Day 3 - API Integration Report

API Integration Process

I integrated the API for Avion, a marketplace, by connecting the backend REST APIs to the frontend using Next.js and Sanity.

1. **APIs Used**:
   - **Products API**:
     Endpoint: hackathon-template02/schema/product.ts at main · bilalmk/hackathon-template02 · GitHub
   - **Categories API**:
     Endpoint: hackathon-template02/schema/category.ts at main · bilalmk/hackathon-template02 · GitHub
2. **Steps Taken**:
   - Fetched data from the APIs using `fetch` in Next.js.
   - Populated the data into Sanity Studio for content management.

Adjustments Made to Schemas

Product Schema - Fields

1. **Category**
2. **Name**
3. **Slug**
4. **Image**
5. **Price**
6. **Quantity**
7. **Tags**
8. **Description**
9. **Features**
10. **Dimensions**
    - Height
    - Width
    - Depth

Category Schema -Fields

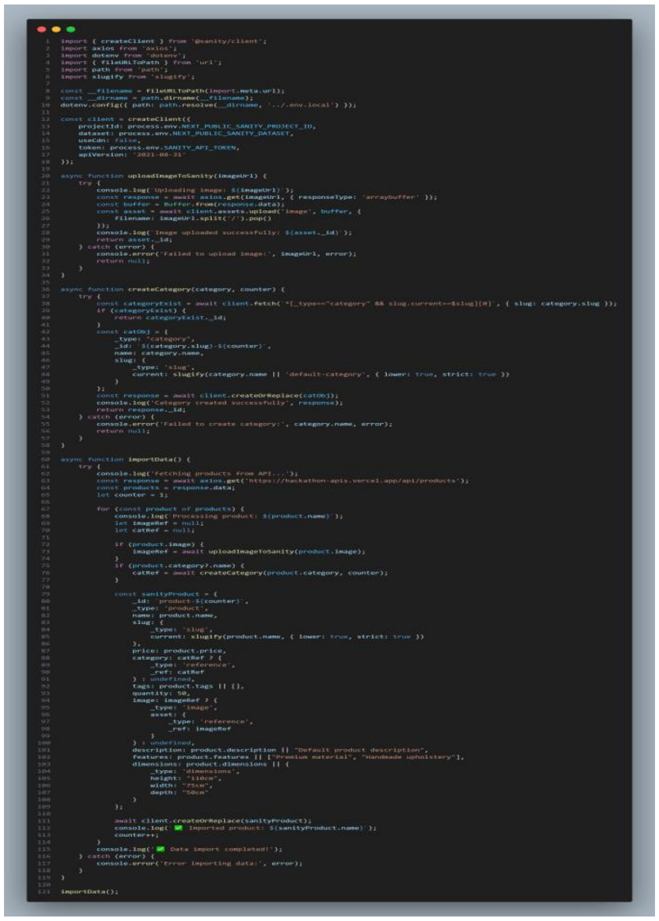
1. **Name**
2. **Slug**

These schemas were added to the `/src/sanity/schemaTypes` directory and imported into the `index.ts` file for use.

Migration Steps and Tools Used

1. **Environment Setup**:
    - Added sensitive data like API keys in a `.env` file:
    - `NEXT_PUBLIC_SANITY_PROJECT_ID="<Project ID>"`
    - `NEXT_PUBLIC_SANITY_DATASET="production"`
    - `NEXT_PUBLIC_SANITY_AUTH_TOKEN="<Auth Token>"`
2. **Migration Script**:
    - Created an `importSanitData.mjs` file in the `/scripts` folder.
    - Used the provided script to automate data transfer to Sanity.
3. **Tools Used**:
    - **dotenv**: To manage environment variables securely.
    - **npm**: To install and run dependencies.
4. **Execution**:
    - Installed the required packages using:
    - `npm install dotenv`
    - Ran the migration script with:
    - `npm run importSanitData.mjs`

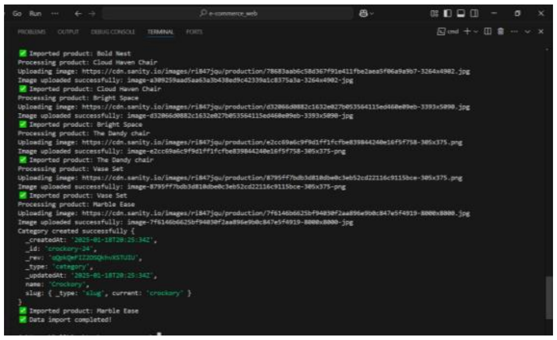
Code Snippets



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

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

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'
});

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}`);
        return asset._id;
    } catch (error) {
        console.error('Failed to upload image:', imageUrl, error);
        return null;
    }
}

async function createCategory(category, counter) {
    try {
        const categoryExist = await client.fetch(`*[_type=="category" && slug.current==$slug][0]`, { slug: category.slug });
        if (categoryExist) {
            return categoryExist._id;
        }
        const catObj = {
            _type: "category",
            _id: `${category.slug}-${counter}`,
            name: category.name,
            slug: {
                _type: 'slug',
                current: slugify(category.name || 'default-category', { lower: true, strict: true })
            }
        };
        const response = await client.createOrReplace(catObj);
        console.log('Category created successfully', response);
        return response._id;
    } catch (error) {
        console.error('Failed to create category:', category.name, error);
        return null;
    }
}

async function importData() {
    try {
        console.log('Fetching products from API...');
        const response = await axios.get('https://hackathon-apis.vercel.app/api/products');
        const products = response.data;
        let counter = 1;

        for (const product of products) {
            console.log(`Processing product: ${product.name}`);
            let imageRef = null;
            let catRef = null;

            if (product.image) {
                imageRef = await uploadImageToSanity(product.image);
            }
            if (product.category?.name) {
                catRef = await createCategory(product.category, counter);
            }

            const sanityProduct = {
                _id: `product-${counter}`,
                _type: 'product',
                name: product.name,
                slug: {
                    _type: 'slug',
                    current: slugify(product.name, { lower: true, strict: true })
                },
                price: product.price,
                category: catRef ? {
                    _type: 'reference',
                    _ref: catRef
                } : undefined,
                tags: product.tags || [],
                quantity: 50,
                image: imageRef ? {
                    _type: 'image',
                    asset: {
                        _type: 'reference',
                        _ref: imageRef
                    }
                } : undefined,
                description: product.description || "Default product description",
                features: product.features || ["Premium material", "Handmade upholstery"],
                dimensions: product.dimensions || {
                    _type: 'dimensions',
                    height: "110cm",
                    width: "75cm",
                    depth: "50cm"
                }
            };

            await client.createOrReplace(sanityProduct);
            console.log(`✅ Imported product: ${sanityProduct.name}`);
            counter++;
        }
        console.log('✅ Data import completed!');
    } catch (error) {
        console.error('Error importing data:', error);
    }
}

importData();
```

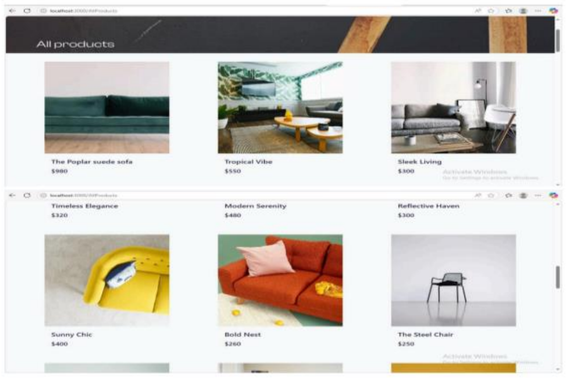
Screenshots

1. **API Calls**:

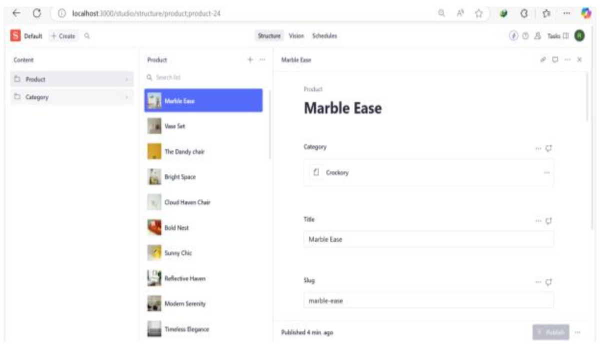


```
✅ Imported product: Bold Nest
Processing product: Cloud Haven Chair
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/78683aab6c58d367f91e411fbe2aea5f06a9a9b7-3264x4902.jpg
Image uploaded successfully: image-a389259aad5aa63a3b438ed9c42339a1c8375a3a-3264x4902-jpg
✅ Imported product: Cloud Haven Chair
Processing product: Bright Space
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/d32066d0882c1632e027b053564115ed460e09eb-3393x5090.jpg
Image uploaded successfully: image-d32066d0882c1632e027b053564115ed460e09eb-3393x5090-jpg
✅ Imported product: Bright Space
Processing product: The Dandy chair
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/e2cc69a6c9f9d1ff1fcfbe839844240e16f5f758-305x375.png
Image uploaded successfully: image-e2cc69a6c9f9d1ff1fcfbe839844240e16f5f758-305x375-png
✅ Imported product: The Dandy chair
Processing product: Vase Set
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/8795ff7bdb3d810dbe0c3eb52cd22116c9115bce-305x375.png
Image uploaded successfully: image-8795ff7bdb3d810dbe0c3eb52cd22116c9115bce-305x375-png
✅ Imported product: Vase Set
Processing product: Marble Ease
Uploading image: https://cdn.sanity.io/images/ri847jqu/production/7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000.jpg
Image uploaded successfully: image-7f6146b6625bf94030f2aa896e9b0c847e5f4919-8000x8000-jpg
Category created successfully {
  _createdAt: '2025-01-18T20:25:34Z',
  _id: 'crockory-24',
  _rev: 'oQpkQeFIZ2OSQkhvXSTUIU',
  _type: 'category',
  _updatedAt: '2025-01-18T20:25:34Z',
  name: 'Crockory',
  slug: { _type: 'slug', current: 'crockory' }
}
✅ Imported product: Marble Ease
✅ Data import completed!
```

2. **Frontend Display**:



3. **Sanity CMS**:



localhost:3000/studio/structure/product;product-24

Product

Marble Ease

Category: Crockory

Title: Marble Ease

Slug: marble-ease

Published 4 min. ago

Best Practices Followed

1. Used `.env` files to store sensitive data securely.
2. Followed clean coding practices with descriptive variable names and modular functions.
3. Validated data during migration using schemas.
4. Thoroughly documented all steps with screenshots and code snippets.
5. Committed changes frequently with clear messages.

Day 3 Checklist

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

Conclusion:

This report summarizes my progress on Day 3, covering API integration, schema adjustments, and data migration for the **AVIO N** marketplace.