List of All Remaining Tasks to Complete by the End of the Semester

Roughly speaking, I believe that the order that these features are listed in are the approximate priority level for getting these features done in. So us trying to get this week's requirements done first, then getting our old features back online, then implementing our new feature ideas, then finally hopefully get a PACMAN working in there for the extra credit. If you disagree with this ordering, let me know.

New Tasks from Week 11's Assignment 9

All new features are DONE!

Migrate the logging system to use the BOOST library. (Mike)

This task & feature has been Cancelled.

Use QFile & a separate dialog box to create a notes dialog. Export to a .txt file. (Jacob)

DONE

Finish Reimplementing Old Features That Were Disabled

Adding Shoppers to Shopper's Table (Mike)

Before we allow the user to purchase their Shopping List, we need to confirm the Shopper information of the user. To accomplish this, we are going to need to make a way to add and create new Shoppers into the Shoppers Table.

Selecting the Current Shopper (Mike)

Before we allow the user to purchase their Shopping List, we need to confirm the Shopper information of the user. We need a way of selecting who the current Shopper is that will be purchasing the Shopping List.

Purchase the Items in the Shopping List (Mike)

Make sure that we have a current Shopper being selected that will be the one to purchase the current Shopping List.

Also make sure to check the Quantity in the database to ensure that we don't accidentally sell what we don't have.

Add a New User to the Database (Chris)

For admins only.

Change a User's Password (Chris)

For admins only.

Import a .CSV File into the Database (Jacob)

Already partially started on this.

Is similar to the 2nd required feature listed for this week.

Add a Book to the User's Book List from the Inventory (Jacob)

This functionality is started, but it currently behaves more like the shopping list than the original functionality, listed below. We still need to add the extra stuff of what is listed here.

When a book is added to the Book List from the inventory, remove that record from the database.

If the user closes the program while the Book List is populated, then add those books back into the database.

May be combined with our 2nd new feature listed down below.

Print the User's Book List to the Screen (Jacob)

Old functionality for this is DONE

May be combined with our 2nd new feature listed down below.

Export the User's Book list to a .CSV file and Empty the Book List (Jacob)

May be combined with our 2nd new feature listed down below.

Begin Developing 3 New Features

1. Dedicated Login Screen (Mike)

So it'll go Splash Screen, then Log In Dialog Box, and if they can log in it'll bring them to the Main Menu Window. If they can't sign in, then the program closes.

2. Favorite Books List(s) (Chris)

Allowing the user to be able to add a book to their Favorites List. We can even save this Favorites List into a file so that the program will "remember" the user's favorites after the program closes and reopens. If this is not enough for Professor Carmon, then we could shift to having it so that the user can create any number of their own Book Lists! (This has the potential to combine with our previous Book List feature and may require changes to that previous feature if we do combine it with this. As in, for example, saving all Book Lists as .CSV files instead of inside a 1 or more variables.)

3. Admins Being Able to Edit Book Records (Jacob)

So, this is technically possible already with what tools the admins have right now with Exporting and Importing Books, but if we made it possible for them to edit the Book Records in the Database more directly without having to go through all those extra steps it'll make it so much easier on them.

Resolve As Many of Our Listed Issues as Possible

On average, I've found that we have about 15 issues listed in our Issues List each week. Some of these are small, others are large, and some we probably don't need to worry about for our final turn-in. But regardless, we should still try to get as many of these resolved as possible both before we turn this in and before we attempt to work on our possible Extra Credit.

ID #’s 5, 9, 13, & 17: Book List, Import, & Export, (Jacob) (Mandatory)

Each of these Issue Items revolves around the same general features of the Book List functionality, so this should be taken care of by the person working on this functionality.

ID #’s 20, 21, & 22: Shopping List & Purchasing, (Mike) (Mandatory)

So, we have some of the most basic functionality of the Shopping List done, but when it comes to the most important parts like purchasing the list that has not been completed yet. After someone completes this functionality, please look over these issues and make sure that these issues are resolved for the new implementation.

ID #’s 24 & 28: Login Screen & Admin Menu Access, (Mike) (Mandatory)

So technically the current Login Screen works about half-way, so it is not completely disabled. But with us working to rework our Login Screen a little bit, just by having it complete it will resolve this issue.

ID # 25: Previous Functionality, (All) (Mandatory)

Same with Issue ID # 24 above, so long as we get our previous functionality complete as listed before in this document, then this issue gets resolved automatically.

ID # 26: Updating the Menu Bar, (Jacob) (Fully Optional)

This Issue was talking about 2 separate things. The first was removing the buttons in our menu that do nothing in our program and adding in some buttons that we ACTUALLY needed. This first part has already been taken care of.

The second thing was about trying to move our entire Hard Coded Menu Bar out of our Main Window Constructor Method and putting it in our actual Main Window .UI Design. Although this is considered an Issue, this can be considered more of an optional Issue and we can probably ignore this issue for our final. (Only work on this issue AFTER getting in our Extra Credit!)

ID # 29: Status Bar Not Auto-Updating, (Jacob) (Partially Optional)

So, the status bar is working as intended but only when the Main Menu is first created. This functionality should be put on a short timer (somewhere between 2-20 seconds) to update the displayed numbers after a user updates the database. This issue is not 100% important though, so it can be worked on after most of these other issues are resolved first.

ID # 30: Splash Screen Copyright, (Mike) (Partially Mandatory)

So, our Splash Screen is using a Copyrighted image, something that we CANNOT use without at least giving credit for it, even if our project is only for an educational purpose. Jacob has already put a credit for the illustration tucked away into our About Menu Option, but that is very likely not good enough. We need to put our artist’s credit where it matters, at the same time and place as where we are using, i.e., our Splash Screen!

ID # 31: Book ISBN Validation, (None) (Fully Optional)

Basically, we were never able to figure out how to ACTUALLY validate an ISBN. Although this is an important point for our application, because of how confusing it is to figure out, this is an Issue we are very likely to leave in our final application turn in.

ID # 32: Boost Library and Boost Logging, (None)

Since including Boost Logging into our project is no longer a requirement, this Issue has been resolved automatically.

ID # 33: QMessageBox Pop-Ups (Mike)

This is a new Issue that I've noticed in our MainWindow.cpp file, so this document is the first time it is being recorded. This should be reported and listed in our Project Documentation after listing it here. For one of our previous assignments, we were given the requirement of needing to display a QMessageBox whenever our application performs a database operation. Well there are certain parts in our Main Window code that do not adhere to this requirement, so if someone can look into this and add those in/replace when we write messages into the textEditLarge ui element with QMessageBoxes.

Potential Extra Credit Features

Pacman!

We probably won't have enough time to make this ourselves, so if we do this it will probably be by porting someone else's C++ PACMAN code into our QT Project as an easter egg.

I've already found at least 1 GitHub Repo online that is C++ PACMAN that we'll probably be able to use, but I have not looked into it at all, let alone test it, to see if it will work for us.

One notable requirement is that this is meant to be an easter egg for ONLY the admins! So this must be a feature that only the admins will have access to, which means it'll have to be hidden somewhere in the admin menu. That, or at the very least the easter egg will only trigger if the user is flagged as an admin.

Clean Up Unnecessary Items in Our Application

This is mostly here because over half of our current top menu bar is filled with menu buttons that don't do anything. We either need to clean them out or give them purpose. This can be extended to also remind us to clean up our code whenever we can, mainly commented out code.