Hackathon Submission Report: API Integration and Data Migration

Introduction

In this hackathon, I developed a functional product listing system using Sanity CMS integrated with a custom API. This document outlines the steps taken, the tools and technologies used, and includes visual evidence of the completed tasks.

Key Achievements

1. **Custom API Implementation**:
    a. I used my own API for the integration process.
2. **Data Migration**:
    a. Leveraged Ali-Jawad's migration repository to transfer data from the API into Sanity CMS.
3. **Schema Design**:
    a. Created detailed Sanity schemas with multiple fields.
    b. Focused only on the product schema for simplicity.
4. **Frontend Integration**:
    a. Utilized GROQ queries to fetch data from Sanity.
    b. Rendered data seamlessly on the frontend.

Process Breakdown

1. Data Migration

- **Repository Used**: Ali-Jawad's migration repository.
- **Steps Taken**:
    - Retrieved data using the custom API.
    - Processed and imported the data into Sanity using migration scripts.

2. Sanity Schema Design

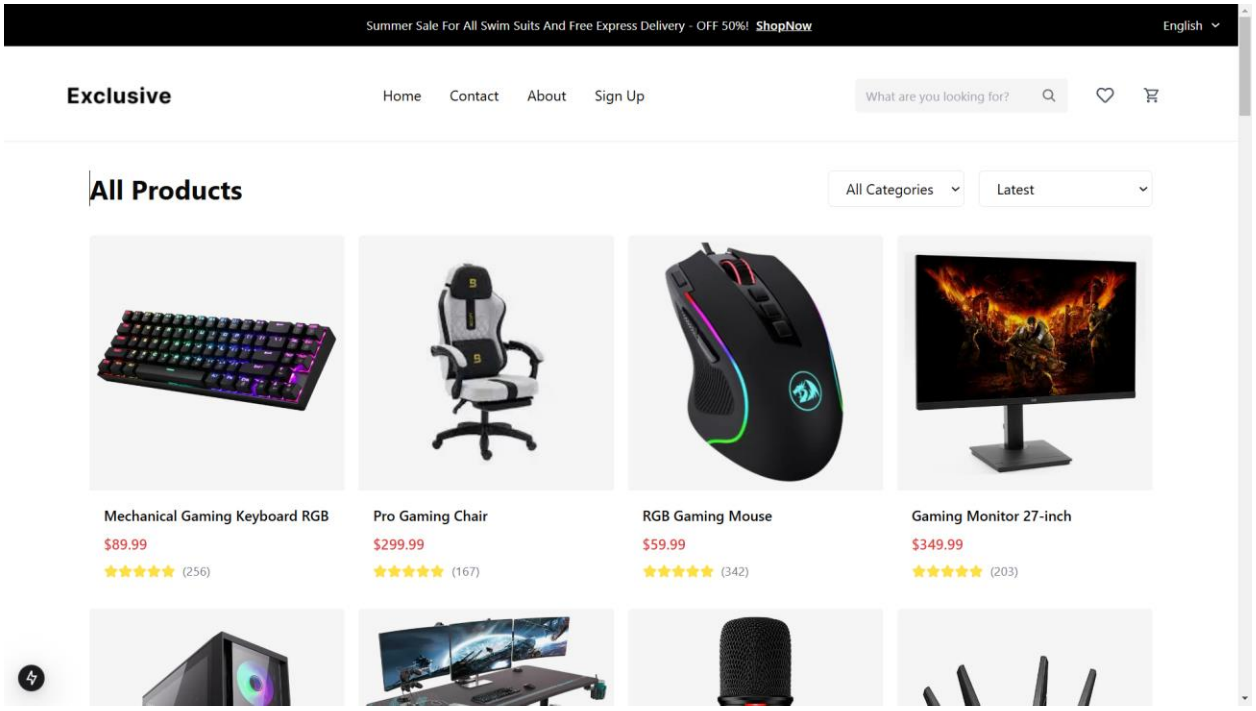
- **Schema Focus**:
  - Designed a product schema.
  - Included fields such as title, description, price, and images.

3. Data Fetching

- **Technology Used**: GROQ Queries.
- **Steps Taken**:
  - Fetched data from Sanity CMS.
  - Displayed the data in the frontend application dynamically.

Screenshots and Visual Proof

1. **Product Rendering**:



Summer Sale For All Swim Suits And Free Express Delivery - OFF 50%! ShopNow
English
Exclusive
Home
Contact
About
Sign Up
What are you looking for?
All Categories
Latest

All Products

Mechanical Gaming Keyboard RGB
$89.99
(256)

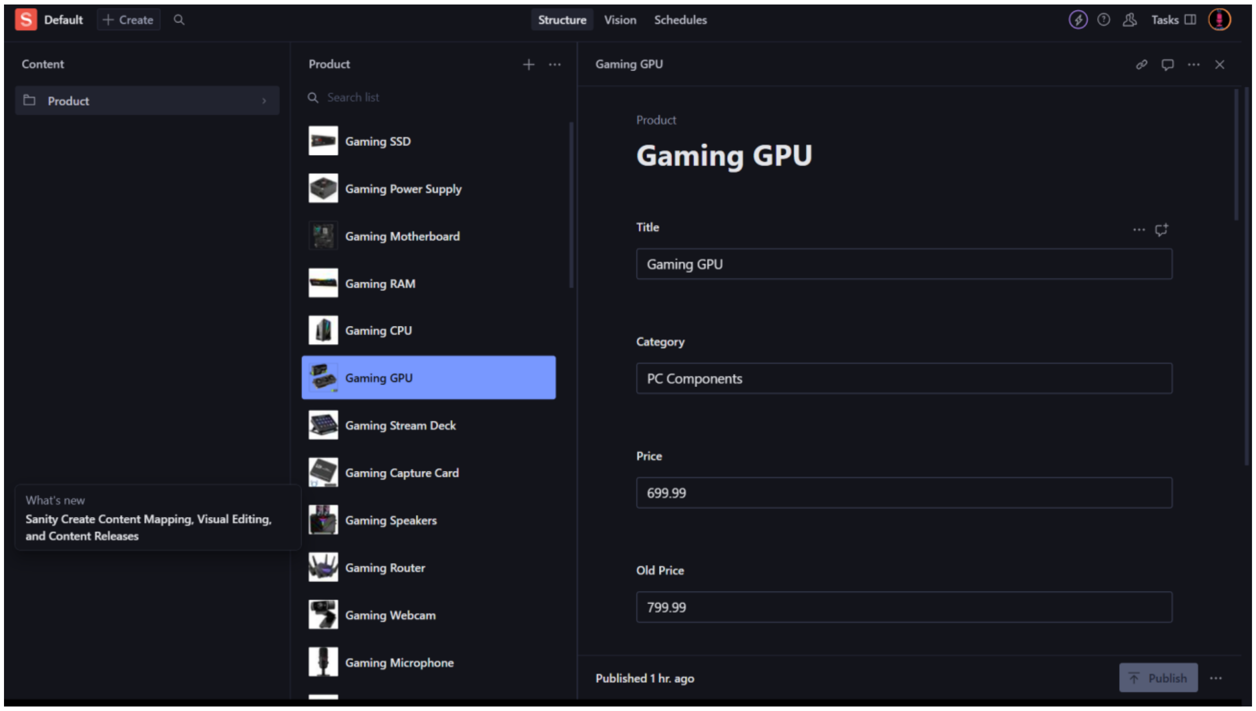
Pro Gaming Chair
$299.99
(167)

RGB Gaming Mouse
$59.99
(342)

Gaming Monitor 27-inch
$349.99
(203)

   a. Screenshot showcasing the frontend rendering of products fetched from Sanity CMS.

2. **Sanity Data Structure**:



Product

Gaming SSD
Gaming Power Supply
Gaming Motherboard
Gaming RAM
Gaming CPU
Gaming GPU
Gaming Stream Deck
Gaming Capture Card
Gaming Speakers
Gaming Router
Gaming Webcam
Gaming Microphone

What's new
Sanity Create Content Mapping, Visual Editing, and Content Releases

Gaming GPU

Product
Gaming GPU

Title: Gaming GPU

Category: PC Components

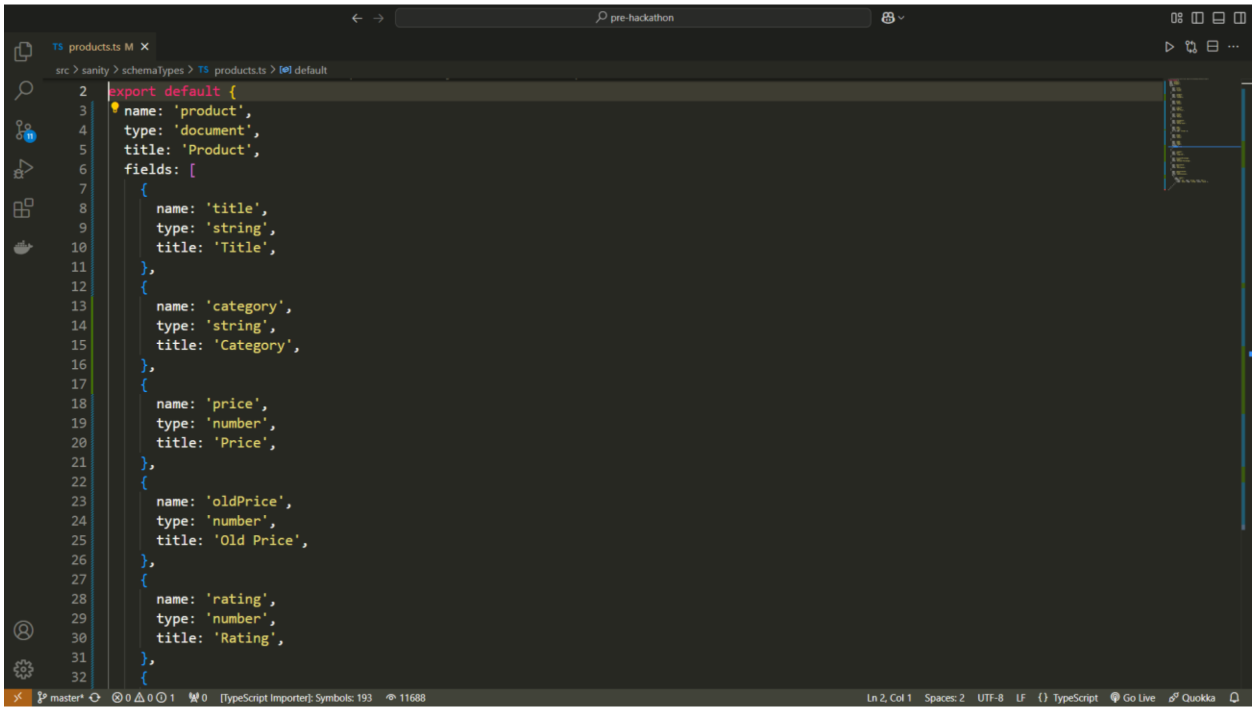
Price: 699.99

Old Price: 799.99

Published 1 hr. ago

a. Image of the populated data in Sanity CMS.

3. **Sanity Schema**:

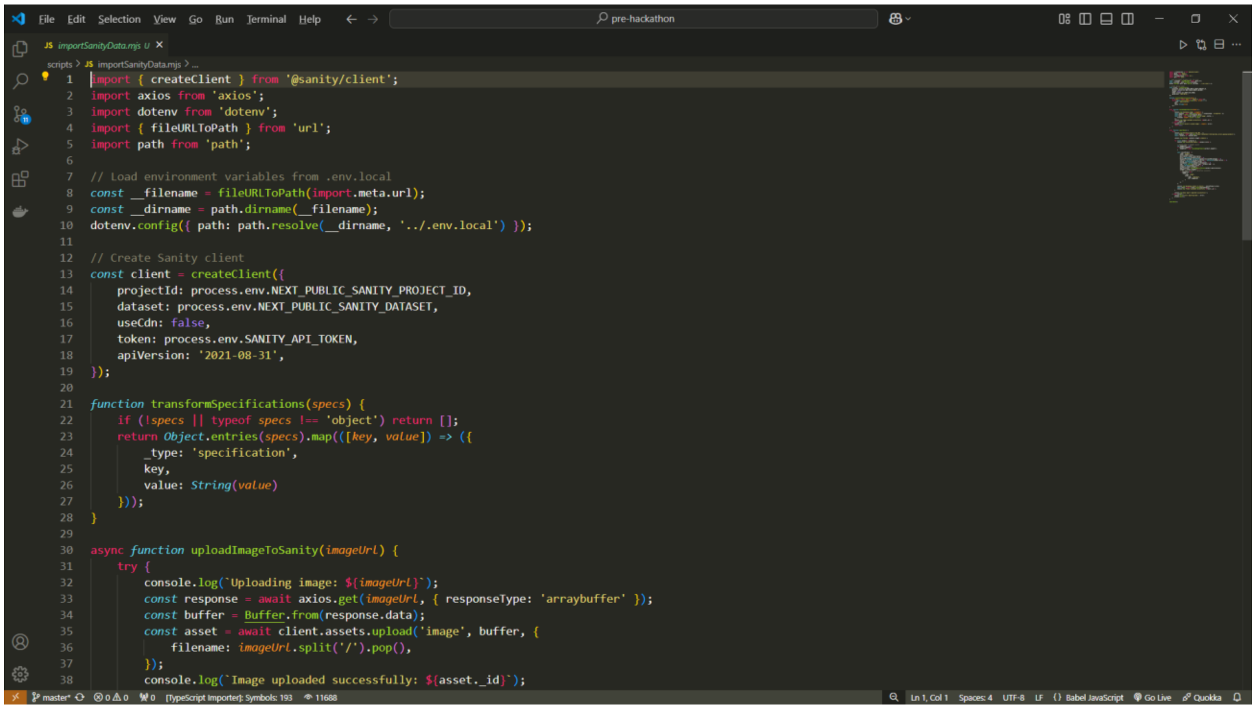

pre-hackathon
products.ts M
src > sanity > schemaTypes > products.ts > default

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'title',
      type: 'string',
      title: 'Title',
    },
    {
      name: 'category',
      type: 'string',
      title: 'Category',
    },
    {
      name: 'price',
      type: 'number',
      title: 'Price',
    },
    {
      name: 'oldPrice',
      type: 'number',
      title: 'Old Price',
    },
    {
      name: 'rating',
      type: 'number',
      title: 'Rating',
    },
    {
```

master* ⊗0 ⚠0 ⓘ1 [TypeScript Importer]: Symbols: 193 11688 Ln 2, Col 1 Spaces: 2 UTF-8 LF {} TypeScript Go Live Quokka

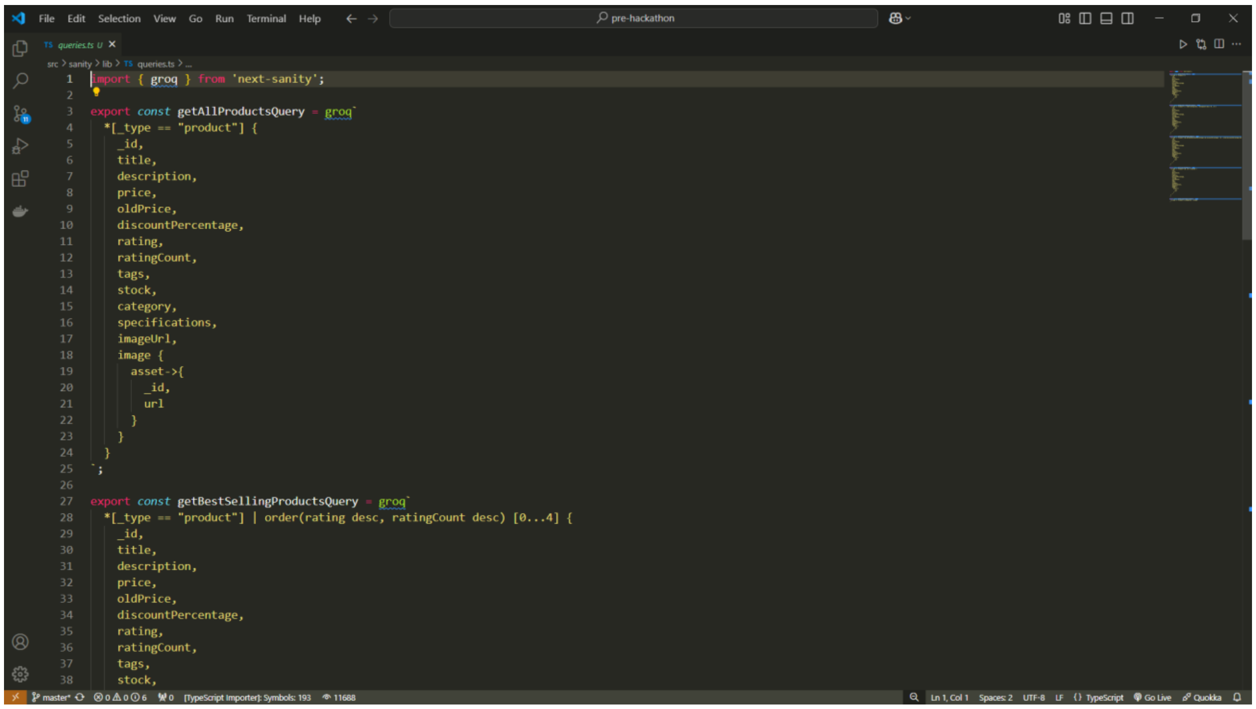
a. A screenshot of the product schema design in Sanity.

4. **Migration File**:


```javascript
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// 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',
});

function transformSpecifications(specs) {
    if (!specs || typeof specs !== 'object') return [];
    return Object.entries(specs).map(([key, value]) => ({
        _type: 'specification',
        key,
        value: String(value)
    }));
}

async function uploadImageToSanity(imageUrl) {
    try {
        console.log(`Uploading image: ${imageUrl}`);
        const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
        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}`);
```

a. Visual evidence of Ali-Jawad’s migration file used in the process.

5. **GROQ Queries**:


```typescript
import { groq } from 'next-sanity';

export const getAllProductsQuery = groq`
  *[_type == "product"] {
    _id,
    title,
    description,
    price,
    oldPrice,
    discountPercentage,
    rating,
    ratingCount,
    tags,
    stock,
    category,
    specifications,
    imageUrl,
    image {
      asset->{
        _id,
        url
      }
    }
  }
`;

export const getBestSellingProductsQuery = groq`
  *[_type == "product"] | order(rating desc, ratingCount desc) [0...4] {
    _id,
    title,
    description,
    price,
    oldPrice,
    discountPercentage,
    rating,
    ratingCount,
    tags,
    stock,
```

a. Screenshot of the GROQ queries used to fetch the data.

Tools and Technologies Used

- **Sanity CMS**: For content management and data storage.
- **Next.js**: As the frontend framework.
- **GROQ**: For querying data from Sanity CMS.
- **Custom API**: Created and utilized to fetch initial product data.
- **Migration Repository**: Ali-Jawad's script for efficient data transfer.
- **Visual Tools**: Screenshots taken for validation and demonstration.

Conclusion

This project demonstrated a complete cycle of API integration, data migration, and frontend rendering using modern tools. The process allowed me to gain hands-on experience with Sanity CMS, GROQ queries, and Next.js, preparing me for real-world challenges in data management and dynamic rendering.

Next Steps

- Extend the schema to include categories and orders.
- Enhance the frontend design for a more polished user experience.
- Optimize API queries for better performance.