CIS 227 Assignment 9

Assignment Details

Migrate your logging system to use the BOOST library.

Use QFile and a separate dialog box to create a notes dialog. Export the note to a .txt file

GANTT Chart

Begin the development of 3 new features for your final project.

Team Roles

Lead Programmer – Chris Rodela

UX/UI Programmer – Jacob Wiles

Functional Programmer – Mike Dolan

Program – 70

UX/UI – 35

Function - 35

Documentation – 30

Total Possible Points – 100

**Version 0.4.0**

| REVISION HISTORY | | | |
| --- | --- | --- | --- |
| DATE | VERSION | DESCRIPTION | AUTHOR |
| 2/1/2023 | 0.0.1 | Initial merging of individual code | Jacob Wiles |
| 2/1/2023 | 0.0.2 | Fixing merged code to behave as intended | Jacob Wiles |
| 2/1/2023 | 0.0.3 | Fixing bugs and reformatting the UX and UI | Jacob Wiles |
| 2/1/2023 | 0.1.0 | “Official Release” and Final turn-in for the week | Jacob Wiles |
| 2/15/2023 | 0.2.0 | Added password encryption by implementing the Boost library, Users have ability to exit program, Users can now add books to the books.csv file, users are able to create a list of books to store in their “cart” | Christopher Rodela |
| 2/22/2023 | 0.2.1 | Change over from csv to SQLite database + update database | Jacob Wiles |
| 2/22/2023 | 0.2.2 | Add admin menu | Jacob Wiles |
| 2/22/2023 | 0.2.3 | Fixing back-end queries used by the admin menu | Jacob Wiles |
| 2/23/2023 | 0.2.4 | Adding random values to QUANTITY\_ON\_HAND in the database, Made the shopping list functionality and menu, Added the SHOPPERS Table to the data. | Jacob Wiles |
| 2/24/2023 | 0.2.5 | Various bug fixes from recent versions and adding a warning for the user to the front-end when they choose to export a file. | Jacob Wiles |
| 2/25/2023 | 0.2.6 | Adding a UI prompt for a user adding a new shopper, Adding a “purchase” option to the shopping list menu, Adding basic version of CalcTotalCost() function | Michael Dolan |
| 2/26/2023 | 0.2.7 | Updating database to not accept as many NULL values and adding more database stability | Jacob Wiles |
| 2/28/2023 | 0.2.7.1 | Adding in the front-end of displaying the user’s shopping receipt and letting the user choose how they wish to search for books in the database. | Michael Dolan |
| 3/1/2023 | 0.2.8 | Adding Back-End Database functions and connecting them to the Front-End UI. | Christopher Rodela |
| 3/1/2023 | 0.2.8.1 | Fixing/Adjusting Integrations between the Front-End and Back-End | Jacob Wiles |
| 3/1/2023 | 0.2.8.2 | Various Bugfixes | Jacob Wiles |
| 3/1/2023 | 0.3.0 | “Official Release” and Final turn-in for week 7 | Jacob Wiles |
| 3/14/2023 | 0.3.1 | Made code compatible with QT. Introduced GUI | Jacob Wiles |
| 3/14/2023 | 0.3.2 | Added back-end GUI functionality | Michael Dolan |
| 3/14/2023 | 0.3.3 | Created a log file for db ops, user log ins, user exits | Michael Dolan |
| 3/14/2023 | 0.3.4 | QT GUI implementation and log file complete | Jacob Wiles/Michael Dolan |
| 3/22/2023 | 0.3.5 | Adding a Splash Screen & Menu Bar and displaying the number of books in the database in the Status Bar | Michael Dolan |
| 3/24/2023 | 0.3.5.1 | Adding in our program’s name that we forgot to include in the previous version | Jacob Wiles |
| 3/28/2023 | 0.3.5.2 | Completing the adjustBookQuantityInInventory() function that was left unfinished last week. Also linking up the signal and slot to the AddBook button that was not included in the previous version. | Jacob Wiles |
| 3/28/2023 | 0.3.5.3 | Fixing the blank splash screen to include our intended image. | Michael Dolan |
| 3/28/2023 | 0.3.6 | Adding Invalid Entry Pop-Up Box | Michael Dolan |
| 3/29/2023 | 0.3.6.1 | Adjusting all of our windows for them to appear on their expected screen for multi-monitor displays | Jacob Wiles |
| 3/29/2023 | 0.3.7 | Implementing checking the user’s current hardware information and displaying it to the user. Also setting up our Book class validation functions. | Chris Rodela |
| 3/29/2023 | 0.3.7.1 | Bug fixing the hardware info display box to display properly. Also relocating the Book class validation functions into the Book class. | Jacob Wiles |
| 3/29/2023 | 0.4.0 | Official Week 10 Release | Jacob Wiles |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# INTRODUCTION

## PURPOSE

Identify and describe scope of product whose technical specifications are being documented and describe desired outcome.

Present the user with a splash screen for our program, then have them attempt to log in. They do this by entering a username and password and our system checks if the provided username/password pair is found in our list of users. If not, then inform the user that their log in attempt has failed and prompt them if they would like to try again or exit the program. Our system also checks if said pair is also listed as an admin. Depending on if they are an admin or not, the user will have access to the appropriate menus and options that is available for their security level.

Once the user successfully logs in, they are prompted with the main menu and possibly the admin menu depending on if they are an admin. The options for both the main menu and admin menu include searching the database of books, adding a book to the database (either manually or by importing a .csv file), editing the user’s “book list” or “shopping list,” adding a new user to the database, or changing a user’s password.

## DOCUMENT CONVENTIONS

Describe any naming or structural conventions employed throughout document and how they benefit reader.

Camel case will be used for all variables, functions, and file names, except for any class constructor functions.

UX/UI will be separated from the back-end/functionality logic in separate files, mainly the files named with mainmindow.

Header files will contain function declarations and library includes for their respective .cpp files, apart from main.cpp.

When possible, all variables will be declared at the top of the relevant file or function, including lambda functions, and listed in the approximate order that they are used in within their scope.

Within our codebase, the word database is often shortened to “db.” Also, “BSI,” which is used for some of out folders’ names, is short for Bookstore Inventory.

## REFERENCES

List any referenced document names or links.

<https://cplusplus.com/reference/> - C++’s documentation and reference.

<https://github.com/d99kris/rapidcsv> - Used to help parse large .csv files.

<https://doc.qt.io/> - Qt’s documentation pages.

# DESCRIPTION

## FEATURES

List main features with brief description.

Splash Screen Login Menu – A small window that appears before/as our program begins its startup process. The user needs to be able to enter in a username and password and entering in a correct pair will gain you access to the rest of the system. Uses MD5 encryption and hashing to protect the passwords. Will be separated out as a splash screen before the main window appears.

GUI Menu Display – A UI that will display the current menu, take in user input, and return the appropriate information when required.

.CSV File Reader – A library that can read a given .csv file and parse its data for specific information. The library also helps us writing and exporting .csv files in a similar style.

Database – A database that stores all the book, user, and shopper data. The database lets us query it to either get & receive data from it or to send & set data in it. We can search for books either by their ISBN, Title, Author, Year of Publication, Publisher, MSRP, or the Quantity of that book we currently have on hand.

User Lists – A user has 1 or more list of books that they can either pull from the database or “purchase” from the database.

Log File – The application will keep track of certain, specific events that occur during its runtime by saving a record of the event in a log file along with a date and time of when they occurred. Some of these events are, but are not limited to, having a successful database operation, a user logging into the application, and a user exiting the application.

Invalid Entry Pop-Up Window – Whenever a user enters in an invalid entry into the program, a pop-up window will inform the user that their current input entry is invalid.

Hardware Information Display – Within the Help section of our top menu bar, there is a menu item that will display some of the user’s hardware information in a separate dialog box if they click on it.

## USER OVERVIEW

Define groups and describe user characteristics.

Bookstore workers, admins, &/or owners using a Windows PC that need to maintain an inventory for a bookstore.

## ASSUMPTIONS / DEPENDENCIES

Detail all assumed factors (not known facts) that could potentially impact technical specifications set forth. Include external factors. (What other things do users need to have to be able to operate this application?)

The user is going to be using a Windows PC with at least 1 or more monitor displays.

User will need both a keyboard and a mouse/trackpad.

The user will need to have the RapidCSV, SQLite3, and MD5 libraries installed onto their machine(s).

Not all input validation has been completed for this project yet, so it is assumed that the user will only enter in valid inputs and data from the input prompts and in the files that this project attempts to import.

Although MSRPs, tax, and total cost are all displayed to the user with a precision of two decimal places, the actual total and then recorded “total spent” in the shoppers table is tracked to a percentage of a cent. It is assumed that this level of precision for the shoppers’ total spent is desired to be tracked to such a degree instead of rounding it.

In the requirements that request that the splash screen “take up no more than 15% of the desktop space,” we are interpreting that to mean the splash screen will take up 15% of the screen’s area as opposed to it being scaled to 15% of the screen, which would be about 2.25% of the area of the screen.

# SYSTEM FEATURES

## SYSTEM FEATURE 1: The Splash Screen Login Menu(s)

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | The user must enter a username and password in order to log in and utilize the system  Will be separated out as a splash screen before the launch of the main window |
| **STIMULUS / RESPONSE SEQUENCES** | First the user selects which login menu they wish to log in with, the regular menu or the admin menu. After the user makes their selection there is a prompt for the username and then it will prompt for the password. After checking the username/password pair against the database’s table of users, the user will either be told that the login was successful and send them off to that login’s next menu or they will be informed that they entered an invalid username/password pair. |
| **FUNCTIONAL REQUIREMENTS** | We take the username and password and after passing the password through a MD5 encryption hashing algorithm we check if the given pair is valid using the database users table. If the pair is valid, which means it was found in the database and if we’re logging into the admin menu that the user record is marked as being an admin, then we “log in” the user. Else if it is not valid we display an error. |

## SYSTEM FEATURE 2: Importing, Exporting, and Parsing .CSV Files

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | We are going to be both importing and exporting .csv files that contains book data. This will occasionally use the RapidCSV library to help with reading and parsing incoming files. |
| **STIMULUS / RESPONSE SEQUENCES** | For importing .CSV files, the user is asked to ensure that the .CSV file that they want to import is named “import.csv” and is in the correct directory and to his confirm or cancel. If they hit confirm, then the program will attempt to find the file named “import.csv” and to parse through the file for importing.  For exporting, the user is warned that if they already have a file named “exportedBookList.csv” in the folder that we export to then that file will be deleted and overwritten with the new data from their books list. If they confirm after this warning, then the program will do exactly this. |
| **FUNCTIONAL REQUIREMENTS** | For importing, once the user hits “confirm,” then then the program will attempt to find the file named “import.csv” and to parse through the file and adding its data to the BOOKS table in the database.  For Exporting, after the user confirms with the warning, then the program will open/create the export file and name it “exportedBookList.csv” and populate the .CSV file with the data from the user’s book list. It’ll then clear the user’s book list of all its books. |

## SYSTEM FEATURE 3: The Database

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | We’ll be using SQLite to make our database to store and track our books, users, and shoppers data. Whenever the user chooses to perform an action that involves querying/searching the database for book data, we will give the user the option of choosing which part of the books’ data they want to search for, namely ISBN, Title, Author, Publication Year, Publisher, MSRP, or the Quantity on Hand. |
| **STIMULUS / RESPONSE SEQUENCES** | Outside of “purchasing” 1 or more books from the database inventory, the only users who will be allowed to edit the database will be those logged in as “Admins.” Admins can add and remove book and user records to/from the database as well as editing user’s passwords. For the general-purpose search, we give the user the freedom to choose to search using any of the above criteria. As for when the user is asking to put a single result into one of their lists, they are limited to only searching by ISBN and Title. |
| **FUNCTIONAL REQUIREMENTS** | We have a BOOKS, USERS, and SHOPPERS tables with all their columns, except for DESCRIPTION and GENRE, requiring that they cannot be NULL. Once the user selects how they want to search and what they want to search for, the back-end will construct the appropriate query to give to the database to search its records for any matches. It’ll then construct a list of Book objects to hold the data for the user to use later. |

## SYSTEM FEATURE 4: User Book List & User Shopping List

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | A user has two lists of books that they interact with, their “book list” and their “shopping list.” Users can add books from the database to these lists to perform various tasks with these lists once they are populated with that book data. |
| **STIMULUS / RESPONSE SEQUENCES** | For either of these lists, the user can search the database for exactly one matching result to add that result’s corresponding data to that list. If the user’s search wasn’t narrow enough and returns 2 or more results, then we tell them to try again with a more specific search time next time. |
| **FUNCTIONAL REQUIREMENTS** | The book list is used to create an exported .CSV file that will maybe in the future be used to remove book records from the database. Meanwhile the shopping list is for the user to “purchase” the listed books thereby lowering those books’ Quantity on Hand within the database. |

## SYSTEM FEATURE 5: QT GUI Menu

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | We’ll be implementing a GUI into our project for better user navigation and usability |
| **STIMULUS / RESPONSE SEQUENCES** | Users will now be able to easily navigate the application easier while enhancing user experience. Improves database management by improving UI features for administrators. Includes a top menu bar with various functionality and a bottom status bar that tells the user how many total books are currently on record in the database. |
| **FUNCTIONAL REQUIREMENTS** | The application requires user authentication to access the database management functionality and allows different levels of access and permissions based on user role. |

## SYSTEM FEATURE 6: Log File

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Instead of us using cout; and qDebug() to debug our program, we are going to be making use of a log file to log various events that occur in our program. This is in order to allow the clients that we are making this program for to be able to track what our program is doing without needing to use a code editor. |
| **STIMULUS / RESPONSE SEQUENCES** | The only noticeable visual que for this feature would be if the user had their Windows File Explorer open to where the log file is saved. Then if they refresh the page while the program is running, or if it auto refreshes, they will see the log file’s size increase. |
| **FUNCTIONAL REQUIREMENTS** | As various and certain events occur in our program, we will call our own customer logging function that will both save a record of what is happening in the program to the log file and if the log was successfully written will write an identical version of the log to the Qt debug menu. |

## SYSTEM FEATURE 7: Invalid Entry Popup Window

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Whenever a user either enters in an invalid entry or attempts to submit an input that is invalid, the program will display a pop-up message box window informing the user that their entry is invalid. |
| **STIMULUS / RESPONSE SEQUENCES** | After the user clicks a button that would submit data into the application to use, if any of the inputted data is invalid, then a new window will pop up telling the user that their current entry is invalid. |
| **FUNCTIONAL REQUIREMENTS** | For checking if an inputted Book is valid, all that needs to be done is attempt to construct the Book. The newly created Book will have a property of isValid to inform the application if the information used to make the Book was valid or not. |

## SYSTEM FEATURE 8: Displaying the User’s Hardware Information

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Within the Help menu, there is an option that will open a new window to display some of the user’s hardware information. |
| **STIMULUS / RESPONSE SEQUENCES** | The user will click and hover over the Help menu to display its options, then click on the Hardware Information option. When they do, a new window will display some of the hardware information to the user. |
| **FUNCTIONAL REQUIREMENTS** | By including and using QSysInfo, we can inspect the current user’s machine hardware information. We then take this information, format it into a readable layout, and then display it to the user. |

# REQUIREMENTS OF EXTERNAL INTERFACE

## USER INTERFACES

Describe product / user interface characteristics, including standards, style guides, constraints, functionality, and sample screens if applicable.

User will need both a keyboard and at least 1 monitor.

Having a mouse is not required, but is highly recommended.

The user will need to have the RapidCSV library installed onto their machine(s).

The user will need to have the SQLite3 library installed onto their machine(s).

The user will need to have the MD5 library installed onto their machine(s).

# APPENDICES

## APPENDIX A: GLOSSARY OF TERMS

Define all terms and unique acronyms employed throughout document and specific to project.

DB – Shorthand for “database”

BSI – Shorthand for “Bookstore Inventory”

## APPENDIX B: ANALYSIS DOCUMENTATION

List file / document names / provided links to all diagrams, models, additional findings pertinent to technical specification development. (Basically, just about any other documentation that we make for this project should be included here. Also include any files/filenames that would be required to compile & run our program that we’ve written ourselves, so our .cpp’s, our .h’s, and any .json and .csv that we’ve made should also be listed here, but do not include any of the files that were given to us as part of the assignment, i.e. the “books.json” and “books.csv” that are the inventory files that we were given.)

Assignment8Documentation.docx (This File)

Week 10 Gantt Chart.xlsx

Week 10 Gantt Chart-End.pdf

Week 10 Gantt Chart-Start.pdf

Week10Assignment8TaskListDetailedDescriptions.docx

BSI (Folder)

* backend.cpp
* backend.h
* book.cpp
* book.h
* CMakeLists.txt
* CMakeLists.txt.user
* dbmanager.cpp
* dbmanager.h
* hardwareinfo.cpp
* hardwareinfo.h
* hardwareinfo.ui
* hashpasswordencryptor.cpp
* hashpasswordencryptor.h
* loginscreen.cpp
* loginscreen.h
* loginscreen.ui
* main.cpp
* mainwindow.cpp
* mainwindow.h
* mainwindow.ui
* md5.cpp
* md5.h
* rapidcsv.h
* utilities.cpp
* utilities.h

Build-BSI-Desktop\_Qt\_6\_4\_3\_MinGW\_64\_bit-Debug (Folder)

* (Various other files and folders that get automatically built)
* bookstoreInventory.db
* logfile.txt
* scrollRackSplash.png

If you are using your own debugger &/or complier to view this project with, then make sure to copy the three explicitly listed files above and paste them into your own appropriate “build” folder for the project to use. Otherwise, your version of this project may run into some unexpected issues.

(If you are having problems with getting our submitted build version to run on your machine, then try following these steps to hopefully solve your issue:

1) Go “Build -> Clean”,

2) “Build -> Clear CMake Configuration”,

3) “Build -> Rescan Project”,

4) “Build -> Run CMake”,

5) “Build -> Rebuild”. If “Rebuild” doesn’t show up for you, then just do “Build”.

These are the steps we use to get our code working again even though some of us are using different compliers to build it with. Hopefully these steps should also work for you.)

## APPENDIX C: ISSUES

List all unresolved issues, TBDs, pending decisions, findings required, conflicts, etc. (REMEMBER: If Professor Carmon runs into a bug in our application and he doesn’t see it listed and documented here, it will count as an automatic 0 points for the total “Functionality” portion for this turn in. So, TEST TEST TEST!!! Everybody, test your code and communicate any problems you find! And the Lead needs to test ALL the code and ask if anyone knows of any problems with their work.)

| ISSUES | | |
| --- | --- | --- |
| ID | DESCRIPTION | PARTY RESPONSIBLE |
| 5 | Importing files into the database using the Rapidcsv library can lead to larger ISBNs getting converted into scientific notation. Recommended that we replace the use of the Rapidcsv library for importing .csv files with the regular ifstream and doing our parsing manually. | Christopher R |
| 9 | The export book list function doesn't include the new fields MSRP and QuantityOnHand. This might not be needed but we can ask Prof. Carmon about if we'd need it or not. | Michael D |
| 13 | There is currently no input validation done for when we import a file into the database. We need to check that all of the data that we receive from the file is valid before attempting to add it in to the database. | Jacob W |
| 17 | The function that is supposed to remove a book from the database currently does nothing. We probably need to implement that. | Jacob W |
| 20 | In our current implementation, the user is only able to purchase a single copy of each book in their shopping list at a time. The back-end was actually mostly already ready to handle this behavior, but our current front-end doesn’t support any way for the user to make use of it. In the future, we will need to figure out a way to allow the user to purchase more than 1 copy of each book at a time. | Jacob W |
| 21 | The “Add a book to your ‘shopping list’ from the database” menu item currently allows the user to add a book to their shopping list when the QUANTITY of that book in the database is 0. There should be a check to make sure that the QUANTITY is 1 or more before adding that book to the user’s shopping list. | Michael D |
| 22 | Our current “purchaseShoppingList()” function in the back-end handles the increasing of our TOTAL\_SPENT and the decrease of our QUANTITY\_ON\_HAND in two separate sub-functions. This can lead to situations where if the shopping list includes a book that we have a 0 for its QUANTITY, then the shopper still gets docked for the full price of the order but we only update all of the books in the list before the book with the 0 QUANTITY and not updating any of the books that come after. The recommended solution is to combine the two sub-functions increaseTotalSpent() and decreaseBoughtBooks() into the purchaseShoppingList() function and to check that the book is available first, then add the two UPDATE queries to the bigQuery string that will run if there is no errors up to that point at the end of the function. | Michael D |
| 24 | Currently the login screen is disabled due to an error we couldn’t fix | Chris R |
| 25 | Still in the process of reenabling previous functionality after switch to GUI from console | All |
| 26 | Need to update the layout of the menu bar | Mike D |
| 28 | Our admin menu is currently available to all users, it is not being disabled for non-admin users. Make it so that this menu is disabled normally but is enabled if the user logs in as an Admin. Only solve this issue AFTER fixing issue 24. | Jacob Wiles |
| 29 | Our status bar at the bottom of the main window does show the number of books is listed in or inventory, but it does not update when we add any new books like when we add a new book record or change the Quantity on Hand for a book. These numbers need to be automatically updated to properly reflect the actual values when they get updated. Current suggestion is to put the status bar on a timer that will periodically check the database’s current amounts, probably about every 5-15 seconds or so. | Jacob Wiles |
| 30 | For our splash screen image, we are currently using a copyrighted art piece without giving the artist any credit for it. We either need to change the image we are currently using for our splash screen or preferably include a proper artist/illustration credit to the artist who made the illustration. | Mike D |
| 31 | For our Book class validation functions, we don't ACTUALLY validate that an ISBN is a valid "ISBN," as what we have is more of an approximate. This is something that should be documented as an issue/bug, but we probably don't need to worry about fixing it since I believe what we have now is good enough to say that we have fulfilled the searching requirements. |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |