Day 3 - API Integration Report - Avion

API Integration:

Step 01: API Endpoint Setup

- API Endpoint: https://hackathon-apis.vercel.app/api/products
- The API provides data for fields like: name, description, image, _id, features, dimensions, width, height, depth, category, slug, price, tags
- Axios was used to fetch the data from the given API.

Step 02: Fetch Data From The API

Code Snippets:

```
    try {
      // Fetch the image from the URL and convert it to a buffer
      const response = await axios.get(imageUrl, { responseType: 'arraybuffer',timeout: 10000 });
      const buffer = Buffer.from(response.data);

      // Upload the image to Sanity
      const asset = await client.assets.upload('image', buffer, {
        filename: imageUrl.split('/').pop(), // Extract the filename from URL
      });

      // Debugging: Log the asset returned by Sanity
      console.log('Image uploaded successfully:', asset);

      return asset._id; // Return the uploaded image asset reference ID
    } catch (error) {
      console.error('❌ Failed to upload image:', imageUrl, error);
      return null
      //throw error;
    }
  }
```

Failed to connect to Pieces OS. Please en
running and up to date. If the issue persi

Changes Made To Schemas:

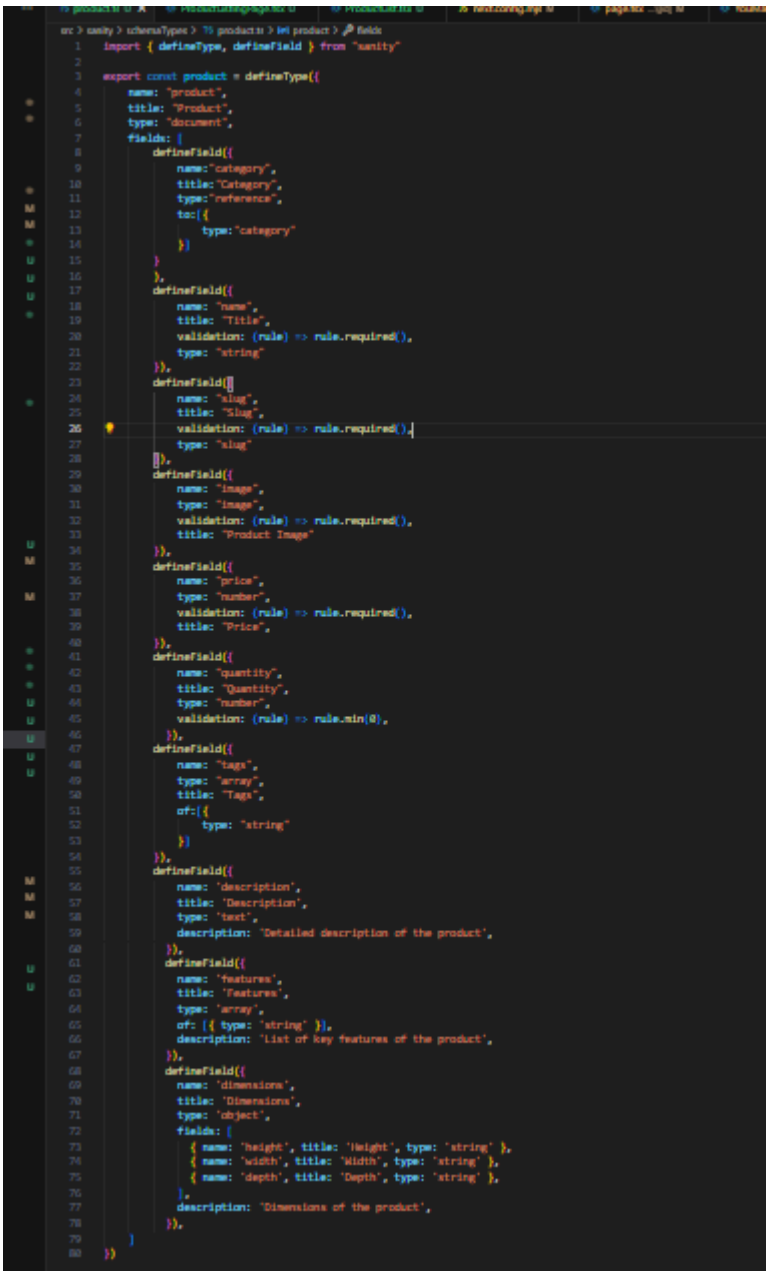
Product Schema Modification:

Updation Of Fields:

Added The following fields in schema:

- Features
- Dimensions

Final Schema:



```
import { defineType, defineField } from "sanity"

export const product = defineType({
    name: "product",
    title: "Product",
    type: "document",
    fields: [
        defineField({
            name:"category",
            title:"Category",
            type:"reference",
            to:[{
                type:"category"
            }]
        }
        ),
        defineField({
            name: "name",
            title: "Title",
            validation: (rule) => rule.required(),
            type: "string"
        }),
        defineField({
            name: "slug",
            title: "Slug",
            validation: (rule) => rule.required(),
            type: "slug"
        }),
        defineField({
            name: "image",
            type: "image",
            validation: (rule) => rule.required(),
            title: "Product Image"
        }),
        defineField({
            name: "price",
            type: "number",
            validation: (rule) => rule.required(),
            title: "Price",
        }),
        defineField({
            name: "quantity",
            title: "Quantity",
            type: "number",
            validation: (rule) => rule.min(0),
          }),
        defineField({
            name: "tags",
            type: "array",
            title: "Tags",
            of:[{
                type: "string"
            }]
        }),
        defineField({
            name: 'description',
            title: 'Description',
            type: 'text',
            description: 'Detailed description of the product',
          }),
          defineField({
            name: 'features',
            title: 'Features',
            type: 'array',
            of: [{ type: 'string' }],
            description: 'List of key features of the product',
          }),
          defineField({
            name: 'dimensions',
            title: 'Dimensions',
            type: 'object',
            fields: [
              { name: 'height', title: 'Height', type: 'string' },
              { name: 'width', title: 'Width', type: 'string' },
              { name: 'depth', title: 'Depth', type: 'string' },
            ],
            description: 'Dimensions of the product',
          }),
    ]
})
```

Migration Steps and Tools Used

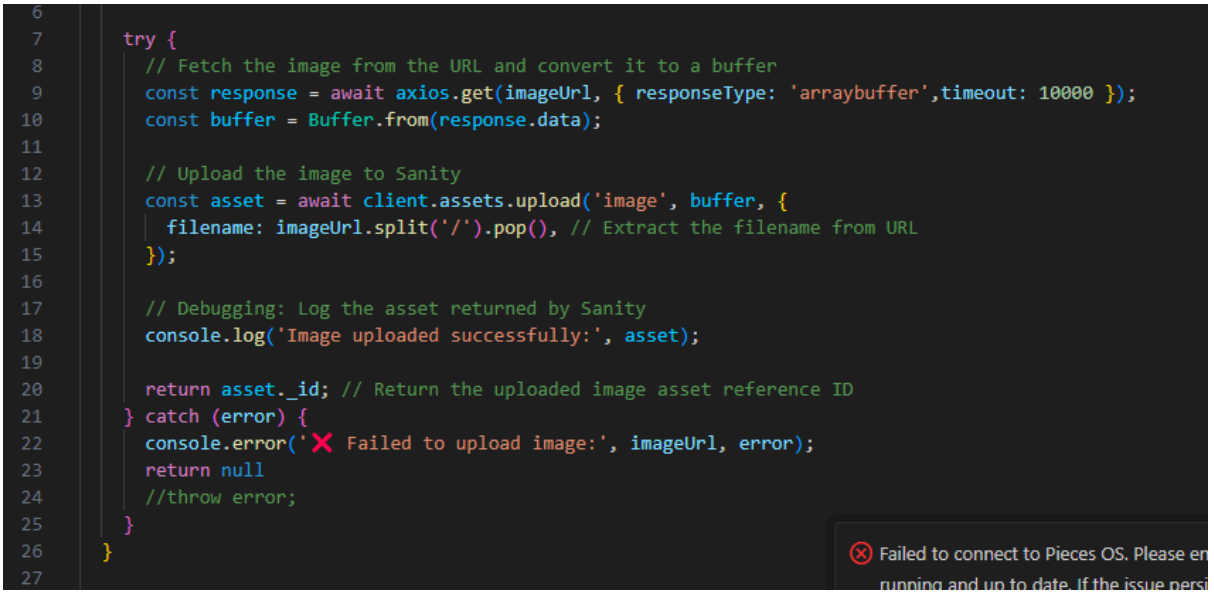
Migration Steps

1. **Environment Setup:**
    - Installed required dependencies: `next-themes`, `@sanity/client`, `axios`, `dotenv`, and any other necessary packages.
    - Created and configured a `.env.local` file to store secure environment variables, such as API keys and other credentials.

2. **Data Fetching:**
    - Retrieved data from an external API or local database, depending on project setup.
    - Utilized `axios` to fetch relevant product or user data and logged the data to verify its structure and integrity.
    - API endpoint: https://hackathon-apis.vercel.app/api/products

3. **Image Upload:**
    - Downloaded images from API response.
    - Uploaded images to an external storage service or database (e.g., Sanity Asset Manager) using the appropriate client SDK.

    Code snippet for image upload:



```
try {
  // Fetch the image from the URL and convert it to a buffer
  const response = await axios.get(imageUrl, { responseType: 'arraybuffer',timeout: 10000 });
  const buffer = Buffer.from(response.data);

  // Upload the image to Sanity
  const asset = await client.assets.upload('image', buffer, {
    filename: imageUrl.split('/').pop(), // Extract the filename from URL
  });

  // Debugging: Log the asset returned by Sanity
  console.log('Image uploaded successfully:', asset);

  return asset._id; // Return the uploaded image asset reference ID
} catch (error) {
  console.error('❌ Failed to upload image:', imageUrl, error);
  return null
  //throw error;
}
}
```


4. Document Creation:

- Created necessary documents (e.g., product listings, user profiles) within the database or content management system (e.g., Sanity).
- Combined API data with uploaded assets (e.g., image URLs) to construct and create records within the platform.

5. Testing and Debugging:

- Thoroughly tested the data fetching, image uploads, and document creation to ensure the end-to-end process worked smoothly.

- Addressed any issues with the API response, data format, or image handling during testing.

6. Final Integration:

- Integrated the migration with the rest of the project to ensure seamless interaction between the frontend and backend.
- Verified data consistency and accuracy, ensuring the migration aligned with the project's functionality.

Tools Used

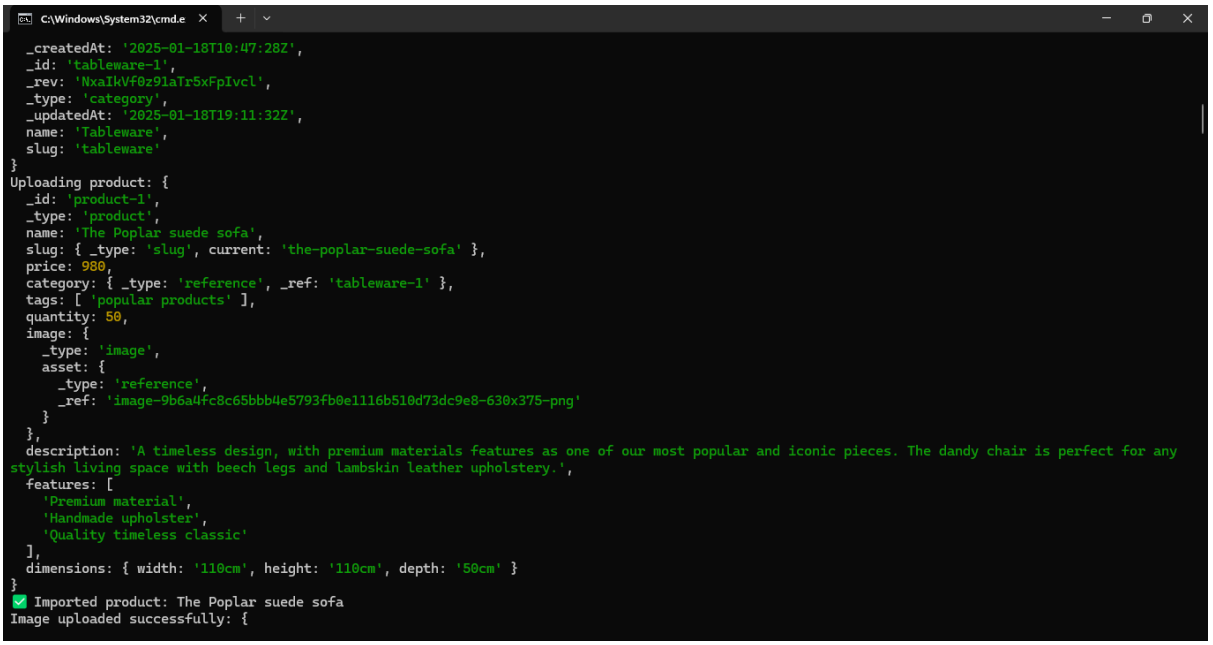
- **Next.js:** Framework for building the web application.
- **Sanity:** Headless CMS for managing product or user data.
- **Axios:** HTTP client for fetching data from external APIs.
- **dotenv:** Environment variables management.
- **Sanity Client:** For interacting with the Sanity CMS and uploading assets.

Code Snippet:

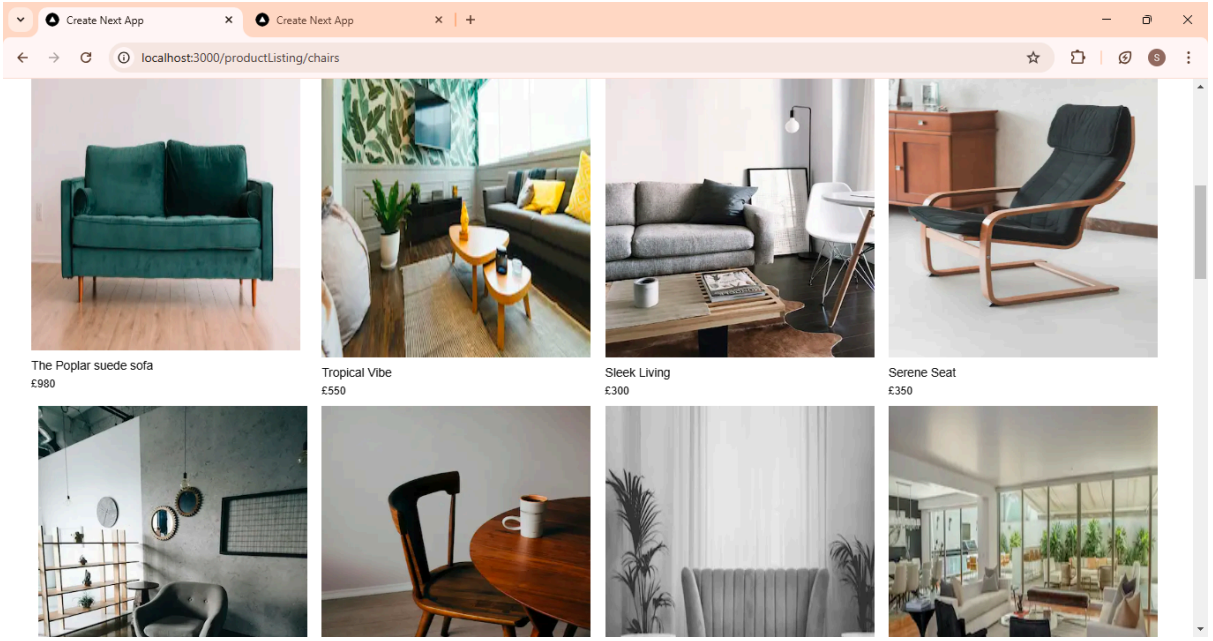
```
const sanityProduct = {
  _id: `product-${counter}`, // Prefix the ID to ensure validity
  _type: 'product',
  name: product.name,
  slug: {
    _type: 'slug',
    current: slugify(product.name || 'default-product', {
      lower: true, // Ensure the slug is lowercase
      strict: true, // Remove special characters
    }),
  },
  price: product.price,
  category:{
    _type: 'reference',
    _ref:catRef?catRef:undefined
  },
  tags: product.tags?product.tags:[],
  quantity:50,
  image: imageRef ? {
    _type: 'image',
    asset: {
      _type: 'reference',
      _ref: imageRef, // Set the correct asset reference ID
    },
  } : undefined,
  description: product.description?product.description: "A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is
  features: product.features?product.features: [
    "Premium material",
    "Handmade upholstery",
    "Quality timeless classic",
  ],
  dimensions: product.dimensions?product.dimensions : {
    _type: 'dimensions', // Custom object type for dimensions
    height: "110cm",
    width: "75cm",
    depth: "50cm",
  }
};
```

API Call Output:

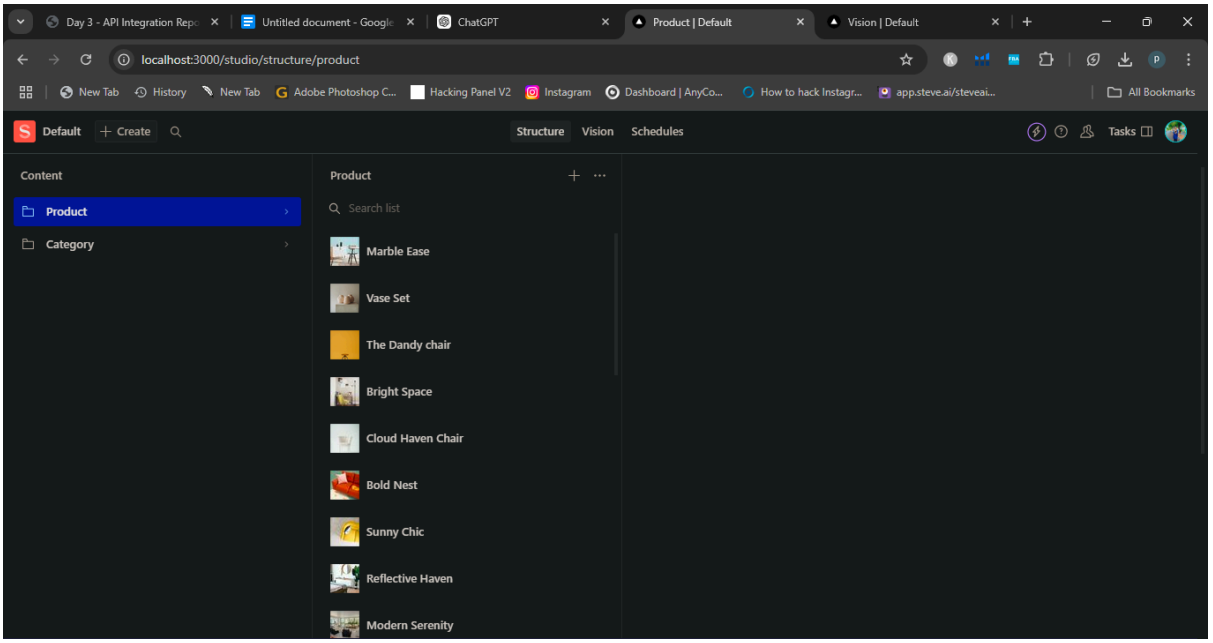
Screenshots:



Frontend Display:



Sanity CMS:



Final Checklist:

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