**HIGH-LEVEL PROBLEM DESCRIPTION OF RESTAURANT INVENTORY MANAGEMENT SYSTEM**

McPearson’s Restaurant in Sometown, NY is an independently owned and operated restaurant (i.e., not part of a restaurant chain like Friendly’s). A major problem they face is that they need to manage the inventory of items from which they supply their daily menus. These items are mainly edible stuff, but of a varied nature, such as pasta, loaves of bread, steaks, different kinds of liquor, vegetables such as lettuce, tomato, potato, etc., bottles of items such as ketchup, mustard, etc. Their main problem is that they order a certain number of items of a particular kind – e.g., they order 100 lbs. of potatoes from a supplier, and it arrives on Monday. They want to have an accurate picture of how many pounds of potatoes they have left by Friday – in order to determine whether they should re-order potatoes for delivery on next Monday. Without a computerized system, they would have to manually go and count/estimate the number of pounds of potatoes left. This would take time off from serving customers, and thus negatively impact their profit margin. If they could, instead, simply look up how many potatoes (i.e., how many pounds of potatoes) are left on the computer, which would be great. Even if the pounds are not exactly accurate, but close enough, they would be helped.

Getting into this situation obviously implies that the inventory items should be ‘trackable’ in some way – e.g., they should be barcoded (it is, of course, not possible to barcode every potato – you barcode a bag of potatoes). More importantly, it implies that whenever restaurant personnel take an item out of inventory – a bag of potatoes, a bottle of vodka, etc. – they scan the related barcode into the computer and that item is removed from inventory.

Doing all of this implies that when the restaurant processes an invoice, which indicates what items came in, and those items are there in front of you, restaurant personnel have to barcode each item that came in and scan them into the computerized system. The system will then record the date and time each item came in, and the barcode. You have to do this for every item you put into inventory, and then, of course, when you take them out, you have to operate as described above.

Thus, the *basic* functionality that the envisaged computerized system should provide include: Barcode an item (manually) and scan it into the inventory database on the computer, scan it out when you take it out of the inventory, allow the restaurant owner to see how many items of a particular kind he has in inventory (e.g., how many bottles of Stolichnaya vodka have I got left?), and see how much it would cost if I want to invoice a supplier (system must keep track of vendors the restaurant deals with, and the prices they charge for various items) order a certain number of units of various items from them.

These are the overall requirements. The initial set of use cases for this system are:

1. Add a new Inventory Item Type – An inventory item type refers to the kind of edible item the restaurant keeps in stock – for example, Idaho Potatoes, Coors beer, etc. As the restaurant decides to change its menu and stock another inventory item, this use case will be executed.
2. Modify an existing Inventory Item Type – The user should be able to modify details of a certain inventory item type.
3. Delete an existing Inventory Item Type – If the restaurant no longer offers a certain kind of item, it should be able to delete it from its inventory item type records.
4. Add a Vendor – The restaurant keeps track of vendors that supply it with certain inventory item types. For example, Idaho Potatoes may be supplied by Kohrer’s Farm and Jackson’s Potato Supply Company. This use case allows the user to add a vendor to the system’s record of vendors.
5. Modify an Existing Vendor’s details – Self explanatory
6. Add a Vendor-Inventory Item Type – This refers to the fact that we are keeping track of which vendor the restaurant has an agreement with to supply a certain item type. In (4) above, we were discussing the need to add a new Vendor itself (e.g., Donald’s Egg Farm) to the system. In this use case, we are discussing the need to record that Donald’s Egg Farm is authorized to supply the restaurant with boxes of 1-doz. Brown eggs (1-doz. Brown eggs, note, is an Inventory Item Type).
7. Delete a Vendor-Inventory Item Type – This refers to handling situations where the restaurant ceases to have an agreement with a certain vendor to supply a certain item type (e.g., Donald’s Egg Farm’s last two shipments of brown eggs were bad, so we are not going to buy 1-doz. Brown eggs from them any more).
8. Process an invoice from a vendor – This use case is executed when a shipment of various items comes in from a certain vendor. The vendor sends an invoice along with the set of items (on paper), and the restaurant has to process all received items. Each received item has to be barcoded and saved in inventory records. For example, suppose vendor named “Jake’s Food Supply” sent in 10 loaves of wheat bread, five cases of beer, 12 steaks of a certain kind and weight, etc. – all items that they are contracted to supply (see use case (6)). The restaurant will verify that the item received is something that “Jake’s Food Supply” is authorized to send over, and accept it. This acceptance involves putting a barcode on the item (barcodes must be unique), and putting in the barcode and other information about the item (such as date received) into inventory.
9. Take an item out of inventory – This use case is executed when a restaurant worker takes a certain item out of inventory. The worker provides the barcode from the item to the person operating the computer, who scans the barcode in and brings up the associated inventory record. The person at the computer then changes the status of the inventory record from “Available” to “Used”. *Please note****:*** Certain items have an expiry period associated with them. If the date received value stored in the inventory record, plus the expiry period (shown in days) is at or beyond today’s date, then the person at the computer should tell the restaurant worker to discard the item and take another one of the same kind. The person at the computer should then change the status of the inventory record to “Expired”.
10. Modify the status of an inventory item – Note that this is different from use case (2). Here, an actual item in inventory is searched for (possibly using the barcode of the item, if known – otherwise, using some other criteria), and its status changed. The status could be changed to “Returned to Vendor”, “Expired” (if the restaurant is dumping the item because it has realized it is too old), “Damaged”, or “Lost”.
11. Obtain re-order list – This refers to generating a list of inventory item types whose current stock is below the threshold value for that type. For example, if the restaurant has a policy of re-ordering Idaho potatoes when the total number of pounds of this inventory item type falls below – say – 15 lbs. – then Idaho potatoes should be included on this list.
12. Obtain full current inventory – This should only include those items whose status is “Available”.

---------------------------------------------------------------------------------------------------------------------

**DATA MODEL**

**Table:** Inventory Item Type

Fields:

1. **ItemTypeName (Text) (Primary Key)**
2. **Units (Integer) – e.g, 1, 24, etc.**
3. **Unit Measure (Text) – e.g., Keg, Bottle, Bag, etc.**
4. **Validity Days (Integer) – e.g, 7, 8, etc. (-1 if the item does not expire)**
5. **Reorder Point (Decimal) – e.g, 2 (if 2 or less bags of 10-lbs potatoes left, order more)**
6. **Notes (Text)**
7. **Status (Active/Inactive – Default: “Active”)**

Examples:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ItemTypeName** | **Units** | **Units Measure** | **Validity Days** | **Reorder Point** | **Notes** | **Status** |
| Guinness | 1 | Keg | -1 | 2.00 |  | Active |
| Harp | 1 | Keg | -1 | 1.00 |  | Active |
| Sam Seasonal | 1 | Keg | -1 | 1.00 |  | Active |
| Bud Light | 24 | Bottle | -1 | 3.00 |  | Active |
| Heineken | 24 | Bottle | -1 | 2.00 |  | Active |
| White Zin | 6 | Bottle | -1 | 2.00 |  | Active |
| 1-doz Brown Eggs | 1 | Carton | 7 | 4.00 |  | Active |
| Brown Bread Loaf | 1 | Loaf | 4 | 8.00 |  | Active |

Explanations: The “Reorder Point” value needs some explaining. See that it is ‘3.00’ for Bud Light above. This means that they have several units of Bud Light – and these units are actually “cases”, containing 24 bottles each. When they take such a case out of inventory, it means that they scan the barcode for the case, and the 24 bottles are considered “used up” (this is probably because the restaurant’s business is such that when they take a case out one evening, they are reasonably sure that 24 bottles – or very close to it – will be consumed that evening). They may have several such cases in inventory. When the number of cases (again, let us emphasize that these cases contain 24 bottles each) reaches 3.00 or lower (as recorded in the “Inventory Item” table below with “Available” status), the restaurant needs to order more Bud Light, from a vendor they have contracted with to supply this item. The restaurant will figure this out when they run the “Obtain re-order list” use case.

**Table:** Vendor

Fields:

1. **Id (Integer) (Auto-generated primary key)**
2. **Name (Text) – e.g, 1, 24, etc.**
3. **Phone Number (Text) – e.g., Keg, Bottle, Bag, etc.**
4. **Status (Active/Inactive – Default: “Active”)**

Examples:

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Name** | **Phone Number** | **Status** |
| 1 | Custom Brew Crafters | 585-624-4386 | Active |
| 2 | Rochester Beer & Beverage | 585-426-0040 | Active |
| 3 | Southern Wine & Spirits | 585-313-7669 | Active |
| 4 | Kreher Farms | 585-555-1458 | Active |
| 5 | Donald’s Supply Co. | 585-555-3479 | Active |

**Table:** Vendor - Inventory Item Type

Fields:

1. **Id (Integer) (Auto-generated primary key)**
2. **VendorId (Integer) (Foreign key to Vendor table)**
3. **InventoryItemTypeName (Text) (Foreign key to Inventory Item Type table)**
4. **Vendor Price (Text) – e.g, 34.25 (Price at which vendor contracts to supply item)**
5. **Date of Last Update (Text) – e.g, 2018/04/16 (MUST be stored in “YYYY/MM/DD” form)**

**Table:** Inventory Item

Fields:

1. **Barcode (Text) (Primary Key) (Usually is numerical (all digits), of a fixed length, like 9 characters)**
2. **InventoryItemTypeName (Text) (Foreign key to Inventory Item Type table)**
3. **VendorId (Integer) (Foreign key to Vendor table)**
4. **Date Received (Text) – e.g, 2018/04/25 (MUST be stored in “YYYY/MM/DD” form)**
5. **Date of Last Use (Text) – e.g, 2018/04/30 (MUST be stored in “YYYY/MM/DD” form)**
6. **Notes (Text)**
7. **Status (Available/Used/Expired/Returned to Vendor/Damaged/Lost – Default: “Available”)**