Case Study

Name : Bhavika Ramchandra Vaishnav

Email : bhavikavaishnav29@gmail.com

Part 1: Code Review & Debugging

The code given for adding a product. There are no syntax errors, and it looks proper at first. But after reviewing it carefully, I noticed some important problems that could cause bugs in real use.

1. Check for Missing Data

I made sure that all required inputs like name, sku, price, etc. are present before doing anything. If any field is missing, the API gives a clear error message

2. Check for Duplicate SKU

I added a check to see if the same SKU is already in the system. If it is, we stop the process and return an error message

3. Handle Price Properly Using Decimal

I converted the price into Decimal format so that values are stored correctly.
Without this, computers might store something like garbage value after the decimal point and that can lead to confusion in pricing.

4. Add Error Handling with Rollback

I used try-except block in the code. So if anything fails, It will get error message. This helps users know what went wrong.

Part 2 : Database Design

Design a database schema for an inventory management system called StockFlow, which is used by companies to:

- Manage multiple warehouses
- Track product quantities
- Link products with suppliers
- Track changes in inventory
- Support product bundles

List of Tables Created

1. **companies**
   Stores company information.

2. **warehouses**
   Stores warehouse data and links each warehouse to a company.

3. **products**
   Stores product details like name, SKU, and price.

4. **inventories**
   Stores the quantity of each product in each warehouse.

5. **suppliers**
   Stores supplier details.

6. **product_suppliers**
   Maps which supplier provides which product (many-to-many relation).

7. **inventory_logs**
   Keeps a history of stock changes (like stock added, sold, or removed).

8. **bundles**
   Represents product bundles (e.g., combo packs, kits, gift boxes).

9. **bundle_items**
   Maps which products are included in which bundle and in what quantity.

Part 3 : API Implementation

This API makes the inventory system smarter by helping companies identify products that may soon be out of stock. It only considers in-demand items and adds supplier info, making the alert actionable. This can help companies avoid delivery delays, lost sales, and customer dissatisfaction.

This case study gave me a chance to apply my backend skills in a real-world scenario.

In Part 1, I found important issues in the product creation API and fixed them by adding proper validation, error handling, and business logic to make it more reliable.

In Part 2, I designed a clean and practical database structure that supports multiple warehouses, suppliers, product bundles, and tracks inventory changes — just like a real inventory system would need.

In Part 3, I built an API that shows low stock alerts based on recent sales and helps companies restock on time by also showing supplier details.