CSC540 – Project 1 (Database Application Development Project)

Inventory Management for a Prepared/Frozen Meals Manufacturer
V2

Modifications will be highlighted in color. Post questions in project questions forum.

Due Dates:

- **Deliverable 1:** Preliminary Report (ER model + Relational schema) - may be partial (**Sep 28**)
- **Deliverable 2:** Full implementation (**Nov 2**)
- **Deliverable 3:** Demos (TBA)

No extensions.

Team size:~ **4 students**. (Use partner finder forum to find partners).

Target DBMS: **MariaDB/MySQL** (departmental server accounts will be provisioned). Client: **JDBC (Java) or Python** menu-driven interface is sufficient. **No ORM tools are allowed as part of your development stack.**

Final Submission Instructions (Important)

Each team will have to book a timeslot to demo the application to the Professor or Teaching Assistants. The details about the demo will be conveyed later.

The following needs to be submitted: **Create a zip file and use the file naming convention** of previous submission.

a. An updated ER model. A short paragraph about what improvements were made from the first design
b. Two SQL files: First one should contain SQL for triggers, tables, constraints, procedure. The second file should contain queries for populating the tables with the sample data. Executable file and source Java /Python Code.

c. README.txt - This should contain the names of the members and any additional instructions that you think will be necessary to compile and execute your code independently by TAs or Instructor.
d. A report that discusses the following items. The main report need not be very long (no longer than 4 pages) but not including figures, tables etc which you can put as an appendix.
e. Functional Dependencies: Functional dependencies that influenced your design e.g. any that led to relational decompositions
f. Comments about normalization (which tables are in BCNF and which are in 3NF). None should be below these. (You can use a table representation for this).
g. Constraints: A description of constraints that were not implemented as part of table definitions and why and how they were implemented in the final design. In particular, a separate subsection here should highlight constraints that couldn't be implemented in the database at all and had to be implemented in application code. Note that a key part of assessing your design is how well you used the DBMS to implement constraints v/s implementing in application code.

Finally, a link will be provided to submit the review of your assessment of your team members contribution to the project.

Application Domain Overview

This project centers on a small manufacturer of **prepared/frozen meals**—items like a *Steak Dinner with mashed potatoes*, *Mac & Cheese* (Side), or a *Chocolate Lava Cake* (Dessert). A **product** is a sellable finished good with a **product number** and **name**, and belongs to a **category** such as *Dinners*, *Sides*, or *Desserts* (categories have their own ids and names). Each product is produced according to a **recipe** (bill of materials, or *BOM*) that lists the **ingredients** and **quantities** required to make a standard amount of the product. For example, a *Steak Dinner* might require *Beef Steak*, *Seasoning Blend*, *Brown Gravy*, *Mashed Potatoes*, and *Herb Butter*, each with explicit quantities and units of measure. Production occurs in **batches**; each **product batch** (lot) has a **lot number**, **expiration date**, and **produced quantity**, and the database must remember **exactly which ingredient lots** were consumed to make it. So a production batch is just a batch of a product being made and all product instances made in that batch have the same lot number. You cannot have different products having the same lot number.

Ingredients come in two kinds. **Atomic ingredients** are indivisible (e.g., *salt*, *sugar*, *beef*). **Compound ingredients** (e.g., *Seasoning Blend*, *Tomato Sauce*) are built from other ingredients—called their **materials** in this document to avoid ambiguity. A compound ingredient's definition is therefore a nested BOM. **Composition limit.** Ingredients may be **atomic** or **compound**. A compound ingredient may list **materials** (its immediate ingredient children). **Materials must not themselves have materials**. In other words, the composition has

one level of children (parent → children; no grandchildren), and self-inclusion/cycles are forbidden.

Ingredients are sourced from **suppliers**, and the same ingredient may be offered by multiple suppliers with different **formulations** and **prices**. A supplier's formulation for an ingredient defines its internal composition (materials and quantities), **pack size**, **unit price**, and **validity dates**. For example, Supplier A's *Seasoning Blend* may include *salt*, *pepper*, and *garlic powder* at one ratio and price per 10-kg pack, while Supplier B offers a slightly different blend, pack size, and price. Over time, suppliers may **version** their formulations; the system preserves versions, selects the active one by date, and prevents overlapping effective periods. Suppliers have a **supplier id** and **name**, and the database records **which ingredients each supplier can provide**.

Both suppliers and the manufacturer organize physical inventory into **batches/lots**. A **received ingredient batch** records the **ingredient id**, **supplier id**, **quantity**, **per-unit (or per-pack) cost**, and **expiration date**. Ingredient batch lots follow a traceable **lot numbering rule** composed from the ingredient id, supplier id, and the supplier's (or facility's) batch identifier—e.g., `ING123-SUP45-B0009`. **Manufacturer product lots** may follow a similar pattern (e.g., `<productId>-<manufacturerId>-<batchId>`) and must be unique.

The inventory application supports the following flows. A **manufacturer creates a product batch** (e.g., 500 trays of *Steak Dinner*). Each product has a **standard batch size** (e.g., 500 units); a production run may produce an integer multiple of this standard size, and the run results in a **single product lot** with the chosen quantity. Standard size is per product and manufacturers can have different standard sizes. During creation, the user selects—for **each ingredient in the product's BOM**—the **specific ingredient batch lots** to consume (thereby fixing supplier and lot expiration). The creation succeeds only if **required quantities exist** and **none of the selected lots are expired**; on success, **ingredient inventories are decremented** accordingly and the product lot is posted. *(Grad)* If multiple batches exist for the same ingredient, the system should prefer the **closest-to-expiration** lot first (FEFO) when making selections, subject to available quantity. (Give priority to choosing the earliest batch to expire. If the number is insufficient, use the next earliest batch. Splitting is NOT allowed)

An **ingredient supplier** can also create **ingredient batches** (their incoming lots); this is a simpler operation that validates that the **ingredient is one they supply** and records the batch's attributes (lot number according to the rule, quantity, cost, expiration, etc.). Suppliers **do not** define product recipes in this system; only manufacturers define **product recipes** by listing the ingredient types and required quantities. A recipe is a template (like we see in recipe books). It is when the batch is created that a manufacturer binds specific ingredients and versions to a recipe's ingredients.

A manufacturer can **calculate the cost** of a production batch based on the **actual ingredient batches selected**. The **per-unit cost** for the product is the batch's total cost divided by its produced quantity. The manufacturer can also run **inventory health reports**, including **nearly-out-of-stock items** (defined as on-hand quantity **below the standard batch size** for that item) and **almost‑expired items** (based on a threshold window (window = 10 days)). The system supports **adding ingredient inventory** by recording a new ingredient batch with an expiration check: **if the expiration date is fewer than 90 days from the intake date, the intake is rejected**.

For labeling and transparency, any user may generate the **ingredient list for a product**, including all **nested materials where applicable**, sorted by **contribution quantity** in the final product from **largest to smallest** (ties broken deterministically).

(Grad) **Recall & Traceability.** If a supplier issues a **recall** for ~~an ingredient—or~~ a specific ingredient lot—the system must identify all **finished product lots** that used the affected inputs within a given time window(window = 20 days) by traversing the (one‑level) ingredient→product relationships. The manufacturer should be able to query **which of their products are at risk** for a given ingredient or lot recall.

Access control note. Each **manufacturer account** owns a specific set of products; a manufacturer **cannot** create production batches for, or add inventory to, products they do not own.

Functional Requirements

Roles

Implement exactly these roles (first screen lets the user select a role):

- **Manufacturer** – owns products and recipes; records ingredient receipts (subject to the 90‑day rule), creates product batches (consumes lots), runs inventory and expiry reports, and views per‑batch cost.
- **Supplier** – defines which ingredients they supply, maintains formulations and prices, and creates **supplier ingredient batches** (lots) for those ingredients.
- **Anyone (Viewer)** – can generate a product's **ingredient list** including nested materials, ordered by quantity contribution.

A user cannot hold multiple roles. Proceed to the menus below for the minimal actions required to support this domain. No support for database level privileges is required. Just regular application level role management using passwords.

Business rule: For this project, a user holds **exactly one** role.

Functional Requirements (by Role)

Supplier — Functions (and how they appear in the UI)

- **Manage Ingredients Supplied**
  Menu: Supplier → Manage Ingredients Supplied.
  Maintain the set of **ingredient types** this supplier can provide.

- **Define/Update Ingredient (Atomic or Compound)**
  Menu: Supplier → Ingredients → Create/Update.
  For each supplied ingredient, define whether it is **atomic** or **compound**. If compound, list its **materials** (one-level children only) and required quantities (in ounces). Include basic pricing (per pack or normalized per ounce). Teams may choose whether to keep simple versioning/validity dates.

- *(Grad) Maintain Do-Not-Combine List*
  Menu: Supplier → Ingredients → Do-Not-Combine.
  For any ingredient they supply, maintain a list of **incompatible ingredients** that should not appear together. Not specific to a supplier, it's a stored given list. For example: Ingredient A can't be combined with ingredient B.

- **Create Ingredient Batch (Lot Intake)**
  Menu: Supplier → Inventory → Receive Ingredient Batch.
  Record a new **ingredient batch** for a supplied ingredient with quantity (oz), cost, expiration date, and a computed **lot number** `<ingredientId>-<supplierId>-<batchId>`. On success, on-hand for that lot increases.Enforce: **lot format rule**, **expiration ≥ intake + 90 days**, and a valid ingredient. On success, on-hand increases.

Manufacturer — Functions (and how they appear in the UI)

- **Create & Manage Product Types**
  Menu: Manufacturer → Products → Create/Update.
  Define a **product type** with name, category, and a manufacturer-scoped id. Each product type belongs to exactly one category and is owned by the current manufacturer account. Specify the **standard batch size (units)** for production planning.

- **Create & Update Recipe Plans (Versioned)**
  Menu: Manufacturer → Products → Recipe Plans.
  For each manufacturer–product-type pair, create a **recipe plan** (BOM) listing ingredients and required quantities (in **ounces**) per **one unit** of product. Updating creates a **new plan version** (new plan id and creation date); previous versions remain

for history. The plan used in production is selected explicitly.

- **Create Product Batch (Production Posting)**
  Menu: Manufacturer → Production → Create Product Batch.
  Select the **product type**, the **recipe plan version** to use, and the **produced units** (must be an **integer multiple** of the product's standard batch size). For each ingredient in the plan, select the **specific ingredient lots** and quantities to consume. Validate **sufficient quantity** and **not expired**; on success:
  • decrement on-hand for the chosen lots,
  • compute **batch cost** and **per-unit cost** from the actual lots consumed, and
  • assign a **unique product batch number** `<productTypeId>-<manufacturerId>-<batchId>`.
  (Grad) Optionally **auto-select lots by FEFO** (prefer earliest-expiring lots that satisfy the required quantity).
  (Grad) **Recall & Traceability:** Given an ingredient or ingredient lot and date window, list finished product batches that used the affected inputs (respect **one-level** composition).

- **Reports & Queries**
  Menu: Manufacturer → Reports.
  a) **On-hand by item/lot**
  b) **Nearly-out-of-stock** (on-hand < product's standard batch size)
  c) **Almost-expired ingredient lots** (within a threshold window)
  d) **Batch Cost Summary** for a selected product batch
  (Grad) **Incompatibility Warning on Save/Post:** When saving a recipe plan or posting a product batch, warn if any **do-not-combine** ingredient pairs appear together (flatten one level). The conflict list is going to be a general list that is not specific to a supplier. Imagine it to be something that a regulatory body provides. This list will be provided to you in the form of a list of pairs (a, b) such that a cannot interact with b.

General (Viewer) — Functions (and how they appear in the UI)

- **Browse Product Types**
  Menu: General → Browse Products.
  View available product types organized by manufacturer and category.

- **Generate Ingredient List**
  Menu: General → Product → Ingredient List.
  For any product type, produce the **flattened one-level ingredient list** from its selected recipe plan, ordered by contribution quantity (desc).

- *(Grad) Compare Two Product Types for Incompatibilities*
  Menu: General → Compare Products.

Given two product types, report whether the union of their flattened ingredient sets contains any **do-not-combine** pairs and list the offending pairs.

Access Control (applies to all)
Manufacturers may act **only** on product types they own; suppliers may create batches **only** for ingredients they supply; general users have read-only access as described above.

Non-Functional / Implementation Notes

- Use **MariaDB/MySQL** features only (portable SQL).
- Implement **stored procedures** and **triggers** where appropriate (see below).
- Application UI can be **menu-driven CLI**; focus is database modeling and logic.

Queries To Be Implemented.

1. List the ingredients and the lot number of the last batch of product type Steak Dinner (100) made by manufacturer MFG001.
2. For manufacturer MFG002, list all the suppliers that they have purchased from and the total amount of money they have spent with that supplier.
3. For product with lot number 100-MFG001-B0901, find the unit cost for that product.
4. Based on the ingredients currently in product lot number 100-MFG001-B0901, what are all ingredients that cannot be included (i.e. that are in conflict with the current ingredient list)
5. Which manufacturers have supplier James Miller (21) NOT supplied to?

Required Procedures & Triggers

- **Trigger: Compute Ingredient Lot Number** on insert of an ingredient batch as `lotNumber = CONCAT(ingredientId, '-', supplierId, '-', batchId)` (or equivalent); enforce uniqueness & pattern.

- **Trigger: Prevent Expired Consumption** when recording consumption; reject if `NOW() > expirationDate` of the ingredient lot.

- **Trigger: Maintain On-Hand**: automatically update inventory balances on receipt/consumption/adjustment.

- **Procedure: Record Production Batch**: creates the batch, consumes declared ingredient lots atomically, computes cost, and writes finished goods inventory.

- **Procedure: Trace Recall**: given ~~ingredientId or~~ lotNumber, compute transitive closure to find affected product batches.

- **Procedure: Evaluate Health Risk**: This procedure runs whenever a new batch is created. It checks whether any pairs appear on a supplier's do-not-combine list. If a conflict is detected, the batch creation is immediately blocked

- **Views**: Current active supplier formulations; Flattened product BOM for labeling; **Health-risk rule violations** (last 30 days).

Appendix A – Minimal CLI Flow (illustrative)

```
None
Login
Select role: [1] Manufacturer  [2] Supplier  [3] General (Viewer)
[4] View Queries

[Manufacturer]
1) Define/Update Product
2) Define/Update Product BOM
3) Record Ingredient Receipt
   - Reject if expiration < today + 90 days
   - Lot number = <ingredientId>-<supplierId>-<batchId>
4) Create Product Batch
   - produced_units must be a multiple of standard_batch_units
   - Select lots to consume (or FEFO auto-select — Grad)
   - Validate not expired & sufficient qty; post inventory;
compute cost
```

```
5) Reports: On-hand | Nearly-out-of-stock | Almost-expired
6) (Grad) Recall/Traceability

[Supplier]
1) Declare Ingredients Supplied
2) Maintain Formulations (materials, price, pack, effective
dates)
3) Create Ingredient Batch (for supplied ingredients)

[General (Viewer)]
1) Product Ingredient List (with nested materials, ordered by
quantity)
```

Grading

Deliverable	**Percentage**
Original ER Diagram and Report	10
Revised ER Diagram	15
Report including discussion on constraints, FDs, normalization choices, SQL files + Readme,etc	35
Application Demo Test Cases	40
Peer Review	Average of peer review grades (individual grade a function of team grade and peer review)