CIS 227 Assignment 2

Assignment Details

Modify Assignment 2 to

Allow the user to add a book to inventory.

Allow the user to create a list of books. And remove the books from inventory.

Allow user to print the list to the screen containing book ISBN, Title, and author and total number of items in the list.

Add a field for book description/plot and genre.

Allow user to export a new book list (csv or JSON)

All inputs must be validated for type.

All passwords must be encrypted

Exit the program on user request

Team Roles

Lead Programmer – Christopher Rodela

UX/UI Programmer – Jacob Wiles

Functional Programmer – Michael Dolan

Program – 70

UX/UI – 35

Function - 35

Documentation – 30

Total Possible Points – 100

**Version 0.1.0**

| 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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# INTRODUCTION

## PURPOSE

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

Have the user attempt to log in by entering a username and password and check 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.

Once the user successfully logs in, prompt them to enter in a book title to search our files for. The program will then take that input, compare it to the book titles in our books.csv file, and output & display a list of up to matching book titles along with the books’ authors, publishers, and publication years. If the program returns a full list of 50 books, then it will inform the user that there is probably more results that can be returned and asks the user if they wish to see the ”next page” of results. If so, then it will repeat the process above and load the next set of results that come after the most recent results.

Once either the user declines to see any more possible results or the program returns less than 50 matching results, then the program will ask the user if they would like to search for a book title again or to exit the program. If the user chooses to use the search again, then the program will follow the above steps starting after the user successfully logs in, allowing them to use the search function without needing to log in again.

## DOCUMENT CONVENTIONS

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

Camelcase will be used for all variable, function, and file names, except for any class constructor functions. UX/UI will be separated from the back-end logic in a separate .cpp file. Header files will contain function declarations and library includes. All variables will be declared at the top of the relevant file and listed in the approximate order that they are used in within that file.

## REFERENCES

List any referenced document names or links.

<https://cplusplus.com/reference/vector/vector/> - Used to help utilize the vector class.

<https://cplusplus.com/reference/string/string/> - Mainly using the find() function section.

# DESCRIPTION

## FEATURES

List main features with brief description.

A login – Being able to enter in a username and password and entering in a correct pair will gain you access to the rest of the system.

A file reader – A library that can read a given .csv file and parse that data for useful information.

A display – A UI that will display the log in process, take in user input, and return the appropriate information when required.

## USER OVERVIEW

Define groups and describe user characteristics.

Bookstore workers &/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 have to have in order to operate this application?)

The user is going to be using a Windows PC.

This is going to be a console-based application, so the user will have to have access to the Windows command prompt. This accessibility requirement is referring both in terms of having the appropriate user permissions within the system as well as having and being able to operate a functioning keyboard or a suitable equivalent that can interact with the command prompt.

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

# SYSTEM FEATURES

## SYSTEM FEATURE 1: The Login Process

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | The user must enter a username and password in order to log in and utilize the system |
| **STIMULUS / RESPONSE SEQUENCES** | First there is a prompt for the username and then it will prompt for the password. It will then check this pair of data against the users.csv file. If the username is found and the correct/matching password is listed for that username, then it will return the message, “Welcome {enter username here}!” Otherwise, it will return with the message, “Invalid username/password,“ along with a prompt for the option of either trying to log in again or to exit the program. |
| **FUNCTIONAL REQUIREMENTS** | We want to take both the username and password at the same time and check if the given pair is valid. We need a user database/file using the .CSV file format. (We need to turn this “database” in along with the rest of our other files.) |

## SYSTEM FEATURE 2: Importing and Reading from .JSON and/or .CSV Files

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | We are going to be importing, reading, and parsing data from multiple external files, the books and the users files, using the .CSV format. This will require the use of the RapidCSV library. |
| **STIMULUS / RESPONSE SEQUENCES** | Once the user gives their input, the program will then open and load the file from the directory, read and parse the file for the relevant data and saving any matches that it finds, and then closing and returning the results of the file parse from those matches. |
| **FUNCTIONAL REQUIREMENTS** | The JSON file to be imported, or the CSV file to be imported, (depending on which we use). |

## SYSTEM FEATURE 3: Searching/Querying Through the Book Inventory File/Database and Displaying the Relevant Search Information

|  |  |
| --- | --- |
| **DESCRIPTION AND PRIORITY** | Given an input of “book title” from the user, the back end will search through the books.CSV file to find any books that contain the word(s) listed in its “Book-Title” column. It will save all that book’s information into an array/vector until either it goes through the entire books.CSV file or until it saves the maximum number of books that can be shown and listed in our display. |
| **STIMULUS / RESPONSE SEQUENCES** | The user will be prompted to enter in a book title to search. Once they enter in that input, they will be notified that the program is loading its results and to please wait patiently. Once the results are loaded, all of the retrieved information will be displayed to the user. If there is any more predicted results, then the user will be asked if the would like to try to load those additional results. If they say yes to this then the program will repeat this process. Otherwise if they no or if there is no expected additional results the program will move on to asking if the user would like to search again for another book title or to exit the program. If they answer yes then it repeats the above functionality, otherwise the program exits. |
| **FUNCTIONAL REQUIREMENTS** | Once the back-end is done parsing the books.CSV file and returns is matching results, then take each of the returned book objects’ information and display it for the user to read. Then if we got our maximum number of books returned from the back-end, default value is 50, then prompt the user for if they would like to try to load any additional possible matches. If they say yes then do so, otherwise ask if they wish to search again. If yes then do so, otherwise exit program. |

# REQUIREMENTS OF EXTERNAL INTERFACE

## USER INTERFACES

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

This program is going to be a console application, which will require suitable user permissions to access and interact with the Windows command prompt as well as a functioning keyboard or a suitable equivalent that can interact with the command prompt. A mouse or a mouse equivalent will not be required for the functional operation of this application but having such equipment will allow the user an additional way to be able to close the program prematurely at their own discretion.

The user will need to have the RapidCSV 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.

## 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.)

Group4AssignmentDocumentationWeek3.docx (This file)

users.csv

CPP2Group4Assignment2BookstoreInventory (Folder)

* include (Folder)
  + rapidcsv.h
* BackEnd.cpp
* BackEnd.h
* Book.cpp
* Book.h
* CPP2Group4Assignment2BookstoreInventory.cpp
* rapidcsv.h

## 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 |
| 0 | Whenever the program asks the user for input, they can press the enter key while no other input is given to move on to the next line in the command prompt window. This makes it so that the user can keep adding new lines in the command window to offset our menu from its intended position. This does not affect the functionality of the program in any way, but it does affect the possible viewing experience of the user. Recommended solution involves implementing cin.get() or getLine(cin, input) to replace the cin << input; lines. | The currently next UX/UI Programmer (Jacob W.) |
| 1 | Whenever the user inputs 2 or more words when we use the line cin << input; in our code, the next time we call cin << input; It takes in the 2nd word the user inputted at that time and uses that as the given input, basically causing our program to usually “skip over” our lines of code. Recommended solution involves replacing all of the cin << input; with getLine(cin, input). | The currently next UX/UI Programmer (Jacob W.) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |