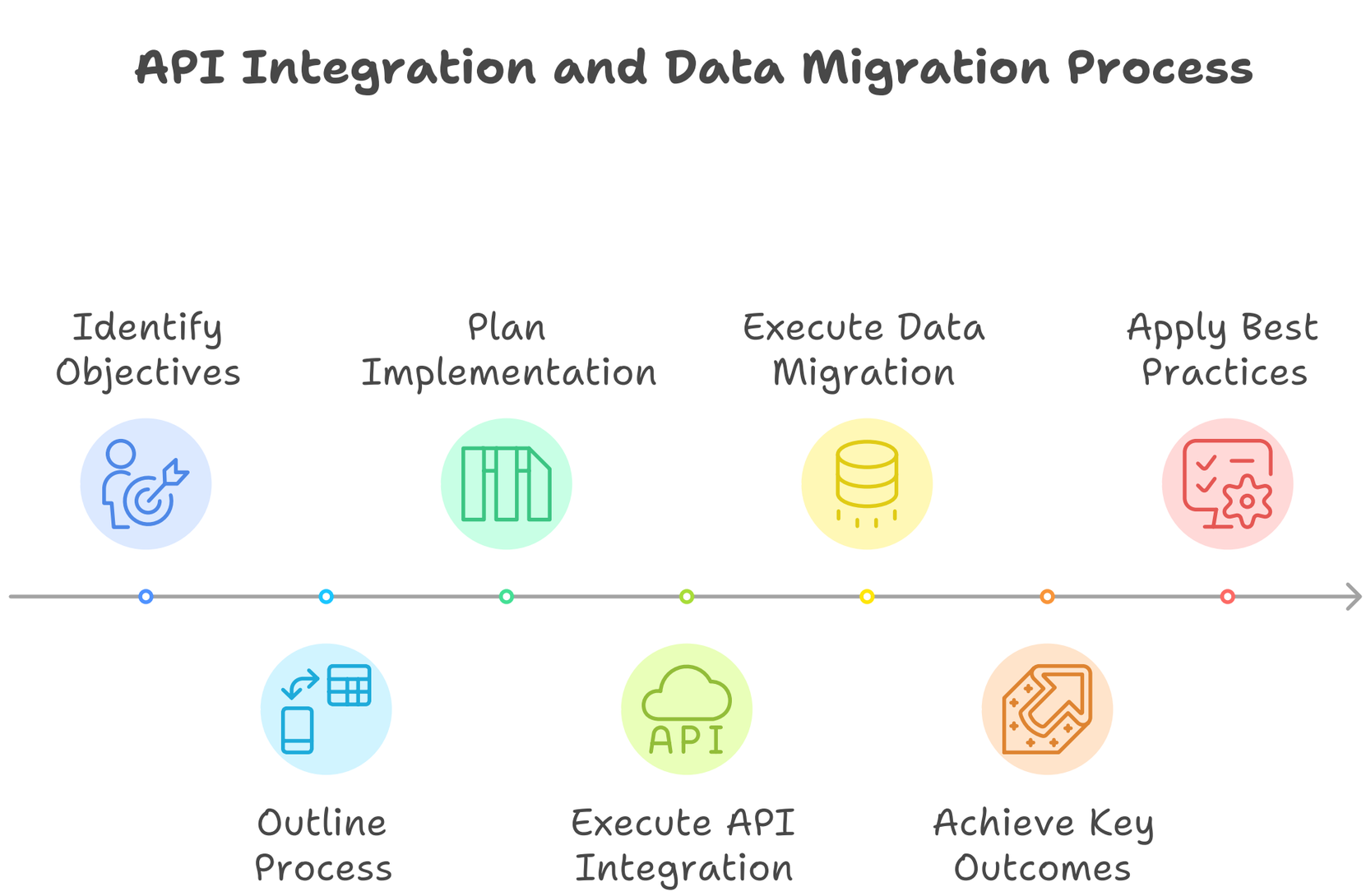
Made by Tayyab Ilyas (Student Leader)

Hackathon : Day 3 - API Integration and Data Migration

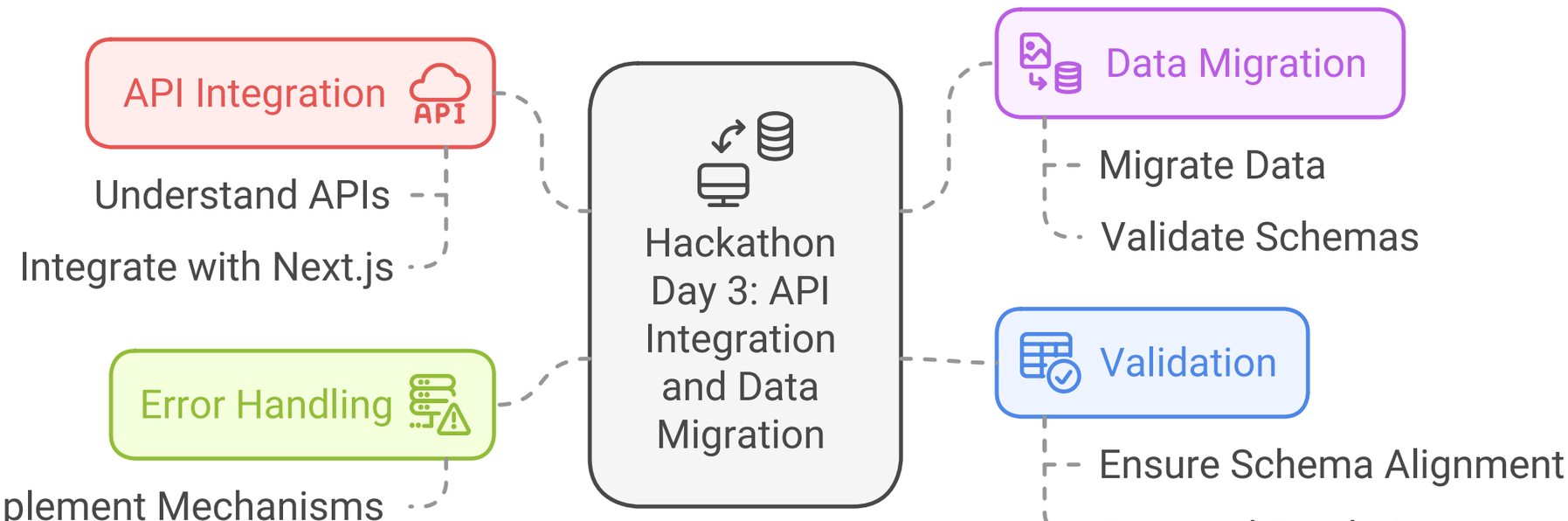
Objective:

The objective of this proposal is to outline the process for integrating APIs and migrating data into Sanity CMS to create a functional backend for the Premii Woods furniture marketplace. This document will provide a detailed plan, key outcomes, and best practices for successfully implementing Day 3 tasks.



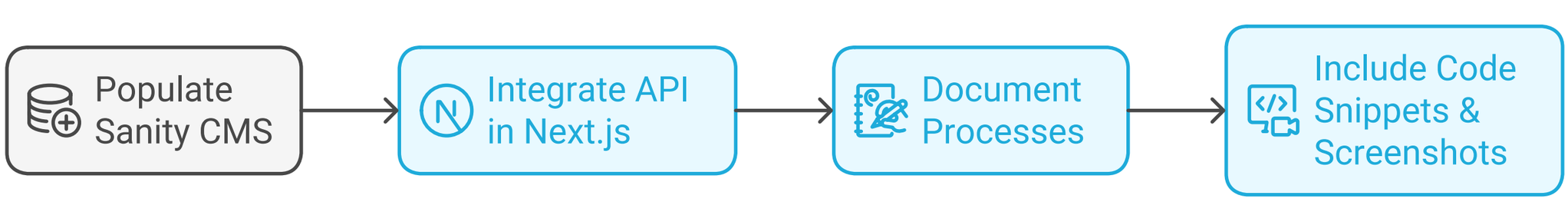
Work:

1. **API Integration:**
    - Understand the provided APIs.
    - Integrate APIs into the Next.js frontend to fetch and display product data.
2. **Data Migration:**
    - Migrate relevant data from APIs to Sanity CMS.
    - Validate and adjust schemas for compatibility.
3. **Error Handling:**
    - Implement robust error-handling mechanisms.
4. **Validation:**
    - Ensure the migrated data aligns with the schema and renders properly on the frontend.



Key Delivered:

1. Populated Sanity CMS with product data.
2. Functional API integration in the Next.js frontend.
3. Detailed documentation of the migration and integration process.
4. Code snippets and screenshots demonstrating the implementation.



API Overview:

Provided APIs:

- I am created my own api and data schema strutured

Steps for API Integration:

1. **Created a api**
2. **Create Utility Functions in Next.js:**
    - Write reusable functions to fetch data from the API.
    - Example:

```
import { client } from './lib/client';

export default async function FetchData() {
    const data = await client.fetch(`*[_type == 'product']`);
    return data;
}
```

3. **Render Data in Components:**
    - Use React components to display API data on the frontend.
    - Example:

```
import { FetchData } from './utils/api';

export default async function page() {
    const data = await FetchData();

    return (
        <section className="py-10 bg-white">
            <div className="container mx-auto px-4">
                {/* Grid Layout */}
                <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3
lg:grid-cols-4 gap-6">
                    {data.map((product, index) => (
                        <Link
                            href={`/Product/`}
                            className="p-3 rounded-lg hover:shadow-lg transition
block border border-gray-200"
                            key={index}
                        >
                            {/* Product Image */}
                            <img
                                src={product.imagePath}
                                alt={product.title}
                                className="w-full h-48 object-contain
rounded-lg"
                            />
                            {/* Product Details */}
                            <h3 className="mt-4 text-lg
font-semibold">{product.title}</h3>
                            <p className="mt-2
text-gray-600">{product.description}</p>
                            <p className="mt-2 text-gray-500 text-sm">
                                Category: {product.category}
                            </p>
                            <p className="mt-2 text-lg font-bold text-gray-800">
                                Rs {product.price}
                            </p>
                        </Link>
                    ))}
                </div>
            </div>
        </section>
    );
}
```

4. **Handle Errors Gracefully:**
    - Log errors in a centralized file for debugging.
    - Display user-friendly error messages in the UI.

Steps for Data Migration:

1. **Validate and Adjust Schema:**
    - Compare the API data structure with the existing Sanity CMS schema.
    - Modify the schema if necessary to ensure compatibility.
    - Example:
        - API Field: **product_title**
        - Schema Field: **name**

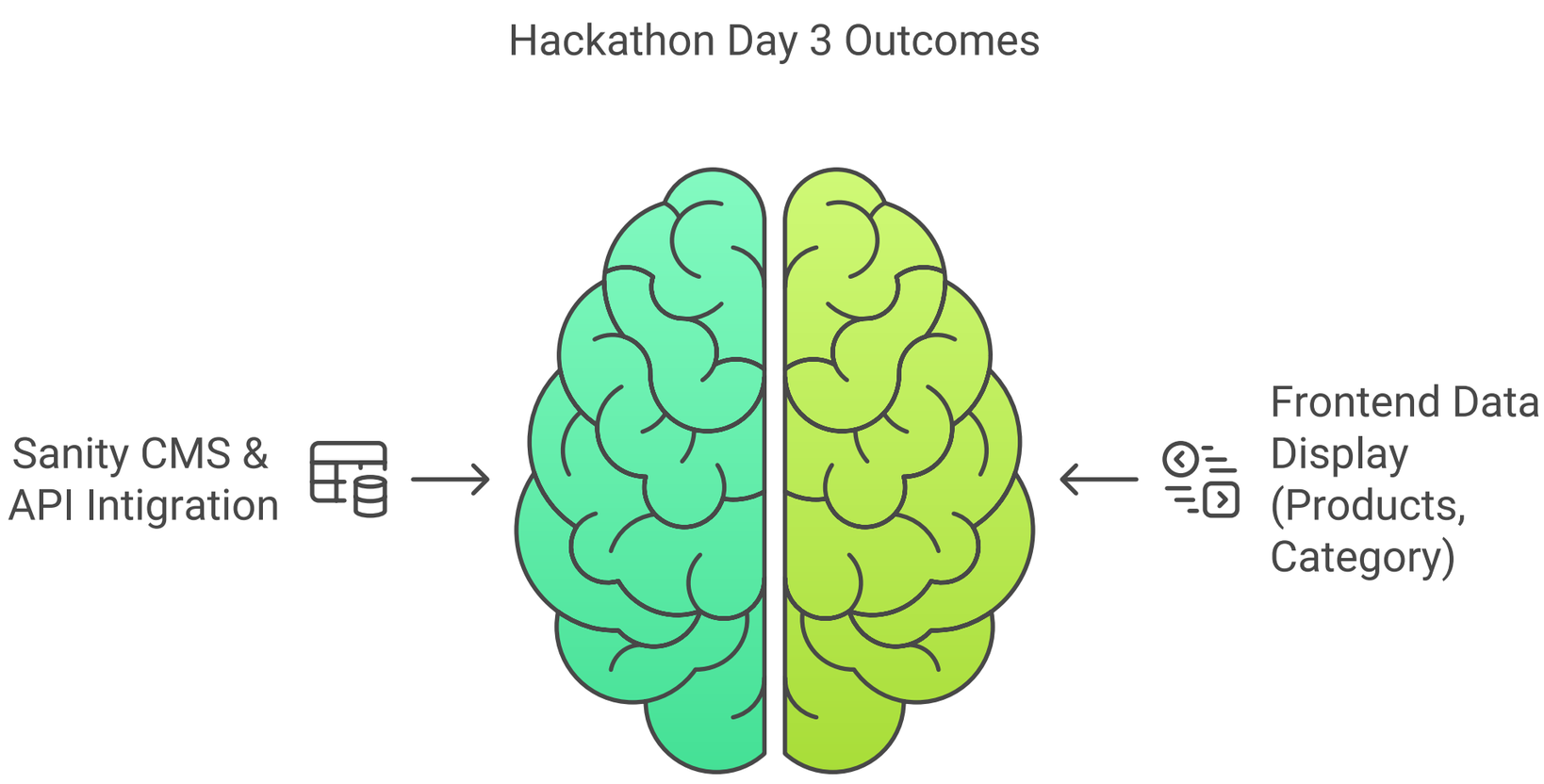
2. **Choose a Migration Method:**
    - **Script-Based Migration:**
        - Use provided scripts to automate the migration process.
        - Example:

```
import sanityClient from '@sanity/client';

const client = sanityClient({
    projectId: 'your_project_id',
    dataset: 'production',
    useCdn: false,
});

const migrateData = async () => {
    const products = await fetchProducts();

    products.forEach(async (product) => {
        await client.createOrReplace({
            _id: product.id,
            _type: 'product',
            name: product.product_title,
            price: product.price,
        });
    });
};

migrateData();
```

    - **Manual Import:**
3. **Validate Data Post-Migration:**
    - The migrated data aligns with the schema and renders properly on the frontend.

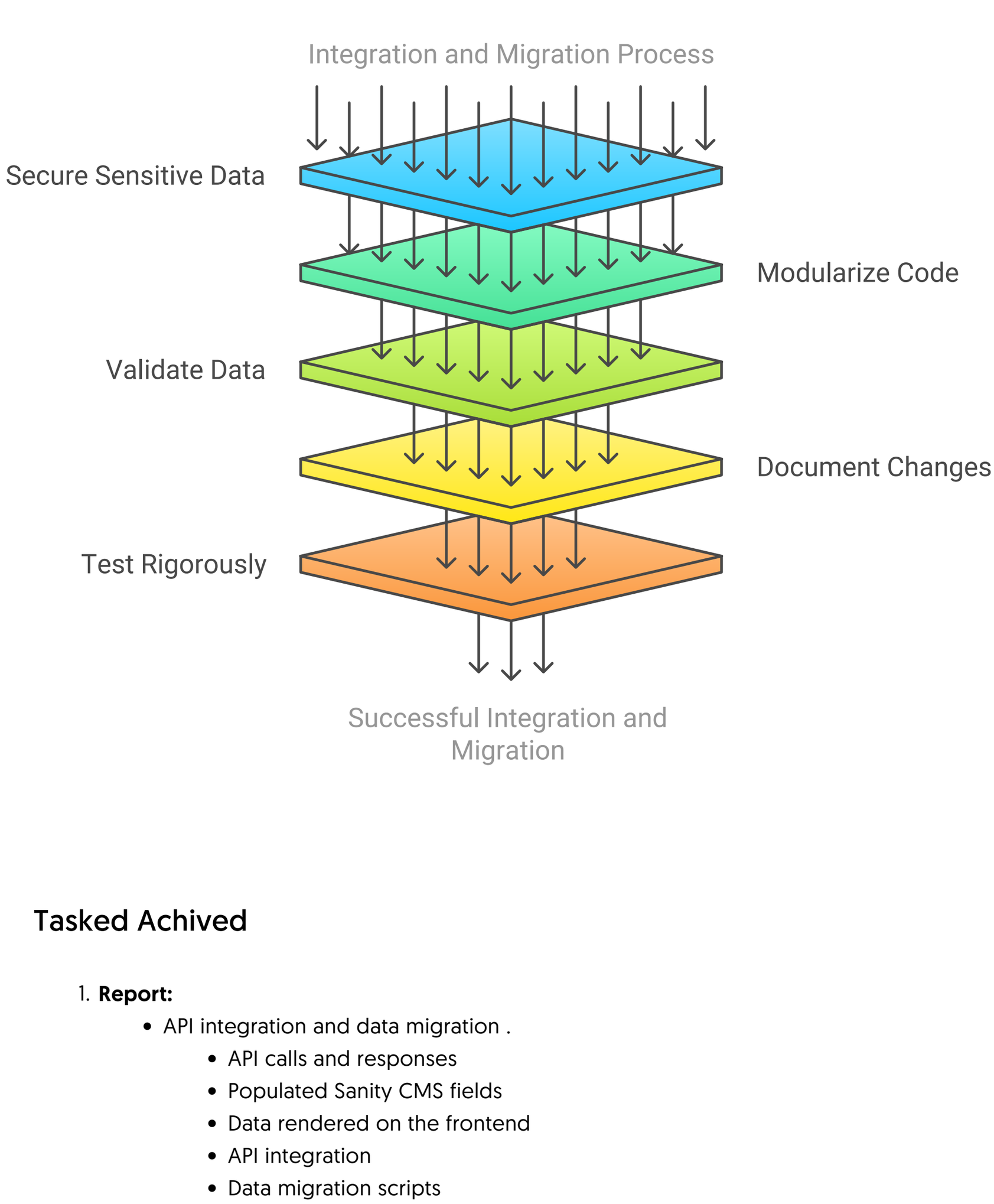
Output:

1. Sanity CMS populated with:
    - Products
    - Categories
2. Frontend displaying data fetched from the API and Sanity CMS.



Best Practices I am Used :

1. Use **.env** files to store sensitive data like API keys securely.
2. Modularize utility functions for better code reusability.
3. Validate all data during migration to avoid discrepancies.
4. Document every step thoroughly, including changes to schemas and scripts used.
5. Test API integration and data migration rigorously.



Tasked Achived

1. **Report:**
    - API integration and data migration .
        - API calls and responses
        - Populated Sanity CMS fields
        - Data rendered on the frontend
        - API integration
        - Data migration scripts
2. **Checklist:**
    - API Understanding: ✓
    - Schema Validation: ✓
    - Data Migration: ✓
    - API Integration in Next.js: ✓
    - Submission Preparation: ✓

Made by Tayyab Ilyas (Student Leader)