DAY 3- API INTEGRATION REPORT- [FURNITURE]

Overview:

This document provides a detailed report of the API integration process for the E-commerce Furniture platform. The integration process was carried out using Sanity, a headless CMS, to streamline content management and improve the dynamic functionality of the website.

Purpose of API Integration:

The primary objectives of this API integration include:

- Enabling dynamic and efficient content updates for the furniture catalog.
- Improving data synchronization between the front-end and back-end systems.
- Ensuring seamless scalability to accommodate a growing inventory.
- Enhancing the user experience through real-time data updates.

Tools and Technologies Used:

- **Sanity CMS**: A headless content management system used as the primary backend for storing and managing furniture catalog data.
- **Next.js**: The front-end framework utilized for rendering dynamic content and pages.
- **Tailwind CSS**: For styling the front end.
- **REST APIs**: Provided by Sanity for data querying and manipulation.
- **React Query**: For efficient data fetching, caching, and synchronization.

API Integration Process:

1. Setting Up Sanity

1. **Project Initialization**:
    - A new project was created in the Sanity dashboard.
    - Schemas were defined to represent the structure of furniture data, including fields such as product name, description, category, price, tags, and images.
2. **Dataset Configuration**:
    - Configured a dataset (e.g., `production`) to store all data.
    - Enabled public read access for non-sensitive data, ensuring secure data queries.

3. **GROQ Queries**:

    - Utilized Sanity's Graph-Relational Object Queries (GROQ) to fetch data efficiently.

2. Backend API Development

1. **API Token Configuration**:

    - Generated API tokens with specific roles (e.g., read-only) for secure access.
    - Stored the tokens securely using environment variables in the Next.js project.

2. **Data Fetching Logic**:

    - Created utility functions to fetch data from Sanity's API endpoints.
    - Example of a GROQ query for fetching furniture products:

    ```
    const query = *[_type == "product"] { _id, title,

     price,

     description,

    discountPercentage,

    "imageUrl": productImage.asset->url,

    tags };
    ```

3. **Pagination and Filtering**:

    - Implemented server-side logic to handle pagination, filtering, and sorting of products.

4. Frontend Integration:

1. **Dynamic Content Rendering**:

    - Integrated data fetched from Sanity into React components.
    - Ensured that the data dynamically updated across the platform when changes were made in Sanity.

2. **Optimized Image Delivery**:

- Leveraged Sanity's built-in Image API for image transformation and optimization.

3. **Real-Time Updates**:

- Configured the front end to subscribe to real-time updates using Sanity's Webhooks, ensuring data reflects changes instantly.

Migration Steps and Tools Used:

1. **Data Export from Legacy System**:

   - Extracted data from the legacy CMS or database in JSON format.
   - Reviewed and cleaned data to ensure consistency and remove duplicates.

2. **Data Transformation**:

   - Used scripts in Node.js to map legacy data to Sanity's schema structure.
   - Verified data against the schema to ensure proper formatting and completeness.

3. **Data Import into Sanity**:

   - Utilized Sanity's CLI tool (`sanity dataset import`) to upload the cleaned and transformed data into the configured dataset.
   - Validated the imported data by performing sample queries in Sanity's Vision tool.

4. **Tooling and Automation**:

   - Used tools such as Postman for testing API responses during migration.
   - Automated repetitive tasks with custom scripts for batch processing and validation.

5. **Final Validation**:

   - Conducted comprehensive testing to ensure the migrated data was accurately displayed on the front end.

Conclusion:

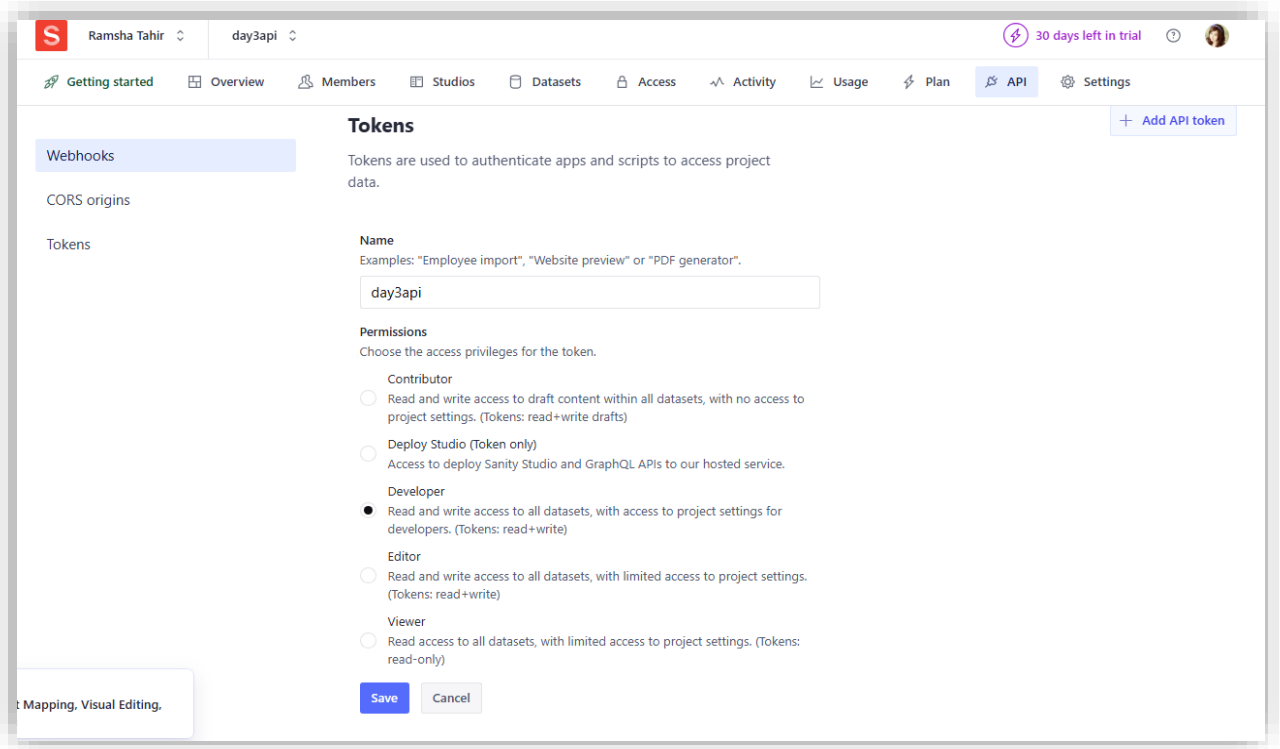
The API integration using Sanity for the E-commerce Furniture platform was successfully implemented. This integration has enhanced the overall functionality and scalability of the platform while providing a seamless user experience. Future improvements may include integrating additional third-party APIs and further optimizing the data delivery pipeline.

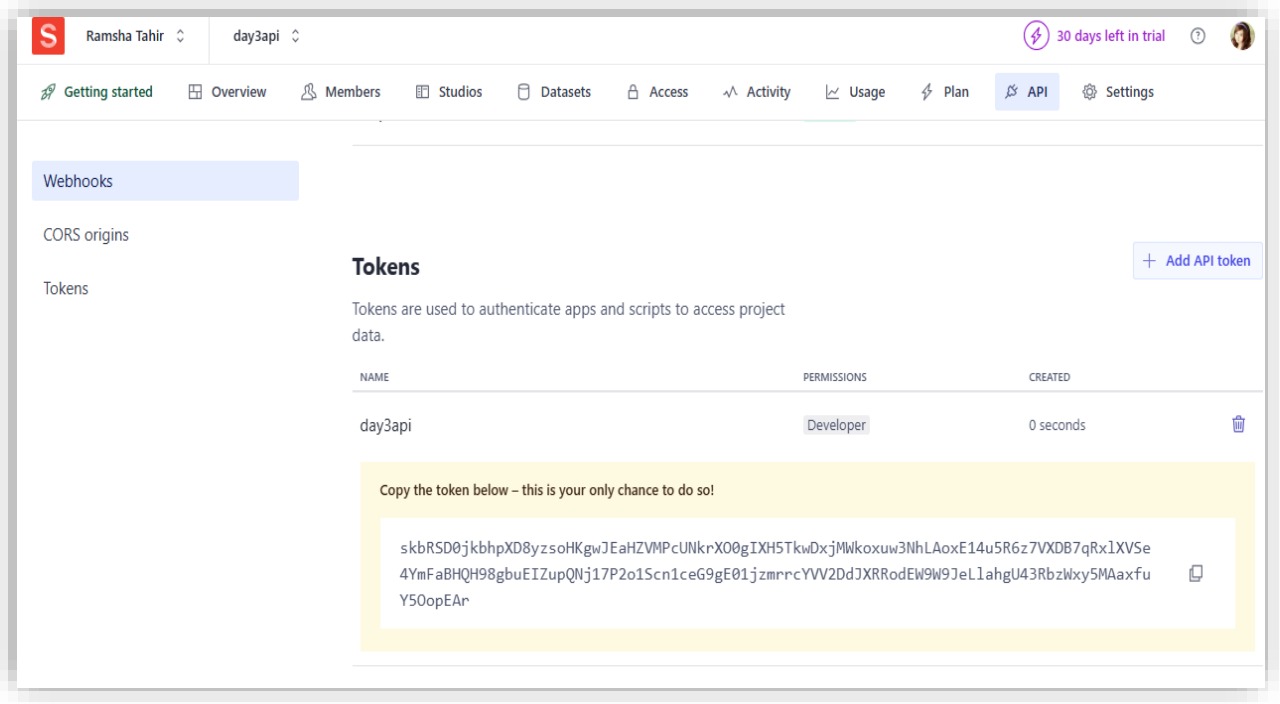
SCREENSHOTS:

SETTING UP SANITY

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.5247]
(c) Microsoft Corporation. All rights reserved.

E:\Next.js\Hackathon 2\e-commerce-furniture>code .

E:\Next.js\Hackathon 2\e-commerce-furniture>npm create sanity@latest
Need to install the following packages:
create-sanity@3.71.1
Ok to proceed? (y) y
 You are logged in as ramshatahir23@gmail.com using Google
 Fetching existing projects

? Create a new project or select an existing one Create new project
? Your project name: day3api
Your content will be stored in a dataset that can be public or private, depending on
whether you want to query your content with or without authentication.
The default dataset configuration has a public dataset named "production".
? Use the default dataset configuration? Yes
 Creating dataset
? Would you like to add configuration files for a Sanity project in this Next.js folder? Yes
? Do you want to use TypeScript? Yes
? Would you like an embedded Sanity Studio? Yes
? What route do you want to use for the Studio? /studio
? Select project template to use Clean project with no predefined schema types
? Would you like to add the project ID and dataset to your .env.local file? Yes
Added http://localhost:3000 to CORS origins
Running 'npm install --legacy-peer-deps --save @sanity/vision@3 sanity@3 @sanity/image-url@1 styled-components@6'

added 913 packages, and audited 1298 packages in 3m

244 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

added 9 packages, and audited 1307 packages in 17s

244 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

Success! Your Sanity configuration files has been added to this project
```

API TOKEN GENERATION





Prepared by *Ramsha Fawad,* Q2, GIAIC Student.

DOTENV.LOCAL FILE:

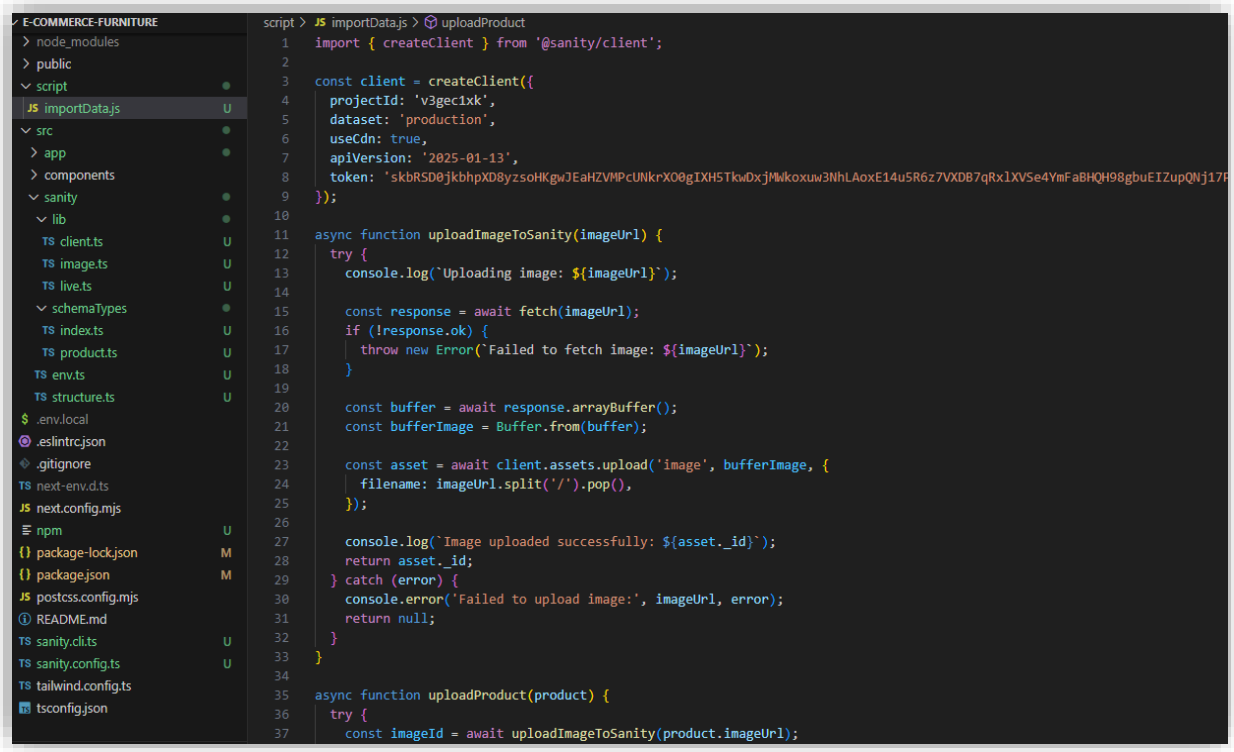
```
$ .env.local
1   NEXT_PUBLIC_SANITY_PROJECT_ID="v3gec1xk"
2   NEXT_PUBLIC_SANITY_DATASET="production"
3   SANITY_API_TOKEN="skbRSD0jkbhpXD8yzsoHKgwJEaHZVMPcUNkrXO0gIXH5TkwDxjMWkoxuw3NhLAoxE14u5R6z7VXDB7qRxlXVSe4YmFaBHQH98
4
```

ADJUSTEMENT MADE IN SCHEMA:

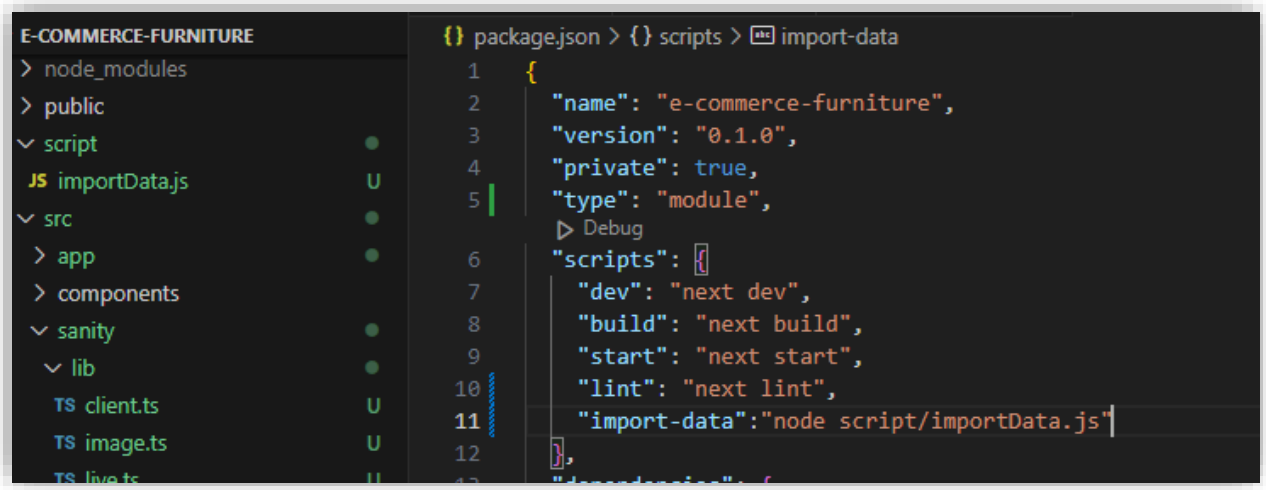
src > sanity > schemaTypes > TS product.ts > product

```ts
import { defineType } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        {
            name: "title",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        },
        {
            name:"description",
            type:"text",
            validation: (rule) => rule.required(),
            title:"Description",
        },
        {
            name: "productImage",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        },
        {
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        },
        {
            name: "tags",
            type: "array",
            title: "Tags",
            of: [{ type: "string" }]
        },
        {
            name:"dicountPercentage",
```

importData.js File:

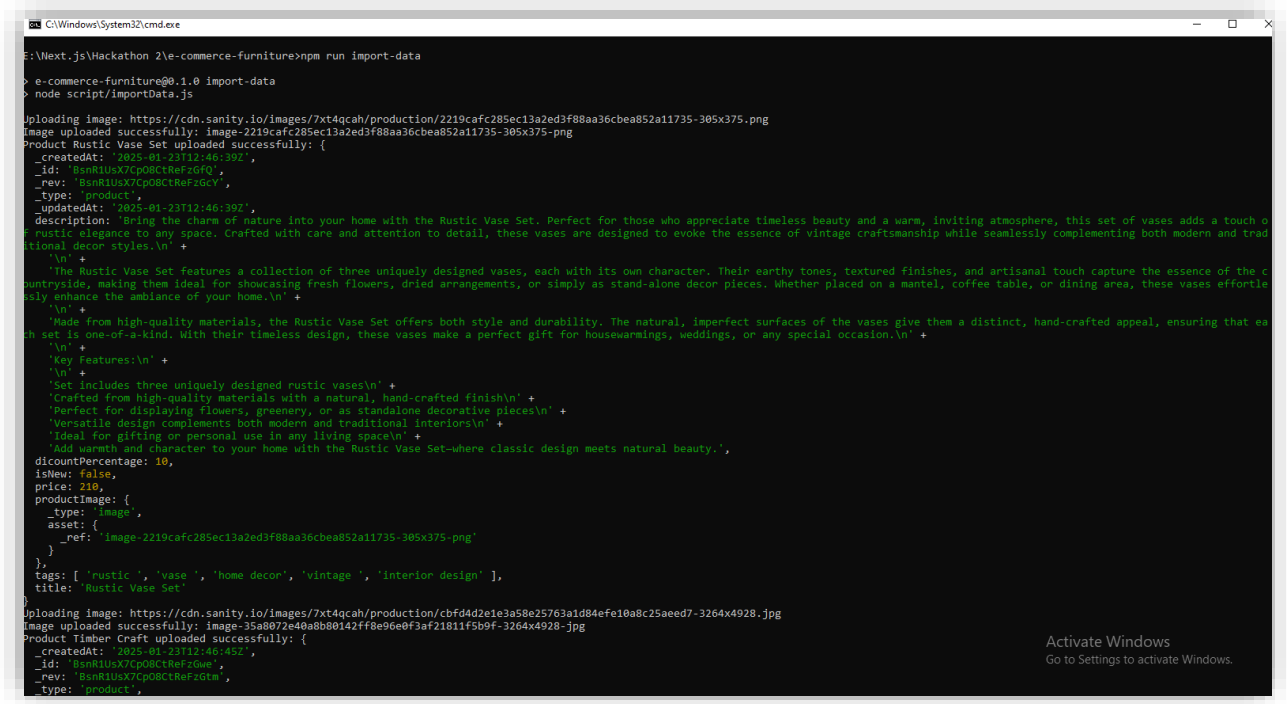


package.json file:



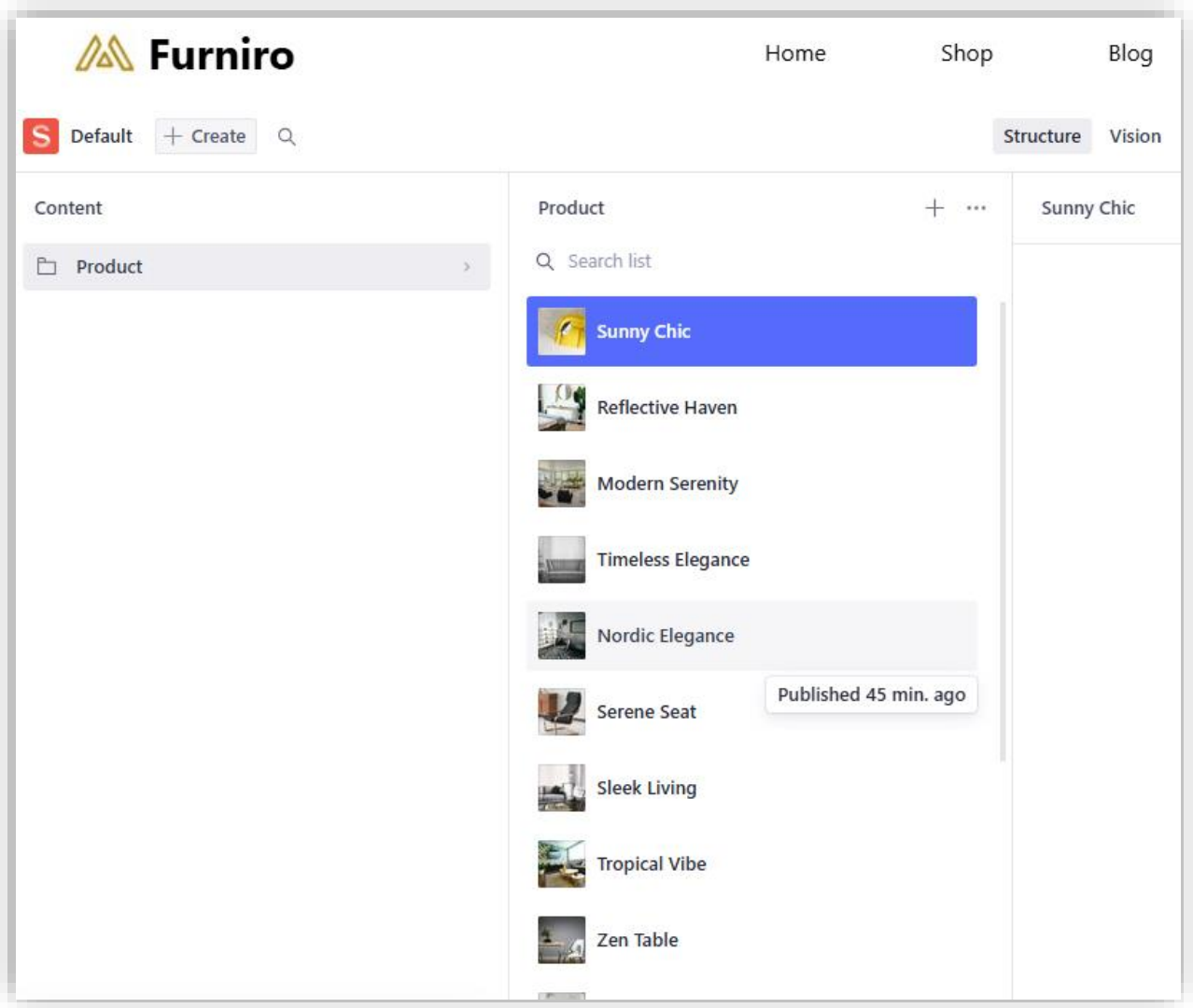
Prepared by *Ramsha Fawad,* Q2, GIAIC Student.

PRODUCT FETCHING IN TERMINAL:

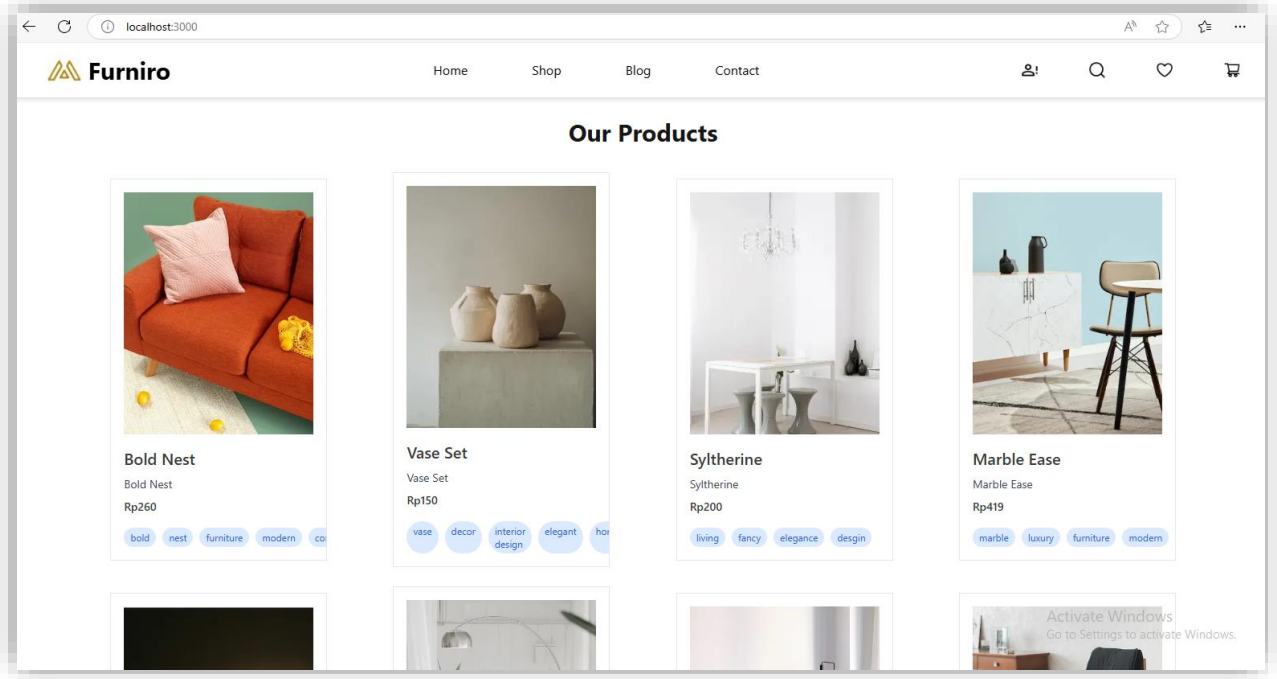


Prepared by *Ramsha Fawad,* Q2, GIAIC Student.

POPULATED SANITY FIELD:



FRONTEND DISPLAY OF DATA:



Prepared by *Ramsha Fawad,* Q2, GIAIC Student.

SELF VALIDATION CHECKLIST:

API Understanding	✓
Schema Validation	✓
Data Migration	✓
API Integration in Next.js	✓
Submission Preparation	✓