CS112B Semester Project

Library Management System Using C++ OOP Principles

Sohaib Nasir
2021609
Faculty of Computer Engineering
Ghulam Ishaq Khan Institute
u2021609@giki.edu.pk

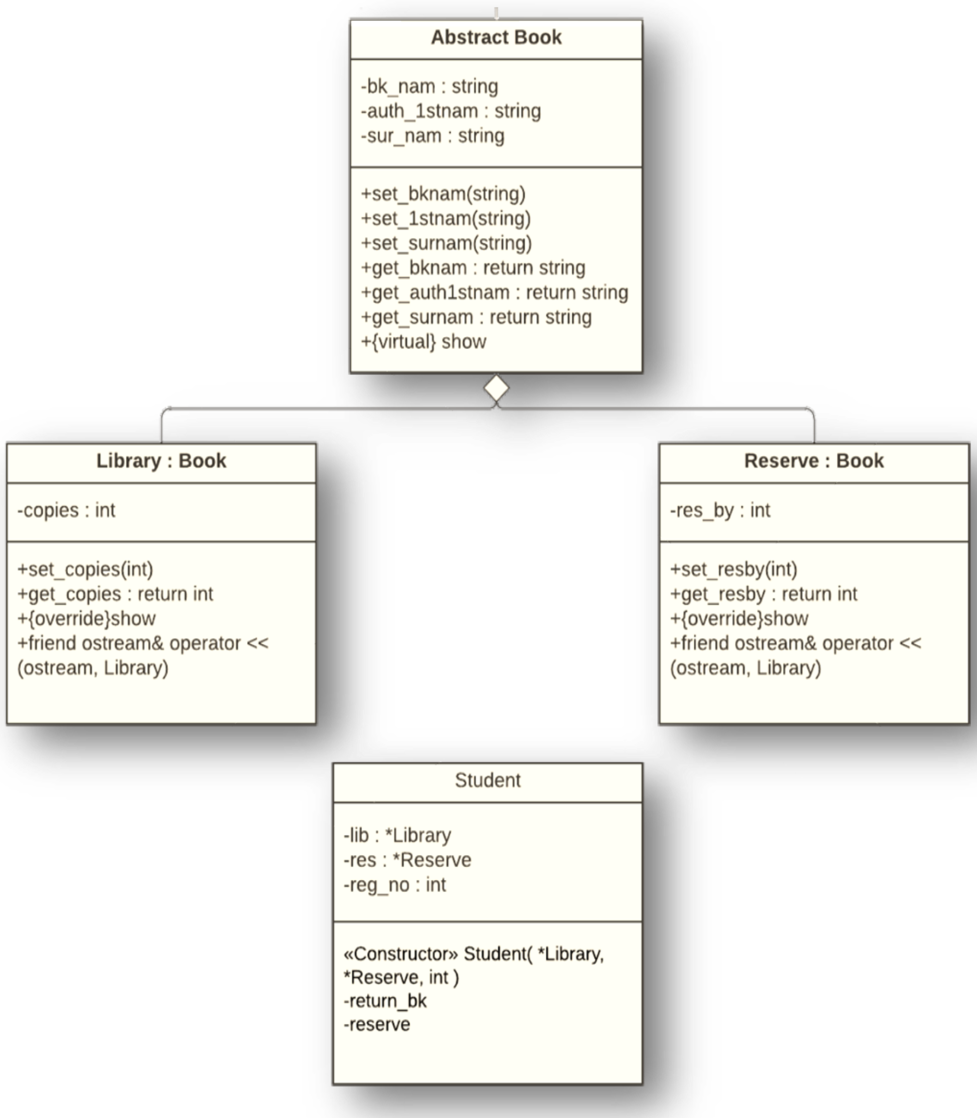
Nauman Asif
2021510
Faculty of Computer Engineering
Ghulam Ishaq Khan Institute
u2021510@giki.edu.pk

Problem statement*— The aim of our project was to develop a reliable library management system on C++ using Object Oriented Programming principles. The algorithm will search the file and if the book available it reserves it and display a message with the book information. The user will also be able to search a book with its title or author's surname.*

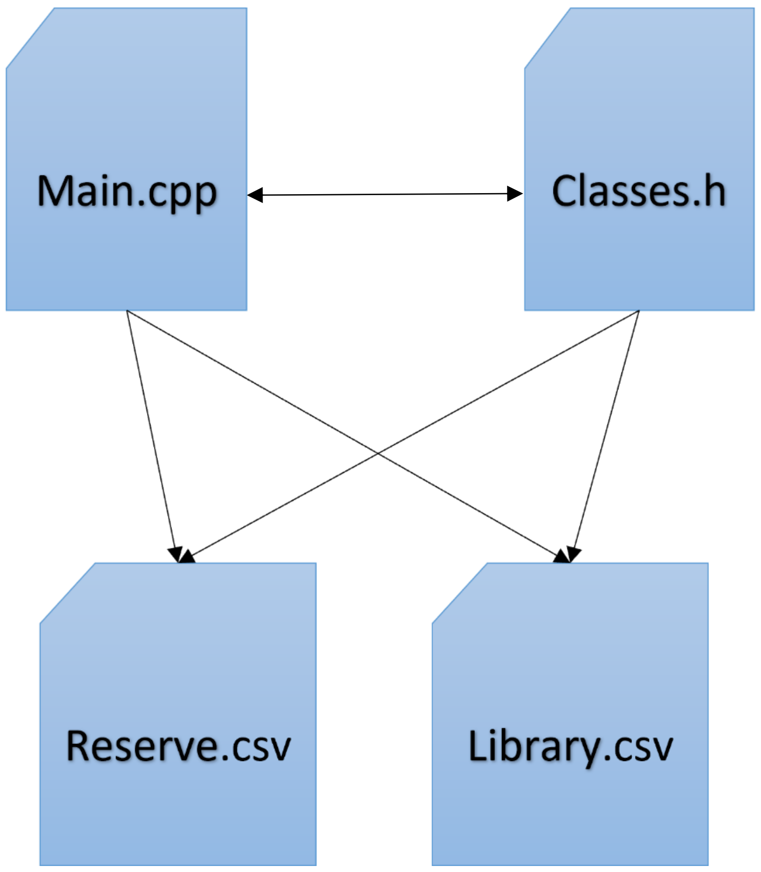
INTRODUCTION:

With the objective being a library management system, our system enables the user to reserve book, return, and utilize other functions relating to the system. The system has been achieved using various OOP principles such as inheritance, operator overloading, friend classes, and polymorphism to achieve this purpose. Also, file handling has played an important role in permanent storage of data about the books and by whom it has been reserved.

CLASS DIAGRAM (UML):



FUNCTIONAL REQUIREMENTS:

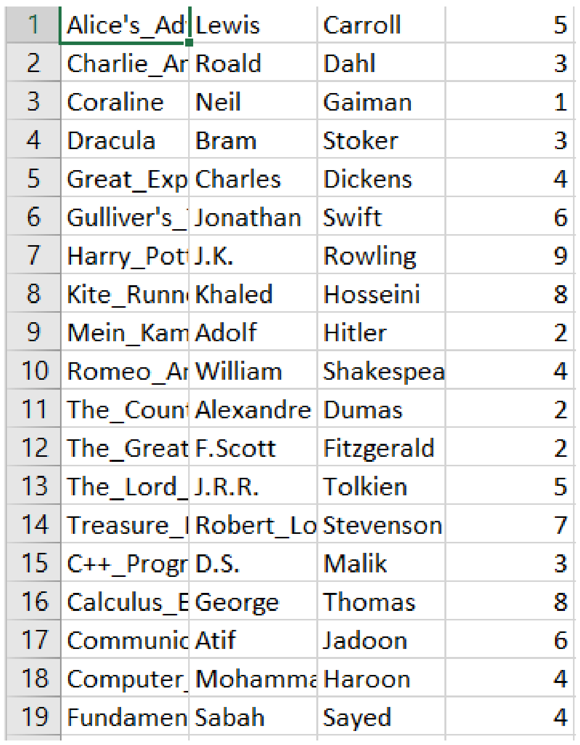


The program consists of one .cpp file and one .h files:

- Main.cpp
- Classes.h (header file containing Book, Library, Reserve and Student classes along with related functions)

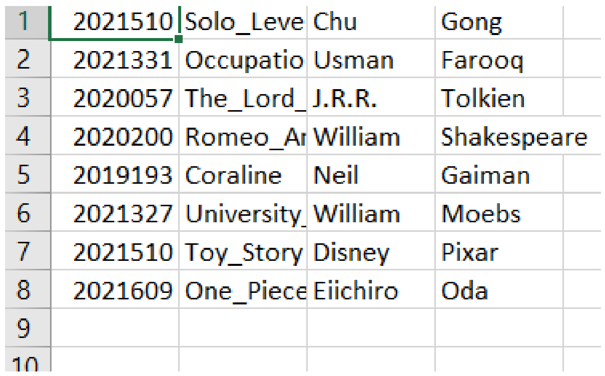
The program needs or utilizes 2 files to perform its functions:

Library.csv: The file includes book name, the authors name information, and the copies of the books remaining. Records are stored in the following format: (S/No., book name, author's 1st name, author's surname, the number of copies remaining).



1	Alice's_Ad	Lewis	Carroll	5
2	Charlie_Ar	Roald	Dahl	3
3	Coraline	Neil	Gaiman	1
4	Dracula	Bram	Stoker	3
5	Great_Exp	Charles	Dickens	4
6	Gulliver's_	Jonathan	Swift	6
7	Harry_Pot	J.K.	Rowling	9
8	Kite_Runn	Khaled	Hosseini	8
9	Mein_Kam	Adolf	Hitler	2
10	Romeo_Ar	William	Shakespea	4
11	The_Count	Alexandre	Dumas	2
12	The_Great	F.Scott	Fitzgerald	2
13	The_Lord_	J.R.R.	Tolkien	5
14	Treasure_I	Robert_Lo	Stevenson	7
15	C++_Progr	D.S.	Malik	3
16	Calculus_E	George	Thomas	8
17	Communic	Atif	Jadoon	6
18	Computer_	Mohamma	Haroon	4
19	Fundamen	Sabah	Sayed	4

Reserve.csv: This csv file contains the registration number by which the book is reserved, the book's name, the author's name information:



1	2021510	Solo_Leve	Chu	Gong
2	2021331	Occupatio	Usman	Farooq
3	2020057	The_Lord_	J.R.R.	Tolkien
4	2020200	Romeo_Ar	William	Shakespeare
5	2019193	Coraline	Neil	Gaiman
6	2021327	University_	William	Moebs
7	2021510	Toy_Story	Disney	Pixar
8	2021609	One_Piece	Eiichiro	Oda
9				
10				

THE CODE:

Main.cpp:

```cpp
// Including the Libraries
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <stdexcept>
// Including the header file
#include "Classes.h"

using namespace std;

// Do-Again function that would be called again and again at multiple times
// so to keep it modular we kept it as a separate function
void do_again(Library* lib, Reserve* res) {
    // Making pointer object of Abstract Class
    Book* bk;

    cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n"
        << "\tBook Title: || Author Name: || Copies Currently Remaining:\n"
        << "\t----------------------------------------------------------\n";

    // Giving the address of Library object array to the Book pointer
    for(int i = 0; i < size_lib; i++) {
        bk = &lib[i];
        bk->show();
        bk++;
    }
    cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n"
        << "Books currently registered by students:\n"
        << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n";

    cout << "\tReg#: || Book Title: || Author Name:\n"
        << "\t----------------------------------------------------------\n";

    // Giving the address of Reserve object array to the Book pointer
    for(int i = 0; i < size_res; i++) {
        bk = &res[i];
        bk->show();
        bk++;
    }
    cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n";

    int reg_no;
    string act;
    bool tag = false;
    char redo;

    cout << "\nEnter your Registration number: ";
    cin >> reg_no;

    for(int i = 0; i < size_res; i++) {
        if(reg_no == res[i].get_resby()) {
            tag = true;
        }
    }
    if(tag == true) {
        cout << "Welcome back, ";
    }
    else if(tag == false) {
        cout << "Welcome ";
    }

    cout << reg_no << " to the library.\n";

    // Making an Object of Class Student
    Student stu(lib, res, reg_no);

    // Asking what to do
    cout << "What would you like to do? [1.Reserve a new book, 2.Return a book, 3.Quit]: ";
    cin >> act;

    if(act == "1") {
        stu.reserve();

        cout << "Would you like to continue? [Y or N]: ";
        cin >> redo;

        if(redo == 'Y' || redo == 'y') {
            cout << "\nAfter changes, the Remaining Books Available are:\n";
            do_again(lib, res);
        }
    }
    else if(act == "2") {
        if(tag == false) {
            cout << "You have no books reserved from the library.\n";
        }
        else if(tag == true) {
            stu.return_bk();
        }

        cout << "Would you like to continue? [Y or N]: ";
        cin >> redo;

        if(redo == 'Y' || redo == 'y') {
            cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n"
                << "\nAfter changes, the Remaining Books Available are:\n";
            do_again(lib, res);
        }
    }
    else if(act == "3") {
        cout << "\nSee you next time.\n";
    }
    else {
        cerr << "\nWrong input, try again.\n";
        do_again(lib, res);
    }
}

// Function to store the Library and Reserve functions into the Library.csv and Reserved.csv files
void store(Library* lib, Reserve* res) {
    fstream fout;

    fout.open("Library.csv", ios::out);

    for(int i = 0; i < size_lib; i++) {
        fout << lib[i].get_bknam() << "," << lib[i].get_auth1stnam() << "," << lib[i].get_surnam() << "," << lib[i].get_copies() << endl;
    }

    fout.close();

    fout.open("Reserved.csv", ios::out);

    for(int i = 0; i < size_res; i++) {
        fout << res[i].get_resby() << "," << res[i].get_bknam() << "," << res[i].get_auth1stnam() << "," << res[i].get_surnam() << endl;
    }

    fout.close();

    cout << "\nThank you for using our Library.\n"
        << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n";
}

// Main function
int main() {
    // Getting the sizes
    get_lib_size();
    get_res_size();

    // Making Arrays
    Library* lib = new Library[--size_lib];
    Reserve* res = new Reserve[--size_res];

    // Iniitializing from the .csv files
    lib = initialize_lib();
    res = initialize_res();

    cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n"
        << "The Books Currently available are:\n";
    // Start the working
    do_again(lib, res);

    // Finally storing the data in the .csv files
    store(lib, res);
}
```

Classes.h:

```
#ifndef CLASSES
#define CLASSES

// Including Libraries
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <stdexcept>
```

```

using namespace std;

// Globally defining the sizes of both Library and the Reserved books
int size_lib = 0;
int size_res = 0;

// Abstract Base Class Book
class Book {
protected:
    string bk_nam, auth_1stnam, sur_nam;

public:
    // Setter and Getter functions
    void set_bknam(string name) {
        bk_nam = name;
    }
    string get_bknam() {
        return bk_nam;
    }

    void set_auth1stnam(string name) {
        auth_1stnam = name;
    }
    string get_auth1stnam() {
        return auth_1stnam;
    }

    void set_surnam(string name) {
        sur_nam = name;
    }
    string get_surnam() {
        return sur_nam;
    }

    // Pure Virtual function show() causing the class to be Abstract
    virtual void show() = 0;
};

// Derived Class Library from Abstract Base Class Book
class Library : public Book {
protected:
    int copies;
    friend ostream &operator << (ostream &, Library &);

public:
    // Setter and Getter functions for the copies count
    void set_copies(int copy) {
        copies = copy;
    }
    int get_copies() {
        return copies;
    }

    // Redefinition of Pure Virtual function show()
    void show() {
        cout << "\t" << bk_nam << " || " << auth_1stnam << " " << sur_nam << " || " <<
copies << endl;
    }
};

```

```cpp
// Friend function ostream
ostream &operator << (ostream &output, Library &temp) {
    output << "\t" << temp.bk_nam << " || " << temp.auth_1stnam << " " << temp.sur_nam << " || " << temp.copies << endl;

    return output;
}

// Derived Class Reserve from Abstract Base Class Book
class Reserve : public Book {
private:
    int res_by;
    friend ostream &operator << (ostream &, Reserve &);

public:
    // Setter and Getter functions for the Registreation numbers
    void set_resby(int reserve) {
        res_by = reserve;
    }
    int get_resby() {
        return res_by;
    }

    // Redifinition of Pure Virtual function show()
    void show() {
        cout << "\t" << res_by << " || " << bk_nam << " || " << auth_1stnam << " " << sur_nam << endl;
    }
};

// Friend function ostream
ostream &operator << (ostream &output, Reserve &temp) {
    output << "\t" << temp.res_by << " || " << temp.bk_nam << " || " << temp.auth_1stnam << " " << temp.sur_nam << endl;

    return output;
}

// Separate Class Student
class Student {
private:
    // Using Library and Reserve Objects
    Library* lib;
    Reserve* res;
    int reg_no;

public:
    // Constructor
    Student(Library* a, Reserve* b, int c) : lib{a}, res{b}, reg_no{c} {}

    //  Getter functions to get Library and Reserve
    Library* get_Library() {
        return lib;
    }

    Reserve* get_Reserve() {
        return res;
    }

    // Function for the student to reserve a book
    void reserve() {
```

```
                string name;
                bool tag = false, flag = true;

                cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n";

                // Display the Reserved books by the entered Registration number
                for(int i = 0; i < size_res; i++) {
                    if(reg_no == res[i].get_resby()) {
                        cout << res[i];
                    }
                }

                cout << "Enter the Book name for the book you wish to reserve or the Author's surname: ";
                cin >> name;

                // This for-loop works and compares the given data
                for(int i = 0; i < size_lib; i++) {
                    // If the book is found then this if statement works
                    if(name == lib[i].get_bknam() || name == lib[i].get_surnam()) {
                        // For-loop to check if a book of this name has been already reserved by the entered Reg #
                        for(int j = 0; j < size_res; j++) {
                            if(reg_no == res[j].get_resby() && (name == res[j].get_bknam() || name == res[j].get_surnam())) {
                                tag = false;
                                flag = false;
                                break;
                            }
                        }
                        // If the book was already registered by the entered Reg #
                        if(flag == false) {
                            break;
                        }
                        tag = true;
                        // If the book has no available copies
                        if(lib[i].get_copies() == 0) {
                            flag == false;
                        }
                        break;
                    }
                }
                string book_name, auth1st_name, auth_surname;

                // If the book is available in the library and has available copies
                if(tag == true && flag == true) {
                    for(int i = 0; i < size_lib; i++) {
                        // Storing the book having the entered name in separate data members and decreasing the number of copies by one
                        if(name == lib[i].get_bknam() || name == lib[i].get_surnam()) {
                            lib[i].set_copies(lib[i].get_copies() - 1);
                            book_name = lib[i].get_bknam();
                            auth1st_name = lib[i].get_auth1stnam();
                            auth_surname = lib[i].get_surnam();
                            break;
                        }
                    }
                    int regs[size_res];
                    string booknames[size_res], auth1stnames[size_res], authsurnames[size_res];
```

```cpp

                // Storing the existing data memebers in separate arrays
                for(int i = 0; i < size_res; i++) {
                    regs[i] = res[i].get_resby();
                    booknames[i] = res[i].get_bknam();
                    auth1stnames[i] = res[i].get_auth1stnam();
                    authsurnames[i] = res[i].get_surnam();
                }
                delete[] res;

                res = new Reserve[size_res + 1];

                // Storing the data back in the newly created array of one size greater
                for(int i = 0; i < size_res; i++) {
                    res[i].set_resby(regs[i]);
                    res[i].set_bknam(booknames[i]);
                    res[i].set_auth1stnam(auth1stnames[i]);
                    res[i].set_surnam(authsurnames[i]);
                }

                res[size_res].set_resby(reg_no);
                res[size_res].set_bknam(book_name);
                res[size_res].set_auth1stnam(auth1st_name);
                res[size_res].set_surnam(auth_surname);

                size_res++;

                cout << "The book has been reserved on your Registration Number.\n";
            }
            // If the book has already been reserved by the entered Reg #
            else if(tag == false && flag == false) {
                cerr << "You already have that book reserved.\n";
            }
            // If all the available copies have already been reserved
            else if(tag == true && flag == false) {
                cerr << "Unfortunately, all the available copies of that book have already been reserved.\n";
            }
            // If there is no book having that name in the library
            else if(tag == false && flag == true) {
                cerr << "There are no books of that name or Author's name in the library.\n";
            }
        }

        // Function for the student to return a book
        void return_bk() {
            string name;
            bool tag = false;

            cout << " *=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*\n";

            // Display the Reserved books by the entered Registration number
            for(int i = 0; i < size_res; i++) {
                if(reg_no == res[i].get_resby()) {
                    cout << res[i];
                }
            }

```

```
                cout << "Enter the Book name for the book you want to return or the Author's surname: ";
                cin >> name;

                // Check if the book has been reserved by the entered Reg # or not
                for(int i = 0; i < size_res; i++) {
                    if(reg_no == res[i].get_resby() && (name == res[i].get_bknam() || name == res[i].get_surnam())) {
                        tag = true;
                        break;
                    }
                }
                // Book Reserved by the entered Reg #
                if(tag == true) {
                    for(int i = 0; i < size_lib; i++) {
                        if(name == lib[i].get_bknam() || name == lib[i].get_surnam()) {
                            lib[i].set_copies(lib[i].get_copies() + 1);
                            break;
                        }
                    }

                    int j = 0;

                    for(int i = 0; i < size_res; i++) {
                        if(name == res[i].get_bknam() || name == res[i].get_surnam()) {
                            continue;
                        }
                        else {
                            res[j] = res[i];
                            j++;
                        }
                    }
                    size_res--;

                    cout << "The book has been returned.\n";
                }
                // Book not Reserved by the entered Reg #
                else if(tag == false) {
                    cerr << "You have no reserved books of that name or Author's name.\n";
                }
            }
        };

        // Function to Initialize the library array from the Library.csv
        Library* initialize_lib() {
            fstream fin;

            Library* a = new Library[size_lib];

            fin.open("Library.csv", ios::in);

            string line, word;
            vector<string> row;

            for(int i = 0; i < size_lib; i++)
            {
                row.clear();

                getline(fin, line);

            stringstream s(line);

            while (getline(s, word, ','))
            {
                row.push_back(word);
            }
            a[i].set_bknam(row[0]);
            a[i].set_auth1stnam(row[1]);
            a[i].set_surnam(row[2]);
            a[i].set_copies(stoi(row[3]));
        }
        fin.close();

        return a;
    }

    // Function to Initialize the Reserve array from the Reserved.csv
    Reserve* initialize_res() {
        fstream fin;

        Reserve* b = new Reserve[size_res];

        fin.open("Reserved.csv", ios::in);

        string line, word;
        vector<string> row;

        for(int i = 0; i < size_res; i++)
        {
            row.clear();

            getline(fin, line);

            stringstream s(line);

            while (getline(s, word, ','))
            {
                row.push_back(word);
            }
            b[i].set_resby(stoi(row[0]));
            b[i].set_bknam(row[1]);
            b[i].set_auth1stnam(row[2]);
            b[i].set_surnam(row[3]);
        }
        fin.close();

        return b;
    }

    // Get the library array size from Library.csv file
    void get_lib_size() {
        fstream fin;

        fin.open("Library.csv", ios::in);

        string line;

        while(fin) {
            getline(fin,line);

            size_lib++;
```

```cpp
    }

    fin.close();
}

// Get the reserve array size from Reserve.csv file
void get_res_size() {
    fstream fin;

    fin.open("Reserved.csv", ios::in);

    string line;

    while(fin) {
        getline(fin,line);

        size_res++;
    }

    fin.close();
}

#endif
```

CONCUSION:

In Conclusion this project was challenging and included almost everything that we have studied until now in CS 112 including inheritance, polymorphism, exception handling, abstraction etc.

Therefore, this task was done slowly but steadily. Thank You!