Specification for Inventory & BOM System

Last updated: November 7, 2018

Description

FlySorter wants to create an SQL database-backed system to manage inventory of parts and assemblies for its products. This will allow FlySorter to track the quantity of parts on-hand, the location of those parts, the parts and quantities that make up sub-assemblies and products, and the cost of each sub-assembly and product.

The system should provide a web interface to allow a user to perform the following actions:

- Log in to gain access[1], add or remove a user to the system
- Add a new part to the system, and assign it a part number, along with other properties, such as the category, manufacturer's part number, cost, and storage location
- Add a sub-assembly to the system, assign it a part number, and choose the parts (and quantities for each part) that comprise that sub-assembly
- Edit parts and sub-assemblies to update properties / quantities / costs / etc.
- "Receive" parts to increase the inventory count
- "Build" a sub-assembly to decrement the parts' inventory counts and increment the inventory count for the sub-assembly
- "Scrap" a part or sub-assembly to decrement the inventory count
- List all parts, or parts by category, displaying all properties including inventory count
- Search for a part
- Display a list of parts within a given subassembly (recursively, if a sub-assembly contains another sub-assembly)

The completed system will be used for our internal production management, and, if the programmer desires, can be published with an open source license for use by others.

Architecture

Backend: MySQL database

Frontend: TBD

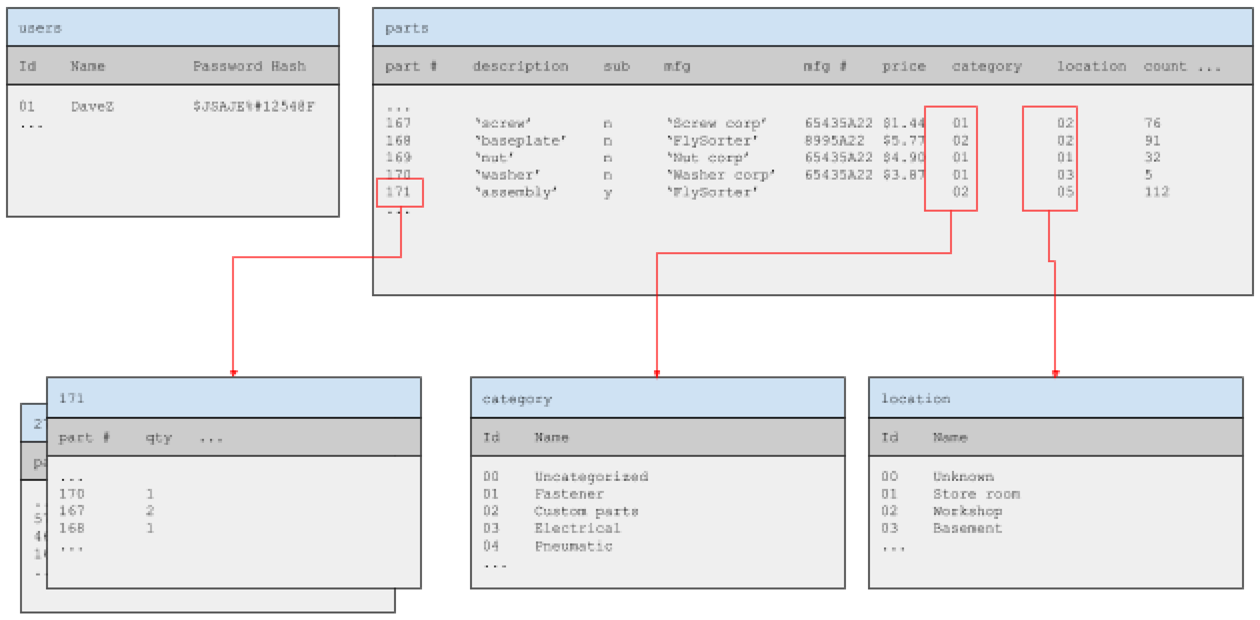
Nice to have: could run on a shared Linux host (via ionos.com). Barring that, a low-cost virtual server is okay.

[1] Not picky about how exactly this happens, as long as we can control access in some way. OAuth, our own database of usernames & hashes, .htpasswd, etc. all fine. All users have the same level of access.

Database Schema

This is the *proposed* database schema. If there is a better way to do things, let's talk!

Table relationships:



users

Id	Name	Password Hash
01	DaveZ	$JSAJE%#12548F
...		

parts

part #	description	sub	mfg	mfg #	price	category	location	count ...
...								
167	'screw'	n	'Screw corp'	65435A22	$1.44	01	02	76
168	'baseplate'	n	'FlySorter'	8995A22	$5.77	02	02	91
169	'nut'	n	'Nut corp'	65435A22	$4.90	01	01	32
170	'washer'	n	'Washer corp'	65435A22	$3.87	01	03	5
171	'assembly'	y	'FlySorter'			02	05	112
...								

171

part #	qty	...
...		
170	1	
167	2	
168	1	
...		

category

Id	Name
00	Uncategorized
01	Fastener
02	Custom parts
03	Electrical
04	Pneumatic
...	

location

Id	Name
00	Unknown
01	Store room
02	Workshop
03	Basement
...	

Table: locations

Column name	Type	Description
loc_id	int	Table ID (unique)
loc_name	text	A physical location where parts are stored

Table: categories

Column name	Type	Description
cat_id	int	Table ID (unique)
cat_name	text	A type of part (e.g. "Fastener" or "Electrical")

Table: parts

Column name	Type	Description
part_id	int	Part number (unique)
part_desc	text	Description of the part or sub-assembly
part_sub	bool	Is this part a sub-assembly or not?
part_src	text	Source (manufacturer/distributor)
part_mfgnum	text	Mfg/Dist part number
part_price	int	Unit price
part_category	id	Id of category
part_location	id	Id of location
part_count	int	Quantity on hand
part_longlead	bool	Is this part a long lead-time item?
part_notes	text	Any additional notes about this part.

Table: [sub-assembly part number, one table per sub-assembly]

Column name	Type	Description
sub_id	int	Table ID (unique)
sub_part	id	Part number
sub_version	text	Part version (if any)
sub_qty	int	Quantity in this sub-assembly
sub_minutes	int	Labor, in minutes

Interface

The application should provide the following interfaces.

Login Screen

When attempting to access any URLs within the app, if the user is not logged in, they should be presented with a login screen.

After entering a valid username and password, they should be presented with the Parts page.

Parts

The Parts page should show a list of all the parts in the database and their associate properties, sortable (up and down) by column header (e.g. group parts by manufacturer, sort by price). Column headers should remain visible when the list is scrolled.

There should be a search box at the top, and once some text is entered (and optionally a button is pressed), the list of parts displayed should be pared down to those that match the search (any field can match). And some way to reset the search to show all parts.

Also at the top, there should be a link to the Category page (e.g. “Edit Categories”) and to the Location page (“Edit Locations”). Optionally an “Edit Users” link, too.

Parts that are actually sub-assemblies should be hyperlinked to the Sub-assembly page, but should still display like other parts. The only difference is that the Price of a sub-assembly is actually a calculated value. The formula is summed over all parts within a sub-assembly:

Price = SUM [((part cost) + (labor minutes * labor cost / 60)) * part qty]

Labor cost (per hour) should be a fixed value for the whole system. Doesn’t need to be editable via the web interface, but should be changeable in code or within the database.

Each part should have an icon or button next to it to edit it. This can happen in place, or in a modal dialog.

There should be a button to add a new part. This can bring up a modal dialog or just add a line to the list and allow the user to fill in fields. Part numbers should start at 100000, and new parts should take the next available part number. It is not necessary to delete a part (it will just be marked obsolete).

When editing or adding a part, input for some fields needs to be sanitized. For example, the Category should provide a means for the user to select from the pre-existing list of Categories. Ditto for Locations.

There should be a button next to each part to increment or decrement the inventory count for that part.

Category List

Should display a list of the existing categories. The first category should be “Uncategorized” and should not be removable. Should be able to add a new category, and remove a category (in which case, all entries with this category should be marked as “Uncategorized”).

There should be a “Done” button that returns the user to the Parts page.

Location List

Should display a list of the existing locations. The first category should be “Unknown” and should not be removable. Should be able to add a new location, and remove a location (in which case, all entries with this location should be marked as “Unknown”).

There should be a “Done” button that returns the user to the Parts page.

Sub-assembly

Each sub-assembly gets its own table in the database. If the table doesn’t exist (like for a newly created sub-assembly), it should be created.

There should be a list of all the parts within the subassembly, and their respective properties (some of which are pulled from the parts table, others of which are pulled from the sub-assembly table).

Sub-assemblies can contain other sub-assemblies (but not themselves), so the display should recurse through and show up to 5 levels of sub-assemblies with some indication of hierarchy (indent, for example).

Each line that is an entry in the primary sub-assembly should have an edit button, and a delete button as well. Deleting a part or sub-assembly from a sub-assembly doesn’t delete the part from the part table, just from this sub-assembly.

The (calculated) total price for the sub-assembly should be displayed at the top of the page.

There should be an “Add Part / Sub-assembly” button that allows the user to add a new line to the current subassembly. Within that interface (again, modal dialog or in-place are fine), there should be an affordance so that the user can search the database for a part to insert into the sub-assembly (so they don't have to just know the part number).

At the top of the sub-assembly page, there should be an additional button: "I BUILT ONE". The button should increment the inventory quantity for this sub-assembly AND decrement the appropriate quantities for each part (the assumption is that building the sub-assembly uses up the individual parts from inventory).

There should be a "Done" button that returns the user to the Parts page.

(Optionally) Add/Remove User

As noted above, we do need to limit access to this application. If it's straightforward and appropriate, it would be helpful to have a way to add & edit users within the app. All users have the same level of access, so there's no need at the moment for different classes of users (administrators vs regular users, for example).