STOCKFLOW – INVENTORY MANAGEMENT SYSTEM CASE STUDY

Candidate Name: Om Shete

PART 1: CODE REVIEW & DEBUGGING

Issues Identified:

1. No input validation on request body.
2. SKU uniqueness not enforced.
3. Incorrect product–warehouse relationship.
4. No database transaction handling.
5. No rollback on failure.
6. Price stored as float instead of decimal.
7. Inventory duplication risk.
8. Optional fields not handled.

Impact in Production:

- API crashes due to missing fields.
- Duplicate SKUs corrupt inventory data.
- Products cannot exist in multiple warehouses.
- Partial data committed on failure.
- Financial precision errors.
- Incorrect stock calculations.

Fix Summary:

- Added validation and error handling.
- Enforced SKU uniqueness.
- Removed warehouse_id from Product.
- Used database transaction with rollback.
- Used Decimal for price.
- Prevented duplicate inventory entries.

PART 2: DATABASE DESIGN

Core Tables:

Companies, Warehouses, Products, Inventory, Inventory_Movements, Suppliers, Product_Suppliers, Product_Bundles

Design Highlights:

- Products independent of warehouses.
- Inventory table maps product–warehouse quantity.
- Inventory movements track audit history.
- Unique constraints prevent duplication.
- Bundles supported via self-referencing table.

Missing Requirements / Questions:

- Can products have multiple suppliers?
- Definition of recent sales?
- Supplier lead times?
- Are warehouses company-specific?
- Bundle nesting rules?

PART 3: LOW-STOCK ALERT API

Endpoint:

GET /api/companies/{company_id}/alerts/low-stock

Assumptions:

- Recent sales = last 30 days.
- Threshold defined per product.
- Sales tracked via inventory movements.

Logic:

- Iterate company warehouses.

- Check inventory below threshold.
- Verify recent sales activity.
- Calculate days until stockout.
- Attach supplier details.

Edge Cases:
- No recent sales.
- Zero daily sales.
- Missing supplier.
- Multiple warehouses.

Conclusion:
This solution focuses on data integrity, scalability, and real-world business constraints while handling ambiguity through explicit assumptions.