Library Management System

- Sushil Bohara (sb7702)

1. Introduction and General Description

1.1. Introduction

"Library Management System" is the program that allows the user to log in and utilize the features related to keeping records of the book (book ID, book name, author, publication, quantity, and price), searching for the book in the system, and keeping track of borrowed/returned books. If a student borrows a book, the book is deleted from the list of available books and is added to the borrowed list. Similarly, the information of the student who borrowed will be linked with the book along with borrowed and returned dates. Only the librarian is able to make changes to the system. This program solves the problem of keeping the records in the hard copy or buying the expensive software to do the same tasks. Moreover, It serves for the better management of the library at a low cost.

1.2. Approach

The objective of this project is to develop a program that can maintain, organize and handle the books in the library systematically. Specifically, this program allows logging into the system as a librarian and as a student or reader. The software presents the user with the main menu with options to sign into the system as a librarian or as a student. The librarian can perform multiple tasks like searching for books in the library, adding and removing books, issuing books on loans, and keeping track of the returned books. On the other hand, the student has access to only one task which is to search for the book. The program is designed by keeping in mind the student can only search for the book and in order to borrow it, he or she must consult the librarian. The program is approached based on the use case UML diagram in figure (i) of section 2.1.

1.3. Solution

The record of the available books in the library is stored in the CSV file named "Data.csv" which keeps information like the name of the book, book ID, author's name, publication, quantity, and price. Book ID is assigned to the books in such a way that multiple copies of the same book will have the same ID; this will help for finding the book faster and easier in the system and the number of copies is reflected as quantity. Similarly, the records of borrowed and returned books are also stored in separate CSV files. When the book is issued or returned, the system automatically stores the corresponding date in the CSV file under the headings "Issued date" and "returned date" respectively. If the book is borrowed by a student and not returned, the returned date in the CSV file is set to 0-0-0.

The main functions used in the program are:

1) searchbook: both students/readers and librarians can search for the book in the system using book id (recommended), book name, or author's name.
2) addbook: The librarian can add a new book to the system. If the book already exists which is found by verifying the book id, only the quantity of the existing book is increased in the CSV file “Data.csv”. Otherwise, the whole book information is written in the new line.
3) deletebook: The librarian can delete the book. If there are multiple copies of the same book, the quantity of the book decreases in the CSV file “Data.csv”. Otherwise, the book is removed from the file.
4) Issuebook: The librarian can issue the book to the student. The student’s name and ID (which is also made string) along with the issue date are recorded in the separate file called “loans.csv”. At the same time, the book is deleted from the “Data.csv”.
5) returnbook: The librarian can return the student's book. The return date is noted automatically and the record is kept in the file “Returnedbooks.csv”.

2. UML Diagrams to show the working of the system

The UML diagram of the library management system is illustrated with (i) the UML use case diagram and (ii) the UML class diagram.

2.1. Use Case UML Diagram to show the working of the system

The use case UML diagram (i) shows the various functionalities that can be performed by the user on the system. The two actors in the system are the librarian and the student. The solid oval diagram shows the functions which are linked with the particular or both actor(s) using the solid line. The rectangular boxes enclose the inputs which must be provided to the system to perform the task inside the oval to which they are connected. Similarly, the dotted oval diagram shows the files used in the system. If the arrow is pointing towards the file, it implies that the data inside the rectangular boxes are written into the file while the outward pointing arrow denotes the data being read from the file.

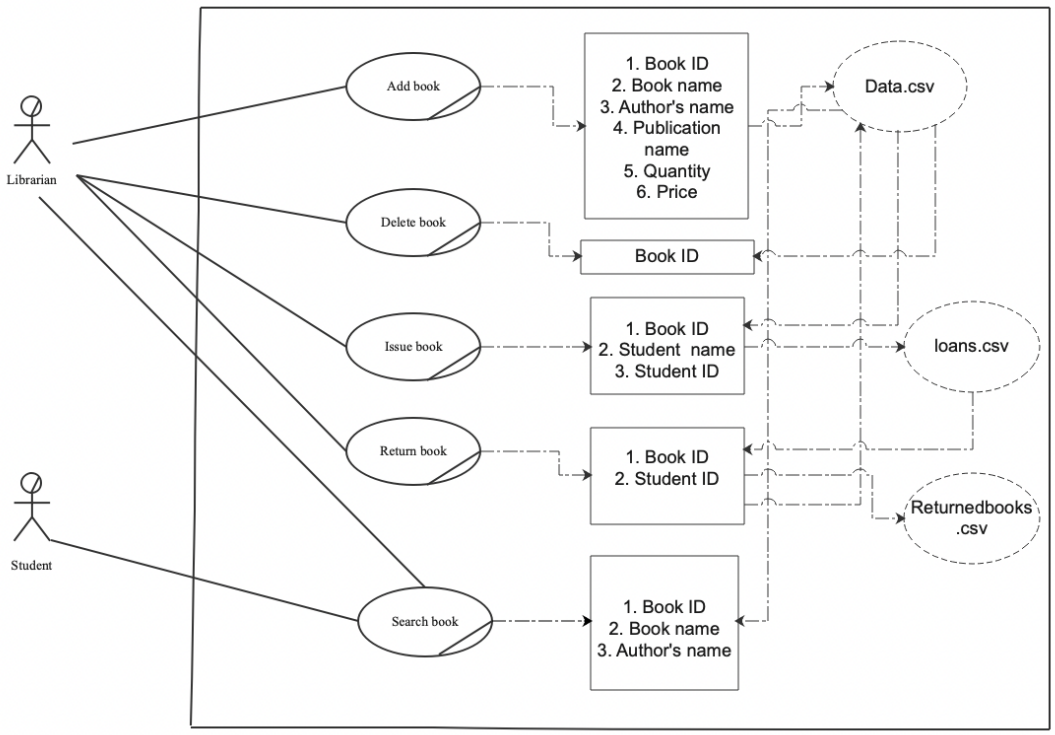


Figure (i): UML use case diagram of the library management system

2.2. UML class Diagram to show the working of the system

Figure (ii) shows the relationship between different classes used in this project. The standard arrows are drawn to show the specific relations. *

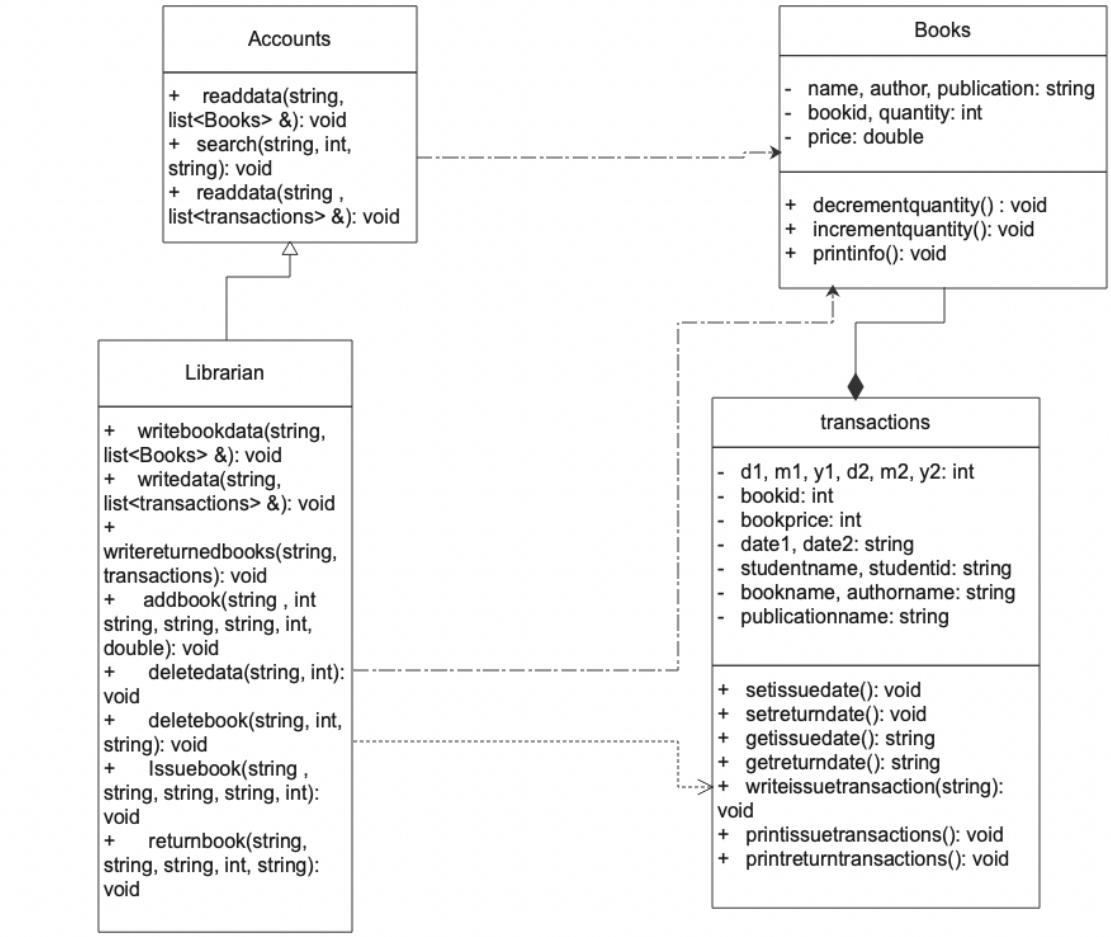


Figure (ii): UML class diagram of the library management system

* 'Librarian' is a subclass of 'Accounts; and friend of 'transactions'.
* Getters and setters of attributes are not shown to save space.

1. OOP paradigms used in the code

This program uses the various paradigms of c++ and object-oriented programming like classes, inheritance, polymorphism, function overloading, etc.

1. Inheritance:
   The program contains the base class called Accounts and since Librarian inherits all characters of class Accounts, it is made into subclasses of class Accounts.

```
class Librarian : public Accounts
```

2. Friendship:
   The class "Librarian" is made friend class of the class "transactions". The class transactions contain the attributes and functions related to issuing and returning the borrowed book. Making the Librarian friend of this class helps to modify the private members of transactions without getters and setters.

```
friend class Librarian;
```

3. Standard template library:
   The 'list' from the standard template library is used multiple times to store the objects of the class "Books". The iterator is used to traverse the list. "push_back" method is used to store data in the list.

```
list<Books> Bookobjects;
list<Books>::iterator itr;
for (itr = Bookobjects.begin(); itr != Bookobjects.end(); it++)
{
    .   .   .    .   .
    .   .   .    .   .
}
```

4. Function overloading
   Some functions with similar functionalities but slightly different implementations are overloaded in the code to improve the readability.

```
void readdata(string infilename, list<Books> &Bookobjects)
{.   .   .    .  .}
void readdata(string infilename, list<transactions> &Bookobjects)
{.   .   .    .   .}
```

4. Optimization

Steps are taken to optimize the code as much as possible. A few instances of optimization in the code are:

1. Inline functions:
   Simple functions like print are made the inline function.
   Example:
```
inline void printissuetransactions()
{.     .      .      .      .}
```

2. Passing the arguments by parameters
   The list is passed as the parameters into the functions.
   Example:
```
void readdata(string infilename, list<Books> &Bookobjects)
{.     .     .      .     .      .}
```

3. Objects are created by analyzing the situation instead of creating the pointers in order to make the compilation efficient. This is because separating the stack memory is easier and faster than separating the heap in large systems.
   Example: In line 314, the object of the class book is created as an alternative to the pointer.
```
Books obj;  // instead of pointer
```

4. While reading from the file, if there is an empty line, the reading is stopped to reduce the garbage collection.
   Example:
```
if (line.empty())
{break;}
```

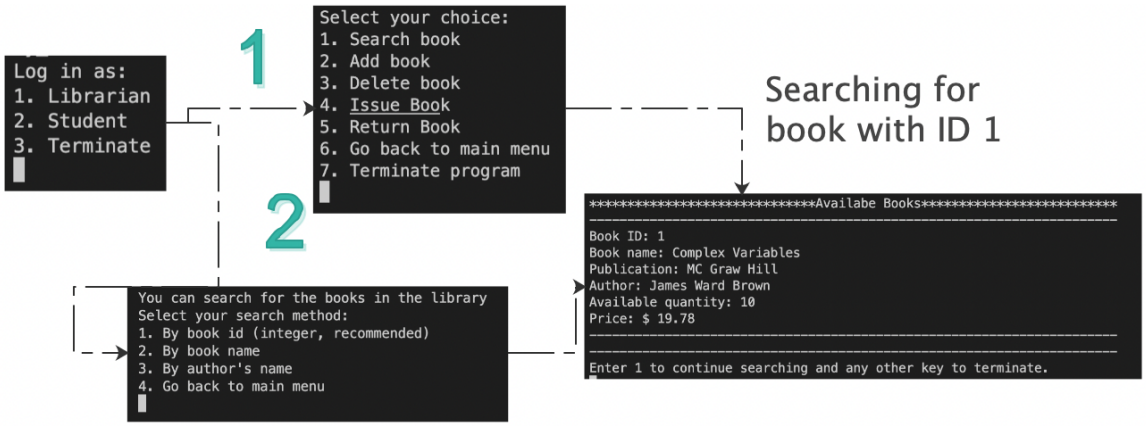
5. Destructors for constructors are created, getters are made constant, and all files are manually closed.
   Example: Some of such codes used are:
```
~Books() {}
string getname () const
{.     .      .      .}
infile.close();
outfile.close();
```

5. User's Guide / Implementation

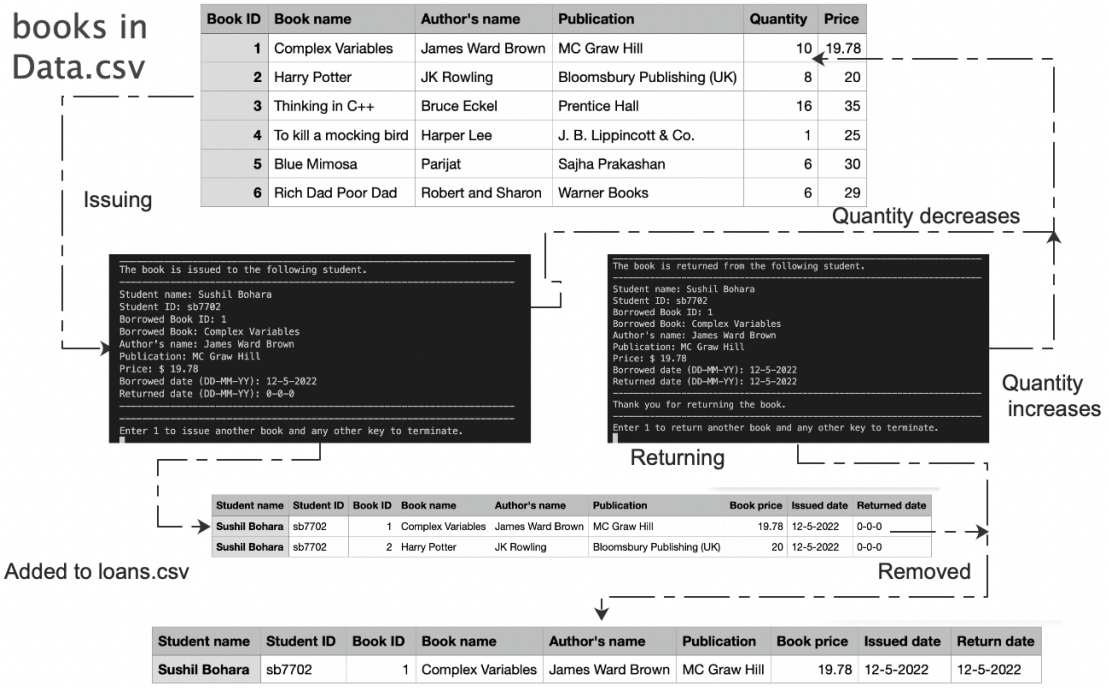
To run the program, compile the CPP file named "Library_sb7702.cpp". Please make that the provided CSV files "Data.csv", "Loans.csv" and "Returnedbooks.csv" are also kept in the same directory as the CPP file. IOS-specific code "clear" is used; so Windows users are advised to find and replace all instances with "CLS". Please do not make edits to these files (like adding the extra spaces, new lines, etc).

As of now, there are 7 books (ID: 1 through 7) with different quantities in the system. Users can use these ID numbers to check the working of various functionalities. They can add books by assigning the ID themselves.

Some snapshots of implementation are:



Issuing / Returning:



Book ID	Book name	Author's name	Publication	Quantity	Price
1	Complex Variables	James Ward Brown	MC Graw Hill	10	19.78
2	Harry Potter	JK Rowling	Bloomsbury Publishing (UK)	8	20
3	Thinking in C++	Bruce Eckel	Prentice Hall	16	35
4	To kill a mocking bird	Harper Lee	J. B. Lippincott & Co.	1	25
5	Blue Mimosa	Parijat	Sajha Prakashan	6	30
6	Rich Dad Poor Dad	Robert and Sharon	Warner Books	6	29

Student name	Student ID	Book ID	Book name	Author's name	Publication	Book price	Issued date	Returned date
Sushil Bohara	sb7702	1	Complex Variables	James Ward Brown	MC Graw Hill	19.78	12-5-2022	0-0-0
Sushil Bohara	sb7702	2	Harry Potter	JK Rowling	Bloomsbury Publishing (UK)	20	12-5-2022	0-0-0

Student name	Student ID	Book ID	Book name	Author's name	Publication	Book price	Issued date	Return date
Sushil Bohara	sb7702	1	Complex Variables	James Ward Brown	MC Graw Hill	19.78	12-5-2022	12-5-2022

6. References

[1]*Bibliosoft-Library Automation Software*, 12-Apr-2016. [Online].Available: https://libraryautomationsoftware.wordpress.com

[2] msschoolsoftware, "School Library Management Software," *YouTube*, 01-Dec-2013. https://www.youtube.com/watch?v=cmckNcCtA_s. [Accessed: 12-May-2022].

[3] N. *, "Library Management System: Library Management Software in India," SkoolBeep, 18-Aug-2021.[Online]. https://www.skoolbeep.com/blog/library-management-system

[4]"Library Management System without file handling" https://www.youtube.com/?watch=. [Accessed: 12-May-2022].