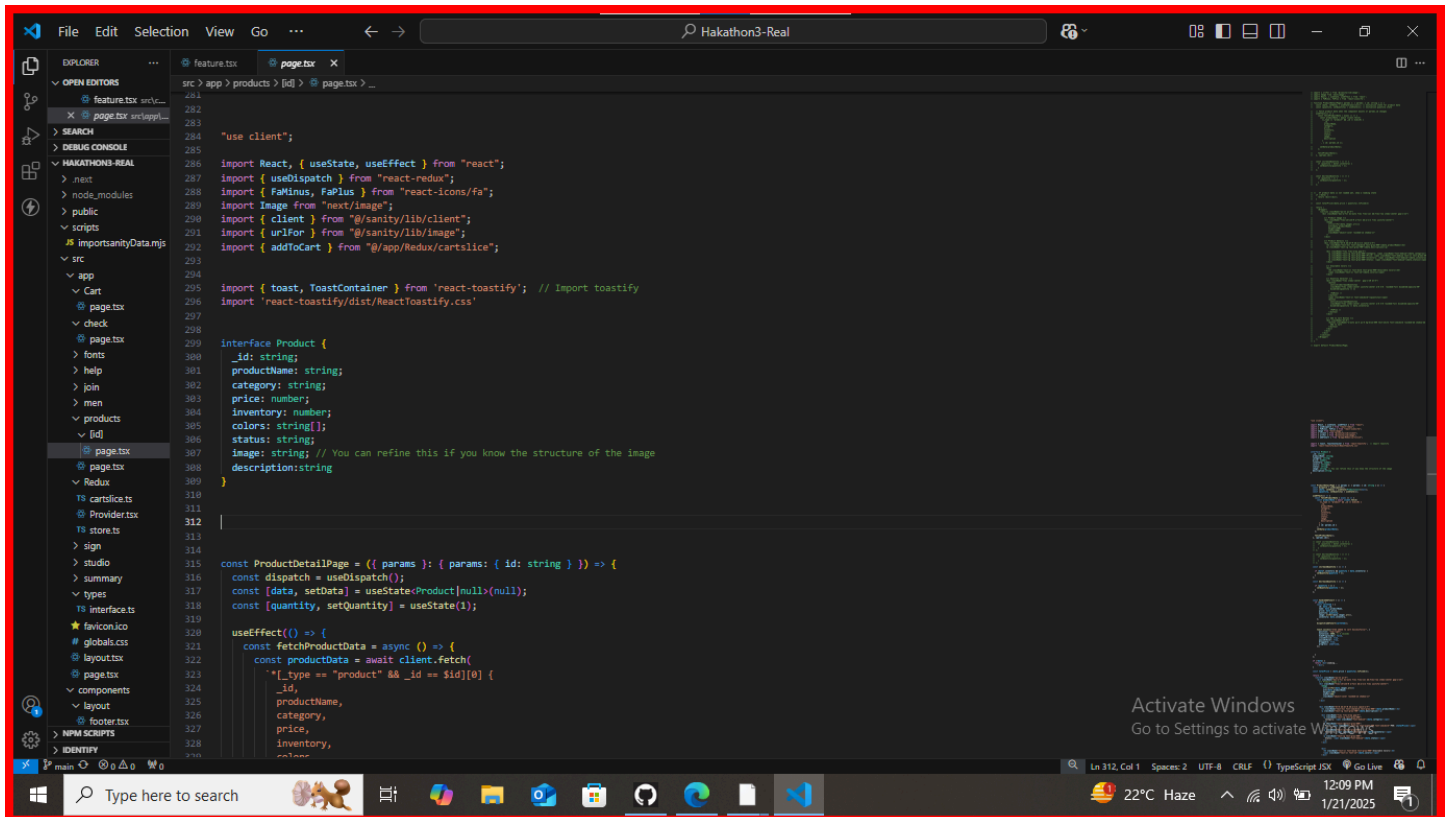
# Displaying Migrated Data in Frontend

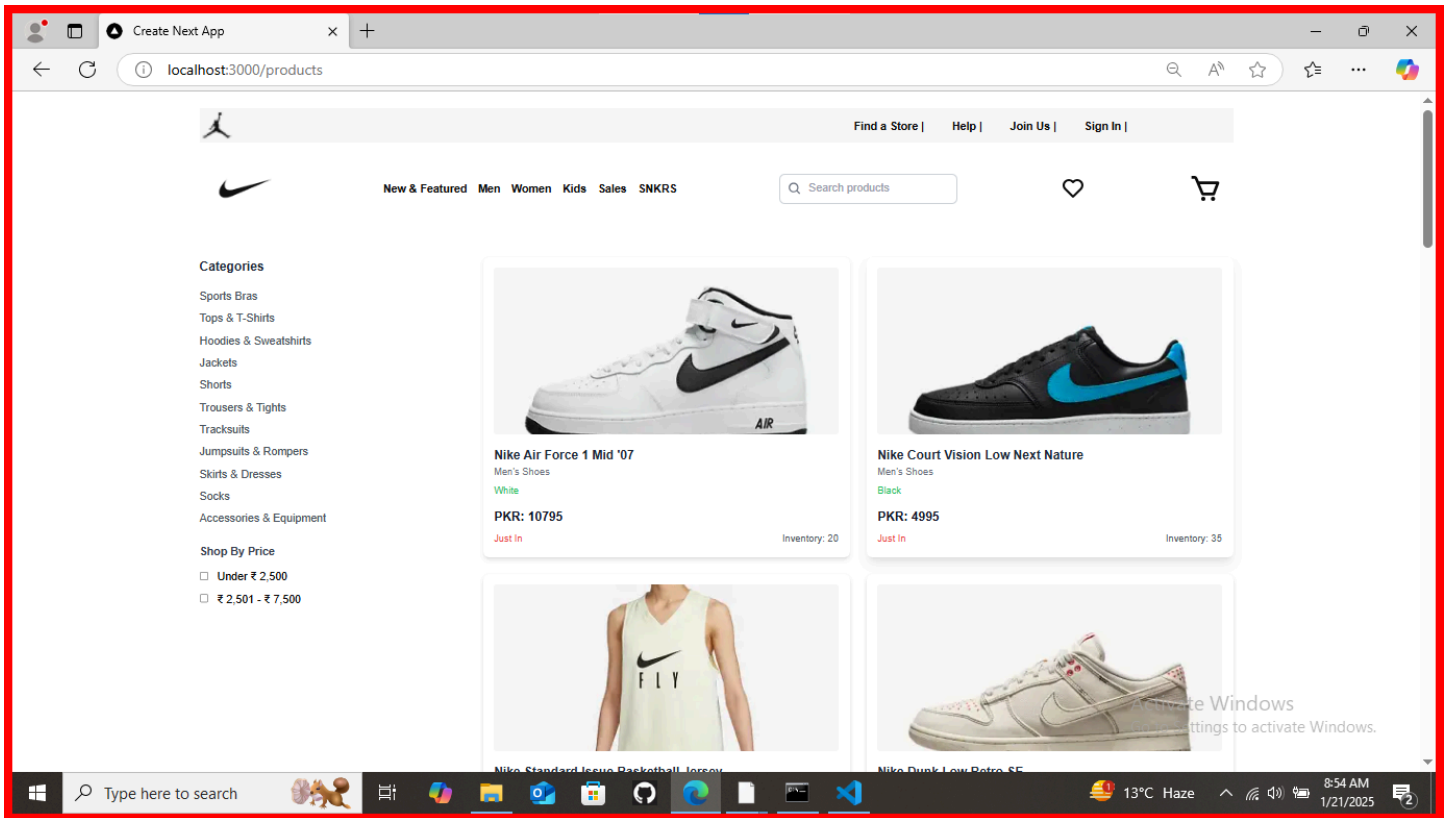
## 1. GROQ Query Implementation

- Constructed GROQ queries to fetch the newly migrated data from Sanity CMS.
- Optimized queries for efficient data retrieval.

## 2. Next.js Frontend Integration .

- Successfully displayed the migrated data on the Next.js application.
- Tested UI to confirm data accuracy and responsiveness.





# Migration Steps and Tools Used .

## • Tools Used:

Sanity CLI for schema management.

Custom scripts for data migration.

GROQ queries for fetching and displaying data.

## • Migration Steps:

1. Prepared the API and mapped its fields to corresponding Sanity schema fields.
2. Ran the migration script to transfer data into Sanity.

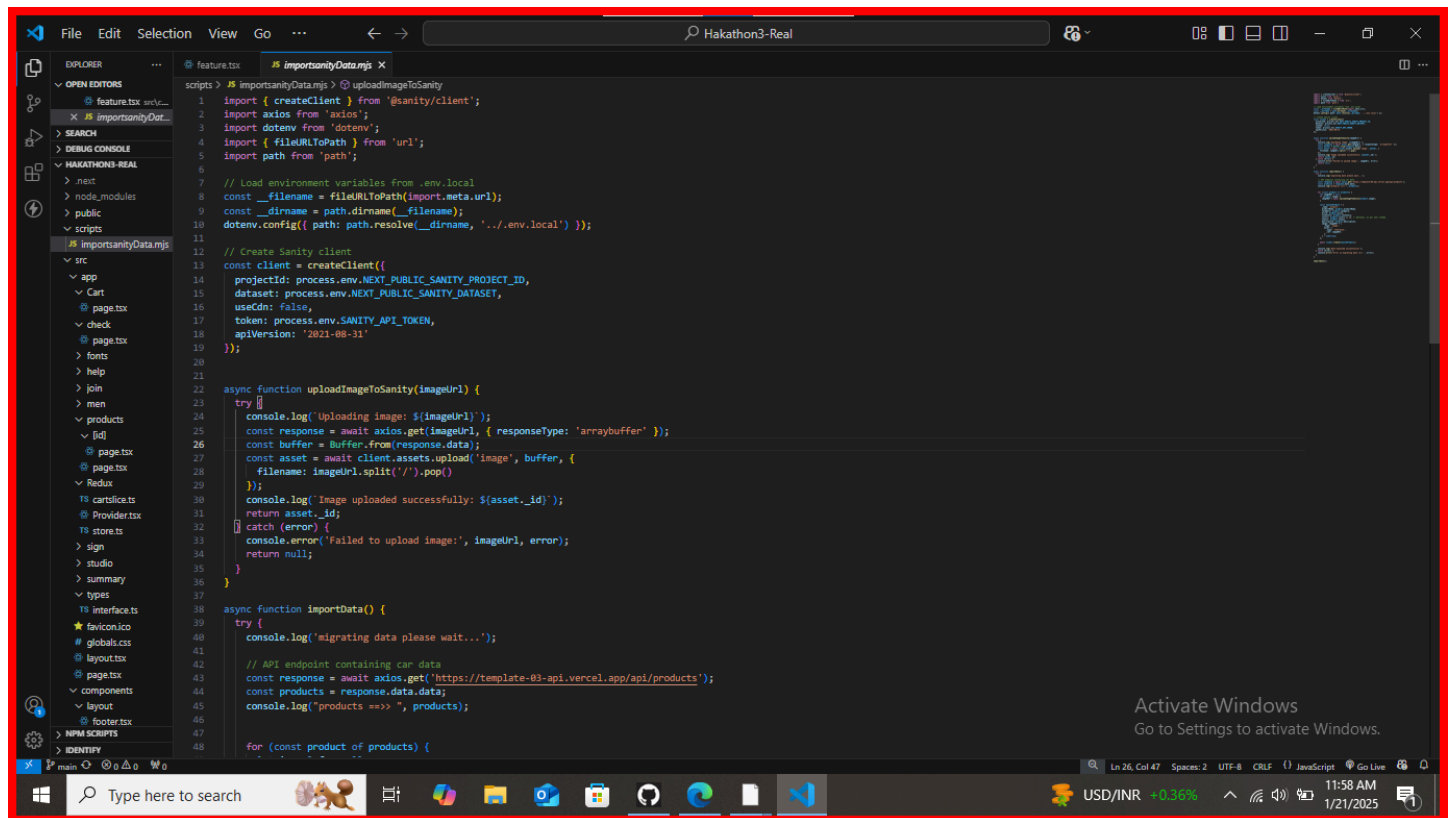
3. Validated the data in Sanity CMS before rendering it in the frontend.

## Screenshots Required :-

1. Script execution and successful migration logs.

2. Data displayed successfully in the frontend.

3. Populated Sanity CMS fields.



The screenshot shows a VS Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders like 'src', 'app', 'Cart', 'check', 'page', 'fonts', 'help', 'join', 'men', 'products', 'id', 'page', 'page', 'Redux', 'TS carts', 'Provider', 'store', 'sign', 'studio', 'summary', 'types', 'interface', 'fav', 'globals', 'layout', 'components', 'layout', 'footer', 'NPM', and 'IDENTITY'. The code editor shows a JavaScript file named 'importSanityData.mjs' with the following content:

```
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 // Load environment variables from .env.local
8 const __filename = fileURLToPath(import.meta.url);
9 const __dirname = path.dirname(__filename);
10 dotenv.config({ path: path.resolve(__dirname, '../.env.local') });
11
12 // Create Sanity client
13 const client = createClient({
14   projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
15   dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
16   useCdn: false,
17   token: process.env.SANITY_API_TOKEN,
18   apiVersion: '2021-08-31'
19 });
20
21 async function uploadImageToSanity(imageUrl) {
22   try {
23     console.log('Uploading image: ' + imageUrl);
24     const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
25     const buffer = Buffer.from(response.data);
26     const asset = await client.assets.upload('image', buffer, {
27       filename: imageUrl.split('/').pop()
28     });
29     console.log('Image uploaded successfully: ' + asset._id);
30     return asset._id;
31   } catch (error) {
32     console.error('Failed to upload image: ' + imageUrl, error);
33     return null;
34   }
35 }
36
37 async function importData() {
38   try {
39     console.log('migrating data please wait...');
40
41     // API endpoint containing car data
42     const response = await axios.get('https://template-03-api.vercel.app/api/products');
43     const products = response.data.data;
44     console.log('products ==> ', products);
45
46     for (const product of products) {
47
48
```

