1 Introduction and Use Cases

Your client, Hypothetical Meals, is a large food company that produces a significant portion of the world’s food. They currently uses a mishmash of spreadsheets and macros to manage food production, including inventory, logistics, sales, etc. They would like a unified system to replace these highly manual procedures. This system will serve the following use cases:

- The system will track ingredients, including vendors which can provide a given ingredient, cost information, and storage information (frozen, refrigerated, etc.). These details are provided by the administrator administrators.

- The system will allow authorized users to document orders for ingredients and to monitor how storage of ingredient stock compares to on-site storage capacity. Ordering is handled by managers or administrators. Ingredient ordering will be done using the “shopping cart” metaphor, with specific vendors being chosen at “check-out”.

- The system will allow the administrator to note when ingredients are consumed by production using the popular “shopping cart” metaphor.

- The system will provide reports on purchasing and ingredient use.

- Users will be able to bulk-import new ingredients and formulas from a simple text format.

- The system will track the production of foods by noting food formulas and allowing managers to record the production of food products according to these formulas. Such requests will provide the user with a list of specific ingredients that will be consumed.

- The system will track ingredient amounts both in terms of storage (measured in sq.ft. based on the type of package) and its use in formulas (e.g., gallons of broth, pounds of potatoes, individual eggs):
– For storage purposes, each ingredient package can be reduced to an effective square footage requirement, *storage footprint*, which takes into account stacking/shelving.

– For formula purposes, each ingredient will have a customizable unit “native” to that ingredient, and the corresponding package will contain a number expressed in such units (e.g., a sack of potatoes may be 25 lb, a pail of broth may be 5 gallon, etc.).

• All users will be able to log in using company single-sign-on. For our purposes, this means support for Duke NetID. Users may be marked as “managers” or “administrators” within the system.

• Managers will have permission to perform the majority of the day-to-day operation of the system, while administrators will be able to override most of the rules of the system to correct erroneous situations. Unprivileged users will have read-only access to the system.

• The system will provide reports on overall production.

• A comprehensive transaction log will track all operations which permute the state of inventory, production, or user accounts.

2 Definitions

• **Ingredient**: A food product purchased by the company for use in production. Available from at least one vendor, typically more. A given ingredient will come in a specific package and temperature state.

• **Ingredient package**: Ingredients may come in a sack or pail (around 50 lbs), a drum (for liquids, around 500 lbs), a “supersack” (a large square sack that sits on a shipping pallet; around 2,000 lbs), or a truckload or railcar (for liquids, 50,000+ lbs).

• **Temperature state**: The kind of storage a given ingredient requires to maintain food safety and/or freshness: either frozen, refrigerated, or room temperature. Ingredients should never be stored at a different temperature state than they come in.

• **Vendor**: A partner company that sells one or more ingredients.

• **Storage capacity**: The total amount the company can store for a given temperature state. Storage capacity is measured in pounds. For packages smaller than railcar/truckload, this is measured in square footage within the warehouse, refrigerator, or freezer (see *storage footprint*, below). Bulk ingredients in railcar or truckload have no maximum capacity as they are either kept in the original vessel or stored in special-purpose tanks.

• **Ingredient stock**: The total amount of a given ingredient owned and stored for use by the company. Measured in pounds. Measured in units specific to the ingredient (see *native units*, below).

• **Production run**: The act of consuming one or more ingredients to produce a product.
• **Storage footprint**: The square footage needed to store a given ingredient package, taking into account stacking and shelving. The footprint is based solely on the package, so the total footprint for an ingredient’s stock is calculated as the integer number of packages remaining (possibly including one partially consumed package) multiplied by the package’s footprint.

• **Native unit**: The unit of measure associated with an ingredient for production purposes. For example, potatoes may be measured in pounds, while broth may be measured in gallons. Ingredient packages provide a configurable number of native units of the ingredient.

• **Food product**: The result of a production run of a given formula. Measured in product units.

• **Product unit**: The indivisible output of production (e.g. a can of soup, a wrapped loaf of bread, etc.). The system does not track what specific form a product unit takes, only that it is an integer count resulting from a production run. This is a different, unrelated concept from native unit, defined above.

• **Formula**: The set of ingredients and their native unit quantities needed to produce a given food product in a specified quantity of product units.

• **Manager**: A user with with the “manager permission” who is able to order ingredients and perform production runs.

• **Administrator**: A user with the “administrator permission” who therefore has all manager rights plus the ability to override system rules to directly correct data, input new ingredients and formulas, and perform other core configuration operations.

• **Unprivileged user**: A user with neither of the above two permissions. Still able to read inventory status, read reports, etc.

### 3 Requirements

*A note on requirements: No set of requirements is perfect, and that is certainly true here. I’m sure that contradictions, under-specified behavior, and unintended consequences will be revealed. Your overriding goal should be to produce a quality system; if you believe that goal would be better served if a requirement were altered or interpreted a certain way, ask about it, and get the conclusion in writing. The result may be a variance in a requirement for a specific team, or even modification of this requirements document for all teams. In short, if unsure, ask.*

Some requirements have attached an informal tip, clarification, or example – these do not alter the requirements themselves, but are meant to answer likely questions about a requirement.

1. **Server**

   1.1. Your software must have a server that supports an arbitrary number of users.

   1.2. During the install/setup process, a special user named “admin” is configured.

   1.3. Users must have their accounts created by the admin-user before being able to use the system. The system shall allow the use of the Duke NetID system to allow all users to login using their Duke credentials in addition to supporting locally created users. The special local “admin” account remains, and has administrator permission.
1.4. Any stored passwords must be kept in a secure manner (i.e., salted + hashed)

1.5. All communication between the clients and server must be encrypted.

   *Tip: For web-based solutions, this means using HTTPS.*

1.6. The server must maintain state in a persistent fashion.

1.7. For all views which show a potentially unbounded number of records, the response time of the interface shall not depend on the quantity of records.

   *Tip: This implies some form of pagination so that only a finite number of records are retrieved at a time. Pagination can be explicit (page 1 of N) or implicit (infinite scrolling). Other UI solutions are likely also possible.*

1.8. The system shall track permission level for each user: unprivileged, manager, and administrator. The special local “admin” user has implicit administrator permission. Permissions are strictly nested: managers can do anything normal users can, and administrators can do anything that managers can.

1.9. Users with administrator permission can create “local” (non-NetID) user accounts.

1.10. Users with administrator permission can grant manager or administrator permission to any existing user (either NetID-based users or local users).

2. Vendor information management

2.1. *The administrator* Administrators will be able to add, edit, or remove vendors. A vendor will be defined by a unique name, a free-form field for contact information, and a unique case-insensitive alphanumeric freight carrier code.

3. Inventory tracking functionality

3.1. *The administrator* Administrators will be able to add or edit ingredients. An ingredient is defined as:

   * A unique name
   * The package the ingredient comes in, which is one of:
     - Sack (50 lbs occupying 0.5 sqft of floorspace)
     - Pail (50 lbs occupying 1 sqft of floorspace)
     - Drum (500 lbs occupying 3 sqft of floorspace)
     - Supersack (2000 lbs occupying 16 sqft of floorspace)
     - Truckload (50000 lbs not occupying any floorspace)
     - Railcar (280000 lbs not occupying any floorspace)
   * The temperature state of the ingredient: frozen, refrigerated, or room temperature
   * The vendor(s) which can provide the ingredient (see req 2) and the price each charges per package
   * The native units of the ingredient (pounds, gallons, or a short custom string)
   * The number of native units that the package provides

3.2. *The administrator* Administrators will be able to remove ingredients. If an ingredient is listed in an existing formula, an appropriate error will be shown indicating the affected formula(s), and the removal will be aborted.
3.3. The administrator Administrators will be able to set the storage capacity for ingredients in each of the three temperature zones: freezer, refrigerator, and warehouse (room temperature), except for truckload/railcar ingredients, for which there is no tracked storage limit.

*Clarification*: Truckload/railcar ingredients are either stored in special-purpose tanks or left in their original vessel until use, so maximum capacity need not be tracked by this software.

3.3.1. Storage capacity per temperature state (freezer/refrigerator/warehouse) will be tracked in pounds. Storage of ingredients other than truckload/railcar will be tracked in sqft of floorspace of the freezer/refrigerator/warehouse.

3.3.2. The system must enforce storage limitations when processing orders (see req 3.4).

3.3.3. Temperature states must always be respected; an ingredient may never be stored at another temperature state.

3.3.4. If the user tries to edit a capacity to a value below the current total inventory storage in that temperature state, an appropriate error will be shown and the operation will be prevented.

3.3.5. As native units of an ingredient are consumed by production, the system will compute the number of packages remaining, including a partially used package. This is used to compute the storage footprint of the ingredient in total. In other words, 

\[
\text{remaining\_packages} = \left\lceil \frac{\text{remaining\_native\_units}}{\text{native\_units\_per\_package}} \right\rceil \quad \text{and} \\
\text{total\_storage\_footprint} = \text{remaining\_packages} \ast \text{package\_storage\_footprint}.
\]

3.4. Users Managers shall be able to log orders of ingredients from vendors.

3.4.1. The user navigates to a given ingredient, identifies a vendor, and requests an order for a given integer number of packages. This is done using a shopping cart metaphor, where the user adds ingredients to cart, can edit quantities or remove items in cart, and finally “check out”.

3.4.2. Orders which would exceed storage capacity must be denied with an appropriate error message.

3.4.3. Successful orders shall immediately log the ingredient into the inventory.

3.4.4. The system shall track money spent on ingredients for use in reporting (see req 4).

3.4.5. The specific vendors for the order will be chosen during the final “check out” stage; the user will be able to conveniently choose a vendor per ingredient with the default being the least expensive for each ingredient.

3.5. Users shall be able to browse the inventory.

3.5.1. All ingredients shall be summarized in an efficient manner.

3.5.2. The user should be able to search by ingredient name and filter based on temperature state and/or package.

3.5.3. The quantity of each ingredient will be tracked in pounds native units for production purposes and sqft for storage purposes.

3.5.4. The user shall be able to view a detail page for any given ingredient showing all tracked information for it.
3.5.5. The administrator **Administrators** shall be able to directly edit ingredient stock quantity to correct errors; this operation shall be logged clearly in the system log (see req 8).

3.6. Users shall be able to note the consumption of ingredients as follows.

3.6.1. The user can add/remove ingredients to a “cart” with quantities specified in pounds.

3.6.2. The user can “check out” the cart, thus indicating that the chosen ingredients are removed from inventory and used in production.

3.6.3. The inventory totals for the selected ingredients should be decremented appropriately.

3.6.4. Attempts to add more to cart than inventory stock allows should be disallowed, as the production must not exceed inventory stock. The inventory cannot go negative.

4. Reporting

4.1. **Spending report**: Users shall be able to view a report of ingredient spending which indicates total spending on each ingredient to date, both overall and only for ingredients used in production.

4.2. **Production report**: Users shall be able to view a report of production which indicates total number of units of each formula produced to date, including the total ingredient cost that went into product.

5. Documentation

5.1. **Developer guide**: A document shall be provided which orients a new developer to how your system is constructed at a high level, what technologies are in use, how to configure a development/build environment, and how the database schema (or equivalent) is laid out.

5.2. **Deployment guide**: A document shall be provided which describes how to install your software entirely from scratch. It should start by describing the platform prerequisites (e.g. Linux distro, required packages, etc.), then mechanically describe every step to deploying your system to production readiness.

6. Bulk import facility

6.1. Administrators shall be able to import new ingredients into the system by means of a text-based import (CSV, JSON, or other simple plaintext format). The customer is accepting proposals on the format.

6.2. The import interface shall include documentation as to the import format.

6.3. The import action shall only occur if the entire input is free of name conflicts or otherwise problematic issues; if such issues arise, the precise nature of the error should be presented to the administrator in enough detail that it can be corrected.

6.4. Administrators shall be able to import new formulas into the system by means of a text-based import (CSV, JSON, or other simple plaintext format). The customer is accepting proposals on the format.
7. Production modeling

7.1. Administrators shall be able to add/edit/delete/view formulas within the system. A formula consists of:

- A unique name
- A long-form description/notes field
- One or more \{(ingredient,quantity)\} tuples, where the quantity is specified in floating-point native units
- The integer number of units of product this formula will produce

*The meaning of ‘unit’ for a food product depends on the product (cans, bags, bushels, etc.), but this distinction need not be modeled by this software. Note that a food product unit is a different concept from an ingredient native unit.*

7.2. Managers shall be able to initiate production of a food product as follows:

7.2.1. The manager will select the formula and input the number of units that will be made. This value defaults to the product unit count listed in the formula and must be an integer greater than this default.

7.2.2. Before committing to production, the manager will be shown a list of the quantities of ingredients to be consumed as compared to the current inventory stock.

7.2.3. If the required ingredients are not present in full in inventory, the user will be alerted to this and production will not be allowed to proceed. However, the manager should be prompted with the option to automatically add to the ordering cart the difference between required ingredients and current stock.

7.2.4. If the production is feasible and confirmed by the manager, ingredient stocks will be appropriately consumed. The quantity of ingredient consumed will be based on how much is called for in the formula scaled by how many units past the base amount are to be produced.

*Example: If a tomato soup formula makes 200 units using 10 gallons of tomato paste and the manager requests to make 300 units, then 300/200 * 10 = 15 gallons would be consumed.*

8. Global system logging

8.1. The system shall record a log of all actions undertaken in the system (i.e., any action that alters the system state, including inventory operations, production runs, user creation/deletion/modification, etc.). Log entries shall include the initiating user, the entities involved, the nature of the event, and the time and date. This includes corrective actions undertaken by administrators (per 3.5.5).

8.2. Managers shall be able to view this log.

8.3. The log view should allow searching by user, ingredient, and/or timespan.

8.4. Users consulting the log shall be able to navigate directly from a reference to an ingredient/product to the relevant detailed view (see reqs 3.5.4, 7.1).

8.5. Users should not be able to tamper with the log in any way, regardless of permission.