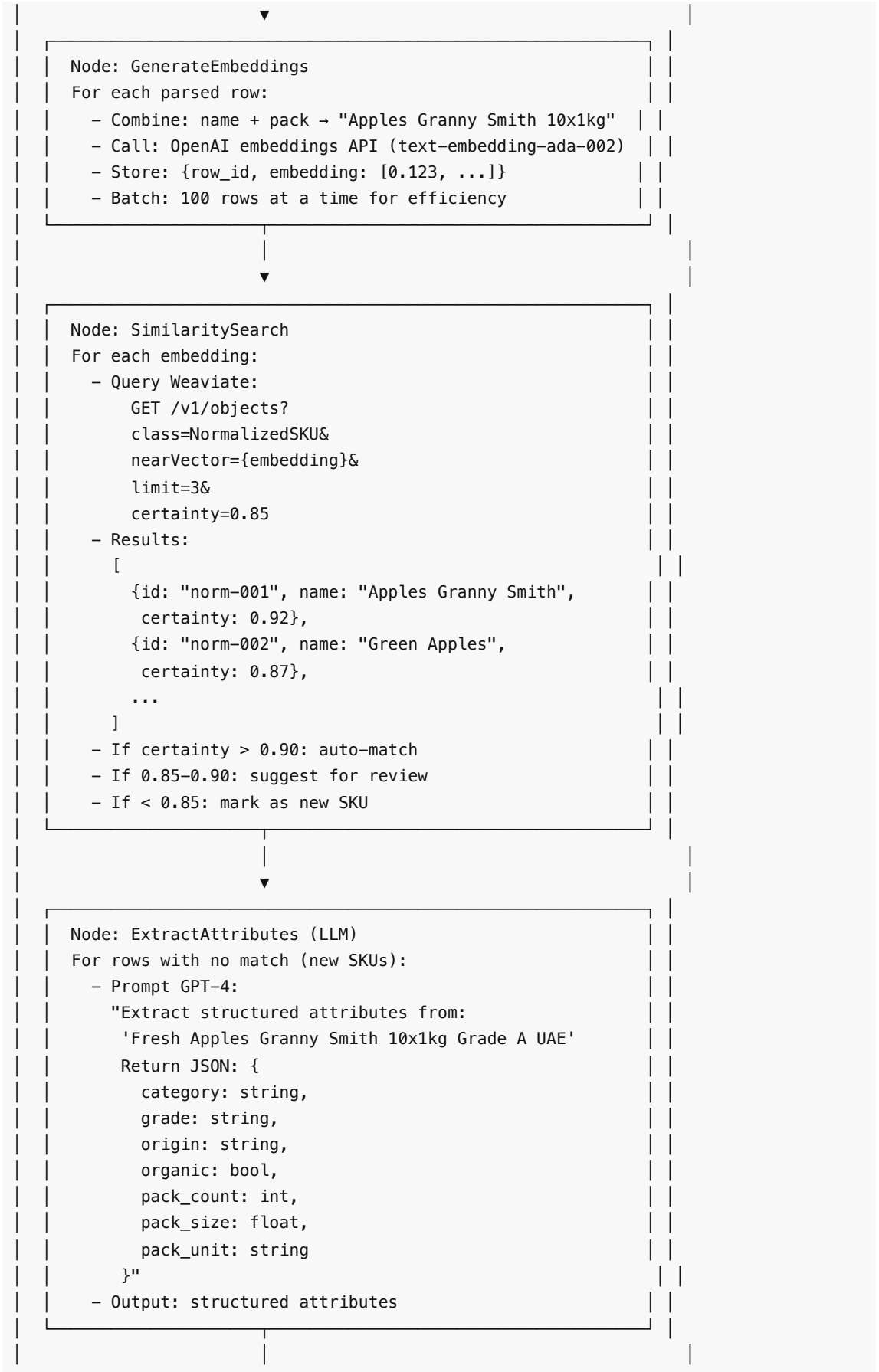
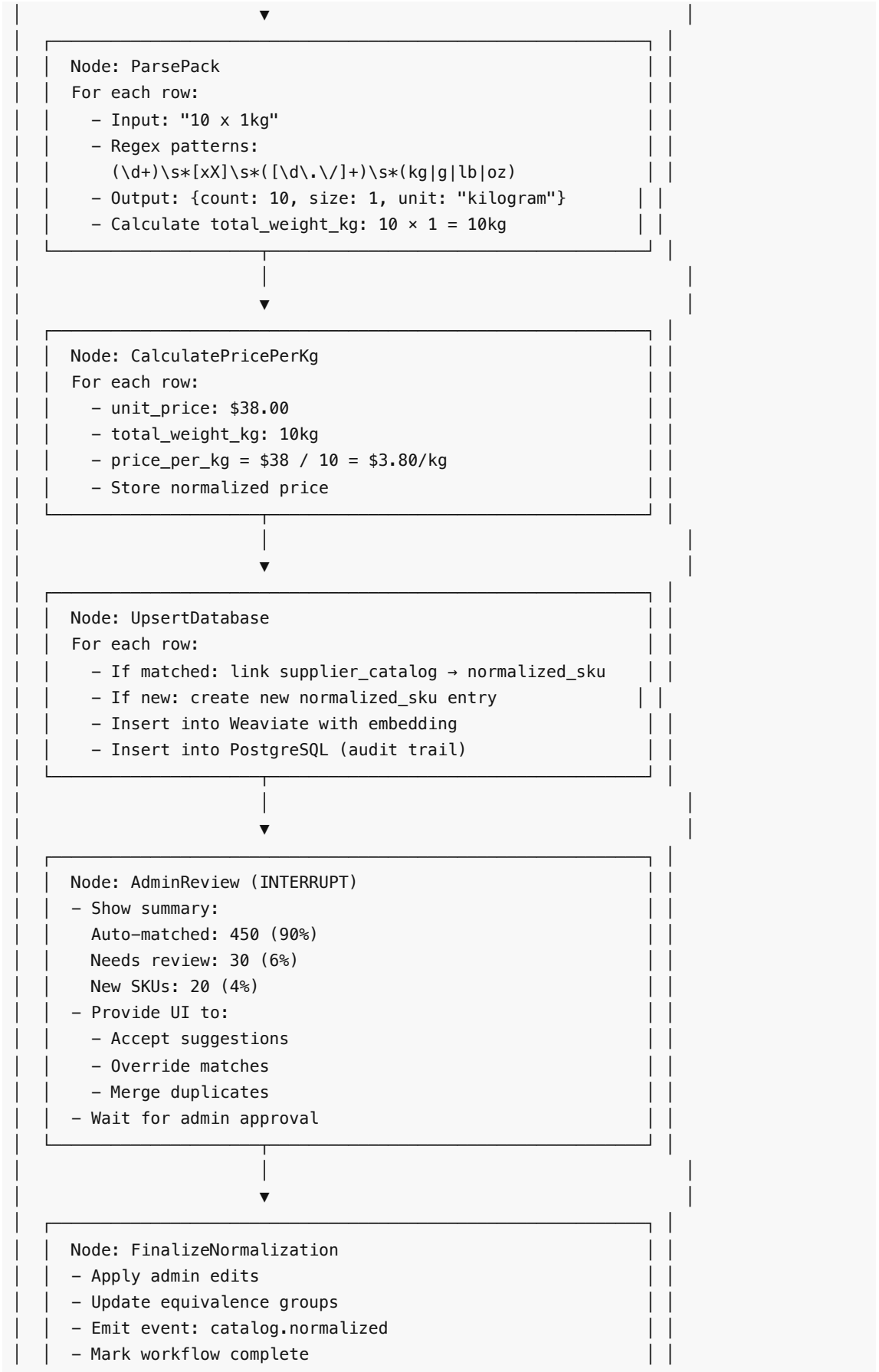
Detailed Data Flows

Flow 1: Supplier Catalog Upload → SKU Normalization


Node: GenerateEmbeddings
For each parsed row:
- Combine: name + pack → "Apples Granny Smith 10x1kg"
- Call: OpenAI embeddings API (text-embedding-ada-002)
- Store: {row_id, embedding: [0.123, ...]}
- Batch: 100 rows at a time for efficiency
Node: SimilaritySearch
For each embedding:
- Query Weaviate:
GET /v1/objects?
class=NormalizedSKU&
nearVector={embedding}&
limit=3&
certainty=0.85
- Results:
[
{id: "norm-001", name: "Apples Granny Smith",
certainty: 0.92},
{id: "norm-002", name: "Green Apples",
certainty: 0.87},
...
]
- If certainty > 0.90: auto-match
- If 0.85-0.90: suggest for review
- If < 0.85: mark as new SKU
Node: ExtractAttributes (LLM)
For rows with no match (new SKUs):
- Prompt GPT-4:
"Extract structured attributes from:
'Fresh Apples Granny Smith 10x1kg Grade A UAE'
Return JSON: {
category: string,
grade: string,
origin: string,
organic: bool,
pack_count: int,
pack_size: float,
pack_unit: string
}"
- Output: structured attributes


▼
Node: ParsePack
For each row:
- Input: "10 x 1kg"
- Regex patterns:
(\d+)\s*[xX]\s*([\d\.\/]+)\s*(kg|g|lb|oz)
- Output: {count: 10, size: 1, unit: "kilogram"}
- Calculate total_weight_kg: 10 × 1 = 10kg
▼
Node: CalculatePricePerKg
For each row:
- unit_price: $38.00
- total_weight_kg: 10kg
- price_per_kg = $38 / 10 = $3.80/kg
- Store normalized price
▼
Node: UpsertDatabase
For each row:
- If matched: link supplier_catalog → normalized_sku
- If new: create new normalized_sku entry
- Insert into Weaviate with embedding
- Insert into PostgreSQL (audit trail)
▼
Node: AdminReview (INTERRUPT)
- Show summary:
Auto-matched: 450 (90%)
Needs review: 30 (6%)
New SKUs: 20 (4%)
- Provide UI to:
- Accept suggestions
- Override matches
- Merge duplicates
- Wait for admin approval
▼
Node: FinalizeNormalization
- Apply admin edits
- Update equivalence groups
- Emit event: catalog.normalized
- Mark workflow complete

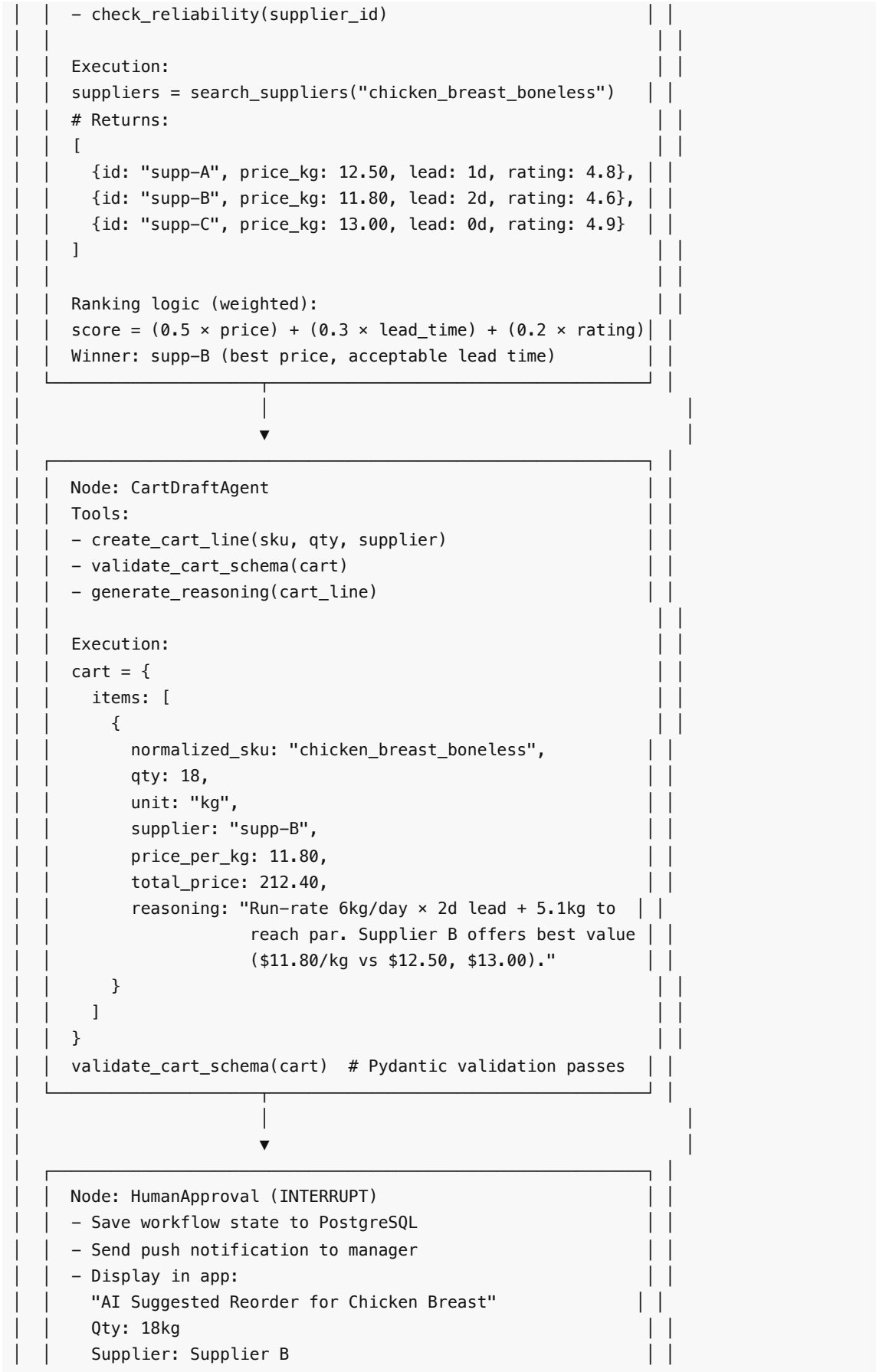
```
│  └──────────────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
                    Normalization Complete
                    500 SKUs processed in ~5 minutes
```

Flow 2: POS Sales → Inventory Depletion → Low Stock Alert → AI Cart

```
┌──────────────────────────────────────────────────────────────┐
│                    Restaurant POS (Foodics)                  │
│  Customer orders:                                            │
│  - 2x Chicken Burger ($15 each)                              │
│  - 1x Caesar Salad ($12)                                     │
│  Total: $42                                                  │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
                    Foodics Webhook: order.created
                    POST https://your-app.com/webhooks/foodics
                              │
                              ▼
┌──────────────────────────────────────────────────────────────┐
│              MedusaJS Webhook Handler                        │
│  {                                                           │
│    event: "order.created",                                   │
│    order_id: "foodics-order-12345",                          │
│    items: [                                                  │
│      {product_id: "food-001", name: "Chicken Burger", qty: 2},  │
│      {product_id: "food-002", name: "Caesar Salad", qty: 1}     │
│    ]                                                         │
│  }                                                           │
│                                                              │
│  Actions:                                                    │
│  1. Map Foodics product_id → internal recipe_id              │
│  2. Fetch recipe BOM (Bill of Materials)                     │
│  3. Deplete inventory                                        │
└──────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────┐
│                   Recipe BOM Expansion                       │
│                                                              │
│  Recipe: Chicken Burger (food-001)                           │
│  Ingredients:                                                │
│  - Chicken Breast: 150g (norm-sku: "chicken_breast_boneless")  │
│  - Burger Bun: 1 unit (norm-sku: "bun_burger_white")         │
│  - Lettuce: 20g (norm-sku: "lettuce_iceberg")                │
│  - Tomato: 30g (norm-sku: "tomato_fresh")                    │
│  - Sauce: 15ml (norm-sku: "mayo_regular")                    │
```

```
│                                                                  │
│  For 2 burgers:                                                  │
│  - Chicken Breast: 300g                                          │
│  - Burger Bun: 2 units                                           │
│  - Lettuce: 40g                                                  │
│  - Tomato: 60g                                                   │
│  - Sauce: 30ml                                                   │
└──────────────────────────────┬───────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────┐
│                 Inventory Depletion (Transactional)              │
│                                                                  │
│  BEGIN TRANSACTION;                                              │
│                                                                  │
│  UPDATE inventory                                                │
│  SET qty_on_hand = qty_on_hand - 0.3  -- 300g = 0.3kg            │
│  WHERE normalized_sku_id = 'chicken_breast_boneless'             │
│    AND branch_id = 'branch-001';                                 │
│                                                                  │
│  UPDATE inventory                                                │
│  SET qty_on_hand = qty_on_hand - 2                               │
│  WHERE normalized_sku_id = 'bun_burger_white'                    │
│    AND branch_id = 'branch-001';                                 │
│                                                                  │
│  ... (repeat for all ingredients)                                │
│                                                                  │
│  INSERT INTO inventory_logs (normalized_sku_id, change_qty,      │
│    reason, created_at)                                           │
│  VALUES ('chicken_breast_boneless', -0.3, 'POS sale', NOW());    │
│                                                                  │
│  COMMIT;                                                         │
│                                                                  │
│  After update:                                                   │
│  - Chicken Breast: 5.2kg → 4.9kg                                 │
│  - Par level: 10kg                                               │
│  - Status: BELOW PAR (4.9kg < 10kg)                              │
└──────────────────────────────┬───────────────────────────────────┘
                               │
                               ▼
┌──────────────────────────────────────────────────────────────────┐
│             Low Stock Detection (Trigger Check)                  │
│                                                                  │
│  IF qty_on_hand < reorder_point THEN                             │
│    Emit event: inventory.low_stock                               │
│                                                                  │
│  Event payload:                                                  │
│  {                                                               │
│    event: "inventory.low_stock",                                 │
│    branch_id: "branch-001",                                      │
│    normalized_sku_id: "chicken_breast_boneless",                 │
│    current_qty: 4.9,                                             │
```

```
│    par_level: 10.0,                                                    │
│    reorder_point: 6.0,                                                 │
│    unit: "kg",                                                         │
│    lead_time_days: 2                                                   │
│  }                                                                     │
└────────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌────────────────────────────────────────────────────────────────────────┐
│        Event Subscriber: AutoReorderSubscriber                         │
│  Listen for: inventory.low_stock                                       │
│  Action: Trigger LangGraph AutoReorderWorkflow                         │
└────────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌────────────────────────────────────────────────────────────────────────┐
│            LangGraph: AutoReorderWorkflow                              │
│                                                                        │
│  State: {                                                              │
│    branch_id: "branch-001",                                            │
│    low_stock_items: [                                                  │
│      {sku: "chicken_breast_boneless", qty: 4.9kg, par: 10kg}           │
│    ],                                                                  │
│    suggested_cart: null,                                               │
│    approval_status: "pending"                                          │
│  }                                                                     │
│                                                                        │
│  ┌──────────────────────────────────────────────────────────────────┐  │
│  │  Node: ProcurementAgent                                          │  │
│  │  Tools:                                                          │  │
│  │  - fetch_sales_history(sku, days=30)                             │  │
│  │  - calc_run_rate(sales_history)                                  │  │
│  │  - get_lead_time(sku)                                            │  │
│  │                                                                  │  │
│  │  Execution:                                                      │  │
│  │  sales = fetch_sales_history("chicken_breast", 30)               │  │
│  │  # Returns: 180kg sold in 30 days                                │  │
│  │  run_rate = 180kg / 30 = 6kg/day                                 │  │
│  │  lead_time = 2 days                                              │  │
│  │  safety_stock = run_rate × 0.5 = 3kg                             │  │
│  │  suggested_qty = (par - current) + (run_rate × lead_time)        │  │
│  │                = (10 - 4.9) + (6 × 2)                            │  │
│  │                = 5.1 + 12 = 17.1kg                               │  │
│  │  Round up: 18kg                                                  │  │
│  └──────────────────────────────────────────────────────────────────┘  │
│                         │                                              │
│                         ▼                                              │
│  ┌──────────────────────────────────────────────────────────────────┐  │
│  │  Node: SourcingAgent                                             │  │
│  │  Tools:                                                          │  │
│  │  - search_suppliers(normalized_sku_id)                           │  │
│  │  - compare_prices(suppliers)                                     │  │
```


- check_reliability(supplier_id)
Execution:
suppliers = search_suppliers("chicken_breast_boneless")
# Returns:
[
{id: "supp-A", price_kg: 12.50, lead: 1d, rating: 4.8},
{id: "supp-B", price_kg: 11.80, lead: 2d, rating: 4.6},
{id: "supp-C", price_kg: 13.00, lead: 0d, rating: 4.9}
]
Ranking logic (weighted):
score = (0.5 × price) + (0.3 × lead_time) + (0.2 × rating)
Winner: supp-B (best price, acceptable lead time)
Node: CartDraftAgent
Tools:
- create_cart_line(sku, qty, supplier)
- validate_cart_schema(cart)
- generate_reasoning(cart_line)
Execution:
cart = {
items: [
{
normalized_sku: "chicken_breast_boneless",
qty: 18,
unit: "kg",
supplier: "supp-B",
price_per_kg: 11.80,
total_price: 212.40,
reasoning: "Run-rate 6kg/day × 2d lead + 5.1kg to
reach par. Supplier B offers best value
($11.80/kg vs $12.50, $13.00)."
}
]
}
validate_cart_schema(cart) # Pydantic validation passes
Node: HumanApproval (INTERRUPT)
- Save workflow state to PostgreSQL
- Send push notification to manager
- Display in app:
"AI Suggested Reorder for Chicken Breast"
Qty: 18kg
Supplier: Supplier B

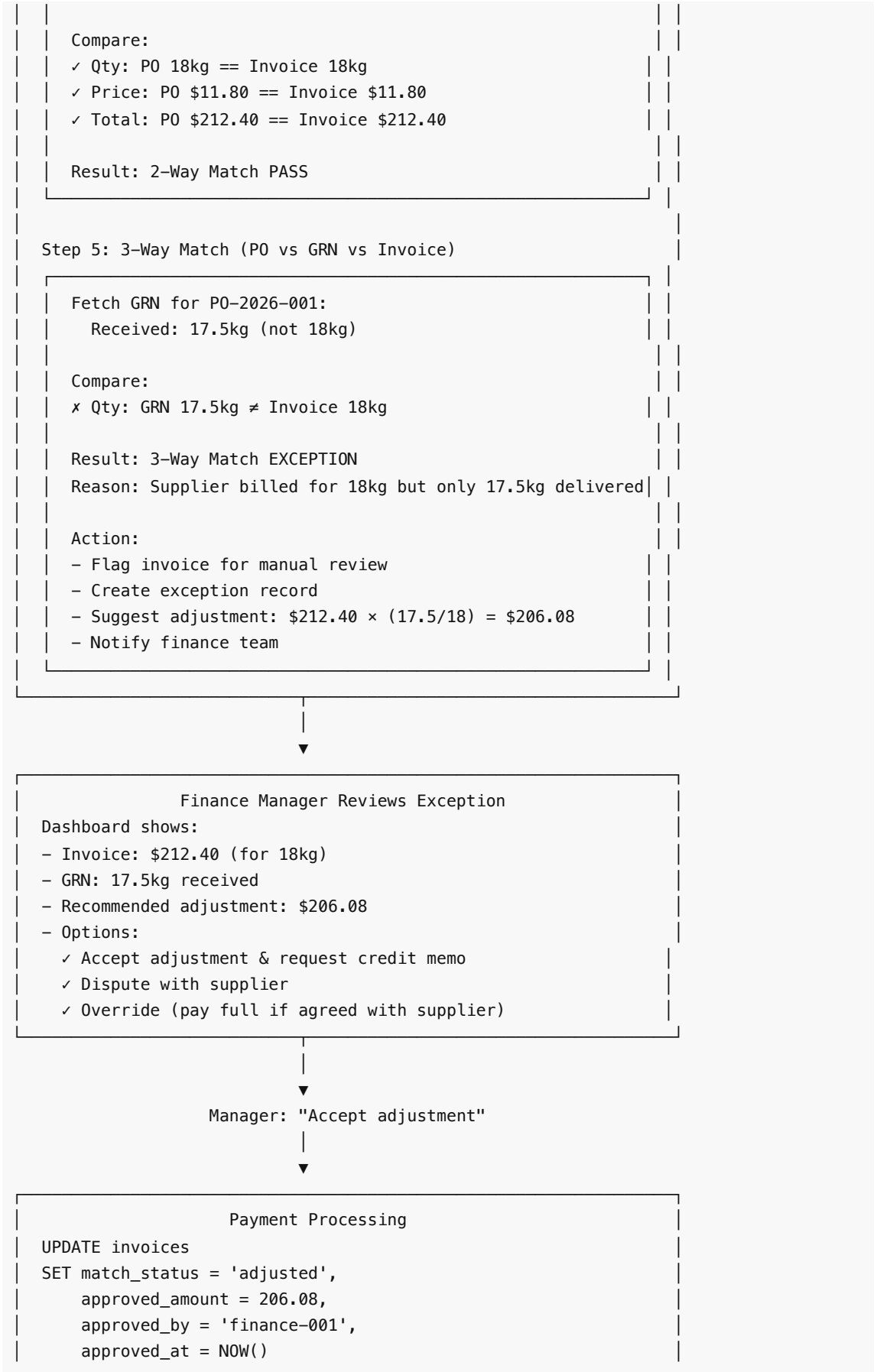
```
│   │    Price: $212.40                                          │ │
│   │    Reasoning: [show above]                                 │ │
│   │  - Wait for user action: approve/edit/reject               │ │
│   └────────────────────────┬──────────────────────────────────┘ │
│                            │                                     │
│                            ▼                                     │
│        Manager clicks "Approve"                                  │
│                            │                                     │
│                            ▼                                     │
│   ┌──────────────────────────────────────────────────────────┐   │
│   │  Node: CreatePOAgent                                     │ │
│   │  Tools:                                                  │ │
│   │  - medusa.orders.create(cart_data)                       │ │
│   │  - notify_supplier(order_id)                             │ │
│   │  - schedule_grn_reminder(delivery_date)                  │ │
│   │                                                          │ │
│   │  Execution:                                              │ │
│   │  order = medusa.orders.create({                          │ │
│   │    branch_id: "branch-001",                              │ │
│   │    supplier_id: "supp-B",                                │ │
│   │    items: [...cart.items],                               │ │
│   │    status: "confirmed",                                  │ │
│   │    created_by: "ai-agent",                               │ │
│   │    approved_by: "manager-001"                            │ │
│   │  })                                                      │ │
│   │  # Returns: {order_id: "PO-2026-001", status: "confirmed"} │ │
│   │                                                          │ │
│   │  notify_supplier(order.id)  # Send email/API notification  │ │
│   │  schedule_grn_reminder(order.delivery_date)              │ │
│   └────────────────────────┬──────────────────────────────────┘ │
│                            │                                     │
│                            ▼                                     │
│   ┌──────────────────────────────────────────────────────────┐   │
│   │  Node: FinalizeWorkflow                                  │ │
│   │  - Log entire workflow execution to audit_logs           │ │
│   │  - Emit event: order.created_by_ai                       │ │
│   │  - Mark workflow state as "completed"                    │ │
│   │  - Send confirmation to manager                          │ │
│   └──────────────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────────────┘
                             │
                             ▼
                  PO Created: PO-2026-001
                  Chicken Breast 18kg @ $11.80/kg
                  Delivery: 2026-02-06 (2 days)

                  Total time: 2 seconds
                  (excluding human approval wait)
```

Flow 3: GRN (Goods Received) → Invoice Match → Payment


Delivery Arrives at Restaurant
Driver brings:
- 18kg Chicken Breast (2 boxes × 9kg)
- PO Reference: PO-2026-001
Storekeeper Opens GRN Module (Mobile App)
- Scan PO barcode or enter PO-2026-001
- App loads PO details:
Expected: 18kg Chicken Breast
Supplier: Supplier B
Receiving Workflow (GRN)
Step 1: Quantity Check
Expected: 18kg
Received: [User weighs] 17.5kg
Discrepancy: -0.5kg (short delivery)
Action: Flag for later claim
Step 2: Quality Check
Condition: [✓] Good [ ] Damaged [ ] Expired
Notes: "Minor packaging wear but product OK"
Photos: [Upload 2 photos of boxes]
Step 3: Signature
Driver signature: [Canvas signature pad]
Receiver signature: [Canvas signature pad]
Timestamp: 2026-02-06 08:15 AM
Step 4: Submit GRN
POST /admin/grn/create
{
po_id: "PO-2026-001",
items: [
{
normalized_sku_id: "chicken_breast_boneless",
qty_expected: 18.0,

```
│  │        qty_received: 17.5,                                    │ │
│  │        unit: "kg",                                            │ │
│  │        quality: "good",                                       │ │
│  │        notes: "Minor packaging wear",                         │ │
│  │        photos: ["s3://grn/photo1.jpg", "..."]                 │ │
│  │      }                                                        │ │
│  │    ],                                                         │ │
│  │    signatures: {driver: "...", receiver: "..."},              │ │
│  │    received_at: "2026-02-06T08:15:00Z"                        │ │
│  │  }                                                            │ │
│  └──────────────────────────────────────────────────────────────┘ │
└──────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────────┐
│                    GRN Processing (Backend)                      │
│                                                                  │
│  BEGIN TRANSACTION;                                              │
│                                                                  │
│  1. Insert GRN record                                            │
│  INSERT INTO grn (po_id, received_by, status, ...)               │
│  VALUES ('PO-2026-001', 'john', 'completed', ...);               │
│                                                                  │
│  2. Update inventory (add received qty)                          │
│  UPDATE inventory                                                │
│  SET qty_on_hand = qty_on_hand + 17.5                            │
│  WHERE normalized_sku_id = 'chicken_breast_boneless'             │
│    AND branch_id = 'branch-001';                                 │
│  # New stock: 4.9 + 17.5 = 22.4kg                                │
│                                                                  │
│  3. Update PO status                                             │
│  UPDATE orders                                                   │
│  SET status = 'delivered', grn_id = [new_grn_id]                 │
│  WHERE id = 'PO-2026-001';                                       │
│                                                                  │
│  4. Create claim if discrepancy                                  │
│  IF qty_received < qty_expected THEN                             │
│    INSERT INTO claims (grn_id, type, qty_diff, status)           │
│    VALUES ([grn_id], 'short_delivery', -0.5, 'open');            │
│  END IF;                                                         │
│                                                                  │
│  5. Emit event: grn.completed                                    │
│  COMMIT;                                                         │
└──────────────────────────────────────────────────────────────────┘
                              │
                              ▼
┌──────────────────────────────────────────────────────────────────┐
│             Supplier Uploads Invoice (Next Day)                  │
│  Supplier logs in, uploads PDF invoice                           │
│  POST /supplier/invoice/upload                                   │
│  File: invoice_B_12345.pdf                                       │
└──────────────────────────────────────────────────────────────────┘
                              │
```

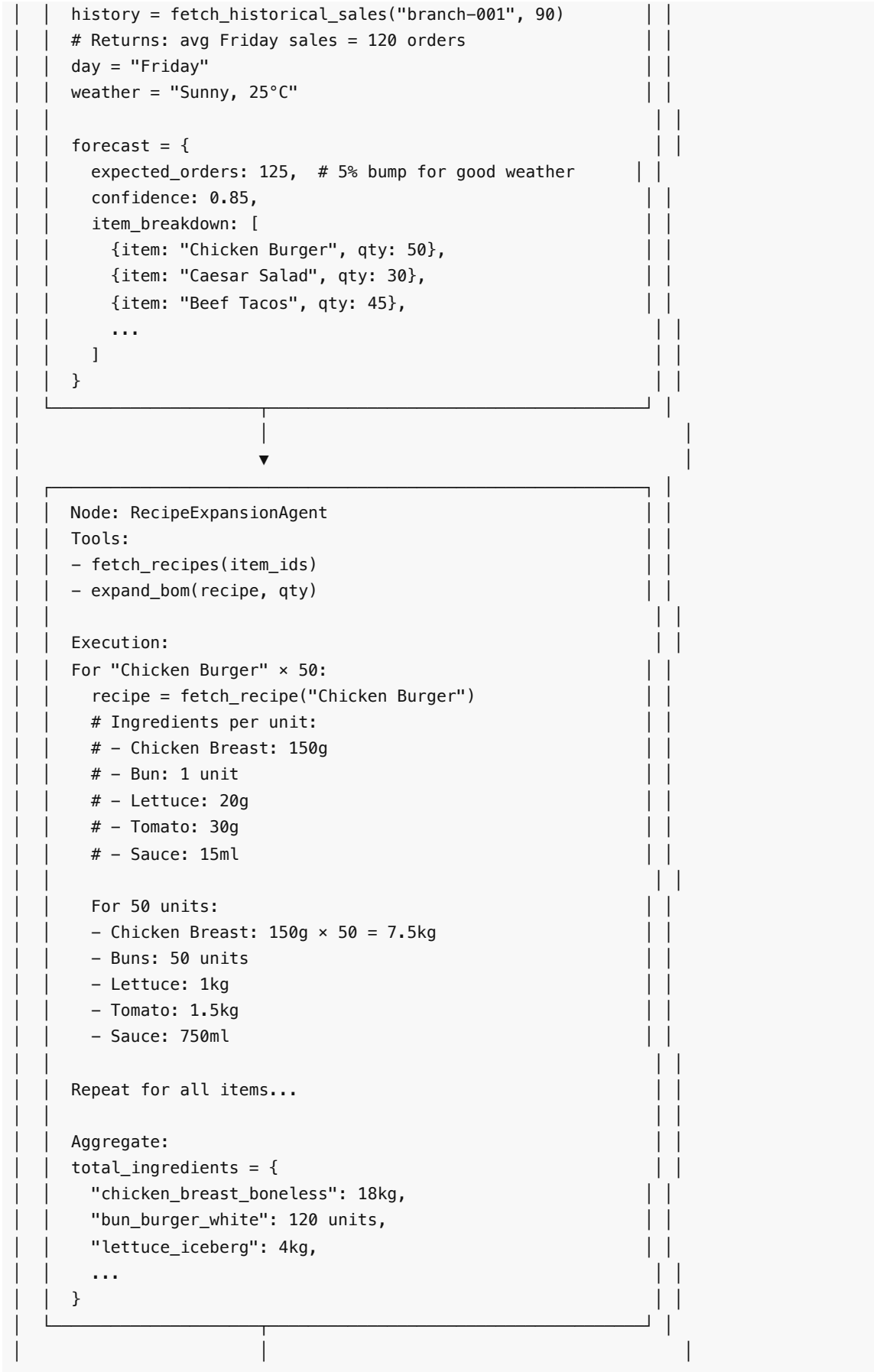

Invoice OCR Pipeline
Step 1: Store file
S3: invoices/supp-B/invoice_B_12345.pdf
Step 2: AWS Textract AnalyzeExpense
API Call:
textract.analyze_expense(document=pdf)
Extracted:
{
invoice_number: "INV-B-12345",
invoice_date: "2026-02-05",
vendor: "Supplier B LLC",
total: "$212.40",
line_items: [
{
description: "Chicken Breast Boneless Fresh",
quantity: "18 KG",
unit_price: "$11.80",
line_total: "$212.40"
}
]
}
Step 3: LangGraph Agent: InvoiceValidationAgent
Tools:
- parse_invoice_line(description)
- map_to_normalized_sku(description)
- validate_price(invoice_price, po_price)
Execution:
line = "Chicken Breast Boneless Fresh 18 KG"
sku = map_to_normalized_sku(line)
# Returns: "chicken_breast_boneless"
Validation:
invoice_qty = 18kg
invoice_price_kg = $11.80
invoice_total = $212.40
Step 4: 2-Way Match (PO vs Invoice)
Fetch PO-2026-001:
Expected: 18kg @ $11.80/kg = $212.40

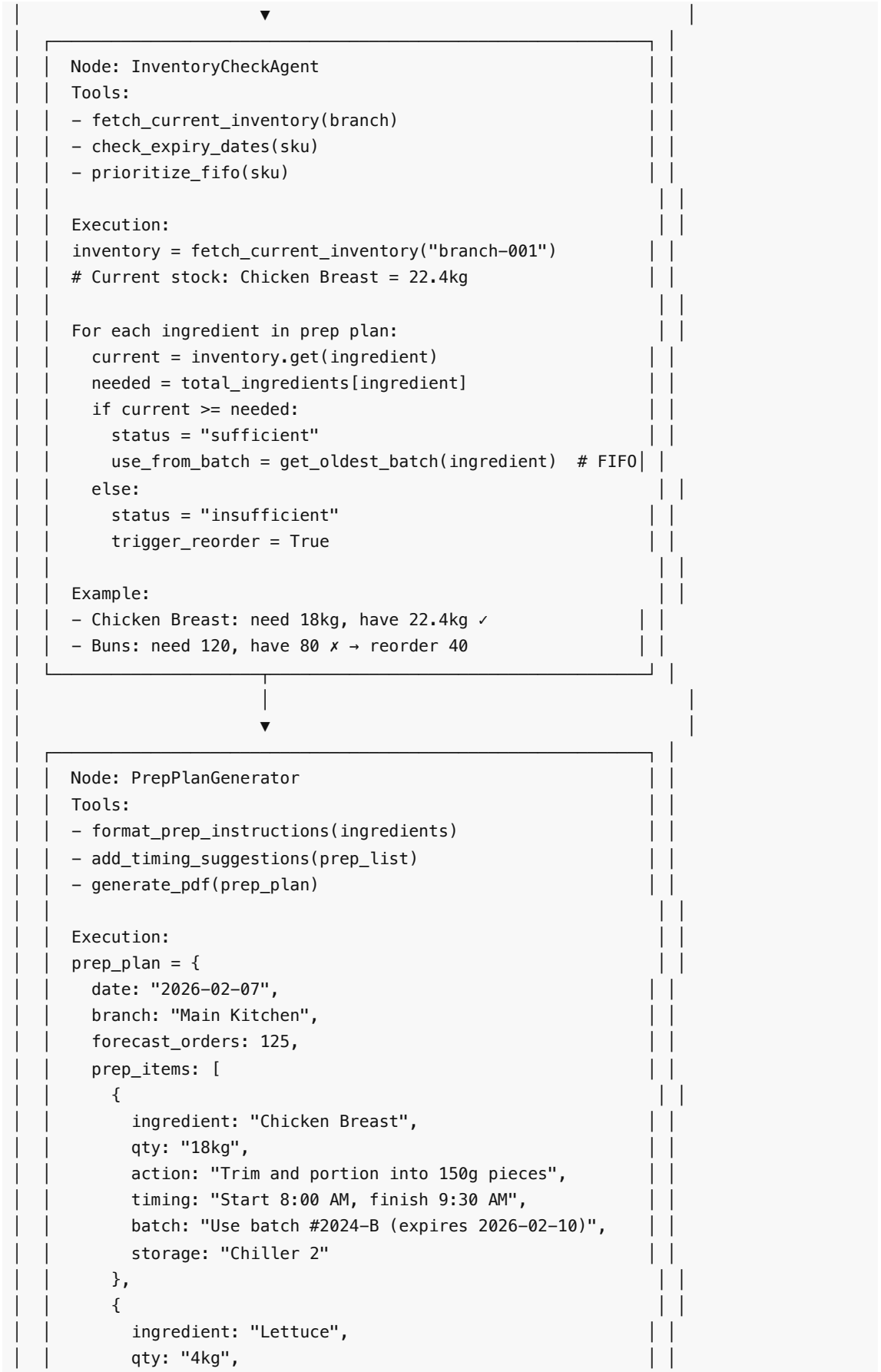

Compare:
✓ Qty: PO 18kg == Invoice 18kg
✓ Price: PO $11.80 == Invoice $11.80
✓ Total: PO $212.40 == Invoice $212.40
Result: 2-Way Match PASS
Step 5: 3-Way Match (PO vs GRN vs Invoice)
Fetch GRN for PO-2026-001:
Received: 17.5kg (not 18kg)
Compare:
✗ Qty: GRN 17.5kg ≠ Invoice 18kg
Result: 3-Way Match EXCEPTION
Reason: Supplier billed for 18kg but only 17.5kg delivered
Action:
- Flag invoice for manual review
- Create exception record
- Suggest adjustment: $212.40 × (17.5/18) = $206.08
- Notify finance team
Finance Manager Reviews Exception
Dashboard shows:
- Invoice: $212.40 (for 18kg)
- GRN: 17.5kg received
- Recommended adjustment: $206.08
- Options:
✓ Accept adjustment & request credit memo
✓ Dispute with supplier
✓ Override (pay full if agreed with supplier)
Manager: "Accept adjustment"
Payment Processing
UPDATE invoices
SET match_status = 'adjusted',
approved_amount = 206.08,
approved_by = 'finance-001',
approved_at = NOW()

```
│  WHERE id = [invoice_id];                                                  │
│                                                                            │
│  Emit event: invoice.approved                                              │
│                                                                            │
│  IF payment_terms = 'net_30' THEN                                          │
│    Schedule payment for 30 days from invoice_date                          │
│  ELSE                                                                      │
│    Process payment immediately via gateway                                 │
│  END IF;                                                                   │
└────────────────────────────────────────────────────────────────────────────┘
```

Flow 4: Kitchen Copilot (Daily Prep Plan Generation)

```
Every day at 5:00 AM (scheduled cron)
         │
         ▼
┌────────────────────────────────────────────────────────────────────────────┐
│           Cron Job: Generate Daily Prep Plans                              │
│  For each branch:                                                          │
│    Emit event: kitchen.prep_plan_requested                                 │
└──────────────────────────────┬─────────────────────────────────────────────┘
                               │
                               ▼
┌────────────────────────────────────────────────────────────────────────────┐
│      Event Subscriber: PrepPlanSubscriber                                  │
│  Listen for: kitchen.prep_plan_requested                                   │
│  Action: Trigger LangGraph PrepPlanWorkflow                                │
└──────────────────────────────┬─────────────────────────────────────────────┘
                               │
                               ▼
┌────────────────────────────────────────────────────────────────────────────┐
│           LangGraph: PrepPlanWorkflow                                      │
│                                                                            │
│  State: {                                                                  │
│    branch_id: "branch-001",                                                │
│    date: "2026-02-07",                                                     │
│    forecast: null,                                                         │
│    recipes: null,                                                          │
│    prep_plan: null                                                         │
│  }                                                                         │
│                                                                            │
│  ┌──────────────────────────────────────────────────────────────────┐      │
│  │  Node: ForecastAgent                                             │      │
│  │  Tools:                                                          │      │
│  │  - fetch_historical_sales(branch, days=90)                       │      │
│  │  - get_day_of_week()                                             │      │
│  │  - get_weather_forecast()  # optional                            │      │
│  │  - apply_forecast_model(history, context)                        │      │
│  │                                                                  │      │
│  │  Execution:                                                      │      │
```


history = fetch_historical_sales("branch-001", 90)
# Returns: avg Friday sales = 120 orders
day = "Friday"
weather = "Sunny, 25°C"
forecast = {
expected_orders: 125, # 5% bump for good weather
confidence: 0.85,
item_breakdown: [
{item: "Chicken Burger", qty: 50},
{item: "Caesar Salad", qty: 30},
{item: "Beef Tacos", qty: 45},
...
]
}
Node: RecipeExpansionAgent
Tools:
- fetch_recipes(item_ids)
- expand_bom(recipe, qty)
Execution:
For "Chicken Burger" × 50:
recipe = fetch_recipe("Chicken Burger")
# Ingredients per unit:
# - Chicken Breast: 150g
# - Bun: 1 unit
# - Lettuce: 20g
# - Tomato: 30g
# - Sauce: 15ml
For 50 units:
- Chicken Breast: 150g × 50 = 7.5kg
- Buns: 50 units
- Lettuce: 1kg
- Tomato: 1.5kg
- Sauce: 750ml
Repeat for all items...
Aggregate:
total_ingredients = {
"chicken_breast_boneless": 18kg,
"bun_burger_white": 120 units,
"lettuce_iceberg": 4kg,
...
}


▼
Node: InventoryCheckAgent
Tools:
- fetch_current_inventory(branch)
- check_expiry_dates(sku)
- prioritize_fifo(sku)
Execution:
inventory = fetch_current_inventory("branch-001")
# Current stock: Chicken Breast = 22.4kg
For each ingredient in prep plan:
current = inventory.get(ingredient)
needed = total_ingredients[ingredient]
if current >= needed:
status = "sufficient"
use_from_batch = get_oldest_batch(ingredient) # FIFO
else:
status = "insufficient"
trigger_reorder = True
Example:
- Chicken Breast: need 18kg, have 22.4kg ✓
- Buns: need 120, have 80 ✗ → reorder 40
▼
Node: PrepPlanGenerator
Tools:
- format_prep_instructions(ingredients)
- add_timing_suggestions(prep_list)
- generate_pdf(prep_plan)
Execution:
prep_plan = {
date: "2026-02-07",
branch: "Main Kitchen",
forecast_orders: 125,
prep_items: [
{
ingredient: "Chicken Breast",
qty: "18kg",
action: "Trim and portion into 150g pieces",
timing: "Start 8:00 AM, finish 9:30 AM",
batch: "Use batch #2024-B (expires 2026-02-10)",
storage: "Chiller 2"
},
{
ingredient: "Lettuce",
qty: "4kg",

```
│  │        action: "Wash, core, and chop",                        │ │
│  │        timing: "Start 9:00 AM, finish 9:45 AM",               │ │
│  │        batch: "Use batch #2024-L (expires 2026-02-08)",       │ │
│  │        storage: "Walk-in cooler"                              │ │
│  │      },                                                       │ │
│  │      ...                                                      │ │
│  │    ],                                                         │ │
│  │    alerts: [                                                  │ │
│  │      "Buns low: only 80 available, need 120. Reorder 40."     │ │
│  │    ]                                                          │ │
│  │  }                                                            │ │
│  │                                                               │ │
│  │  generate_pdf(prep_plan)                                      │ │
│  │  # Save to: s3://prep-plans/branch-001/2026-02-07.pdf         │ │
│  └──────────────────────┬────────────────────────────────────────┘ │
│                         │                                          │
│                         ▼                                          │
│  ┌──────────────────────────────────────────────────────────────┐  │
│  │  Node: NotifyChef                                            │  │
│  │  - Send push notification to chef app                        │  │
│  │  - Send email with PDF attachment                            │  │
│  │  - Display in-app: "Prep Plan Ready for Feb 7"               │  │
│  │  - Mark workflow complete                                    │  │
│  └──────────────────────────────────────────────────────────────┘  │
└────────────────────────────────────────────────────────────────────┘
                              │
                              ▼
                Chef receives prep plan at 6:00 AM
                Ready to start prep at 8:00 AM
```

Summary: Key Integration Points

Flow	Trigger	LangGraph Workflow	Outcome
Catalog Upload	Supplier uploads CSV	`CatalogNormalizationWorkflow`	SKUs normalized, ready for comparison
Low Stock	POS sale depletes inventory	`AutoReorderWorkflow`	AI-suggested cart → Manager approval → PO
GRN	Delivery received	Manual entry → 3-way match	Inventory updated, invoice matched
Invoice OCR	Supplier uploads PDF	`InvoiceValidationAgent`	Extracted, matched, payment scheduled
Prep Plan	Daily cron (5 AM)	`PrepPlanWorkflow`	Chef receives prep list by 6 AM

All workflows are **stateful**, **interruptible**, **auditable**, and **explainable**.