Day 3 - API Integration Report - EasyMart

- API integration process:

  API integration is the process of connecting your application to external services or data sources.

  - **Setting Up Environment Variables in '.env.local':**
    - NEXT_PUBLIC_SANITY_PROJECT_ID=your project id
    - NEXT_PUBLIC_SANITY_DATASET=production
    - SANITY_API_TOKEN=your sanity token
  - **Identify APIs**: First, I identified the APIs I need, such as product data.
    - **Migration script:** https://github.com/OkashaTanoli/template-03-api/blob/master/scripts/data-migration.mjs
    - **Api:** https://template-03-api.vercel.app/api/products
    - **Sanity schema:** https://github.com/OkashaTanoli/template-03-api/blob/master/src/sanity/schemaTypes/products.ts
  - **Setup API Endpoints /products**: I configure endpoint. i.e. /products endpoint fetches all product details
  - **Authentication**: Most APIs require authentication. We set up API keys or OAuth tokens to ensure only authorized users can access the data.
    - SANITY_API_TOKEN=your sanity token
  - **Making Requests**: The frontend sends requests to the API, asking for data. For example, when a user browses products, the frontend sends a GET request to the /products endpoint.
  - **Handle Responses**: Once the API sends back data (like product details or order information), we process and display it on the website or app in a user-friendly way.

- ## Adjustments Made to Schemas:

Schemas define the structure of data within the application. During the project, we made adjustments to our schemas to ensure they fit the business needs.

i. **In sanity/schemaTypes/products.ts:**

```
export default {
  name: 'product',
  type: 'document',
  title: 'Product',
  fields: [
    {
      name: 'name',
      type: 'string',
      title: 'Product Name',
    },
    {
      name: 'description',
      type: 'string',
      title: 'Description'
    },
    {
      name: 'price',
      type: 'number',
      title: 'Product Price',
    },
    {
      name: 'discountPercentage',
      type: 'number',
      title: 'Discount Percentage',
    },
    {
      name: 'priceWithoutDiscount',
      type: 'number',
      title: 'Price Without Discount',
      description: 'Original price before discount'
    },
    {
      name:'rating',
      type:'number',
      title:'Rating',
      description:'Rating of the product'
    },
    {
      name: 'ratingCount',
      type: 'number',
      title: 'Rating Count',
      description: 'Number of ratings'
    },
    {
      name: 'tags',
      type: 'array',
      title: 'Tags',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags'
      },
      description: 'Add tags like "new arrival", "bestseller", etc.'
    },
    {
      name: 'sizes',
      type: 'array',
      title: 'Sizes',
      of: [{ type: 'string' }],
      options: {
        layout: 'tags'
      },
      description: 'Add sizes like S , M , L , XL , XXL'
    },
    {
      name: 'image',
      type: 'image',
      title: 'Product Image',
      options: {
        hotspot: true // Enables cropping and focal point selection
      }
    }
  ]
};
```

ii. In sanity/schemaTypes/index.ts

```
import { type SchemaTypeDefinition } from 'sanity'
import product from './product'

export const schema: { types: SchemaTypeDefinition[] } = {
  types: [product],
}
```

3. Migration Steps and Tools Used

Migration involves moving data from one system to another, often from a local database to a sanity system or between different platforms.

- **In scripts/importSanityData.mjs**

```
import { createClient } from '@sanity/client';

import axios from 'axios';

import dotenv from 'dotenv';

import { fileURLToPath } from 'url';

import path from 'path';


// Load environment variables from .env.local

const __filename = fileURLToPath(import.meta.url);

const __dirname = path.dirname(__filename);

dotenv.config({ path: path.resolve(__dirname, '../.env.local') });


// Create Sanity client

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 importData() {
  try {
    console.log('migrating data please wait...');

    // API endpoint containing car data
    const response = await axios.get('https://template-03-api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==>> ", products);

    for (const product of products) {
      let imageRef = null;
    if (product.image) {
      imageRef = await uploadImageToSanity(product.image);
    }

    const sanityProduct = {
      _type: 'product',
      productName: product.productName,
      category: product.category,
      price: product.price,
      inventory: product.inventory,
      colors: product.colors || [], // Optional, as per your schema
      status: product.status,
      description: product.description,
      image: imageRef ? {
        _type: 'image',
        asset: {
          _type: 'reference',
          _ref: imageRef,
        },
      } : undefined,
    };

    await client.create(sanityProduct);
  }

  console.log('Data migrated successfully!');
} catch (error) {
    console.error('Error in migrating data ==>> ', error);
  }}
importData();
```

- **In package.json:**

```
"scripts": {
  "dev": "next dev",
  "build": "next build",
  "start": "next start",
  "lint": "next lint",
  "import-data": "node scripts/importSanityData.mjs"
}
```

- **Now you can run the import script using:**

```
npm run import-data
```

Day 3 Checklist:

Self-Validation Checklist:

i. API Understanding: ✓
ii. Schema Validation: ✓
iii. Data Migration: ✓
iv. API Integration in Next.js: ✓
v. Submission Preparation: ✓