**LIBRARY MANAGEMENT SYSTEM**

**-SANDHYA VAIDYANATHAN**

**THIS DOCUMENT INCLUDES, CODE, SAMPLE INPUT AND OUTPUT SCREENSHOTS AND PRESENTATION SLIDES.**

**MAIN**

```
// LibraryManagementSystem.cpp : Defines the entry point for the console application.


#include "stdafx.h"
#include "Book.h"
#include"ReadFile.h"
#include "LibraryMember.h"
#include"Librarian.h"

using namespace std;

int main()
{
	string ID;
	vector<Book> books;
	vector<Journal> journals;
	vector<LibraryMember*> members;
	bool isValidUser = false;
	cout << "" << setw(50) << "LIBRARY MANAGEMENT SYSTEM " << setw(30) << "" << '\n';
	ReadFile readFile;
	readFile.ReadBooks(books);
	readFile.ReadJournals(journals);
	readFile.ReadMembers(members);
	Book book;
	LibraryMember *librarymember;
	MemberOfStaff *memberOfStaff;
	Librarian *librarian;

	while (!isValidUser)
	{
		cout << "Enter MemberID:    ";
		cin >> ID;
		for (int i = 0; i < members.size(); i++)
		{
			librarymember = dynamic_cast <LibraryMember*> (members[i]);
			if (librarymember->getmemberId() == ID)
			{
				isValidUser = true;
				if (librarymember->getmemberType() == "Student")
				{
					librarymember = dynamic_cast <LibraryMember*>
(members[i]);
					break;
				}
				else if (librarymember->getmemberType() == "Staff")
				{
```

```
                                        memberOfStaff = dynamic_cast <MemberOfStaff*>
(members[i]);
                                        break;
                                }
                                else if (librarymember->getmemberType() == "Librarian")
                                {
                                        librarian = dynamic_cast <Librarian*> (members[i]);
                                        break;
                                }
                        }
                }
                if (!isValidUser)
                        cout << "The user does not exist in our records. Please try again"
<< endl;
        }

        if (isValidUser)
        {
                if (librarymember->getmemberType() == "Student")
                {
                        librarymember->Operations(books);
                }

                if (librarymember->getmemberType() == "Staff")
                {
                        memberOfStaff->Operations(journals, books);
                }

                if (librarymember->getmemberType() == "Librarian")
                {
                        librarian->Operations(journals, books);
                }
                readFile.WriteBooks(books);
                readFile.WriteJournals(journals);
                readFile.WriteMembers(members);
        }

        cin.get();
        return 0;
        }
```

---

STDAFX

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once
#include "targetver.h"
#include<fstream>
#include <tchar.h>
#include<iostream>
#include<vector>
#include<string>
#include<sstream>
#include<iomanip>
```

```
using namespace std;


// TODO: reference additional headers your program requires here

// stdafx.cpp : source file that includes just the standard includes
// LibraryManagementSystem.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STDAFX.H
// and not in this file
```

---

**BOOK CLASS**

Book.h

```
#pragma once
class Book
{
protected:
	string BookId;
	string Title;
	int NoOfAvailableCopies;
	int MaxAvailableCopies = 3;
public:
	Book();
	Book(string , string, int);
	~Book();
	void BorrowBook();
	void ReturnBook();
	void ShowBook();
	string getTitle();
	string getBookId();
	int getAvailableCopies();
	Book* findBook(vector<Book>& bookCollection);
	void ListAllBooks(vector<Book>& bookCollection);
};
```

Book.cpp

```
#include "stdafx.h"
#include "Book.h"

Book::Book()
{
}

Book::Book(string bookId, string title, int noOfCopies)
{
```

```
	BookId = bookId;
	Title = title;
	NoOfAvailableCopies = noOfCopies;
}


Book::~Book()
{
}

void Book::BorrowBook() {
	if (NoOfAvailableCopies > 0)
	{
		NoOfAvailableCopies--;
		cout << "Borrow Successful." <<endl;
	}

	else
		cout << "The book is not available to borrow" << endl;
}

void Book::ReturnBook() {
	NoOfAvailableCopies++;
}

void Book::ShowBook() {
	cout <<endl << BookId << "\t" << Title << "\t" << NoOfAvailableCopies<<endl;
}

string Book::getTitle()
{
	return Title;
}

string Book::getBookId()
{
	return BookId;
}

int Book::getAvailableCopies()
{
	return NoOfAvailableCopies;
}

Book* Book::findBook(vector<Book> &bookCollection)
{
	string bookinfo;
	Book* tempBook = NULL;
	cout <<"\n" << "Enter Book Title or Book ID:"<<endl;
	cin >> bookinfo;
	for (int i = 0; i < bookCollection.size(); i++)
	{
		if (bookCollection[i].getTitle() == bookinfo ||
bookCollection[i].getBookId() == bookinfo)
		{
			tempBook = &bookCollection[i];
			break;
		}
```

```
	}
	return tempBook;
}

void Book::ListAllBooks(vector<Book> &bookCollection)
{
	cout << endl << "BookId" << "\t" << "Title" << "\t" << "Available copies" << endl;
	for (int i = 0; i < bookCollection.size(); i++)
		{
			bookCollection[i].ShowBook();
		}
}
```

---

**JOURNAL CLASS**

Journal.cpp

```
#pragma once
class Journal
{
private:
	string IssueId;
	string JournalName;
	string IssueName;
	int NoOfAvailableJournals = 5;
	int NoOfAvailableIssues = 4;
	int MaxNoOfIssues = 4;
	int CopyOfIssue = 1;
public:
	Journal();
	Journal(string, string, string, int);
	~Journal();
	void BorrowJournal();
	void ReturnJournal();
	void ShowJournal();
	string getJournalName();
	string getIssueName();
	string getIssueId();
	int getCopyofIssue();
	Journal* findJournal(vector<Journal>& journalCollection);
	void ListAllIssues(vector<Journal>& journalCollection);
};
```

Journal.cpp

```
#include "stdafx.h"
#include "Journal.h"


Journal::Journal()
{
}
```

```
Journal::~Journal()
{
}

Journal::Journal(string journalName, string issueName, string issueId, int copyOfIssue)
{
	IssueId = issueId;
	JournalName= journalName;
	IssueName= issueName;
	CopyOfIssue= copyOfIssue;

}

void Journal::BorrowJournal()
{
	if (CopyOfIssue > 0)
	{
		CopyOfIssue--;
		cout << "Borrow Successful";
	}
	else
		cout << "Issue is not available";
}

void Journal::ReturnJournal()
{
	CopyOfIssue++;
}

void Journal::ShowJournal()
{
	cout << endl << JournalName << "\t\t\t" << IssueName << "\t\t\t" << IssueId <<
"\t\t\t" << CopyOfIssue << endl;
}

string Journal::getJournalName()
{
	return JournalName;
}

string Journal::getIssueName()
{
	return IssueName;
}

string Journal::getIssueId()
{
	return IssueId;
}

int Journal::getCopyofIssue()
{
	return CopyOfIssue;
}

Journal * Journal::findJournal(vector<Journal>& journalCollection)
{
	string journalinfo;
```

```
	Journal* tempJournal = NULL;
	cout << "\n" << "Enter Issue Name or Issue ID";
	cin >> journalinfo;
	for (int i = 0; i < journalCollection.size(); i++)
	{
		if (journalCollection[i].getIssueName() == journalinfo ||
journalCollection[i].getIssueId() == journalinfo)
		{
			tempJournal = &journalCollection[i];
			break;
		}
	}
	return tempJournal;
}

void Journal::ListAllIssues(vector<Journal>& journalCollection)
{
	cout << endl << "JournalName" << "/t/t/t" << "Issue Name" << "/t/t" << "Issue Id"
<< "/t/t" << "Available Copies";
	for (int i = 0; i < journalCollection.size(); i++)
	{
		journalCollection[i].ShowJournal();
	}
}
```

---

**LIBRARY MEMBER CLASS**

LibraryMember.h

```
#include "Book.h"
#pragma once
class LibraryMember
{
protected:
	string MemberName;
	string MemberId;
	string MemberType;
	int NoOfBooksOnLoan;
	int MaxNoOfBooks=6;
	int NoOfJournalsOnLoan = 0;
	vector<Book> bookCollection;

public:
	LibraryMember();
	LibraryMember(vector<Book>& bookCollection);
	~LibraryMember();
	LibraryMember(string, string, string, int, int);
	virtual bool OktoBorrow();
	string getmemberId();
	string getmemberName();
	string getmemberType();
	int getnoofBooksonloan();
	int getnoOfJournalsonloan();
	void ShowRecord();
	void ReturnBookCase(vector<Book>& bookCollection);
	void BorrowBookCase(vector<Book>& bookCollection);
	void PrintOperations();
```

```
	void Operations(vector<Book> &bookCollection);
};
```

Librarymember.cpp

```
#include "stdafx.h"
#include "LibraryMember.h"


LibraryMember::LibraryMember()
{
}

LibraryMember::LibraryMember(vector<Book> &bookCollection)
{
	this->bookCollection = bookCollection;
}

LibraryMember::~LibraryMember()
{
}

LibraryMember::LibraryMember(string memberId, string memberName , string memberType, int
noOfBooksonloan, int noofJournalsonloan)
{
	MemberId = memberId;
	MemberName = memberName;
	MemberType = memberType;
	NoOfBooksOnLoan = noOfBooksonloan;
	NoOfJournalsOnLoan = noofJournalsonloan;
}

bool LibraryMember::OktoBorrow()
{

	if (NoOfBooksOnLoan < MaxNoOfBooks)
		return true;
	else
		return false;
}

string LibraryMember::getmemberId()
{
	return MemberId;
}

string LibraryMember::getmemberName()
{
	return MemberName;
}

string LibraryMember::getmemberType()
{
	return MemberType;
}

int LibraryMember::getnoofBooksonloan()
{
```

```
	return NoOfBooksOnLoan;
}

int LibraryMember::getnoOfJournalsonloan()
{
	return NoOfJournalsOnLoan;
}

void LibraryMember::ShowRecord()
{
	cout << MemberId <<"\t" << MemberName << "\t" << MemberType << "\t" <<
NoOfBooksOnLoan << "\t" << NoOfJournalsOnLoan <<endl;
}

void LibraryMember::ReturnBookCase(vector<Book>& bookCollection)
{
	Book book;
	Book* tempbook = book.findBook(bookCollection);
	if (tempbook == NULL)
	{
		cout << "The mentioned book is not in the library records. Please contact
the librarian if you need any assisstance.";
	}
	else
	{
		tempbook->ReturnBook();
		NoOfBooksOnLoan--;
		cout << "Copy of Book Returned";
	}
}

void LibraryMember::BorrowBookCase(vector<Book>& bookCollection)
{
	Book book;
	Book* tempbook = book.findBook(bookCollection);
	if (tempbook == NULL)
	{
		cout << "Book Not Found";
	}
	else if (tempbook->getAvailableCopies() > 0)
	{
		tempbook->BorrowBook();
		NoOfBooksOnLoan++;

	}
	else
		cout << "No Available copies to Borrow";
}

void LibraryMember::PrintOperations()
{
	cout << "1.	Borrow Book" << endl;
	cout << "2.	Return Book" << endl;
	cout << "3.	List All Books" << endl;
	cout << "4.	Exit" << endl;
}

void LibraryMember::Operations(vector<Book>& bookCollection)
```

```
{
	int operationChoice = 0;
	cout << endl << "Welcome " + MemberName << endl << endl;
	while (operationChoice != 4)
	{
		cout << endl;
		PrintOperations();
		cout << endl << "Please enter your choice : ";
		cout << endl << endl;
		cin >> operationChoice;
		switch (operationChoice)
		{
		case 1:
			{
				if (OktoBorrow())
					BorrowBookCase(bookCollection);
				else
					cout << "Sorry!Copy of Book cannot be borrowed as the
limit exceeds ";
				break;
			}
		case 2 :
			{
				ReturnBookCase(bookCollection);
				break;
			}
		case 3:
			{
				Book book;
				book.ListAllBooks(bookCollection);
				break;
			}
		}
	}

}
```

---

**MEMBER OF STAFF CLASS**

MemberOfStaff.h

```
#pragma once
#include "LibraryMember.h"
#include "Journal.h"

class MemberOfStaff :public LibraryMember
{
	int MaxItemsOnLoan = 12;
public:
	MemberOfStaff();
	~MemberOfStaff();
	MemberOfStaff(string memberId, string memberName, string memberType, int
noOfBooksonloan, int noofJournalsonloan);
	bool OktoBorrow();
	void BorrowJournal(vector<Journal>& journalCollection);
	void ReturnJournal(vector<Journal>& journalCollection);
```

```
	void PrintOperations();
	void Operations(vector<Journal>& journalCollection, vector<Book>& bookCollection);
};
```

MemberOfStaff.cpp

```
#include "stdafx.h"
#include "MemberOfStaff.h"


MemberOfStaff::MemberOfStaff()
{
}


MemberOfStaff::~MemberOfStaff()
{
}


MemberOfStaff::MemberOfStaff(string memberId, string memberName, string memberType, int
noOfBooksonloan, int noofJournalsonloan)
{
	MemberId = memberId;
	MemberName = memberName;
	MemberType = memberType;
	NoOfBooksOnLoan = noOfBooksonloan;
	NoOfJournalsOnLoan = noofJournalsonloan;
}

bool MemberOfStaff::OktoBorrow()
{
	if (NoOfBooksOnLoan + NoOfJournalsOnLoan < MaxItemsOnLoan)
		return true;
	else
		return false;
}

void MemberOfStaff::BorrowJournal(vector<Journal> &journalCollection)
{
	Journal journal;
	Journal* tempjournal = journal.findJournal(journalCollection);
	if (tempjournal == NULL)
	{
		cout << "Issue Not Found";
	}
	else if(tempjournal->getCopyofIssue() >0)
	{
			tempjournal->BorrowJournal();
			tempjournal->ShowJournal();
			NoOfJournalsOnLoan++;
			ShowRecord();
		}
	else
	{
		cout << "Issue not available right now";
```

```
	}
}

void MemberOfStaff::ReturnJournal(vector<Journal> &journalCollection)
{
	Journal journal;
	Journal* tempjournal = journal.findJournal(journalCollection);
	if (tempjournal == NULL)
	{
		cout << "The mentioned issue is not in the library records. Please contact
the librarian if you need any assisstance.";
	}
	else
	{
		tempjournal->ReturnJournal();
		NoOfJournalsOnLoan--;
		cout << "Return Success.";
	}
}

void MemberOfStaff::PrintOperations()
{
	cout << "1. BorrowBook" << endl;
	cout << "2. Return Book" << endl;
	cout << "3. List All Books" << endl;
	cout << "4. Borrow Journal" << endl;
	cout << "5. Return Journal" << endl;
	cout << "6. List All Journals" << endl;
	cout << "7. Exit" << endl;
}

void MemberOfStaff::Operations(vector<Journal>& journalCollection,vector<Book>&
bookCollection)
{

	int operationChoice = 0;
	cout << endl << "Welcome " + MemberName;
	while (operationChoice != 7)
	{
		cout << endl;
		PrintOperations();
		cout << endl << "Please enter your choice : ";
		cout << endl << endl;
		cin >> operationChoice;
		switch (operationChoice)
		{
		case 1:
		{

			if(OktoBorrow())
				BorrowBookCase(bookCollection);
			else
				cout<< "Sorry!Book cannot be borrowed as the limit exceeds ";
			break;
		}
		case 2:
		{
```

```
                ReturnBookCase(bookCollection);
                break;
        }
        case 3:
        {
                Book book;
                book.ListAllBooks(bookCollection);
                break;
        }

        case 4:
        {
                if (OktoBorrow())
                        BorrowJournal(journalCollection);
                else
                        cout << "Sorry!Journal cannot be borrowed as the limit exceeds
";
                break;
        }
        case 5:
        {
                ReturnJournal(journalCollection);
                break;
        }
        case 6:
        {
                Journal journal;
                cout << endl << "JournalName" << "\t\t\t\t" << "IssueName" <<
"\t\t\t\t" << "IssueId" << "\t\t\t\t" << "CopyOfIssue" << endl;
                journal.ListAllIssues(journalCollection);
                break;
        }
        }
    }
}
```

---

**LIBRARIAN CLASS**

Librarian.h

```
#pragma once
#include "MemberOfStaff.h"

class Librarian :public MemberOfStaff
{
private:
    int operationChoice;
public:
    Librarian();
    Librarian(string memberId, string memberName, string memberType, int
noOfBooksonloan, int noofJournalsonloan);
    void AddBook(vector<Book>& bookCollection);
    ~Librarian();
    void AddJournal(vector<Journal>& journalCollection);
    void DeleteBook(vector<Book>& bookCollection);
```

```
        void DeleteJournal(vector<Journal>& journalCollection);
        void PrintOperations();
        void Operations(vector<Journal>& journalCollection,vector<Book>& bookCollection);

};
```

Librarian.cpp

```
#include "stdafx.h"
#include "Librarian.h"
#include"Journal.h"


Librarian::Librarian()
{
}


Librarian::~Librarian()
{
}

Librarian::Librarian(string memberId, string memberName, string memberType, int
noOfBooksonloan, int noofJournalsonloan)
{
        MemberId = memberId;
        MemberName = memberName;
        MemberType = memberType;
        NoOfBooksOnLoan = noOfBooksonloan;
        NoOfJournalsOnLoan = noofJournalsonloan;
}

void Librarian::AddBook(vector<Book> &bookCollection)
{

        string bookid, title;
        int copies;
        cout << "Enter details of the book you would like to add" << endl;
        cout << "Book ID:    ";
        cin >> bookid;
        cout << "Book Title: ";
        cin >> title ;
        cout << "Copies:     ";
        cin >> copies;
        ofstream out;
        Book b(bookid, title, copies);
        bookCollection.push_back(b);
        cout << " Book added" << endl;
        b.ShowBook();

}

void Librarian::AddJournal(vector<Journal> &journalCollection)
{
        string jname, iname,iid;
        int copies;
        cout << "Enter details of the journal you would like to add" << endl;
```

```
        cout << "Journal Name:      ";
        cin >> jname;
        cout << "Issue Name: ";
        cin >> iname;
        cout << "Issue ID:   ";
        cin >> iid;
        cout << "Copies:     ";
        cin >> copies;
        Journal j(jname, iname, iid, copies);
        journalCollection.push_back(j);
        cout << " Journal added" << endl;
        j.ShowJournal();

}

void Librarian::DeleteBook(vector<Book> &bookCollection)
{
        string bookid;
        string line;
        cout << "Enter the Book ID to delete" <<endl;
        cin >> bookid;
        bool flag = false;
                for (int i = 0; i < bookCollection.size(); i++)
                {
                        if (bookCollection[i].getBookId() == bookid)
                        {
                                bookCollection[i].ShowBook();
                                swap(bookCollection[i], bookCollection.back());
                                bookCollection.pop_back();
                                cout << " Book Deleted";
                                flag = true;
                        }
                }
                if (flag == false)
                        cout << "Book Not Found. Please enter a valid ID";
        }

void Librarian::DeleteJournal(vector<Journal> &journalCollection)
{
        string issueid;
        cout << "Enter the Issue ID to delete";
        cin >> issueid;
        bool flag = false;
        for (int i = 0; i < journalCollection.size(); i++)
        {
                if (journalCollection[i].getIssueId() == issueid)
                {
                        journalCollection[i].ShowJournal();
                        swap(journalCollection[i], journalCollection.back());
                        journalCollection.pop_back();
                        cout << " Journal Deleted";
                        flag = true;
                }
        }
        if (flag == false)
                cout << "Failed! Please enter a valid ID";

}
```

```
void Librarian::PrintOperations()
{
	cout << "Choose an Operation: (1-11)" << endl;
	cout << "1. BorrowBook" << endl;
	cout << "2. Return Book" << endl;
	cout << "3. List All Books" << endl;
	cout << "4. Borrow Journal" << endl;
	cout << "5. Return Journal" << endl;
	cout << "6. List All Journals" << endl;
	cout << "7. Add Book" << endl;
	cout << "8. Delete Book" << endl;
	cout << "9. Add Journal" << endl;
	cout << "10. Delete Journal" << endl;
	cout << "11. Exit" << endl;
}

void Librarian::Operations(vector<Journal> &journalCollection, vector<Book>
&bookCollection)
{
	cout << endl << "Welcome " + MemberName;
	while (operationChoice != 11)
	{
		cout << endl;
		PrintOperations();
		cout << endl << "Please enter your choice : ";
		cout << endl << endl;
		cin >> operationChoice;
		switch (operationChoice)
		{
		case 1:
		{

			if (OktoBorrow())
				BorrowBookCase(bookCollection);
			else
				cout << "Sorry!Book cannot be borrowed as the limit exceeds ";
			break;

		}
		case 2:
		{
			ReturnBookCase(bookCollection);
			break;
		}
		case 3:
		{
			Book book;
			book.ListAllBooks(bookCollection);
			break;
		}

		case 4:
		{
			if (OktoBorrow())
				BorrowJournal(journalCollection);
			else
				cout << "Sorry!Book cannot be borrowed as the limit exceeds ";
```

```
                        break;
                }
                case 5:
                {
                        ReturnJournal(journalCollection);
                        break;
                }
                case 6:
                {
                        Journal journal;
                        journal.ListAllIssues(journalCollection);
                        break;
                }
                case 7:
                        AddBook(bookCollection);
                        break;
                case 8:
                        DeleteBook(bookCollection);
                        break;
                case 9:
                        AddJournal(journalCollection);
                        break;
                case 10:
                        DeleteJournal(journalCollection);
                        break;
                }
        }
}
```

---

**READFILE**

ReadFile.h

```
#pragma once
#include "Book.h"
#include "Journal.h"
#include "LibraryMember.h"

class ReadFile
{
public:
        ReadFile();
        ~ReadFile();
        void ReadBooks(vector<Book>& bookCollection);
        void ReadJournals(vector<Journal>& journalCollection);
        void WriteBooks(vector<Book>& bookCollection);
        void ReadMembers(vector<LibraryMember*>& Memberslist);
        void WriteJournals(vector<Journal>& journalCollection);
        void WriteMembers(vector<LibraryMember*>& members);
};

#include "stdafx.h"
#include "ReadFile.h"
#include"Book.h"
#include"Journal.h"
```

```
#include"LibraryMember.h"
#include"MemberOfStaff.h"
#include"Librarian.h"


ReadFile::ReadFile()
{
}


ReadFile::~ReadFile()
{
}

void ReadFile::ReadBooks(vector<Book>& bookCollection)
{
	string temp1, temp2;
	int    temp3;
	string input;
	ifstream fin("books.txt");

	while (true)
	{
		getline(fin, input);
		if (!fin) break; //check for eof
		istringstream buffer(input);
		buffer >> temp1 >> temp2 >> temp3;
		Book book(temp1, temp2, temp3);
		bookCollection.push_back(book);
	}

}

void ReadFile::ReadJournals(vector<Journal>& journalCollection)
{
	string jname, iname, iid;
	int copy;
	string inputs;
	ifstream fin("journals.txt");
	while (true)
	{
		getline(fin, inputs);
		if (!fin) break; //check for eof
		istringstream buffer(inputs);
		buffer >> jname >> iname >> iid >> copy;
		Journal journal(jname, iname, iid, copy);
		journalCollection.push_back(journal);
	}
}
void ReadFile::WriteBooks(vector<Book> &bookCollection)
{
	ofstream out("output.txt");
	for (int i = 0; i <bookCollection.size(); i++)
	{
		out << bookCollection[i].getBookId() << "\t" <<
bookCollection[i].getTitle() << "\t" << bookCollection[i].getAvailableCopies() << endl;
	}
	out.close();
```

```
	remove("books.txt");
	rename("output.txt", "books.txt");
}

void ReadFile::WriteJournals(vector<Journal> &journalCollection)
{
	ofstream out("output.txt");
		for (int i = 0; i <journalCollection.size(); i++)
		{
			out << journalCollection[i].getJournalName() << "\t" <<
journalCollection[i].getIssueName() << "\t" << journalCollection[i].getIssueId() <<"\t"<<
journalCollection[i].getCopyofIssue()<< endl;
		}
	out.close();
	remove("journals.txt");
	rename("output.txt", "journals.txt");
}

void ReadFile::WriteMembers(vector<LibraryMember*> &members)
{
	ofstream out("outputMembers.txt");
	for (int i = 0; i <members.size(); i++)
	{
		out << members[i]->getmemberId() << "\t" << members[i]->getmemberName() <<
"\t" << members[i]->getmemberType() << "\t" << members[i]->getnoofBooksonloan() << "\t"
<< members[i]->getnoOfJournalsonloan()<<endl;
	}
	out.close();
	remove("members.txt");
	rename("outputMembers.txt", "members.txt");

}

void ReadFile::ReadMembers(vector<LibraryMember*>& Memberslist)
{
	string mid, name, mtype;
	int books, journals;
	string input;
	ifstream fin("members.txt");
	while (true)
	{
		getline(fin, input);
		if (!fin) break; // check for eof
		istringstream buffer(input);
		buffer >> mid >> name >> mtype >> books >> journals;
		LibraryMember *member;
		if (mtype == "Staff") {
			MemberOfStaff* staff = new MemberOfStaff(mid, name, mtype, books,
journals);
			Memberslist.push_back(staff);

		}
		else if (mtype == "Student") {
			LibraryMember *student= new LibraryMember(mid, name, mtype, books,
journals);
			Memberslist.push_back(student);
		}
		else if (mtype == "Librarian") {
```

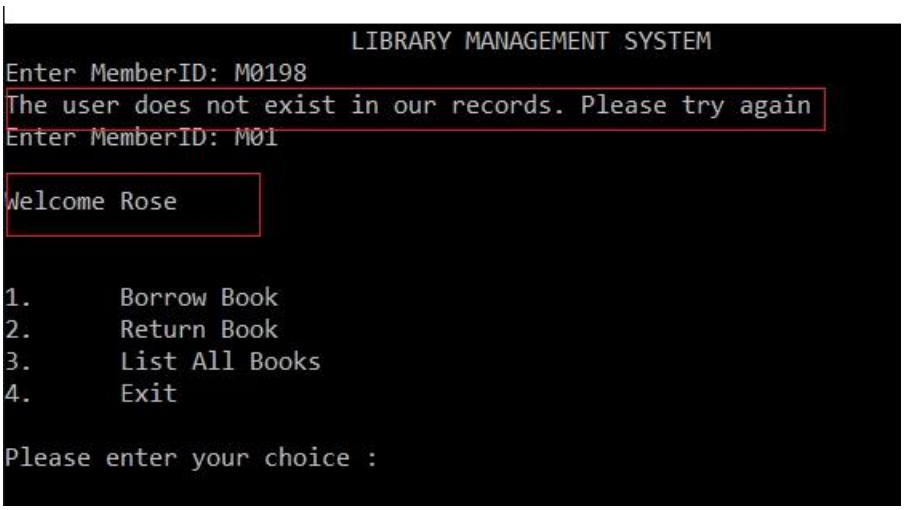
```
                    LibraryMember *librarian = new Librarian(mid, name, mtype, books,
journals);
                    Memberslist.push_back(librarian);
                }
        }
}
```

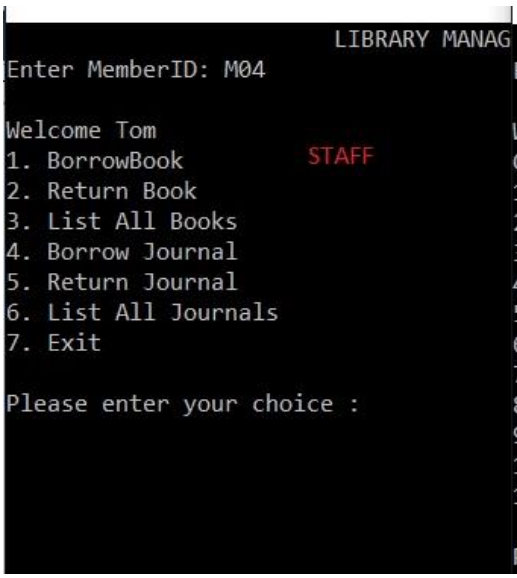
---

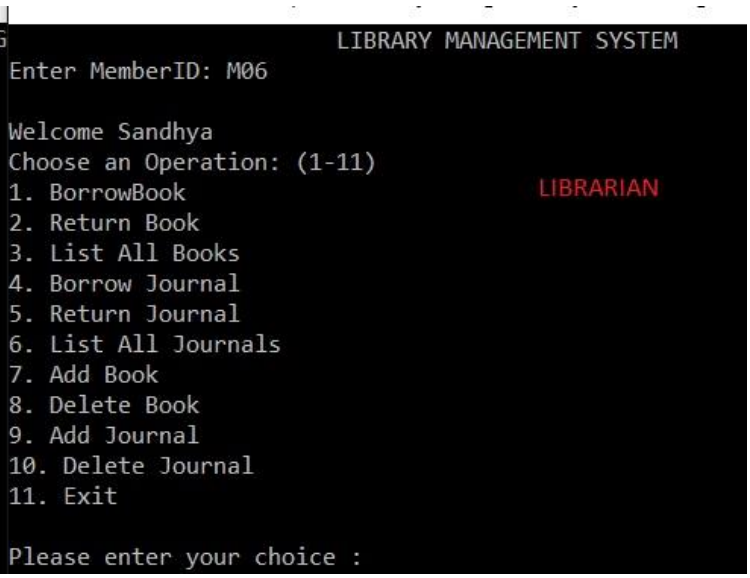
**SAMPLE INPUT/OUTPUT**

LOGIN:

Fails when an invalid ID is entered.
Enters the system when a valid ID is given as an input.
Welcomes the user and displays functions according to the User type.



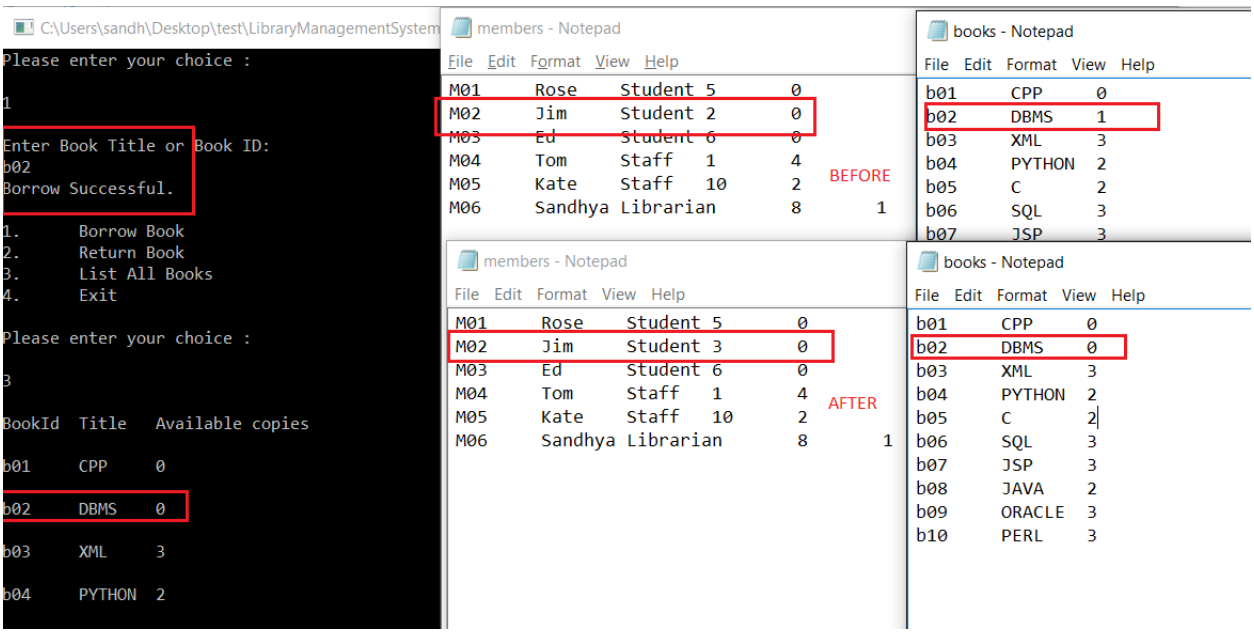




---

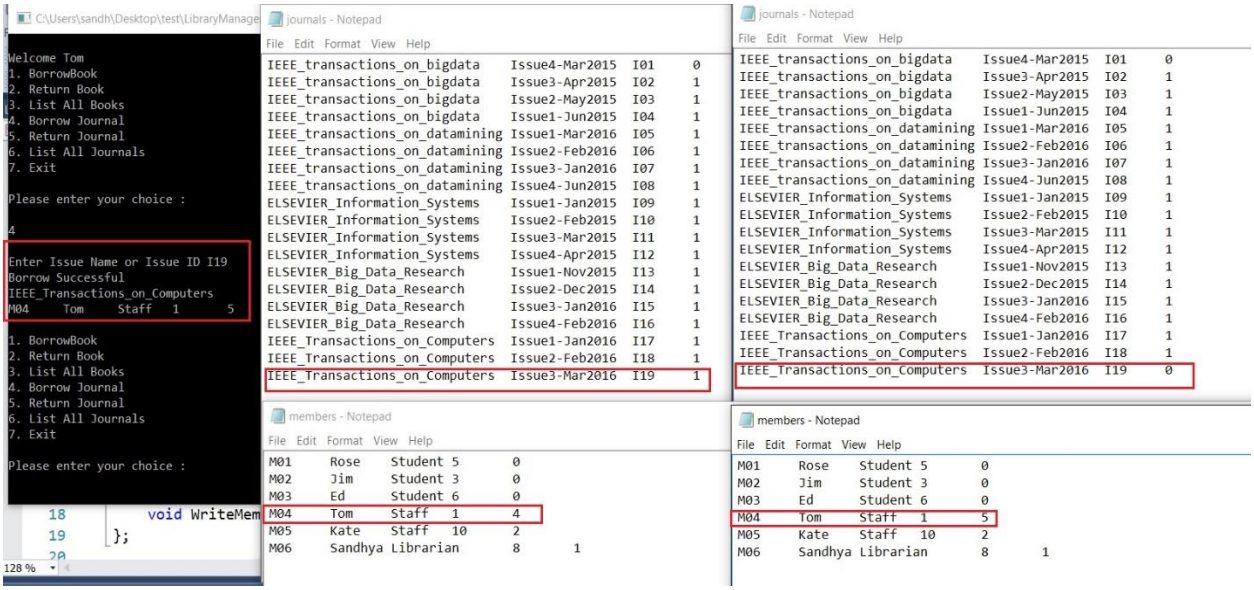
**The data is first read into the memory from three files books.txt, journals.txt and members.txt, and when the session terminates the new values are updated in the corresponding text files.**

Books.txt has 3 fields – book id, book name, no. Of copies Available
Journals.txt has 4 fields – journal name, issue name, issue id, copy of issues
Members.txt has 5 fields – member id, member name, member type, no. of books on loan, no. of journals on loan

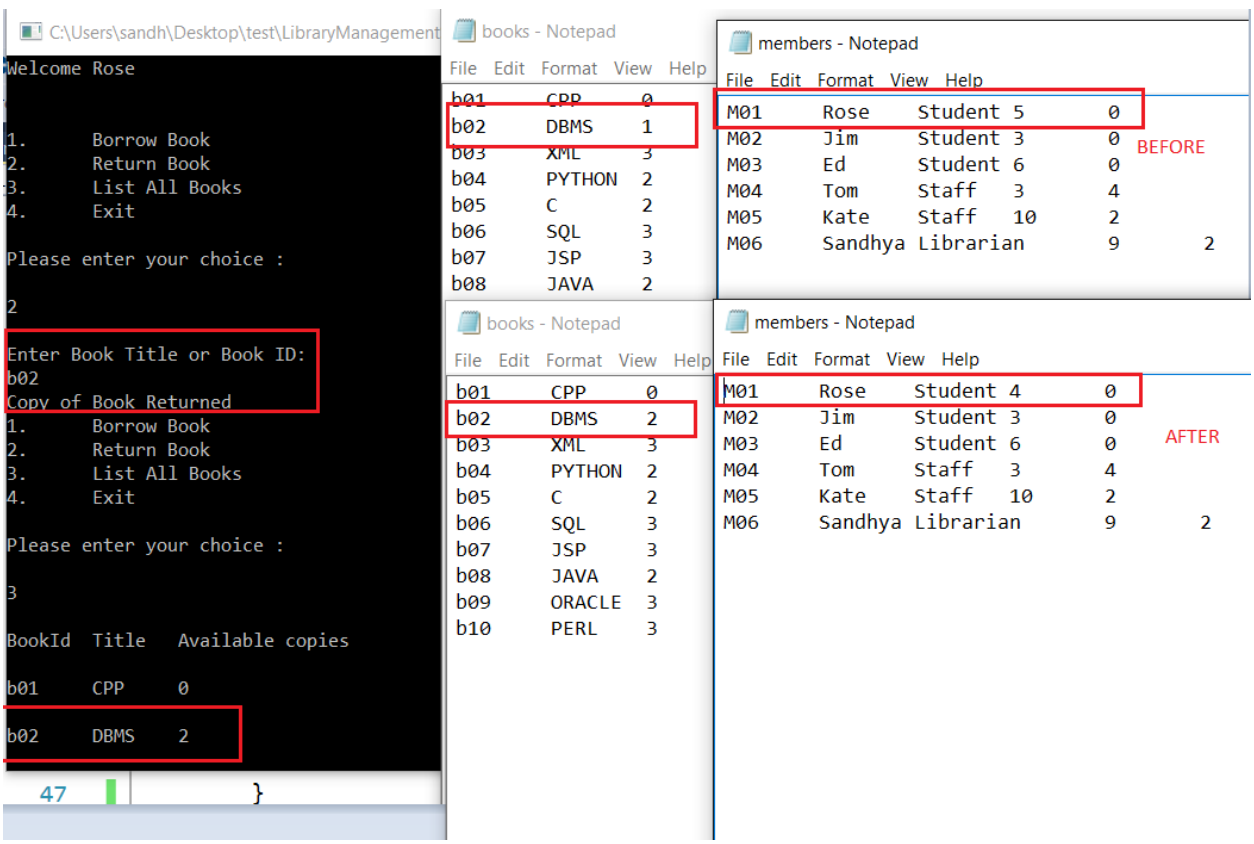
When a book is borrowed,
The book count in books.txt is reduced against the borrowed book.
The noOfbooksonloan field is increased for the person who has borrowed the book in members.txt




Similarly
When a journal is borrowed,
The copy of issue count in journal.txt is reduced against the borrowed copy of issue.
The noOFjournalsonloan field is increased for the person who has borrowed the journal in members.txt

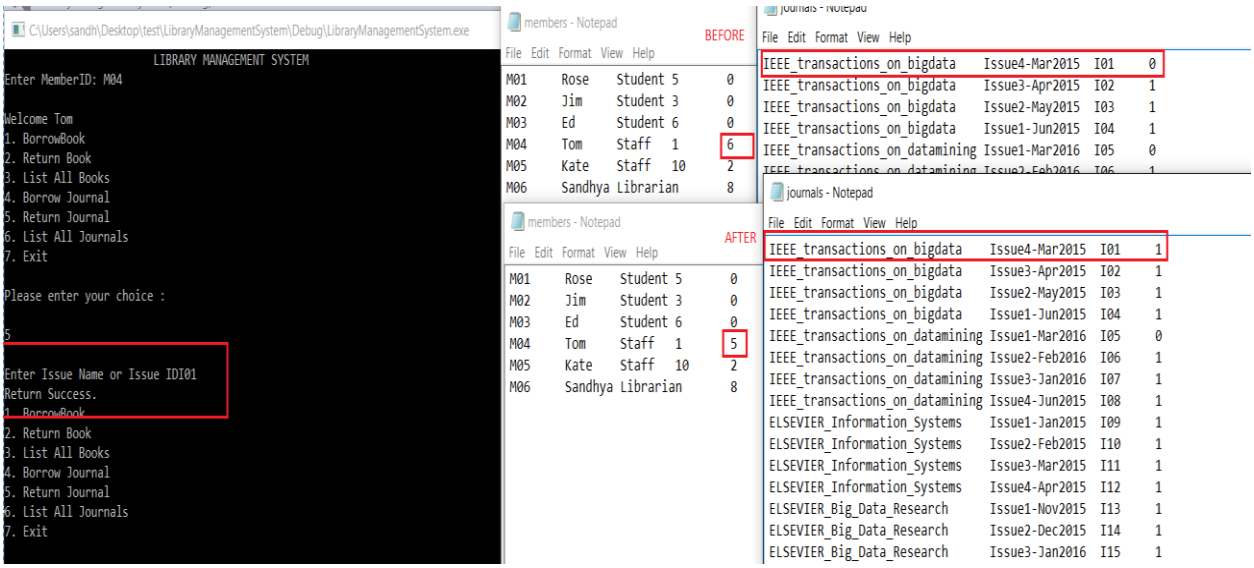



When a copy of book is returned,
The noofcopies in books.txt is increased against the returned book.
The noOfbooksonloan field is reduced for the person who has returned the book in members.txt file



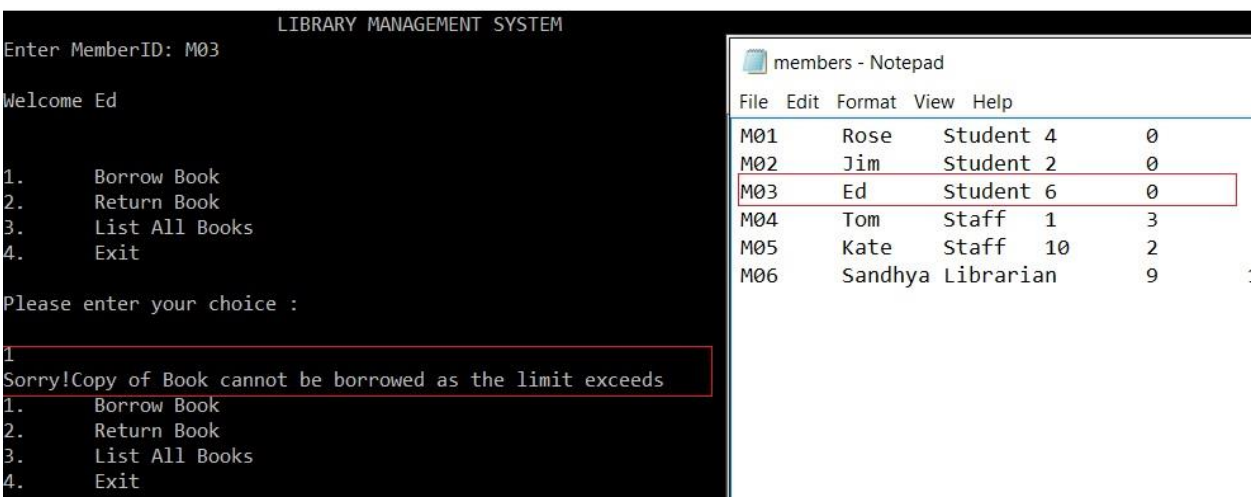

Similarly,
When a journal is returned,

The noofcopies in journals.txt is increased against the returned journal.
The noOfjournalsonloan field is reduced for the person who has returned the journal in members.txt file




Student is allowed to borrow only 6 books.

**Borrow book fails** when it **goes beyond 6 FOR students.**




**Borrow fails when max items borrowed** (books+journals) count is **more than 12 for member of staff or librarian.**

```
C:\Users\sandh\Desktop\test\LibraryManagementSystem\Debug\Library

                    LIBRARY MANAGEMENT SYSTEM
Enter MemberID: M05

Welcome Kate
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :

1
Sorry!Book cannot be borrowed as the limit exceeds
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit
```

members - Notepad

File Edit Format View Help

```
M01     Rose    Student 5       0
M02     Jim     Student 3       0
M03     Ed      Student 6       0
M04     Tom     Staff   1       4
M05     Kate    Staff   10      2
M06     Sandhya Librarian       8       1
```

**Borrowing journal unavailable in the library system**

```
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :

4

Enter Issue Name or Issue ID random
Issue Not Found
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :
```

**Borrowing book unavailable in the library system**

```
                    LIBRARY MANAGEMENT SYSTEM
Enter MemberID: M01

Welcome Rose


1.      Borrow Book
2.      Return Book
3.      List All Books
4.      Exit

Please enter your choice :

1

Enter Book Title or Book ID:
HARRYPOTTER
Book Not Found
1.      Borrow Book
2.      Return Book
3.      List All Books
4.      Exit

Please enter your choice :
```

**Borrow fail when there are no copies (available copies =0)**

```
BookId  Title   Available copies

b01     CPP     0

b02     DBMS    2

b03     XML     3

b04     PYTHON  2

b05     C       2

b06     SQL     3

b07     JSP     3

b08     JAVA    2

b09     ORACLE  3

b10     PERL    3

1.      Borrow Book
2.      Return Book
3.      List All Books
4.      Exit

Please enter your choice :

1

Enter Book Title or Book ID:
b01
The book is not available to borrow
```

**Borrowing issue of a journal when available copies is 0**

C:\Users\sandh\Desktop\test\LibraryManager

```
4

Enter Issue Name or Issue ID random
Issue Not Found
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :

4

Enter Issue Name or Issue IDI19
Issue not available right now
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :
```

journals - Notepad

File Edit Format View Help

```
IEEE_transactions_on_bigdata     Issue4-Mar2015  I01     1
IEEE_transactions_on_bigdata     Issue3-Apr2015  I02     1
IEEE_transactions_on_bigdata     Issue2-May2015  I03     1
IEEE_transactions_on_bigdata     Issue1-Jun2015  I04     1
IEEE_transactions_on_datamining Issue1-Mar2016  I05     0
IEEE_transactions_on_datamining Issue2-Feb2016  I06     1
IEEE_transactions_on_datamining Issue3-Jan2016  I07     1
IEEE_transactions_on_datamining Issue4-Jun2015  I08     1
ELSEVIER_Information_Systems     Issue1-Jan2015  I09     1
ELSEVIER_Information_Systems     Issue2-Feb2015  I10     1
ELSEVIER_Information_Systems     Issue3-Mar2015  I11     1
ELSEVIER_Information_Systems     Issue4-Apr2015  I12     1
ELSEVIER_Big_Data_Research       Issue1-Nov2015  I13     1
ELSEVIER_Big_Data_Research       Issue2-Dec2015  I14     1
ELSEVIER_Big_Data_Research       Issue3-Jan2016  I15     1
ELSEVIER_Big_Data_Research       Issue4-Feb2016  I16     1
IEEE_Transactions_on_Computers   Issue1-Jan2016  I17     1
IEEE_Transactions_on_Computers   Issue2-Feb2016  I18     1
IEEE_Transactions_on_Computers   Issue3-Mar2016  I19     0
```

**Returning journal not available in library records**

```
Please enter your choice :

5

Enter Issue Name or Issue ID RANDOM
The mentioned issue is not in the library records. Please contact the librarian if you need any assisstance.
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :
```

**Returning book not available in library records**

```
3.      List All Books
4.      Exit

Please enter your choice :

1

Enter Book Title or Book ID:
HARRYPOTTER
Book Not Found
1.      Borrow Book
2.      Return Book
3.      List All Books
4.      Exit

Please enter your choice :

2

Enter Book Title or Book ID:
hp
The mentioned book is not in the library records. Please contact the librarian if you need any assisstance.
1.      Borrow Book
2.      Return Book
3.      List All Books
4.      Exit
```

## Add book:

```
C:\Users\sandh\Desktop\test\LibraryManagementSystem\Debug\LibraryManagementSystem.exe
Enter details of the book you would like to add
Book ID:        b20
Book Title:     ECLIPSE
Copies: 2
 Book added

b20     ECLIPSE 2

Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

3

BookId  Title   Available copies

b01     CPP     0

b02     DBMS    1

b03     XML     3

b04     PYTHON  2

b05     C       2

b06     SQL     3

b07     JSP     3

b08     JAVA    2

b09     ORACLE  3

b10     PERL    3

b20     ECLIPSE 2

Choose an Operation: (1-11)
```

## Add journal:

```
C:\Users\sandh\Desktop\test\LibraryManagementSystem\Debug\LibraryManagementSystem.exe

Please enter your choice :

9
Enter details of the journal you would like to add
Journal Name:   ACM
Issue Name:     ISSUE_7
Issue ID:       I50
Copies: 1
 Journal added

ACM                     ISSUE_7                 I50                     1

Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

10
Enter the Issue ID to deleteI50

ACM                     ISSUE_7                 I50                     1
 Journal Deleted
Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :
```

**Deletion of books/journals can also be done by the librarian:**

```
Enter the Book ID to delete
b20

b20     eclipse 3
 Book Deleted
Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

3

BookId  Title   Available copi

b01     CPP     0

b02     DBMS    1

b03     XML     3

b04     PYTHON  2

b05     C       2

b06     SQL     3

b07     JSP     3

b08     JAVA    2

b09     ORACLE  3

b10     PERL    3

Choose an Operation: (1-11)
```

**JOURNAL DELETION:**

```
Please enter your choice :

10
Enter the Issue ID to delete I05

IEEE_transactions_on_datamining                Issue1-Mar2016                I05                    0
 Journal Deleted
Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

6

IEEE_Transactions_on_Computers                 Issue3-Mar2016                I19                    0

IEEE_transactions_on_bigdata                   Issue3-Apr2015                I02                    1

IEEE_transactions_on_bigdata                   Issue2-May2015                I03                    1

IEEE_transactions_on_bigdata                   Issue1-Jun2015                I04                    1

IEEE_Transactions_on_Computers                 Issue2-Feb2016                I18                    1

IEEE_transactions_on_datamining                Issue2-Feb2016                I06                    1
```

```
journals - Notepad
File  Edit  Format  View  Help
IEEE_Transactions_on_Computers  Issue3-Mar2016  I19  0
IEEE_transactions_on_bigdata    Issue3-Apr2015  I02  1
IEEE_transactions_on_bigdata    Issue2-May2015  I03  1
IEEE_transactions_on_bigdata    Issue1-Jun2015  I04  1
IEEE_transactions_on_datamining Issue1-Mar2016  I05  0
IEEE_transactions_on_datamining Issue2-Feb2016  I06  1
IEEE_transactions_on_datamining Issue3-Jan2016  I07  1
IEEE_transactions_on_datamining Issue4-Jun2015  I08  1
ELSEVIER_Information_Systems    Issue1-Jan2015  I09  1
ELSEVIER_Information_Systems    Issue2-Feb2015  I10  1
ELSEVIER_Information_Systems    Issue3-Mar2015  I11  1
ELSEVIER_Information_Systems    Issue4-Apr2015  I12  1
ELSEVIER_Big_Data_Research      Issue1-Nov2015  I13  1
ELSEVIER_Big_Data_Research      Issue2-Dec2015  I14  1
ELSEVIER_Big_Data_Research      Issue3-Jan2016  I15  1
ELSEVIER_Big_Data_Research      Issue4-Feb2016  I16  1
IEEE_Transactions_on_Computers  Issue1-Jan2016  I17  1
IEEE_Transactions_on_Computers  Issue2-Feb2016  I18  1
```

The issue ID I05 is not present anymore

**Deletion fails when the librarian is trying to delete a book that's not present in the system.**

```
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

8
Enter the Book ID to delete
BUIO
Book Not Found. Please enter a valid ID
Choose an Operation: (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :
```

**Issue of a Journal delete with invalid issue ID**

```
C:\Users\sandh\Desktop\test\LibraryManagementSysten
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :

10
Enter the Issue ID to deleteHJKLL
Failed! Please enter a valid ID
Choose an Operation. (1-11)
1. BorrowBook
2. Return Book
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Add Book
8. Delete Book
9. Add Journal
10. Delete Journal
11. Exit

Please enter your choice :
```

**LIST ALL BOOKS:**

```
3. List All Books
4. Borrow Journal
5. Return Journal
6. List All Journals
7. Exit

Please enter your choice :

3

BookId  Title   Available copies

b01     CPP     0

b02     DBMS    0

b03     XML     3

b04     PYTHON  2

b05     C       2

b06     SQL     3

b07     JSP     3

b08     JAVA    2

b09     ORACLE  3

b10     PERL    3

1. BorrowBook
2. Return Book
3. List All Books
```

**LIST ALL JOURNALS:**

```
7. Exit

Please enter your choice :

6

IEEE_Transactions_on_Computers          Issue3-Mar2016          I19          0

IEEE_transactions_on_bigdata            Issue3-Apr2015          I02          1

IEEE_transactions_on_bigdata            Issue2-May2015          I03          1

IEEE_transactions_on_bigdata            Issue1-Jun2015          I04          1

IEEE_Transactions_on_Computers          Issue2-Feb2016          I18          1

IEEE_transactions_on_datamining         Issue2-Feb2016          I06          1

IEEE_transactions_on_datamining         Issue3-Jan2016          I07          1

IEEE_transactions_on_datamining         Issue4-Jun2015          I08          1

ELSEVIER_Information_Systems            Issue1-Jan2015          I09          1

ELSEVIER_Information_Systems            Issue2-Feb2015          I10          1

ELSEVIER_Information_Systems            Issue3-Mar2015          I11          1

ELSEVIER_Information_Systems            Issue4-Apr2015          I12          1

ELSEVIER_Big_Data_Research              Issue1-Nov2015          I13          1

ELSEVIER_Big_Data_Research              Issue2-Dec2015          I14          1

ELSEVIER_Big_Data_Research              Issue3-Jan2016          I15          1

ELSEVIER_Big_Data_Research              Issue4-Feb2016          I16          1

IEEE_Transactions_on_Computers          Issue1-Jan2016          I17          1
```

**SLIDES PRESENTED DURING THE DEMO ARE ATTACHED BELOW.**

**[Removed slides which had sample input/output, as I have already attached them above]**

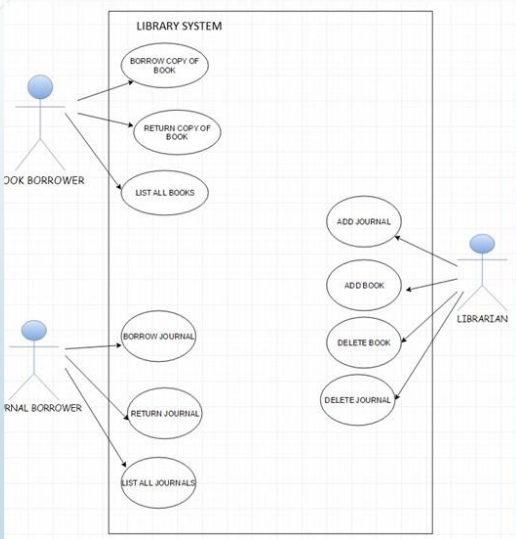
# LIBRARY MANAGEMENT SYSTEM

-SANDHYA VAIDYANATHAN
Under the supervision of Prof. Uday K Chakraborty

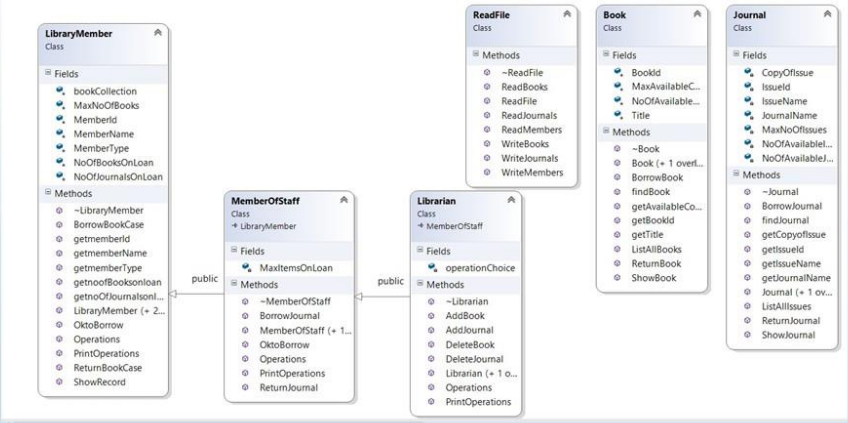
## Agenda

- Use case diagram for Library
- Class Model with Attributes and Operations
- Library Class model
- How the program works – Sample Input / Output
- Demo
- Design Decisions and Implementation
- References

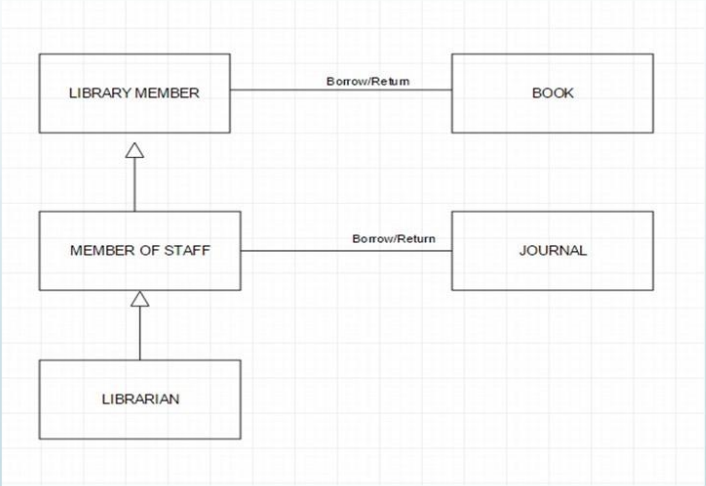
# Use Case Diagram for Library




# Class Model with Attribute and Operations



# Library Class Model




# Design decisions and Implementation

- Language Used : C++
- IDE – Microsoft Visual Studio
- Data loaded in memory for faster execution
- Usage of Text file to store data( 3 text files – books, journals, members)
- Each row in a text file is loaded as vectors
- Multilevel Inheritance
- Dynamic casting – finds object class at run time
- Doing the operations in their respective classes instead of main function makes the code more modular.

# References:


- Using UML SOFTWARE ENGINEERING WITH OBJECTS AND COMPONENTS – Stevens Pooley Second Edition.
- http://www.tutorialspoint.com/
- http://stackoverflow.com/