StockFlow Technical Assessment Report

Mahesh Shinde

July 23, 2025

Question 1: Fix the Bug in the API

Problem Statement:
The existing Python code returns a list of inventory items for a given store, sorted by quantity. However, it has performance and readability issues, and lacks scalability.

Fixes Implemented in Spring Boot:

- Rewrote the logic using Spring Boot REST API standards.
- Used JPA Repository with a custom query to retrieve sorted inventory.
- Eliminated hardcoded values for better reusability.
- Added ‘@PathVariable‘ for dynamic store ID.

Spring Boot Code:

```
// InventoryController.java
@GetMapping("/store/{storeId}/inventory")
public ResponseEntity<List<Inventory>> getInventoryByStore(@PathVariable
Long storeId) {
    List<Inventory> inventoryList =
    inventoryRepository.findByStoreIdOrderByQuantityAsc(storeId);
    return ResponseEntity.ok(inventoryList);
}

// InventoryRepository.java
List<Inventory> findByStoreIdOrderByQuantityAsc(Long storeId);
```

Question 2: Design the Inventory Database

Entities:

- **Store:** id, name, location
- **Product:** id, name, description
- **Inventory:** id, product_id, store_id, quantity

ER Diagram Description:

- One-to-many: A store can have multiple inventory entries.
- One-to-many: A product can appear in inventories of different stores.

Why This Structure?

- Ensures data normalization.
- Avoids redundancy and supports scalability.
- Improves performance for both read and write operations.

Question 3: Design Low Stock Alert API

Requirement:
Design an API that alerts when the stock of any product in a store goes below a defined threshold.

Spring Boot Solution:

```
// InventoryController.java
@GetMapping("/store/{storeId}/low-stock")
public ResponseEntity<List<Inventory>> getLowStockItems(
    @PathVariable Long storeId,
    @RequestParam(defaultValue = "5") int threshold) {
    List<Inventory> lowStock = inventoryRepository.findByStoreIdAndQuantityLessThan
    (storeId, threshold);
    return ResponseEntity.ok(lowStock);
}

// InventoryRepository.java
List<Inventory> findByStoreIdAndQuantityLessThan(Long storeId, int quantity);
```

Edge Case Handling:

- If no low stock items exist, return an empty list with HTTP 200.
- Threshold value is customizable via query parameter.
- Supports pagination and sorting if required.

Conclusion

All three tasks were implemented using clean, scalable Spring Boot APIs with performance-focused and industry-ready solutions. The database was normalized to ensure efficient relationships, and the alert mechanism was built with edge-case handling in mind.