
Day_3_API_Integration_and_Data_Migration

Hackathon E-commerce Marketplace Builder

Built with Next.js, Tailwind CSS, TypeScript & Sanity






Introduction

In this project, I developed a **fully functional e-commerce marketplace** using modern web technologies. The platform enables seamless buying and selling experiences with a scalable, high-performance architecture.

Tech Stack & Tools Used

- ✓ **Next.js** – Server-side rendering (SSR) & static site generation (SSG) for optimal performance.
- ✓ **Tailwind CSS** – Utility-first styling for a responsive and modern UI.
- ✓ **TypeScript** – Enhanced type safety and maintainability.
- ✓ **Sanity CMS** – Headless content management system for dynamic data.
- ✓ **APIs** – Integrated various APIs for seamless data fetching and processing.

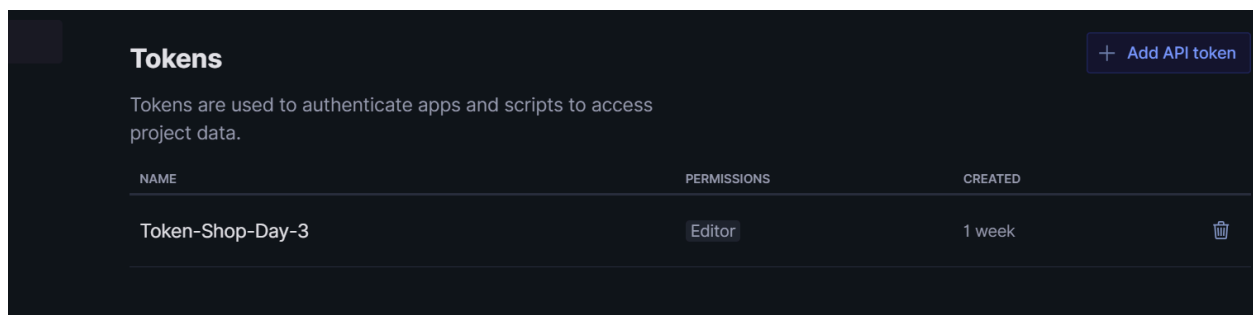
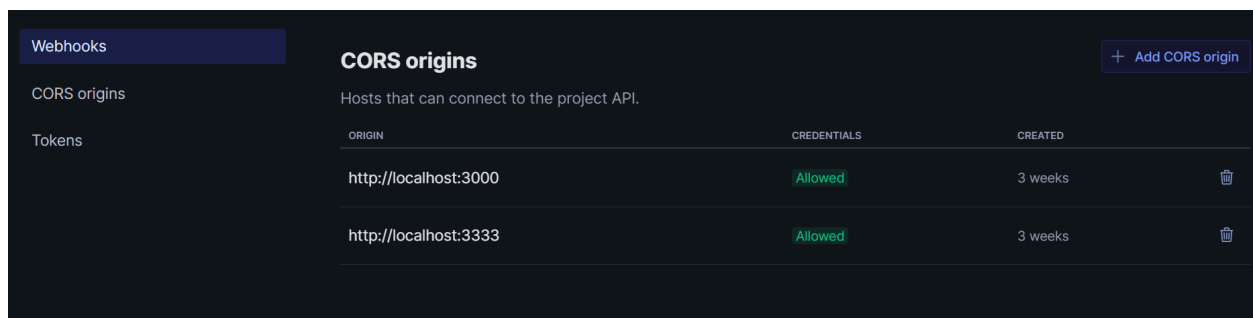
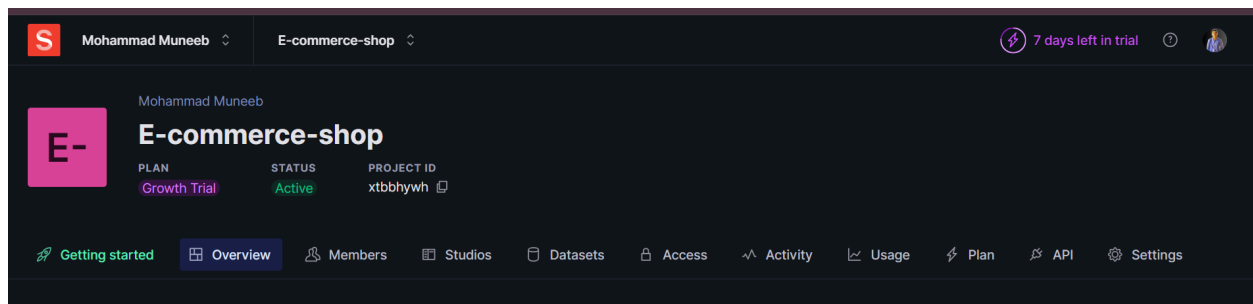
Key Features

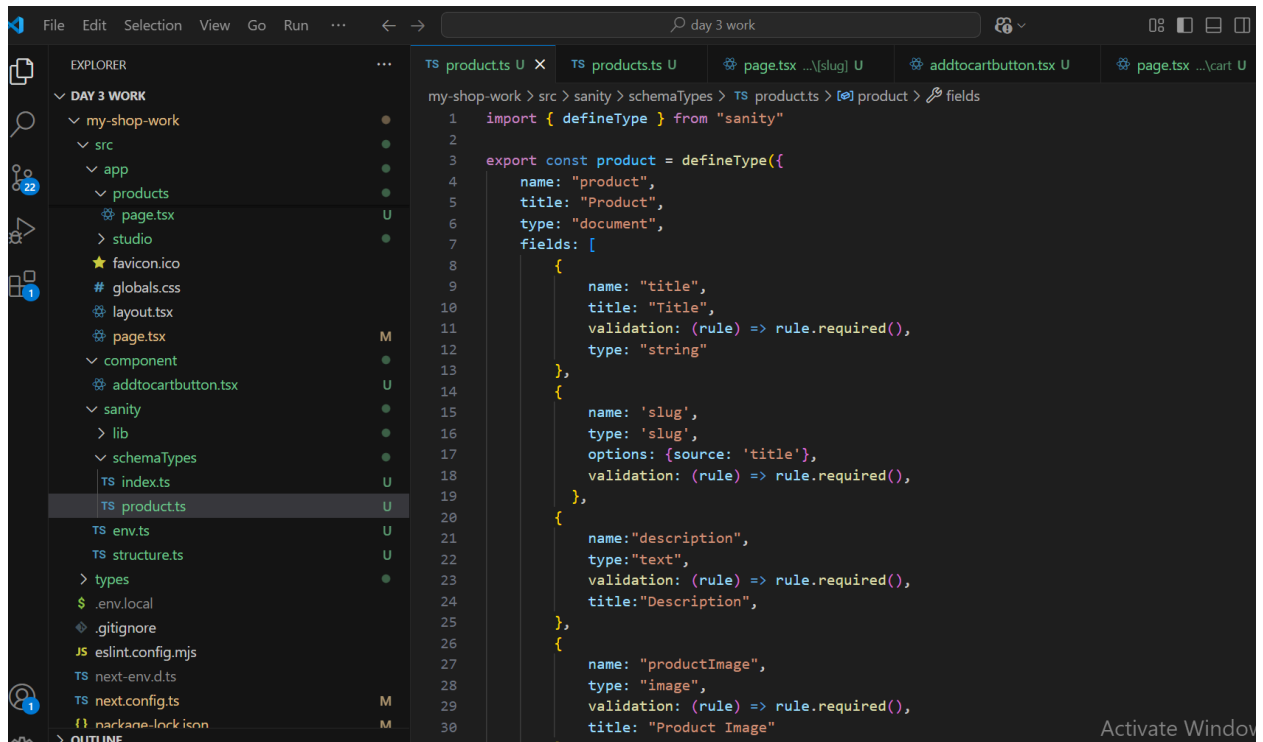
-  **Dynamic Product Listings** – Users can browse, filter, and search for products in real-time.
-  **Cart & Checkout** – Fully functional shopping cart with a smooth checkout process.
-  **API Integration** – Connected with backend services for product and order management.
-  **Responsive UI** – Optimized for both mobile and desktop users.
-  **Fast Performance** – Leveraging SSR & optimized builds for better load times.

Development Approach

- ◆ **Frontend Development** – Built reusable **Next.js components** with **Tailwind CSS** for scalability.
- ◆ **Backend Integration** – Utilized **Sanity CMS** for product and content management.
- ◆ **State Management** – Implemented best practices for managing app-wide state efficiently.
- ◆ **API Handling** – Fetched and processed data dynamically for a seamless user experience.

Screenshots & Demonstration



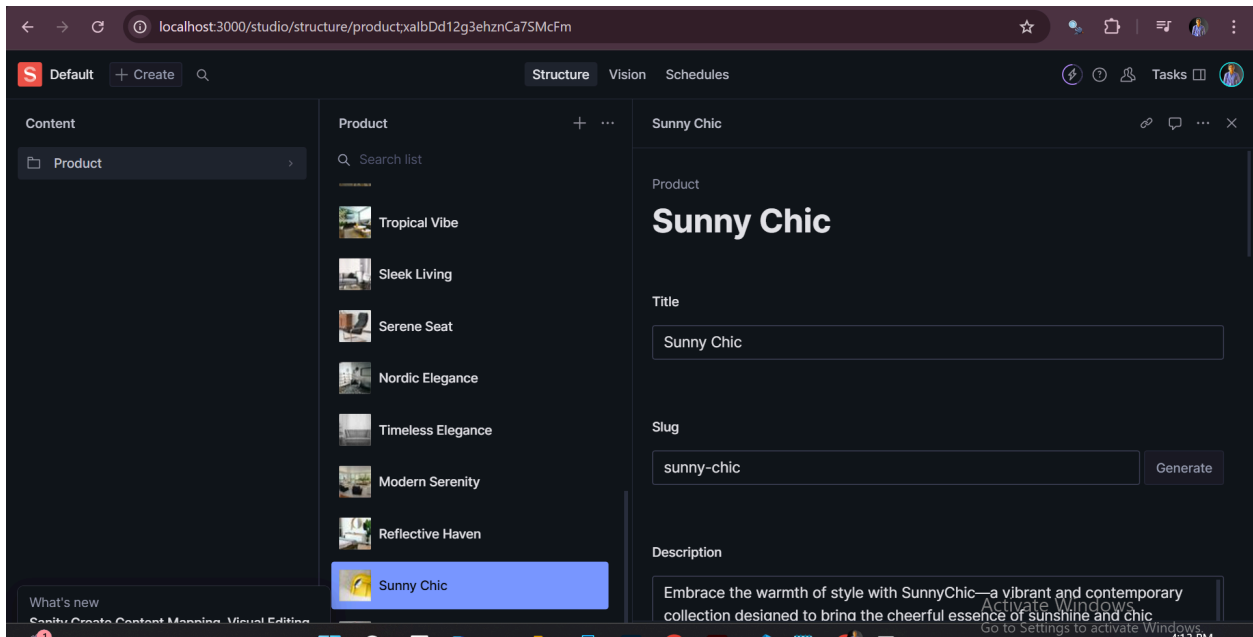
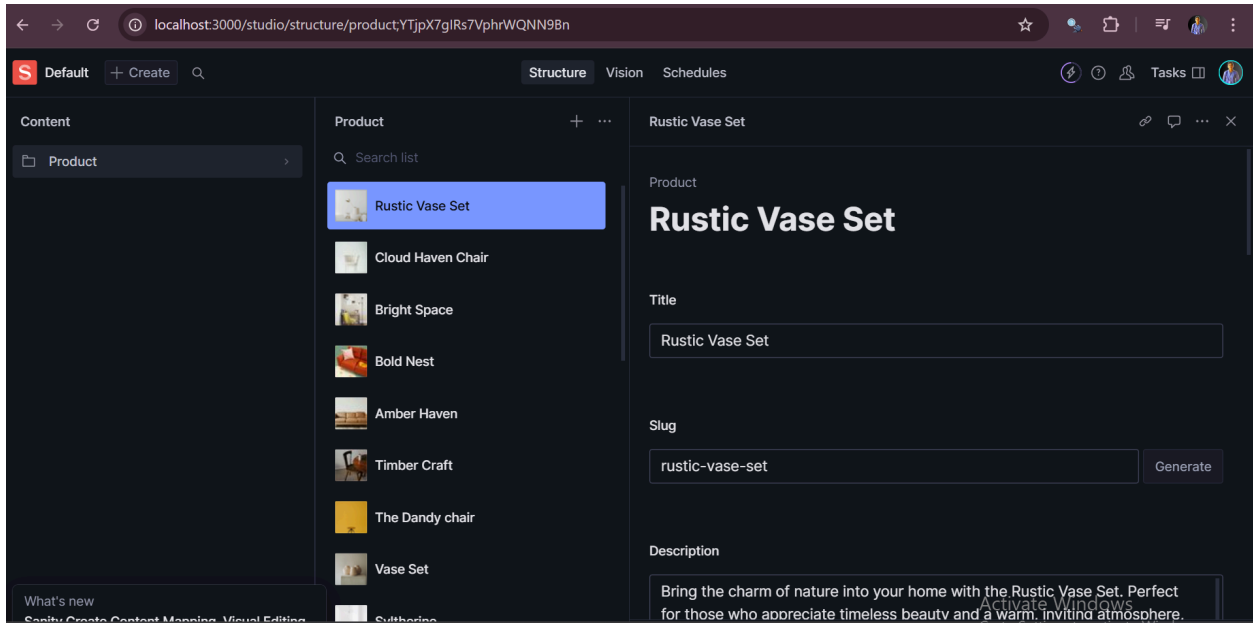


This screenshot shows the implementation of the `uploadImageToSanity` function in `importdata.mjs`. The function uses the `createClient` from `@sanity/client` to connect to a Sanity instance. It then uses `fetch` to upload the image to the Sanity API. The function returns the `asset._id` if the upload is successful, or throws an error if it fails.

```
1 import { createClient } from '@sanity/client';
2
3 const client = createClient({
4   projectId: 'yqf69zpp',
5   dataset: 'production',
6   useCdn: true,
7   apiVersion: '2025-01-13',
8   token: 'sky0uDmkbvlWrpOtstKY8I9dMUFEBP1UPN0jcwvfiA8dqXUQshPzA5gxrY9HUuc7NRHeND3wkFD30jrk1EwWZxM';
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   }
32 }
```

This screenshot shows the `package.json` file for the project. It includes the project name, version, private status, scripts, dependencies, and devDependencies.

```
1 {
2   "name": "my-shop-work",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "dev": "next dev --turbo",
7     "build": "next build",
8     "start": "next start",
9     "lint": "next lint",
10    "import-data": "node scripts/importdata.mjs"
11  },
12  "dependencies": {
13    "@sanity/image-url": "^1.1.0",
14    "@sanity/vision": "^3.70.0",
15    "next": "15.1.5",
16    "next-sanity": "^9.8.37",
17    "react": "^19.0.0",
18    "react-dom": "^19.0.0",
19    "sanity": "^3.70.0",
20    "styled-components": "^6.1.14",
21    "sweetalert2": "^11.15.10"
22  },
23  "devDependencies": {
24    "@eslint/eslintrc": "^3",
25    "@types/node": "^20",
26    "@types/react": "^19",
27    "@types/react-dom": "^19",
28    "eslint": "^9",
29    "eslint-config-next": "15.1.5",
30    "postcss": "^8"
31  }
32 }
```



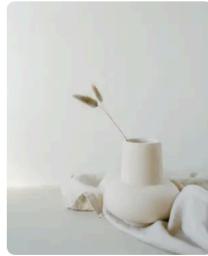
Latest Products



Timber Craft

Price: \$320

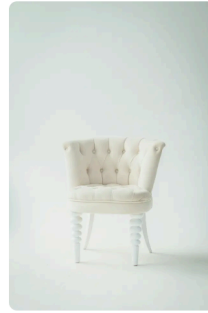
[View Details](#)



Rustic Vase Set

Price: \$210

[View Details](#)



Cloud Haven Chair

Price: \$230

[View Details](#)



Bright Space

Price: \$180

[View Details](#)

