**Day 3 - API Integration Report – [Bazaar Nest]**

**Prepared by: Ibad Uddin**

**1. Introduction**

This document provides details on the API integration process, schema adjustments, and migration steps implemented for our marketplace. Instead of using the provided API, a custom backend server was developed using **Express (Node.js) and MongoDB** and deployed on **Railway**. The product data was then fetched and integrated into **Sanity CMS** to maintain consistency between the backend and frontend.

**2. API Integration Process**

**Backend Development**

* **Framework:** Express (Node.js)
* **Database:** MongoDB (Mongoose for schema management)
* **Deployment:** Railway
* **API Testing:** Postman

**API Endpoints Created**

* **GET /api/products** - Fetch all products
* **GET /api/products/:id** - Fetch a single product by ID
* **POST /api/products** - Add a new product
* **PUT /api/products/:id** - Update product details
* **DELETE /api/products/:id** - Delete a product

**Example Product Schema in MongoDB**

{

"id": "product001",

"name": "Cotton T-Shirt",

"description": "High-quality cotton t-shirt with a comfortable fit.",

"price": 19.99,

"stock": 50,

"category": "Men's Clothing",

"category\_slug": "mens-clothing",

"images": [

"https://teetall.pk/cdn/shop/products/5142dcd15ae1fab547b502b043d65f66.webp",

"https://teetall.pk/cdn/shop/products/2a0e874ae076990cc9450472683df228.webp"

],

"size": ["S", "M", "L", "XL"],

"qcom\_availability": true,

"brand": "ComfortWear",

"tags": ["summer", "casual", "cotton"],

"ratings": 4.5,

"created\_at": "2024-01-31T10:00:00Z"

}

**Sanity CMS Integration**

The same schema was used in Sanity CMS to ensure seamless data fetching and rendering.

**3. Adjustments Made to Schemas**

**MongoDB Schema Adjustments**

* Added **category\_slug** for SEO-friendly URLs.
* Used **created\_at** to track product entry timestamps.
* Ensured **price** is stored as a floating-point number.

**Sanity CMS Schema Adjustments**

* Mirrored MongoDB schema to ensure consistency.
* Data transformation was required to format **ObjectId** as a string.
* Image URLs stored as an array for multiple images.

**4. Migration Steps and Tools Used**

**Migration Workflow**

1. **Add Data**: in local file and seed it
2. **Extract Data from MongoDB**: Used Mongoose to fetch products.
3. **Verify Data** : On railway server
4. **Transform Data**: Adjusted object format to match Sanity CMS schema.
5. **Load Data into Sanity**: Used Sanity API for importing data.

**Tools Used**

* **Node.js & Express**: Backend development
* **Mongoose**: MongoDB schema management
* **Axios**: API requests
* **Sanity API**: Data migration
* **Postman**: API testing

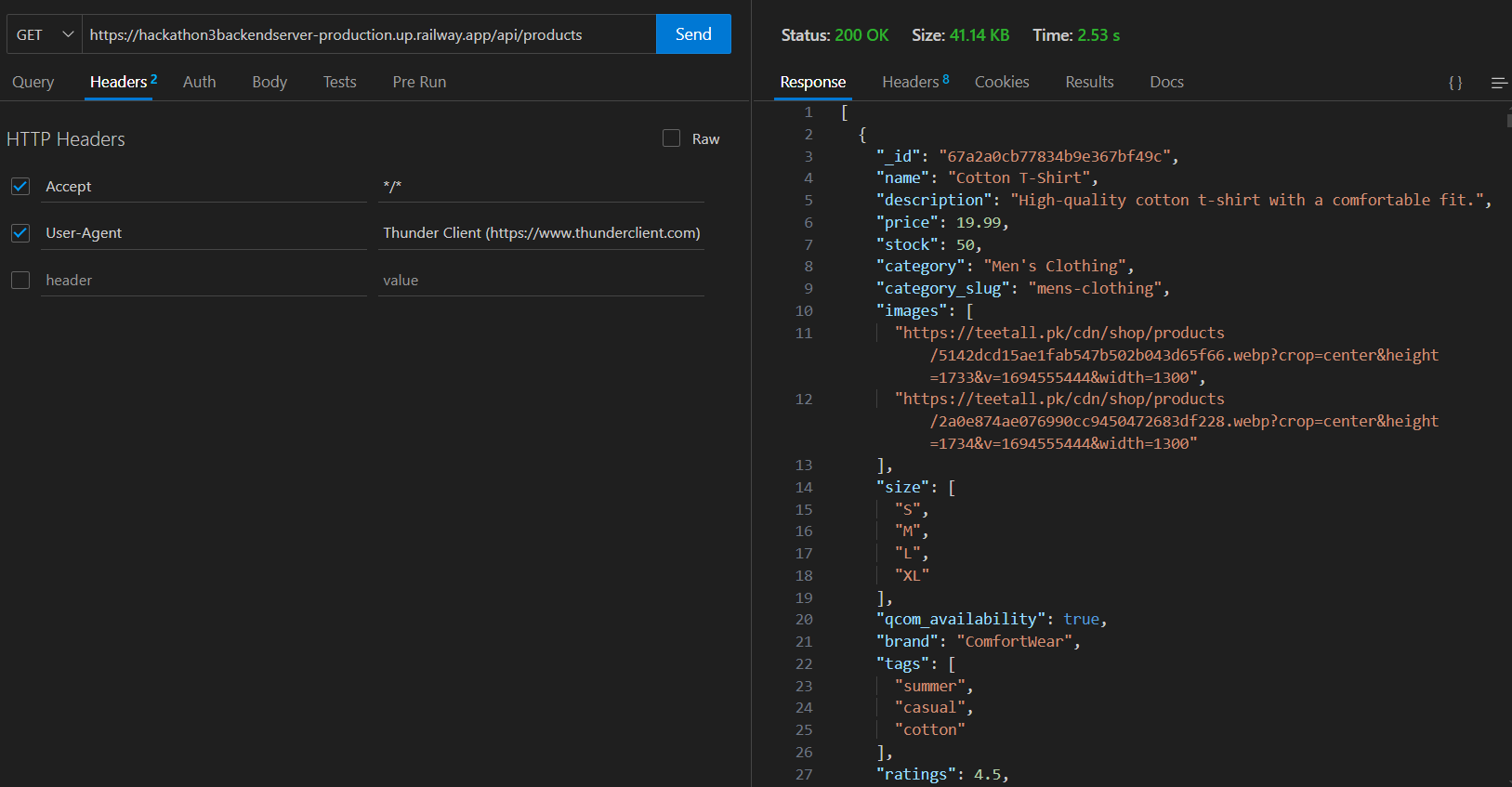
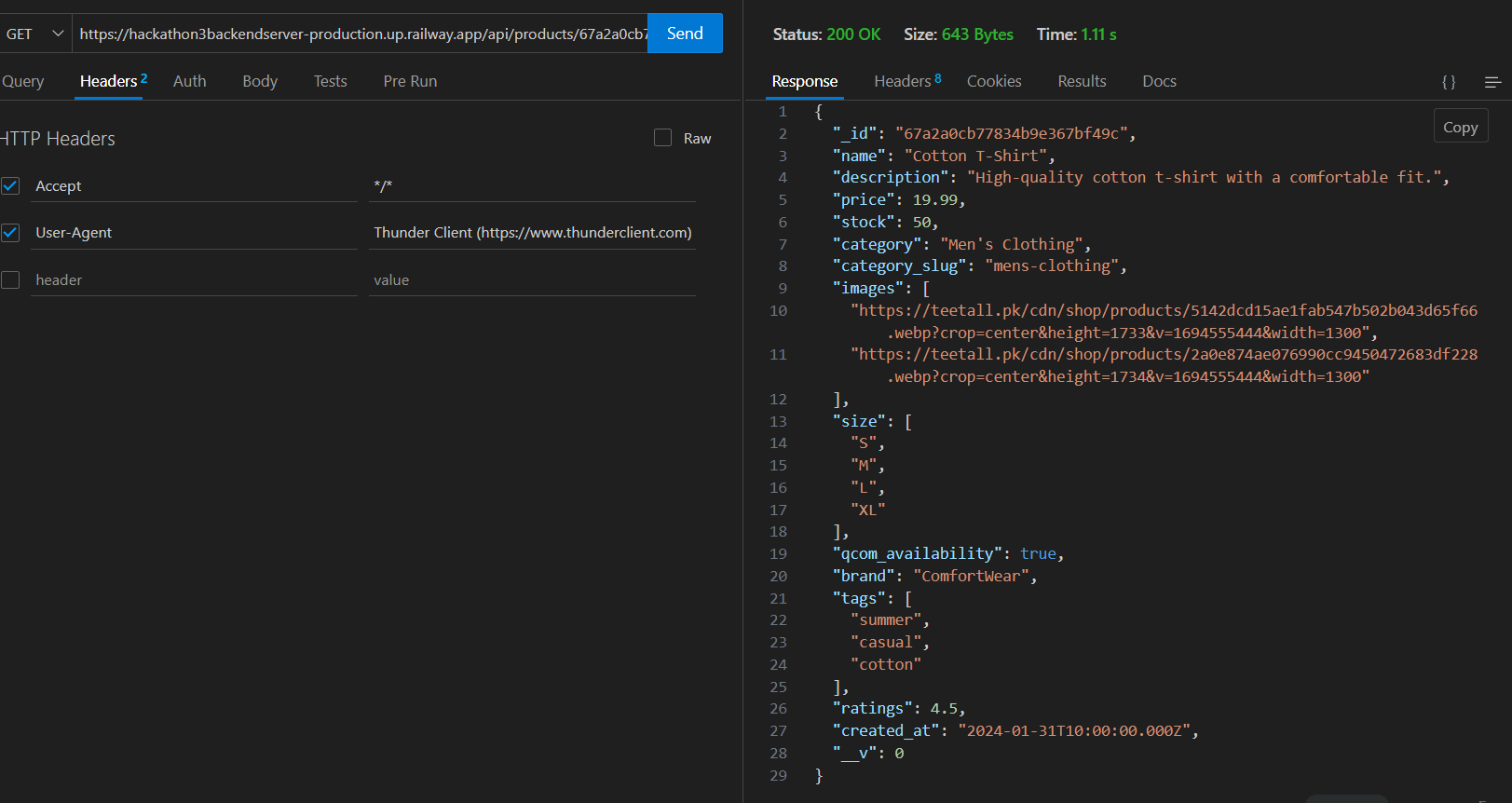
**Migration Script (Node.js)**

**Link:**

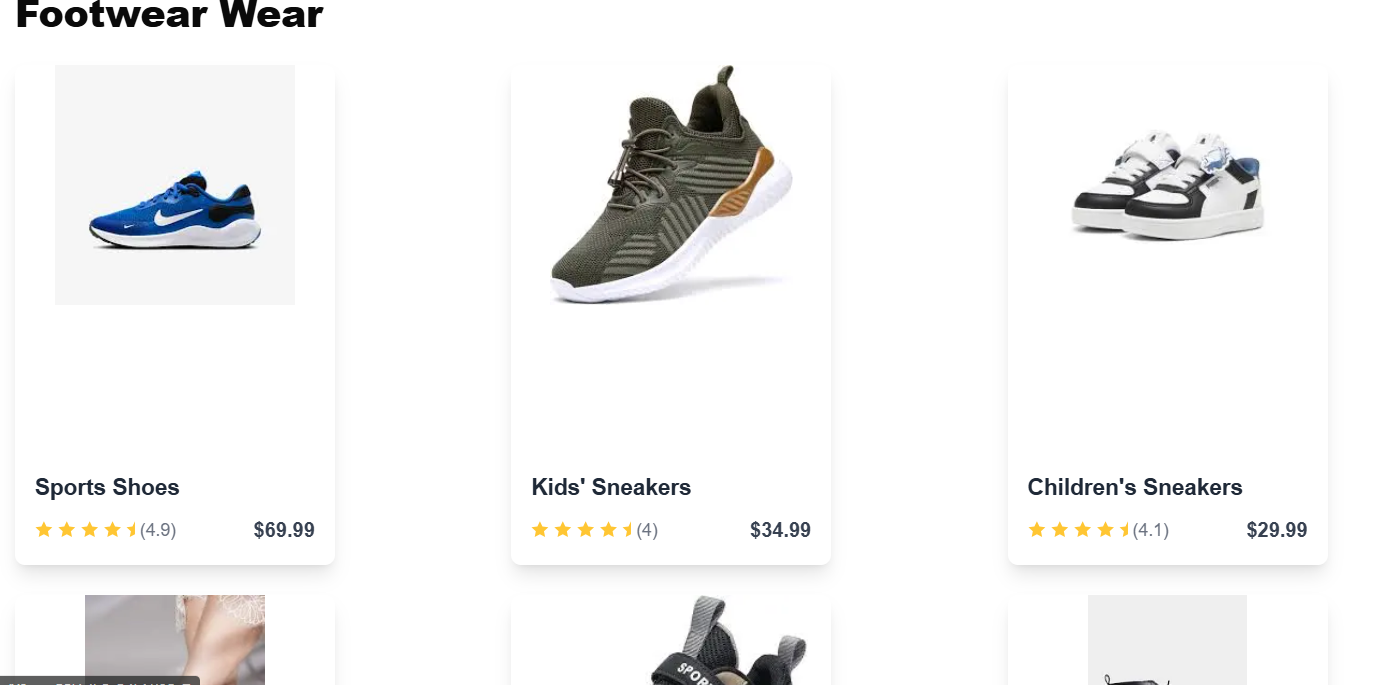
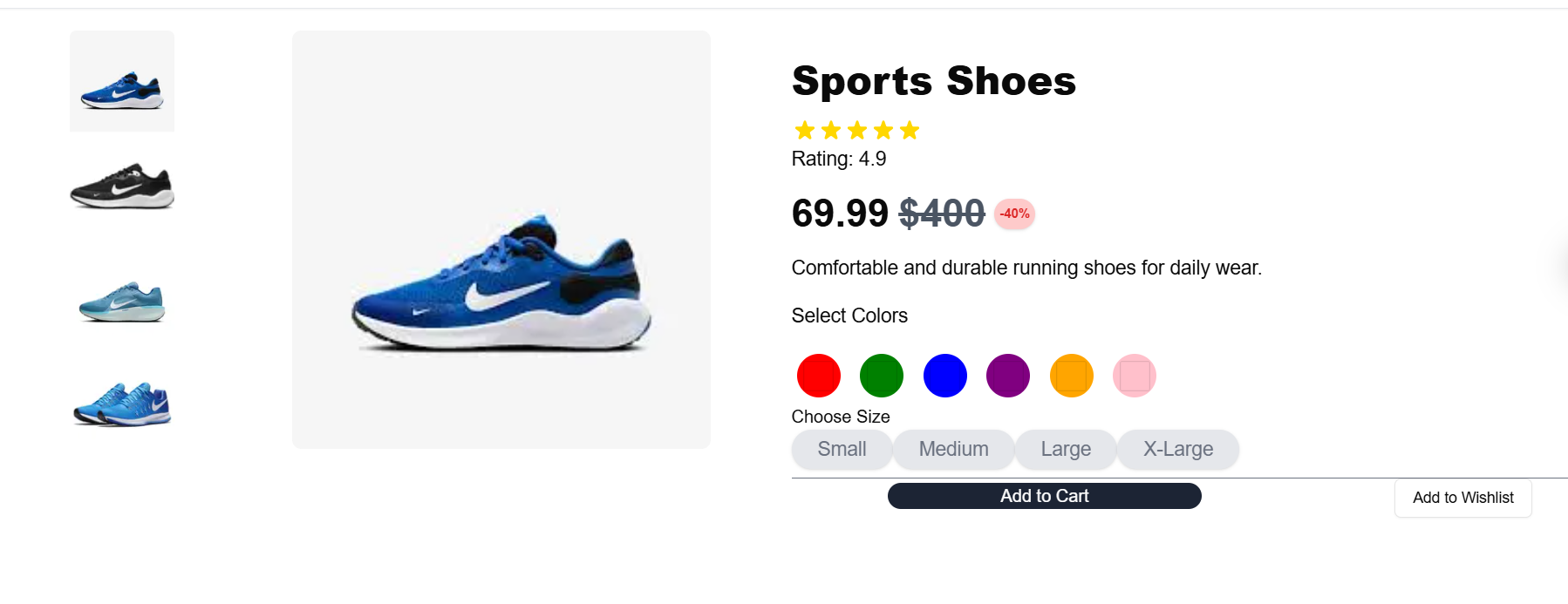
**https://github.com/IbadUddinSiddqui/hackathon-2/blob/main/scripts/importData.mjs**

**5. Screenshots**

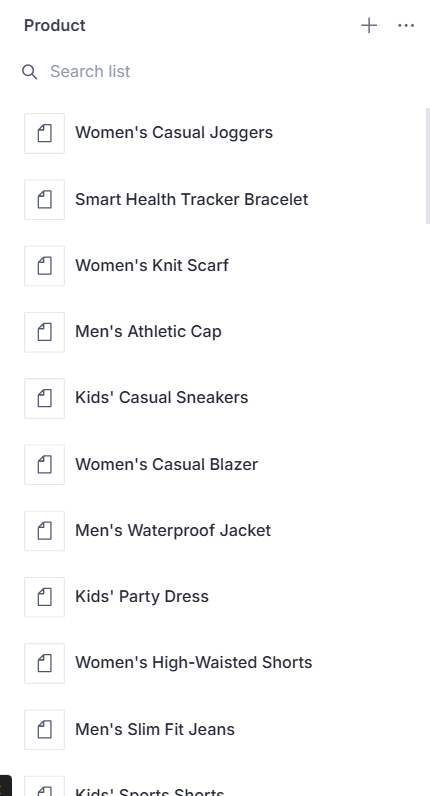
**API Calls**

* Screenshot of GET /api/products API call response.
* 
* Screenshot of GET /api/products/:id API call response.
* 

**Frontend Data Display**

* Screenshot of product listings rendered in the frontend.
* 
* Screenshot of individual product details displayed.
* 

**Sanity CMS Data**

* Screenshot of populated Sanity CMS fields.
* 

**6. Conclusion**

By developing a custom backend and deploying it on Railway, we ensured maximum flexibility for our marketplace. Our API integration strategy maintained a consistent product schema across MongoDB, Sanity CMS, and the frontend, leading to a seamless user experience.

This document provides a step-by-step overview of the API integration, data migration, and schema adjustments, ensuring clarity in the implementation process.