Game Time Gear - Day 3: API Integration and Data Migration

Prepared by: Syed Huzaifa Ahmed Hashmi

Introduction

Welcome to Day 3 of my journey in building **Game Time Gear**. Today's focus is on integrating APIs and migrating data into Sanity CMS to build a functional backend for the marketplace. I will use provided API references to populate Sanity with product data and integrate these APIs into the Next.js frontend, ensuring that my data schemas are aligned and functional.

Objective

By the end of today, I will have:

- Reviewed and understood the provided API documentation.
- Validated and confirmed that my Sanity CMS schema (from Day 2) is compatible with the API data.
- Drafted and executed a migration script using the provided API (via `npm run migrate`).
- Integrated API calls into my Next.js project with utility functions.
- Rendered fetched product data in the frontend components.
- Identified areas where error handling is needed for robust API integration.
- Prepared a report detailing the migration steps, API integration process, and relevant code snippets for submission.

API Overview

Provided APIs

- **Assigned Template API**:
  - **Key Endpoint**: `/products`
  - *Note*: No endpoints for order history or additional data are being used.
- **API Documentation**:
  - I have reviewed the API documentation for the assigned template and confirmed that the data structure matches my Sanity CMS schema.
  - Since there are no differences between the API and my schema, no adjustments or field mappings were necessary.

Data Migration

Data Migration Method

- **Method Used**: API script migration
- **Command**: `npm run migrate`
- **Status**:
  - I have drafted the migration scripts.
  - The migration script fetches product data from the `/products` endpoint and imports it into Sanity CMS.
  - I have cross-checked the migrated data; while some minor errors exist, they are not critical.

API Integration in Next.js

Utility Functions & Component Rendering

- **Utility Functions**:
  - Located in `src/lib/api.ts`, these functions use `client.fetch` to retrieve product data from Sanity.
- **Component Rendering**:
  - The `ProductCard.tsx` component is set up to display product information.
  - The `FeaturedSection.tsx` component is ready but currently does not render product data.
  - The `products/` page directory is empty, indicating that the product listing page still needs to be developed further.

Testing

- **Tools Used**:
    - I have used browser developer tools and Postman to test API endpoints.
- **Current State**:
    - Fetched data is visible in Sanity CMS and appears correct when viewed on the Next.js frontend.
    - However, no robust error handling mechanisms (like fallback data or skeleton loaders) have been implemented yet.
    - I have not yet tested scenarios where the API returns an error or empty response.

Error Handling

- **Implemented**:
    - Currently, there is no centralized error logging or fallback mechanism in place.
- **To Do**:
    - Implement error handling in API utility functions.
    - Add fallback UI elements (e.g., skeleton loaders) for a better user experience in case of errors.

Best Practices

1. **Security**:
    - Use `.env` files to securely store API keys and sensitive data.
2. **Code Quality**:
    - Follow clean coding practices with descriptive variable names and modular functions.
    - Comment complex logic and maintain a changelog for any schema adjustments.
3. **Data Validation**:
    - Validate all migrated data against your Sanity CMS schema.
    - Log any discrepancies for further review.
4. **Testing**:
    - Thoroughly test API integration using tools like Postman.
    - Implement error handling to manage edge cases such as empty responses or API failures.
5. **Version Control**:
    - Commit changes frequently with meaningful commit messages and tag significant milestones.

6. **Peer Review**:
    - Share your integration process with peers or mentors for feedback and incorporate improvements.

Day 3 Checklist

Self-Validation Checklist

- **API Understanding**: ✔
- **Schema Validation**: ✔
- **Data Migration**: ✔ (Minor errors observed, but acceptable)
- **API Integration in Next.js**: Partially ✔ (Product listing page still to be developed)
- **Submission Preparation**: ✘ (Documentation of migration steps and screenshots pending)

End Note

Day 3 focused on integrating external APIs and migrating data into Sanity CMS. With the migration scripts drafted and initial API integration in Next.js underway, the backend foundation for **Game Time Gear** is growing stronger. The next steps involve enhancing error handling, completing the product listing pages, and finalizing the migration report for submission.

Prepared by Syed Huzaifa Ahmed Hashmi
