

# *Day 3 Implementation Report*

## Furniro - Dynamic Online Furniture Marketplace

Hackathon Project Report - Marketplace Builder Hackathon 2025

### Project Overview

Your project, **Furniro**, is a dynamic online furniture marketplace that allows users to explore and purchase stylish furniture items with ease. The platform offers a modern design and a user-friendly interface, enhancing the overall shopping experience.

**Day 3** of the **Marketplace Builder Hackathon 2025** focuses on **API Integration and Data Migration**. The primary goal is to integrate product data from an external API into Sanity CMS for the project. This integration enables automatic population of product details such as images, titles, descriptions, prices, and tags, eliminating the need for manual data entry and making the process more efficient and scalable for the marketplace.

#### Key Objectives:

- **API Integration:** Connect the marketplace application with external APIs to fetch product data.
- **Data Migration:** Transfer and map the fetched data into Sanity CMS, ensuring that all product details are accurately populated.
- **Automation:** Set up processes to automatically update the CMS with new or updated product information from the external API.
- **Scalability:** Ensure that the integration can handle a growing amount of data efficiently.

By the end of Day 3, participants should have a system where product data is seamlessly integrated into Sanity CMS from external sources, laying a robust foundation for managing dynamic content in the marketplace.

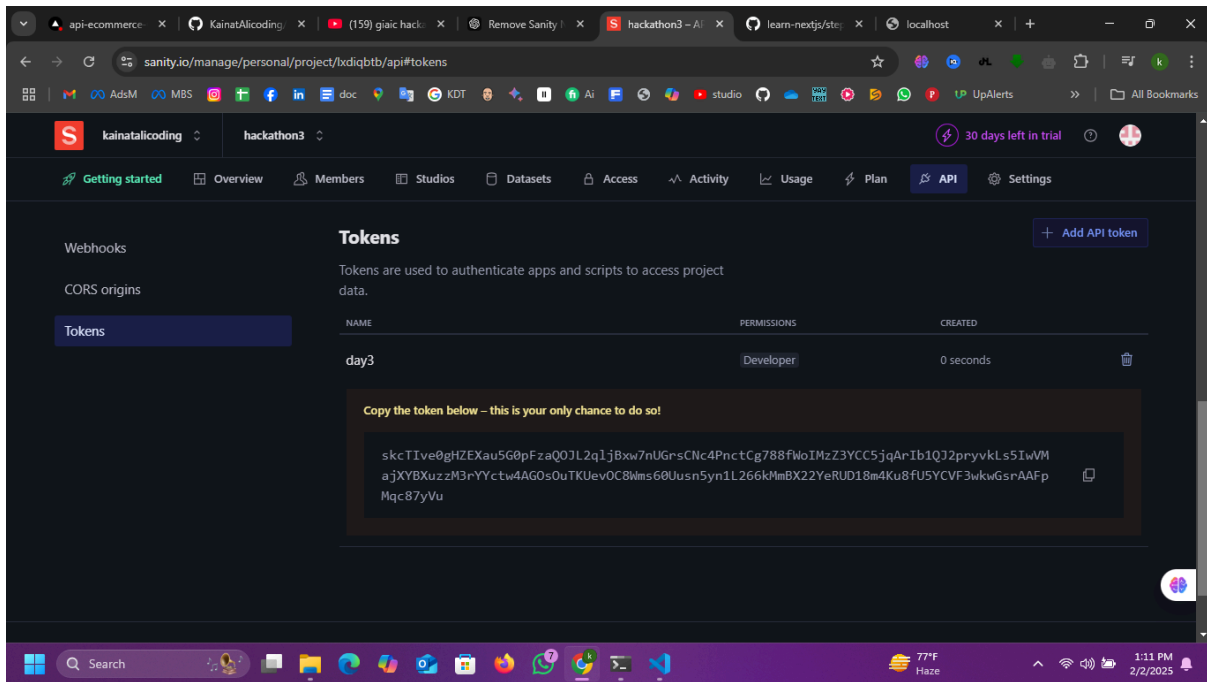
# Sanity integration

The screenshot shows the Sanity.io dashboard for a project named 'hackathon3'. The user 'kainatalcoding' is logged in, with a 30-day trial period remaining. The dashboard includes a sidebar with navigation links: Getting started, Overview (selected), Members, Studios, Datasets, Access, Activity, Usage, Plan, API, and Settings. The main content area is divided into two sections: 'Next steps' and 'Project members'. The 'Next steps' section contains a card titled 'Initialize your project with the CLI' with the command `npm create sanity@latest -- --project lxdqbtb --dataset produ` and a 'Copy' button. The 'Project members' section shows a list of team members and an 'Invite project members' button. The bottom of the dashboard shows a 'Usage' section with a 'View more' link.

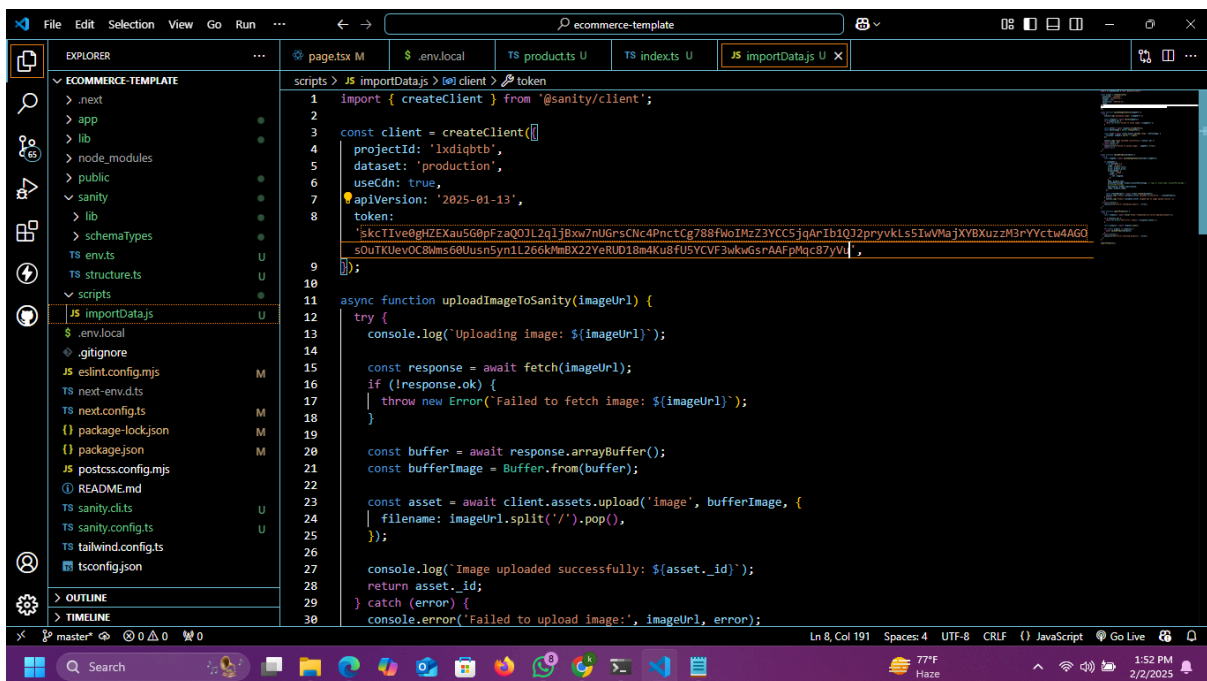
The screenshot shows the 'API' section of the Sanity.io dashboard. It includes a sidebar with navigation links: Getting started, Overview, Members, Studios, Datasets, Access, Activity, Usage, Plan, API (selected), and Settings. The main content area is divided into two sections: 'Webhooks' and 'Tokens'. The 'Webhooks' section includes a 'CORS origins' table and a 'Tokens' section. The 'CORS origins' table lists the origins that can connect to the project API, with columns for 'ORIGIN', 'CREDENTIALS', and 'CREATED'. The 'Tokens' section includes a 'Tokens' table and a 'Tokens' section.

ORIGIN	CREDENTIALS	CREATED
http://localhost:3000	Allowed	just now
http://localhost:3333	Allowed	2 hours

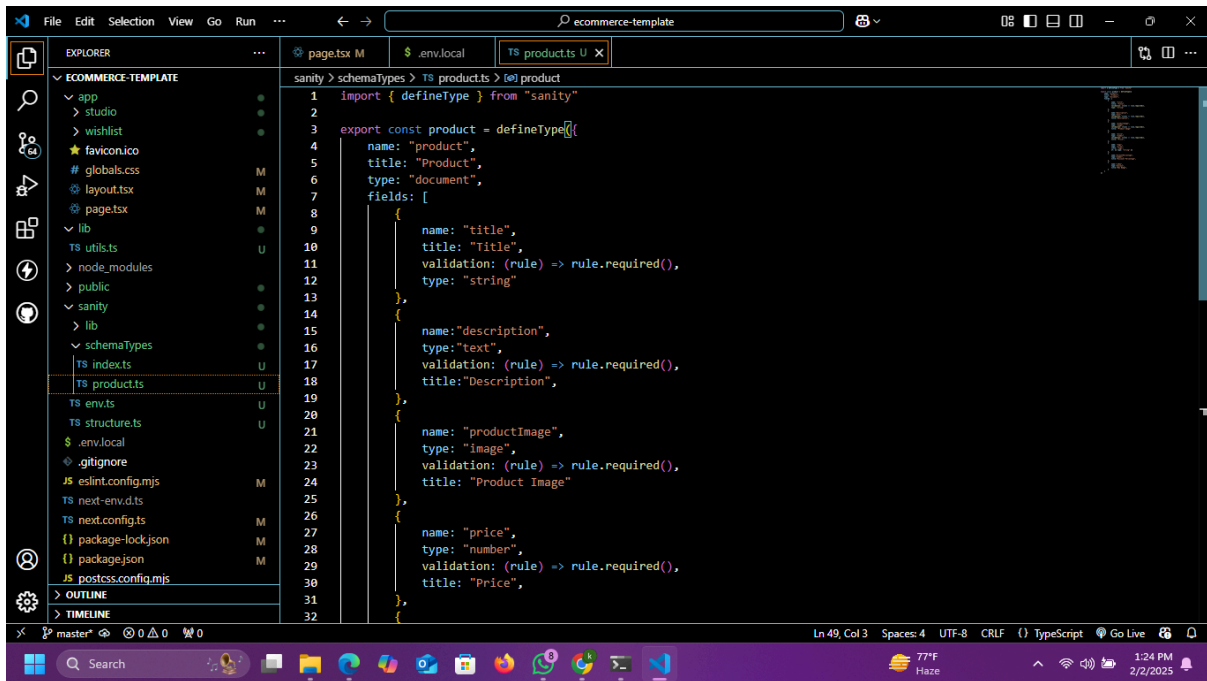
# TOKEN



## API Integration Import data



# Products

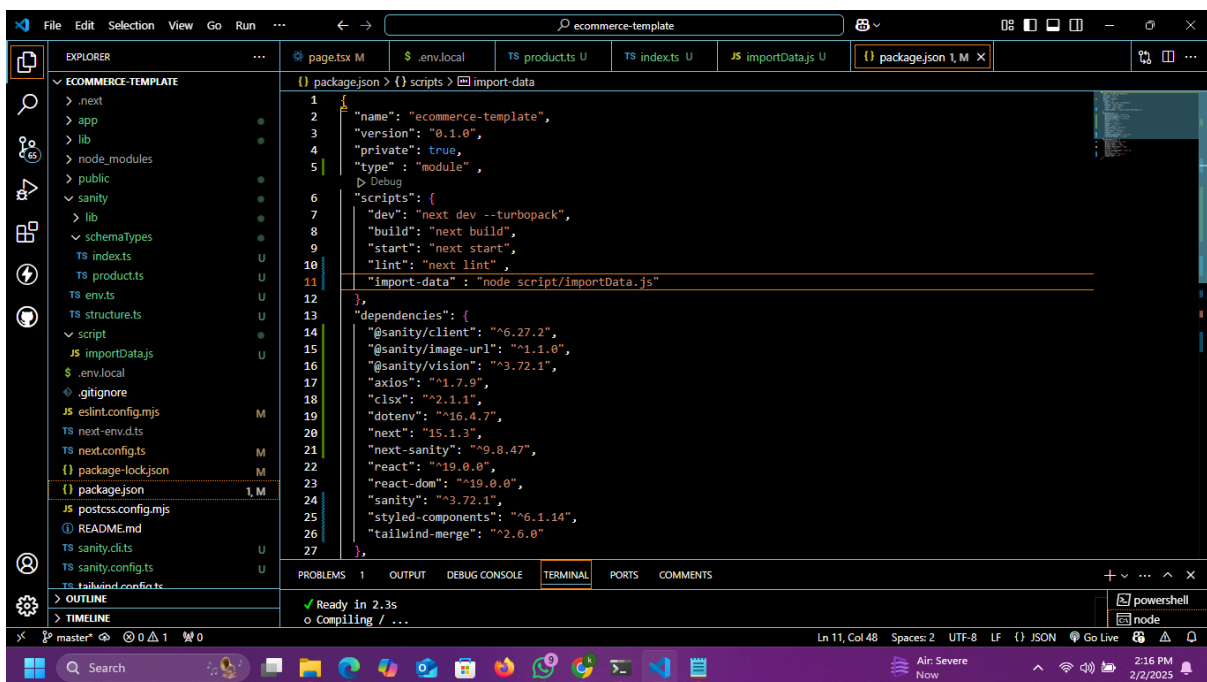


The screenshot shows the VS Code editor with the 'product.ts' file open. The file defines a Sanity schema for a product. The Explorer sidebar on the left shows the project structure, including 'app', 'lib', 'sanity', and 'schemaTypes'. The main editor area displays the following TypeScript code:

```
1 import { defineType } from "sanity"
2
3 export const product = defineType({
4   name: "product",
5   title: "Product",
6   type: "document",
7   fields: [
8     {
9       name: "title",
10      title: "Title",
11      validation: (rule) => rule.required(),
12      type: "string"
13    },
14    {
15      name: "description",
16      title: "Description",
17      type: "text",
18      validation: (rule) => rule.required()
19    },
20    {
21      name: "productImage",
22      title: "Product Image",
23      type: "image",
24      validation: (rule) => rule.required()
25    },
26    {
27      name: "price",
28      title: "Price",
29      type: "number",
30      validation: (rule) => rule.required()
31    }
32  ]
33 })
```

The status bar at the bottom indicates the file is at line 49, column 3, with 3 spaces and UTF-8 encoding. The system tray shows the date and time as 2/2/2025, 1:24 PM.

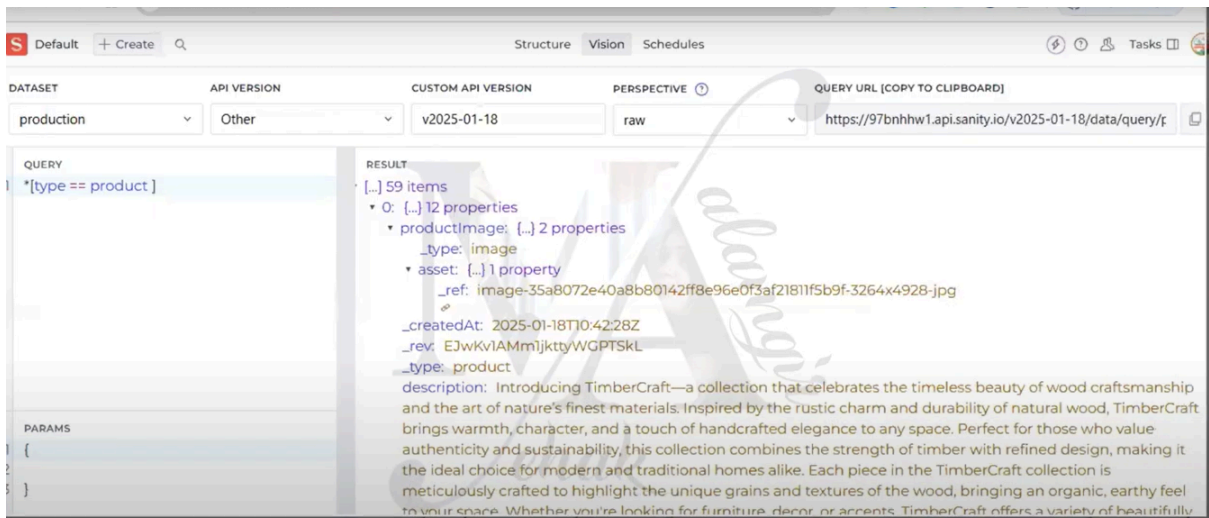
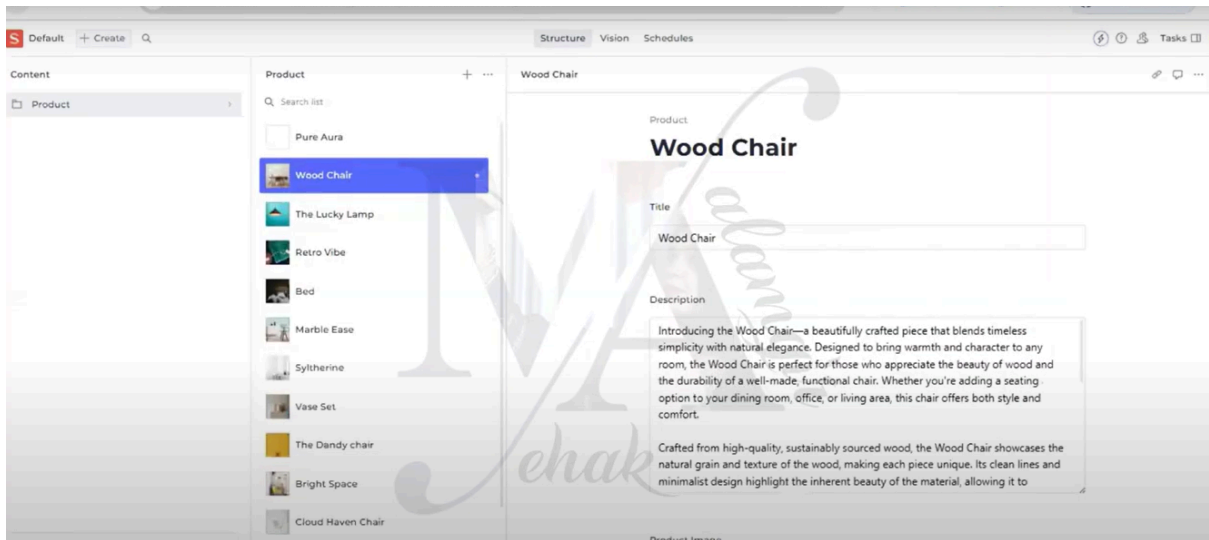
## Changes in package json file



The screenshot shows the VS Code editor with the 'package.json' file open. The file contains the following JSON content:

```
1 {
2   "name": "e-commerce-template",
3   "version": "0.1.0",
4   "private": true,
5   "type": "module",
6   "scripts": {
7     "dev": "next dev --turbo",
8     "build": "next build",
9     "start": "next start",
10    "lint": "next lint",
11    "import-data": "node script/importData.js"
12  },
13  "dependencies": {
14    "@sanity/client": "^6.27.2",
15    "@sanity/image-url": "^1.1.0",
16    "@sanity/vision": "^3.72.1",
17    "axios": "^1.7.9",
18    "clsx": "^2.1.1",
19    "dotenv": "^16.4.7",
20    "next": "15.1.3",
21    "next-sanity": "^9.8.47",
22    "react": "^19.0.0",
23    "react-dom": "^19.0.0",
24    "sanity": "^3.72.1",
25    "styled-components": "^6.1.14",
26    "tailwind-merge": "^2.6.0"
27  },
28}
```

The Explorer sidebar on the left shows the project structure, including 'package.json'. The main editor area displays the JSON content. The status bar at the bottom indicates the file is at line 11, column 48, with 2 spaces and UTF-8 encoding. The system tray shows the date and time as 2/2/2025, 2:16 PM.



Products integrate in fronthand

