Day 3 - API Integration Report - E-commerce

Overview of API Integration

API Integration Process

1. **Data Migration Script Execution:**
    - Executed the data-migration script provided by Sir Ali Jawad.
    - Configured the script with my Sanity project ID and token to establish a secure connection with the Sanity CMS.
    - Successfully migrated data into the specified Sanity dataset using the script.
2. **Integration with Next.js Project:**
    - Configured the sanity.query.ts file to handle Sanity queries efficiently.
    - Integrated the migrated data seamlessly into the Next.js e-commerce project.
    - Adjusted queries to utilize GROQ syntax for fetching product data.
3. **Implementation of Categorization Functionality:**
    - Enhanced schemas to support categorization based on API specifications.
    - Enabled dynamic filtering of products by category (e.g., "men," "women," "kids") through partial match queries.
    -
4. **Debugging and Verification:**
    - Identified an issue where migrated data was not visible using the Sanity Vision tool.
    - Verified data retrieval through a basic Next.js API route:

```
export async function GET() {
  const data = await sanityClient.fetch(`*[_type=="product"]`);
  return NextResponse.json(data, { status: 200 });
}
```

    - Confirmed data fetching using the ThunderClient VS Code extension, isolating the problem to the Vision tool.

5. **Handling Legacy Data Migration Errors:**
    - Unintentionally migrated old data, which required removal.
    - Wrote the following script to delete redundant entries:

```
export async function DELETE() {
    const productsToDelete = await sanityClient.fetch(`*[_type == "product" && defined(ratingCount)]._id`);

    if (productsToDelete.length === 0) {
        console.log("No products with 'ratingCount' field found.");
        return;
    }

    for (const productId of productsToDelete) {
        await sanityClient.delete(productId);
        console.log(`Deleted product with ID: ${productId}`);
    }
    console.log("All products with 'ratingCount' field deleted successfully.");

    return NextResponse.json("success", { status: 200 })}
```

    - Executed DELETE requests via ThunderClient and successfully cleared unwanted data.

Adjustments Made to Schemas

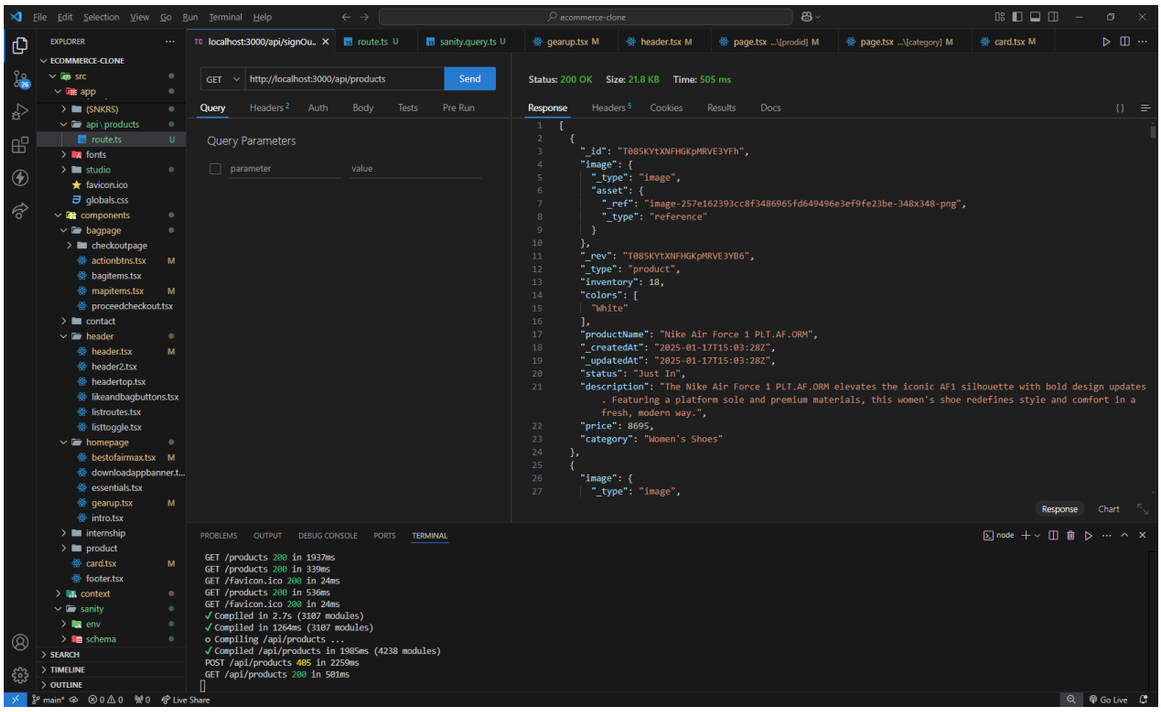
- Updated product schema to match the specified API structure.
- Structured categories to support partial matching for improved filtering capabilities.

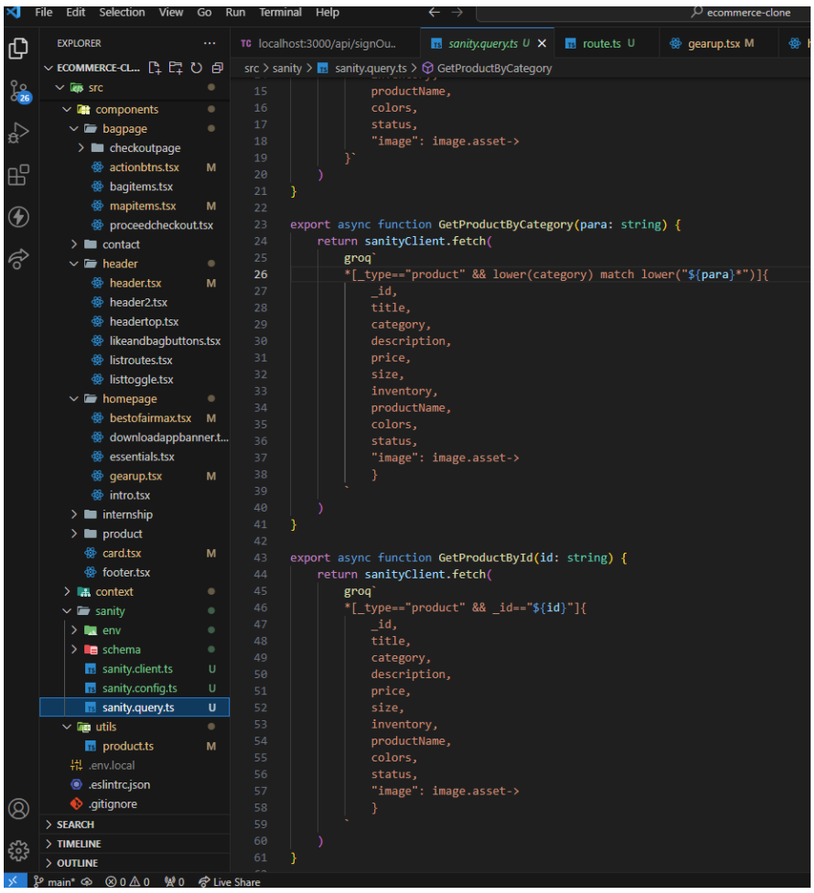
Migration Steps and Tools Used

1. Utilized the data-migration script provided by Sir Ali Jawad.
2. Configured Sanity project ID and token to ensure proper integration.
3. Migrated data into the dataset without errors.
4. Validated successful data migration with GROQ queries tested through API routes.
5. Displayed the fetched product data on the frontend UI for verification.

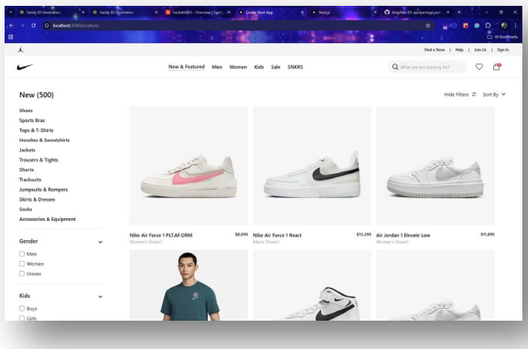
Screenshots

1. **API Calls Verification:**
    - Successful API fetch operations captured using GROQ and ThunderClient.

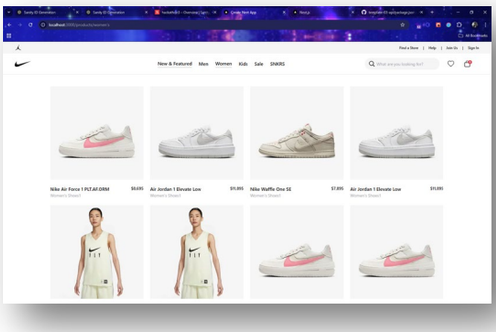




2. **Frontend Data Display:**
    - Below is a representation of the frontend UI displaying the migrated product data:

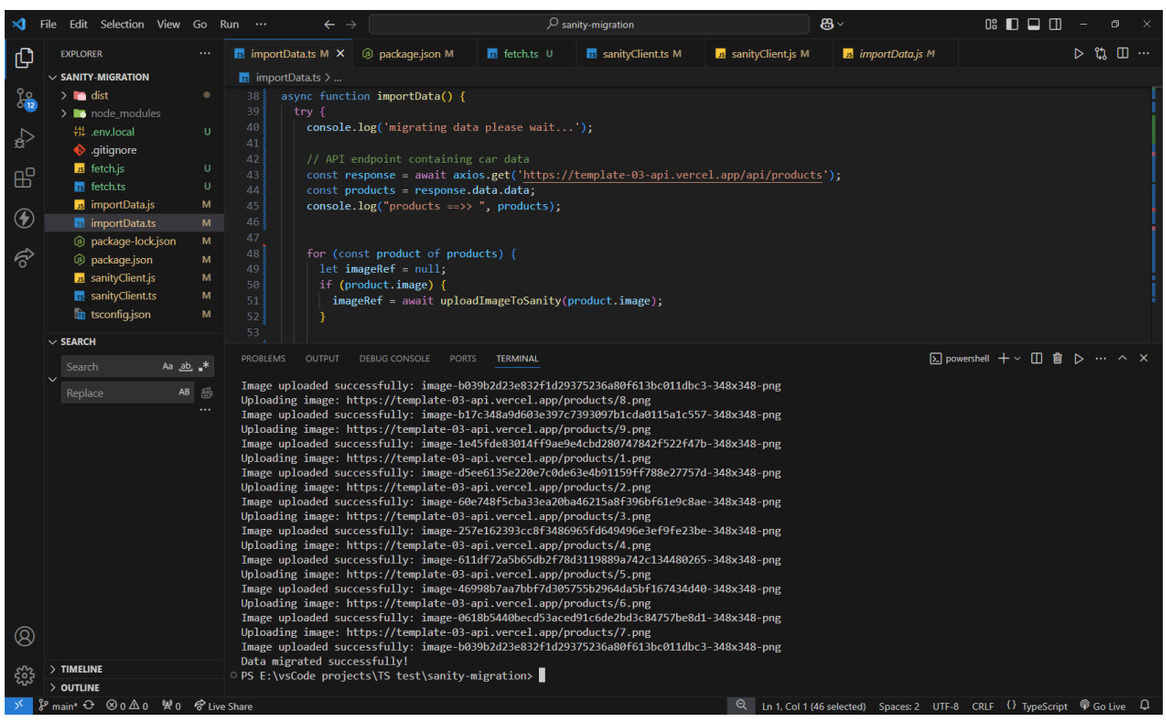


All Data



Data by Category

3. **Sanity CMS Data Migration:**
    - Screenshots of the populated data in Sanity CMS, confirming successful migration.



Conclusion

The integration process was executed successfully, with data migrated, validated, and displayed accurately in the frontend. Debugging steps ensured proper functionality, and schema adjustments enabled enhanced filtering and categorization.