**1.1 Solution 1: Inventory Management with Angular and Spring Boot**

* **Description:** This basic solution would leverage Angular for a user-friendly interface and Spring Boot for robust backend functionality. Users could add items with basic details (ID, name, cost, price, quantity) using Angular forms. Spring Boot would handle data persistence through Spring Data JPA and expose REST APIs for CRUD operations.
* **Limitations:** This solution lacks features like:
  + Transaction tracking (selling, inserting)
  + Item history view
  + Filtering by ID or name
  + While functional, it wouldn't provide a comprehensive inventory management experience.

**1.2 Solution 2: Inventory Management with Basic Transactions and Angular Filtering**

* **Description:** Building on Solution 1, this approach would introduce functionalities for basic transactions and leverage Angular for item filtering:
  + Selling items: Users could sell items at set prices or custom prices through the Angular interface. Spring Boot would handle the sale logic and update inventory levels.
  + Inserting items: This would allow recording returned items or adding previously missing stock, again using Angular forms and Spring Boot APIs.
  + Filtering by ID or name: We would implement search functionality using keywords to filter the displayed item list. This could utilize Spring Boot's APIs for efficient data retrieval based on filter criteria.
* **Limitations:** While improved, this solution still lacks:
  + Item history view for detailed transaction tracking
  + This might not be sufficient for businesses needing advanced features like purchase orders or low-stock alerts.