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 use a mishmash of spreadsheets and macros to manage food manufacturing scheduling. They would like a unified system to replace these highly manual procedures. Further, they would like to gain a better understanding of the costs and revenues involved to improve business decision-making. This system will serve the following use cases:

- The system will track SKUs (Stock-Keeping Units) for the food items produced by the company. These details are provided by administrators.
- The system will track the ingredients that are used to manufacture each SKU.
- Manufacturing staff will use the system to calculate of ingredient needs based on manufacturing goals.
- Product managers will use the system to produce reports showing information about ingredients and the SKUs they are used to make.
- Users will be able to bulk-import and bulk-export ingredients and SKU information from a simple text format.
- For multiple SKUs with a common set of ingredients, a SKU can be said to be “based on” another such that it inherits that SKU’s formula.
- The system will track manufacturing lines and the SKUs they can be used to produce.
The system will be able to tie manufacturing goals to deadlines and to allow the administrator to map these goals to a manufacturing schedule using a graphical interface.

The system will furnish reports on manufacturing as well as printable per-manufacturing-line schedules.

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 “administrators” within the system.

The system will track the costs of manufacturing.

The system will integrate with existing sales record-keeping to track sales to customers on a per-SKU basis.

The system will furnish configurable Sales Reports detailing sales trends across product lines, SKUs, customers, and timespans.

To allow for use worldwide, all text inputs and outputs must be fully Unicode-capable.

IT administrators will be able to restore the state of the system from a robust backup system using a clearly documented procedure.

2 Definitions

• **Ingredient**: A food product purchased by the company for use in manufacturing. Ingredients come in packages of varying size (e.g. a 55 gallon drum, 20 lb. sack, etc.). Ingredients are tracked by ingredient number, commonly written as Ingr#.

• **SKU**: Stock-Keeping Unit. Refers to a product made by the company. The term “SKU” may also be used to refer to the SKU number, commonly written as SKU#.

• **Customer**: A food retailer or distributor, such as a grocery store, convenience store, food service company, etc.

• **Consumer**: The individuals who purchase SKU units from our customers.

• **Product line**: A group of related SKUs, such as “rich and hearty soup”. Every SKU belongs to a product line.

• **Unit**: An individually purchasable food item (e.g. one can of soup). Units are what are sold to consumers.

• **Case**: A carton or box of units (e.g. a box of 24 cans of soup). Cases are what get sold to customers.

• **UPC**: Universal Product Code. A standard for assigning numbers to items purchased or traded in the worldwide economy. Hypothetical Meals products use the UPC-A standard.

• **Manufacturing goal**: A list of SKUs and case quantities that the company may want to manufacture. Includes a deadline date.
• **Input assistance:** A user input is said to be *assisted* if it is a user-selected reference to an existing record (ingredient, SKU, etc.) where the UI provides a listing, inline search, autocomplete, and/or other means to allow easy and efficient selection. *Unless otherwise specified, all selections of an existing record should be assisted.*

• **Unit of measure:** The unit used to measure an ingredient. The system will need to be able to understand and convert units internally.
  
  – For weight-based items, one of:
    * Ounce (oz.)
    * Pound (lb.)
    * Ton (ton) – defined as a U.S. imperial ton, or 2000 lb.
    * Gram (g)
    * Kilogram (kg)
  
  – For volume-based items, one of:
    * Fluid ounce (fl.oz.)
    * Pint (pt.)
    * Quart (qt.)
    * Gallon (gal.)
    * Milliliter (mL)
    * Liter (L)

  – For individual discrete items, a *count* (e.g., “a 96-count crate of eggs”).

• **Manufacturing line:** A cohesive set of systems in a factory used to manufacture a particular set of SKUs. Different SKUs are manufactured at differing rates.

• **Manufacturing activity:** The act of producing a single SKU on a manufacturing line, usually to meet one entry in a manufacturing goal. Scheduled for a particular timespan on a particular manufacturing line.

• **Formula:** The set of ingredients and quantities used to create one or more SKUs. Many SKUs will have a 1:1 mapping to formula, but some formulas service multiple SKUs (e.g., a formula for salsa being used for SKUs differing by unit or case size). Ingredient quantities in formulas will be expressed in *units of measure*. When connected to a SKU, a formula’s ingredients are adjusted by a *scale factor*. Formulas are tracked by formula number, commonly written as Formula#.

• **Sale:** When a customer buys one or more cases of a SKU from Hypothetical Meals. Sales are tracked in a separate system. Sale prices can vary over time and per customer.

• **Ingredient cost:** The per-package cost of an ingredient. It is possible to compute a per-case ingredient cost for a SKU using its formula; in this case, ingredient cost is pro-rated to fractional packages. For example, if 10 lbs. of carrots costs 20 USD, and a case of carrot soup uses 15 lbs., then the case costs \(\frac{20}{10} \times 15 = 30\) USD in carrots alone; the total ingredient cost of a case would be the sum of all its formula ingredients computed in this way.
• **Manufacturing setup cost**: The fixed retooling cost to prepare a manufacturing line to make a particular SKU (includes draining, cleaning, and run-out of excess ingredient stock).

• **Manufacturing run cost**: The cost per case to manufacture a SKU separate from ingredient cost (includes manufacturing payroll, packaging, shipping, wear and tear on equipment, etc.)

• **Cost Of Goods Sold (COGS)**: The sum of ingredient cost, manufacturing setup cost, and manufacturing run cost for a SKU, divided to give a cost per case.

• **Revenue**: The money collected from the customer for one or more sales. Can be expressed as a total for a number of sales or averaged per-case.

• **Profit**: The difference between revenue and cost. Can be expressed as a total for a number of sales or averaged per-case.

• **Margin**: The percent difference between revenue per case and COGS for a SKU, i.e., $\frac{\text{avg revenue}}{\text{cogs}} - 1$.

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. 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 at minimum)

   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 unless a full listing is explicitly requested by the user.

   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).
1.8. The system shall track permission level for each user: regular versus administrator users. The special local “admin” user has implicit administrator permission.

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

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

1.11. All text inputs and outputs must be fully Unicode-compatible. This means that users should be able to input data using non-English characters without data loss or corruption.

2. SKU Data Management

2.1. Administrators shall be able to add, modify, review, and remove ingredients within the system.

2.1.1. An ingredient consists of:

2.1.1.1. Ingredient name (required, unique)

2.1.1.2. Ingredient number (required, unique, auto-generated unless supplied by user). Written as “Ingr#.”

2.1.1.3. Vendor information (optional free-form multi-line text field)

2.1.1.4. Package size (required, a number plus its unit of measure (one of those given in the definitions) – the system must understand these units of measure to perform conversions elsewhere)

2.1.1.5. Cost per package in USD (required – represents the cost of the ingredient per “package size” as defined above)

2.1.1.6. Comment (optional free-form multi-line text field)

2.1.2. Users may review ingredients with the following view options.

2.1.2.1. The view should be filterable by any combination of:

- Keyword search by name or Ingr#.
- SKU(s) produced with the ingredient. SKU selection shall be assisted.

2.1.2.2. The interface should show the number of SKUs that use a given an ingredient and provide some kind of detail view allowing the user to review a list of the SKUs that use an ingredient.

2.1.2.3. The view should be sortable by all shown fields.

2.2. Administrators shall be able to add, modify, review, and remove product lines within the system to help organize SKUs.

2.2.1. A product line consists of a name (required, unique).

2.2.2. All SKUs are a member of exactly one product line.

2.3. Administrators shall be able to add, modify, review, and remove SKUs within the system.

2.3.1. A SKU consists of:
2.3.1.1. Name (required, max 32 characters)

*Note: When a SKU is shown elsewhere in the system, its display name should be shown as "<NAME>: <SIZE_PER_UNIT> * <CASE_COUNT>"*, e.g. "Tomato soup: 28oz * 128". The SKU# should also be shown, either inline in parenthesis or by other means.

2.3.1.2. SKU# (required, unique, numeric, auto-generated unless supplied by user)

2.3.1.3. Case UPC# (required, conforming to the UPC-A standard (12-digit) for consumer products (starts with 0-1 or 6-9) with a valid check digit, unique)

2.3.1.4. Unit UPC# (required, confirming to same UPC standards as Case UPC#, not necessarily unique (as two SKUs could be for different case-sizes of the same consumer item))

2.3.1.5. Unit size (required, free-form short text field describing how much comes in a single package of the item, e.g. “28oz.”, “1qt.”, etc.) Note: Unlike ingredients, the system should not parse or enforce constraints on this measure – the system will not need to do math on SKU unit size.

2.3.1.6. Count per case (required, the integer number of units in one case)

2.3.1.7. Product line (required, a choice of exactly one product line to which this SKU belongs)

2.3.1.8. (defunct)

2.3.1.9. Comment (optional free-form multi-line text field)

2.3.1.10. Formula (required, a reference to a formula that will list the {ingredient,quantity} as defined in req 2.4). The interface for adding/editing SKUs should make it easy to create/edit formulas inline, as many SKUs will map 1:1 to a formula, so it would be inefficient to force users to create the formula separately first. If selecting an existing formula, the selection shall be assisted.

2.3.1.11. Formula scale factor (required, floating-point value, defaults to 1.0). Formula ingredient quantities are adjusted by multiplying by this factor; the result gives the amounts needed to produce one case of the SKU.

*Example: A case of 20 small 8oz jars of salsa might use a salsa formula with scale 1.0, whereas a case of 20 large 16oz jars might use the same formula with scale factor of 2.0.*

2.3.1.12. Manufacturing lines (required, a selection of zero or more existing manufacturing lines capable of producing the SKU.) See req 4.3. When showing a manufacturing line mapping in summary, a list of shortnames will suffice.

2.3.1.13. Manufacturing rate (required, a floating-point number of cases per hour that can be produced).

2.3.1.14. Manufacturing setup cost in USD (required, numeric)

2.3.1.15. Manufacturing run cost in USD per case (required, numeric)

2.3.2. Users may review SKUs with the following view options.

2.3.2.1. The view should be filterable by any combination of:

- Keyword search by name, SKU#, case UPC#, or unit UPC#.
- Ingredient(s) used. Ingredient selection shall be assisted.
- Selected product line(s). Product line selection shall be assisted.
2.3.2.2. SKUs may be sorted by any shown field.

2.3.2.3. SKUs may be optionally grouped by product line. (defunct)

2.3.2.4. It should be possible to navigate from a SKU to a detail view of its associated formula.

2.3.3. Administrators should be able to bulk-edit the mapping of SKUs to manufacturing lines. This means selecting large numbers of SKUs by a combination of filters and manual selection, seeing the lines that all, some, or none of the selected SKUs can currently be made on, and changing this mapping so that lines are mapped or unmapped to all chosen SKUs.

2.4. Administrators shall be able to add, modify, review, and remove formulas within the system.

2.4.1. A formula consists of:

2.4.1.1. Name (required, max 32 characters)

2.4.1.2. Formula# (required, unique, numeric, auto-generated unless supplied by user)

2.4.1.3. Zero or more \{ingredient,quantity\} tuples, where ingredient refers to an existing ingredient record (with assisted selection) and quantity is a measured amount. The units employed can be any that are logically compatible with the ingredient’s package size (e.g., if an ingredient package is measured by weight in pounds, its use in a formula can be in ounces, pounds, tons, grams, or kilograms).

2.4.1.4. Comment (optional free-form multi-line text field)

2.4.2. Users may review formulas with the following view options.

2.4.2.1. The view should be filterable by any combination of:

- Keyword search by formula name or formula#.
- Ingredient(s) used. Ingredient selection shall be assisted.

2.4.2.2. Formulas may be sorted by any shown field.

2.4.2.3. It should be possible to navigate from a formula to a detail view of its associated SKU(s).

3. Bulk import/export facility

3.1. Administrators shall be able to import new ingredients, SKUs, formulas, and product lines into the system by means of an import compatible with modern spreadsheet software (CSV, XLSX, or similar). The customer is accepting proposals on the format.

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

3.3. 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.

3.4. If an import contains identical record(s) to those already in the system, such records should be ignored.

3.5. If an import contains record(s) that match on name or a unique numeric identifier, the user should be warned about all such records in detail, and if the user approves, the records should be modified to match the imported data.

Updated March 2, 2019 7 ECE 458, Spring 2019
3.6. After a successful import, a count and list of records that were added, updated, and ignored should be provided.

3.7. The system shall be able to export any of the above data in a format compatible with import. The specific records exported should be filterable by the same means defined in the “view options” described for each record in the requirements above (reqs 2.1.2.1, 2.3.2.1, 2.4.2.1).

Note: This allows for an export/modify/import workflow when large-scale changes are needed.

4. Manufacturing

4.1. Users shall be able to input a **manufacturing goal** as follows:

4.1.1. Users shall be able to input a set of SKUs, each with a desired case quantity.

4.1.2. The selection of SKUs shall be **assisted**, further, selection will be aided by the ability to navigate using search and filtering by product line.

4.1.3. Users shall be able to save such manufacturing goals by name for future loading and reference. Manufacturing goals are private to each user.

4.1.4. Users shall be able to export a manufacturing goal to CSV format.

4.1.5. A manufacturing goal shall also have a deadline associated with it. This deadline is met if manufacturing is completed by the close of business on that date.

4.2. Users shall be able to access a **manufacturing calculator**:

4.2.1. Users shall be able to select a saved manufacturing goal and produce a list showing all ingredients required and their quantities as measured in both the units their packages are measured in as well as a floating-point number of packages.

*Clarification: this means if making 3 cases of a SKU with formula scale factor 1.5, an ingredient listed as using 4 gallons would be shown needing 3\times1.5\times4 = 18 gallons; if the ingredient comes in 5 gallon pails, you’d also indicate it needs \frac{18}{5} = 3.6 packages.*

4.2.2. Users shall be able to export the calculation result in CSV format.

4.2.3. Users shall be able to produce a well-formatted printed document suitable for sending to a manufacturing plant. This can be achieved either with native “print” capability (e.g. within a web browser) or by an export-to-PDF feature.

4.3. Administrators shall be able to add, modify, review, and remove **manufacturing lines**.

*Note: The number of manufacturing lines will be relatively small (~20).*

4.3.1. A manufacturing line consists of:

4.3.1.1. A name (required, max 32 characters)

4.3.1.2. A shortname (required, unique, max 5 characters) Used to quickly identify a manufacturing line, e.g. “BMP1” for Boise Manufacturing Plant line number 1.

4.3.1.3. Comment (optional free-form multi-line text field)

4.4. Administrators shall be able to produce a **manufacturing schedule**. This schedule will map SKU production as described in manufacturing goals to manufacturing activities (specific SKUs in specific timespans on specific manufacturing lines). This schedule is to be stored persistently.
Note: The customer does not have a clear vision for the interface on this part and has done their best to articulate their thoughts below; teams are encouraged to work with the customer as they design their UI.

4.4.1. The administrator will “enable” or “disable” existing manufacturing goals created on the system by any user. This selection shall be assisted, allowing search by goal name and/or creator username. For a goal to be enabled, it means the administrator intends to satisfy it by scheduling its production.

4.4.2. The main UI will show time on one axis (as measured in work hours) and manufacturing lines on another. The time dimension should be easily navigable to any reasonable span.

4.4.3. The unscheduled SKUs in the enabled goals will form a palette of manufacturing activities that can be placed onto the timeline. If a SKU appears in two or more selected goals, those manufacturing activities will be treated separately.

4.4.4. The administrator will place manufacturing activities onto manufacturing lines (respecting the mapping between SKU and appropriate lines), scheduling them in time. The duration of these will be dictated by the manufacturing rate and cases to be produced ($hours\_needed = cases\_needed/mfg\_rate$), rounded up to the next highest integer number of hours. To translate hours to days, note that manufacturing lines operate 10 hours per day, 8am to 6pm. It is not necessary to be able to “split” a single activity among two manufacturing lines.

4.4.5. Manufacturing activities can also be removed from the timeline, thus returning them to the palette of unscheduled activities.

4.4.6. To aid with placement, the interface will provide graphical feedback as to the start time and duration of manufacturing activities and the deadlines from the underlying manufacturing goals.

4.4.7. It should be possible to place a manufacturing activity such that its completion exceeds its deadline, but this violation should be shown in a way that is unmistakably apparent.

4.4.8. It should be unmistakably apparent if not all manufacturing goals have been mapped onto the timeline.

4.4.9. If a goal is disabled after one or more of its manufacturing activities have been scheduled, it has no effect on the schedule, but such “orphaned” manufacturing activities should be shown in a visually distinct way, and any warnings relating to deadline or unscheduled activities for disabled goals should be suppressed.

4.4.10. The choices made by the administrator establish the plan of record for manufacturing, and will be subject to modification in ways the original manufacturing goals may not represent. As such, the administrator should be able to override activity durations. Activities whose duration have been overridden should be visually distinct.

Note: editing something in the past is just as valid as editing something in the future (i.e., there is no need to consider the current date in this interface other than to help in navigating the timeline).

4.4.11. A listing of all warning conditions should be available in detail; these conditions are: manufacturing activities that are enabled but not scheduled, scheduled to complete
5. Reporting

5.1. **Ingredient dependency report**: Users shall be able to create a tabular report showing a set of ingredients (the selection of which follows the same rules as the “view options” described in req 2.1.2). For each ingredient, all SKUs made with the ingredient shall be shown. This report should be viewed or exported in CSV format.

5.2. **Manufacturing schedule report**: Users shall be able to create a schedule of manufacturing for any timespan (as measured in days) for any single manufacturing line.

5.2.1. This schedule should show the sequence of manufacturing tasks with all available information (SKU details, formula/ingredient details, case quantity, start/end time and date, and duration in hours).

5.2.2. At the end of the report, a summation of all ingredients needed in total should be presented in tabular form, including both measurement units (e.g. pounds) and number of packages.

5.2.3. In addition to being displayed on-screen, the user should be able to obtain a printed document suitable for sending to a manufacturing plant. This can be achieved either with native “print” capability (e.g. within a web browser) or by an export-to-PDF feature.

5.3. **Sales report**: Users shall be able to create a report breaking down sales in a variety of ways. See req 7 for the source of sales data.

5.3.1. Two main views should be available: Summary and SKU Drilldown.

5.3.2. For summary view, users should be able to narrow the report as follows:

5.3.2.1. Users shall be able to specify the product line(s) to be included, and it should be easy to select all product lines.

5.3.2.2. Users shall be able to specify either all customers or one in particular; customer selection shall be assisted.

5.3.3. The resulting summary report shall include:

5.3.3.1. Grouped headers for product lines, under which each SKU is shown as described below.

5.3.3.2. For each selected SKU, a tabular listing showing sales per calendar year for the last 10 years. Each row should show the SKU, the total revenue (the sum of sales * price_per_case for each sales record in the period), and the average revenue per case.

*Note: “Calendar year” means Jan 1 to Dec 31. This means that the current year will likely show lesser numbers than previous as not all sales have yet occurred; this is normal.*

5.3.3.3. A total row shall be shown per SKU that summarizes all 10 shown years. It should indicate the sum of yearly revenues, average manufacturing run size*, ingredient cost per case*, average manufacturing setup cost per case*, manufacturing run cost per case*, total COGS per case*, average revenue per case*.
(revenue_sum/num_sales), the average profit per case (avg.revenue – cogs), and the profit margin as a percentage (avg.revenue/cogs – 1).

* Regarding average manufacturing run size and costs: The system will not have a full historical record of past manufacturing, so it will not have an accurate measure of manufacturing setup costs for past runs. We will employ the following heuristic to estimate: Over the 10-year period, find the average number of cases per manufacturing activity in the schedule; this is the average manufacturing run size. Assume that all past runs were of that size and use that to determine the number of manufacturing runs needed to meet the noted sales numbers. If no manufacturing activities are recorded in the period, just assume a one-day (10 hour) manufacturing duration to derive the average run size. Either way, this average run size can be used to compute the rest of the cost information. For example, if a SKU has just two manufacturing activities on record, one for 40 cases and one for 60 cases, report an average manufacturing run size of 50 cases. If the SKU has a manufacturing setup cost of 1000 USD, then the average manufacturing setup cost per case is 1000/50 = 20 USD. For the COGS calculation, sum the ingredient cost per case (computed based on the formula and its ingredient costs), the average manufacturing setup cost per case (20 in our example), and the average manufacturing run cost per case (stored as part of the SKU definition).

5.3.3.4. The output above should be available for download in CSV format.

5.3.4. By selecting a SKU in the summary view or picking a SKU via assisted selection, the user should be able to navigate to a detailed sales report for that SKU.

5.3.4.1. Users shall be able to specify either all customers or one in particular; customer selection shall be assisted.

5.3.4.2. Users shall be able to specify the timespan for this detail view; this timespan will default to the last 365 days.

5.3.4.3. The view will show a tabular listing showing sales records (filtered to a specific customer if one is selected). Each row should show the year, week number, customer number, customer name, number of sales, price per case, and the revenue (sales * price_per_case).

5.3.4.4. The view will include a total of the selected timespan, showing all the fields specified in req 5.3.3.3 (though based on the selected timespan rather than a 10-year period).

5.3.4.5. The view will show a line graph that charts weekly revenue over time for the selected timespan (limited to a specific customer if one is selected).

5.3.4.6. The tabular part of the output described above should be available for download in CSV format.

6. Documentation

6.1. 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.
6.2. **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.

6.2.1. In addition to covering how to install the system with “stock” default data, the procedure to install the system from scratch using backed up data should also be included (i.e., disaster recovery).

6.3. **Backup admin guide**: A document shall be provided which explains the backup solution so that a system administrator unfamiliar with your software could configure it from scratch, restore the database to any given backup, and test a backup for validity. See req 8.

7. Sales tracking

7.1. The system must integrate to the existing internal sales tracking system. However, the sales system is based on mainframe software written in COBOL and accessed via telnet, so it is considered too insecure to provide direct access to. Instead, your software will integrate to a thin web-based front end developed by the CTO’s son in 2001. It has no explicit API, but is simple enough to “screen-scrape”. It is available at http://hypomeals-sales.colab.duke.edu:8080/. Data is requested per-SKU per-year, and sales are shown with weekly granularity.

*Note: Obviously these aren’t real sales records. This system will provide realistic sales records for any SKU number provided for years starting 1999. Data is generated in real time, i.e. the records for 2019 will grow as time goes by.*

7.1.1. Each request hits a single-threaded server that has to do a telnet transaction, so it is important to keep requests to a minimum. Your software should cache data that is retrieved.

7.1.2. Data relating to the current year should be refreshed daily so that new records will automatically be picked up as they are created.

7.1.3. Data should be retrieved asynchronously from the front end interface and at a rate of no more than five requests per second (i.e., 200ms between requests). This is especially true for bulk import of SKUs, where the number of requests may be very large. The interface should gracefully handle the state where record retrieval is queued but not yet complete.

7.1.4. For testing purposes, you must be able to flush this cache and trigger re-retrieval of records if requested. This could be via an administrator interface or through direct manipulation of the backend by a developer.

7.2. A sales record consists of:

7.2.1. A reference to a SKU#
7.2.2. A date expressed as an ISO week date (a \{year,week\} tuple, see ISO 8601)
7.2.3. Customer number
7.2.4. Customer name
7.2.5. Sales, expressed in number of cases sold
7.2.6. Price per case in USD

8. Backups: You must deploy a backup solution for your system's database.

8.1. Backups shall be automatic and taken daily.
8.2. Backups shall be kept with a staggered retention (7 daily backups, 4 weekly backups, 12 monthly backups).
8.3. Backups must be stored on a separate system.
8.4. The backup system must require separate credentials to access.
8.5. The backup system should report on progress and alert on failure; this could be via email or another directed communication mechanism.
8.6. The backup system may be built either out-of-band from the main software (e.g. a background database dump restored manually by a sysadmin) or in-band (e.g. the software itself exporting its database using internal automation). If it is in-band, all backup operations should be restricted to administrators.