Day 3 - API Integration Report – [Comforty]

1. API integration process.

- **Set up state variables**:
  - `products`: Holds the fetched product data.
  - `loading`: Tracks loading state.
  - `error`: Stores error messages.
- **Create the `fetchingData` function**:
  - Set `loading` to `true` and `error` to `null` before fetching data.
  - Use `client.fetch()` to get data from the Sanity API:
    - Query: `*[_type == "products"]` to fetch product details (ID, image URL, price, title, etc.).
  - On success:
    - Set the `products` state with the fetched data.
  - On failure:
    - Set the `error` state with an error message.
- **Handle loading and error states**:
  - If `loading` is `true`, display the `SkeletonLoader` component.
  - If an error occurs, show an error message and a retry button to re-fetch data.
- **Use `useEffect` to fetch data**:
  - Trigger `fetchingData` once on component mount.
- **Render product data**:
  - If no errors and data is loaded, display products using the `ProductCard` component.

2. Adjustments made to schemas.

Adjustments Between Old and New Schema:

i. Field Additions:

- **Slug**:

    - Added a new `slug` field to store the product URL slug, which is derived from the title and has a `maxLength` of 96.
    - Validation: Slug is now required.
- **GreenTag**:
    - Added a `greenTag` field to store a tag for environmentally friendly products.
- **Price without Discount**:
    - Added a `priceWithoutDiscount` field to store the price before any discounts.
- **Badge**:
    - Added a `badge` field, presumably for labels like "new", "sale", etc.
- **Category**:
    - Added a reference field for `category` to link to a `categories` document.
- **Description**:
    - Added a `description` field to store a product description.
- **Inventory**:
    - Added an `inventory` field to manage the product's stock count.
- **Tags**:
    - Added a `tags` array field that stores predefined tags, such as "Featured", "Instagram", "Gallery", etc.

ii. Field Modifications:

- **Stock** (old schema) is now **Inventory Management** in the new schema, renamed to be more descriptive.
- **Price** remains the same but has been preserved alongside the new `priceWithoutDiscount`.

iii. Field Removal:

- The **id** field, which may have been auto-generated in the old schema, is not explicitly present in the new schema.
- The **name** field is replaced by **title** for better clarity and consistency with the rest of the fields.

iv. Automatic `_id` Field in Sanity:

- In Sanity, the `_id` field is automatically generated for each document, even if it's not explicitly defined in the schema.
- This field serves as a unique identifier for each document.
- The `_id` field is available by default when you query for documents, so even if it's not listed in the schema definition, it will be included in the fetched data.

3. Migration steps and tools used.

Migration Steps:

i.Create Migration Folder:

- Create a folder named `script` in the root directory of your project.

2. **Create Migration Files**:
    - Inside the `script` folder, create three files:
        - Two files were provided by **Hamza Syed** for importing all product data from the API into Sanity.
        - One additional file was created to handle specific tasks like generating slugs for products.
3. **Set Up Sanity Project**:
    - Create a project in Sanity.
    - Generate a **token** for authentication.
    - In the CORS origin settings, allow:
        - `localhost:3000` for local development.
        - `vercel.com` for deployment on Vercel.
4. **Create `.env` File**:
    - Add a `.env` file in the root directory and store the following values:
        - `PROJECT_ID`: Sanity project ID.
        - `DATASET`: The dataset used for the project.
        - `TOKEN`: The generated Sanity token for authentication.
5. **Update `package.json`**:
    - In the `package.json` file, add the scripts to run the migration:
        - `npm run scripts`
        - `npm run migrate`
        - `npm run cleanup`
6. **Migration Function**:
    - The `migrate` function was specifically created to handle the process of generating slugs for products and adding them to the Sanity dataset.
7. **API Testing Tool**:
    - Used the **Thunder** API testing extension for testing the API during the migration process.