**StockFlow Take-Home:**

**Part 1 — Code Review & Fixes**

**Issues I Found**

* **SKU not checked unique** → can cause duplicate products.
* **Product has warehouse\_id** → wrong, product can be in many warehouses.
* **Price stored as float** → may cause rounding errors.
* **No input validation** → breaks if fields missing.
* **Two commits** → if 2nd fails, DB inconsistent.
* **Inventory row duplication** → should check if already exists.
* **Always returns 200** → should return 201 or errors.

**Simple Fix**

* Remove warehouse\_id from Product.
* Use one transaction (flush, then commit once).
* Validate fields before saving.
* Use Decimal for price.
* Handle duplicate SKU with try/except.

@app.route('/api/products', methods=['POST'])

def create\_product():

data = request.get\_json() or {}

if 'sku' not in data: return {"error": "Missing sku"}, 400

try:

product = Product(name=data['name'], sku=data['sku'], price=Decimal(str(data['price'])))

db.session.add(product)

db.session.flush()

inv = Inventory(product\_id=product.id, warehouse\_id=data['warehouse\_id'], quantity=data.get('initial\_quantity',0))

db.session.add(inv)

db.session.commit()

except IntegrityError:

db.session.rollback()

return {"error": "SKU already exists"}, 409

return {"message": "Product created", "product\_id": product.id}, 201

**Part 2 — Database Design**

* **companies(id, name)**
* **warehouses(id, company\_id, name)**
* **products(id, sku UNIQUE, name, price, type)**
* **inventories(id, product\_id, warehouse\_id, quantity)**
* **suppliers(id, name, email)**
* **supplier\_products(supplier\_id, product\_id, price)**
* **bundle\_items(bundle\_product\_id, component\_product\_id, qty)**

**Questions I’d Ask**

* Are SKUs unique globally or per company?
* How to handle bundles (reduce child stock?)
* Do we need audit trail of stock changes?
* Who owns warehouses? Do they have addresses?

**Part 3 — Low Stock API**

**My Assumptions**

* Low stock threshold is saved in products.threshold or default 10.
* "Recent sales" = sales in last 30 days.
* Days until stockout = current\_stock / avg\_daily\_sales.
* Take first supplier if many.

**Simple Implementation (Flask)**

@app.route('/api/companies/<int:company\_id>/alerts/low-stock')

def low\_stock(company\_id):

default\_threshold = 10

recent\_from = datetime.utcnow() - timedelta(days=30)

# find products with recent sales + low stock

q = db.session.query(Product, Inventory, Warehouse).join(Inventory).join(Warehouse)

q = q.filter(Warehouse.company\_id==company\_id, Inventory.quantity < default\_threshold)

alerts = []

for p, inv, wh in q:

avg\_daily = 1 # beginner simplification

days\_left = inv.quantity // avg\_daily

supplier = Supplier.query.join(SupplierProduct).filter(SupplierProduct.product\_id==p.id).first()

alerts.append({

'product\_id': p.id,

'product\_name': p.name,

'sku': p.sku,

'warehouse\_id': wh.id,

'warehouse\_name': wh.name,

'current\_stock': inv.quantity,

'threshold': default\_threshold,

'days\_until\_stockout': days\_left,

'supplier': {'id': supplier.id, 'name': supplier.name, 'contact\_email': supplier.contact\_email} if supplier else None

})

return {"alerts": alerts, "total\_alerts": len(alerts)}

**Final Note**

I kept it simple because I’m a fresher and honestly speaking I don’t have that much of experience in coding. I want to work as an intern to improve my knowledge and gain hands on experience and learn how things work in the company itself. I know it’s not perfect but I showed how I think and what I’d ask if I was in a real project.