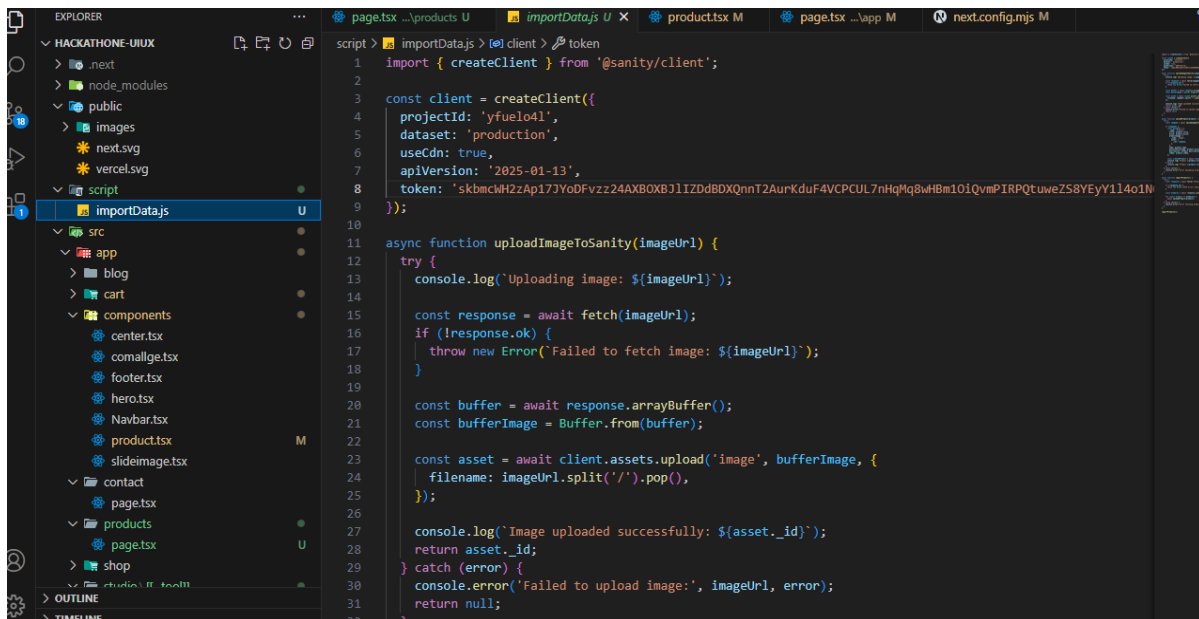


Day 3: Integration and Data Migration

Introduction: On Day 3, the focus is on API integration and data migration. These are critical steps in ensuring the seamless flow of information between different parts of a web application. The key components covered include integrating APIs, creating a Sanity schema, importing data using `importdata.js`, and displaying the imported data on the frontend.

API Integration: API integration involves connecting the frontend of an application to external or internal APIs. This allows the application to fetch and display dynamic data. For this task:

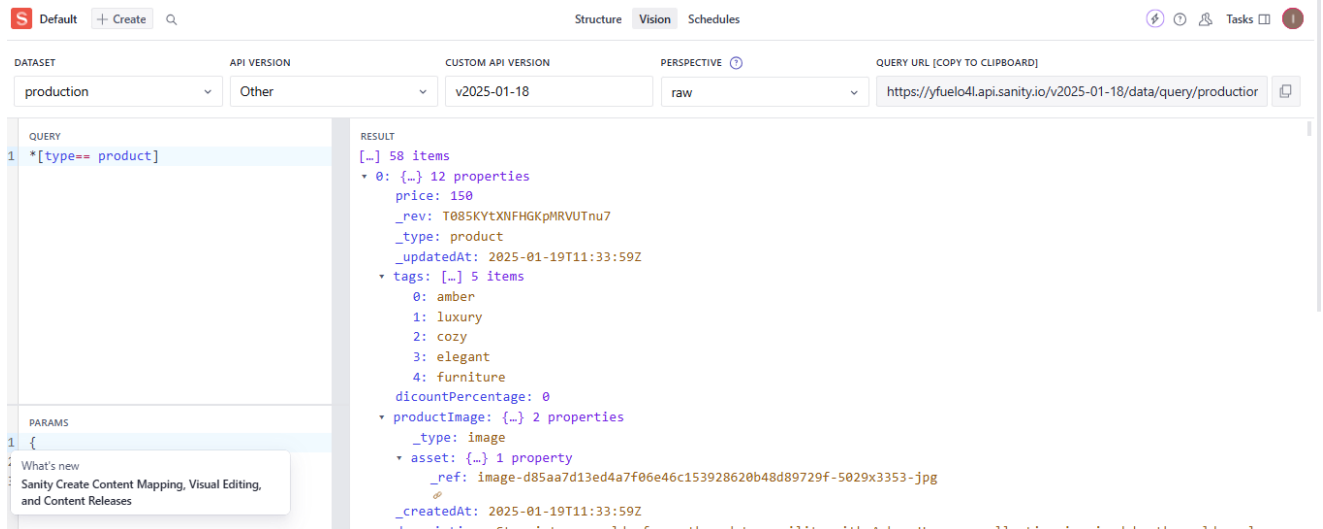
1. The required API endpoints were identified and tested.
2. Authentication (if needed) was set up to ensure secure communication.
3. Data was fetched using libraries like Axios or Fetch.



```
1 import { createClient } from '@sanity/client';
2
3 const client = createClient({
4   projectId: 'yfuelo4l',
5   dataset: 'production',
6   useCdn: true,
7   apiVersion: '2025-01-13',
8   token: 'skbmchH2zAp173YoDFvzz24AXB0XBj1IZDdBDXQnnT2AurKduF4VCPUL7nHqMq8wH8m101QvmPIRPQtuweZS8YEyY114o1M
9 });
10
11 async function uploadImageToSanity(imageUrl) {
12   try {
13     console.log(`Uploading image: ${imageUrl}`);
14
15     const response = await fetch(imageUrl);
16     if (!response.ok) {
17       throw new Error(`Failed to fetch image: ${imageUrl}`);
18     }
19
20     const buffer = await response.arrayBuffer();
21     const bufferImage = Buffer.from(buffer);
22
23     const asset = await client.assets.upload('image', bufferImage, {
24       filename: imageUrl.split('/').pop(),
25     });
26
27     console.log(`Image uploaded successfully: ${asset._id}`);
28     return asset._id;
29   } catch (error) {
30     console.error(`Failed to upload image:`, imageUrl, error);
31     return null;
32   }
33 }
```

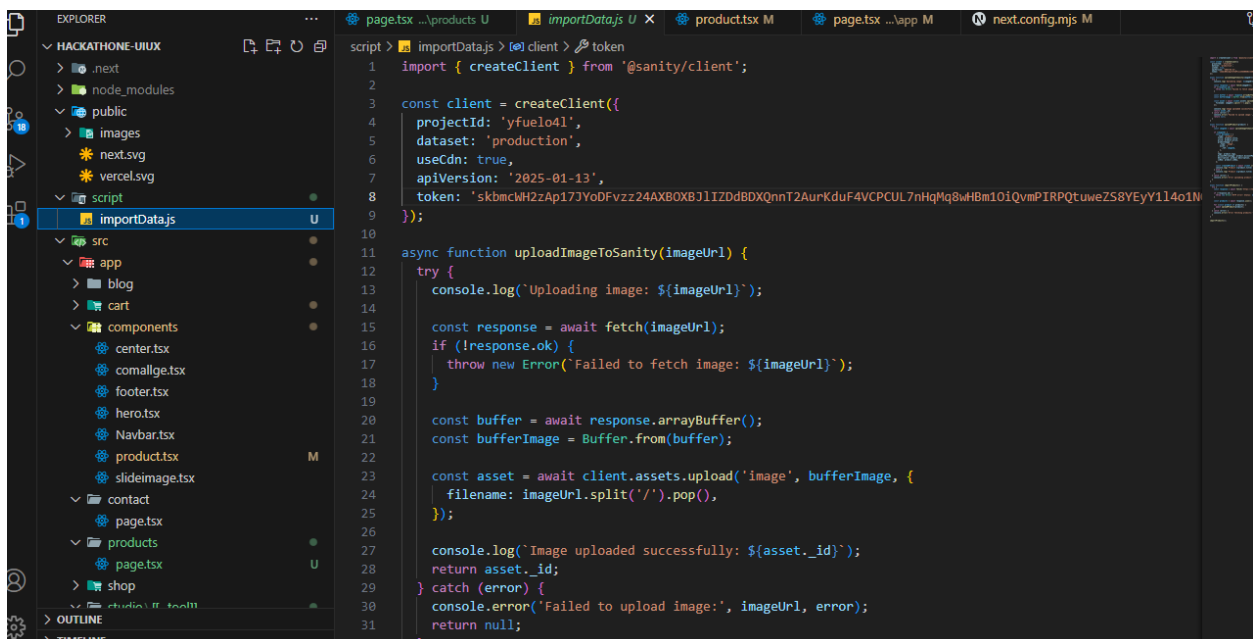
Sanity Schema Creation: Sanity CMS provides a structured way to store and manage content. A schema was created to define the structure of the data. Steps included:

1. Defining schema types for the specific data model (e.g., products, posts).
2. Adding necessary fields such as title, description, image, and price.
3. Validating the schema to ensure compatibility with the data being imported.



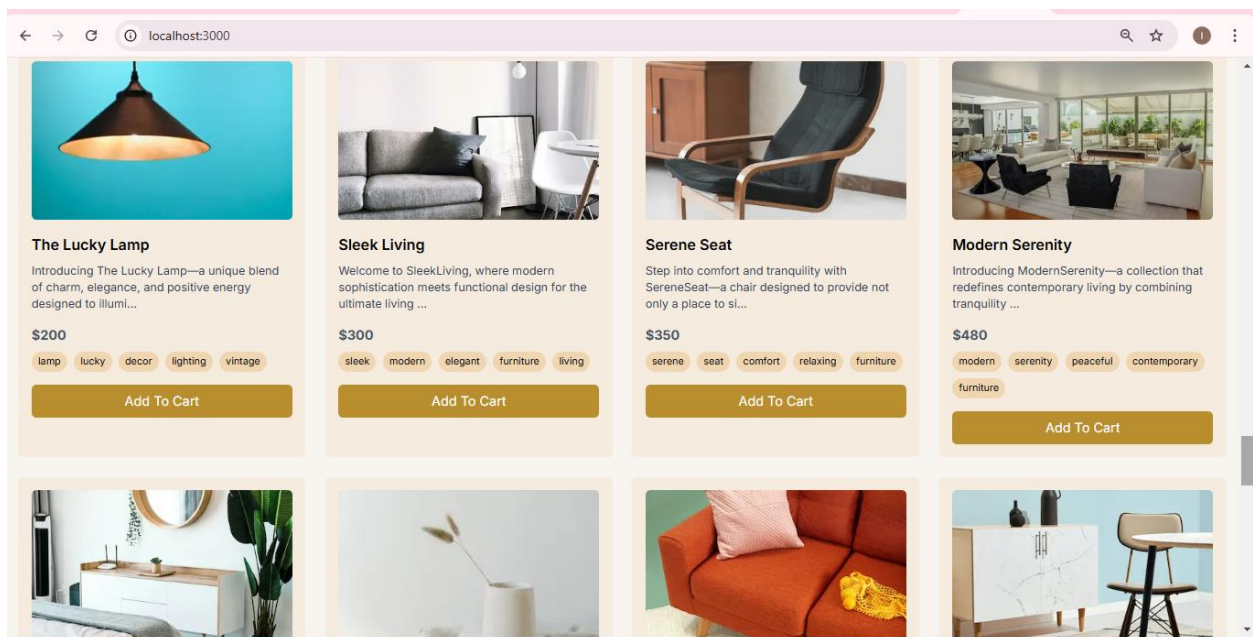
Importing Data (importdata.js): A script named `importdata.js` was written to import existing data into the Sanity dataset. The script:

1. Established a connection with the Sanity project using the project ID and dataset name.
2. Used Sanity's client library to upload JSON or CSV data.
3. Handled errors and ensured that duplicate data was not imported.



Display in Frontend: The imported data was displayed on the frontend using React or Next.js components. Steps included:

1. Fetching data from the Sanity API using GROQ queries or Sanity's client.
2. Mapping over the data to render it dynamically on the UI.
3. Styling the components using Tailwind CSS or another CSS framework to ensure a polished appearance.



Conclusion: Day 3's tasks covered the essential aspects of API integration and data migration. The process involved creating a structured backend with Sanity, successfully importing data, and dynamically displaying it on the frontend. Screenshots of the API integration process, the Sanity schema, and the final UI presentation can be attached to visually demonstrate the workflow and outcomes.