Topic: Library Management system

Group no: MLB_03.01.05

Campus : Malabe

Submission Date :

We declare that this is our own work and this Assignment does not incorporate without acknowledgment any material previously submitted by anyone else in SLIIT or any other university/Institute. And we declare that each one of us equally contributed to the completion of this Assignment.

Registration No	**Name**	**Contact Number**
IT21229398	A.M.B.S Bandara	0701117153
IT21233876	Silva R.L.G	0765672300
IT21233876	Hasanka G.S	0714370503
IT21228926	G.R.S Perera	0704279949
IT21204548	Senevirathna G.A	0701836888

Requirements of library Managements system
Assignments 02

- As a New Member , He need to first register to the system and become a registered Member in the system. If he is un-registered Member , He cant carry a book.

- As a Registered member, He would be able to view available Book categories in the system.

- As a Unregistered member, He would be able to view Reference Items.

- As a registered Member , He can Borrow Items

- A Member can see details of borrowed books and date the book should be returned.

- A Member can pay Membership Fee , Fines using credit card, Debit card, PayPal for each other.

- The ADMIN can add new book to the database and remove books and manage Members.

- As the Librarian , He should login to the system as Librarian and he can manage payment and update the monthly fee Report.

- As the Admin he should login to system as the admin and he can check feedback and give response to members.

- As a Member , He should login to the system as Registered member and the check borrowed books lists and extend the returned date.

Identified Classes

- registered member
- unregister member
- Borrow Items.
- Books
- Reference Items.
- Payment
- Admin

CRC Cards

Class Name : Registered Members	
Responsibilities	**Collaboration**
View available book categories	Books
See details	
Login to System	

Class name : Unregistered Members	
Responsibilities	**Collaboration**
View Reference Items	Reference Items

Class name: Reference Items	
Responsibilities	**Collaboration**
Store Reference Items	

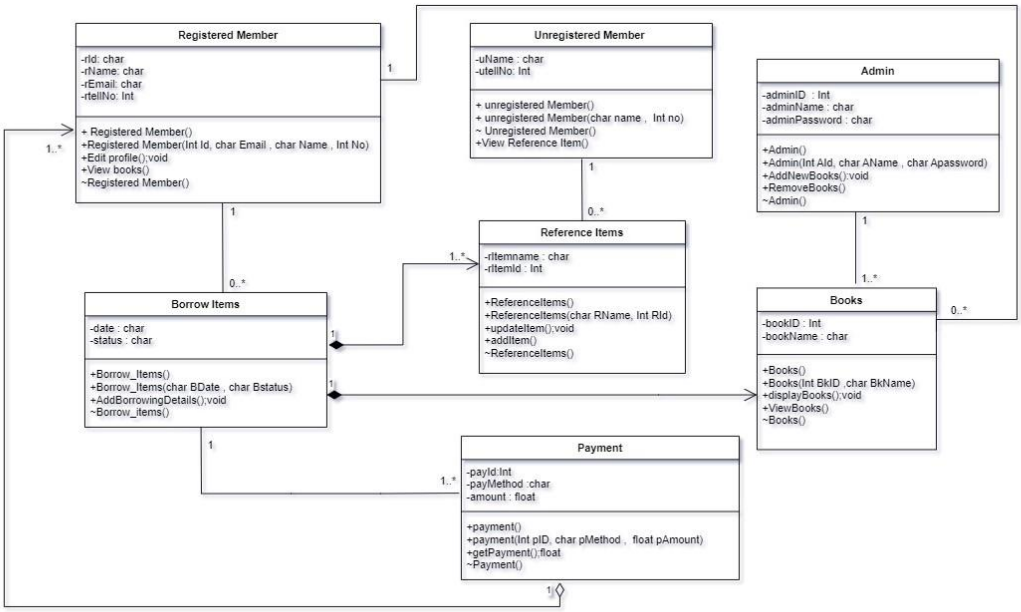
Class name : Borrow Items	
Responsibilities	**Collaboration**
Borrow Books	Reference Items

Class name : Books	
Responsibilities	**Collaboration**
Add Books	Admin
Add Books	Admin

Class name: Payment	
Responsibilities	**Collaboration**
Pay Membership Fee Pay Fine	Registered Member

Class name: Admin	
Responsibilities	**Collaboration**
Manage Members	
Check Feedback	
Give Response	

Class Diagram



Coding

```
//Registerd_Member Aggregation Relationship to payment
#include<iostream>
#include<cstring>

using namespace std;

//Implement Registered Member class

class Registered_Member
{
private:
        char rID[10];
        char rName[20];
        char rEmail[20];
        int rTellNo;
public:
        Registered_Member(){

        };
        EditProfile(){

        };
        ViewBooks(){

        };
        Registered_Member(const char id[],const char email[],const char name[],int no)
        {
```

```
            strcpy(rID,id);
            strcpy(rName,name);
            strcpy(rEmail,email);
            rTellNo = no;
        };

        ~Registered_Member(){
            cout<<" Regidtered Member Destructor called!!"<<endl;
        };
};

class payment
{
        private:
            char payId;
            char payMethod[20];
            float amount;
            Registered_Member *regMember[2];

        public:
            payment(){
                regMember[0] = new
Registered_Member("001","Beesara","beesara@gmail.com",0771234567);
                regMember[1] = new
Registered_Member("002","Sahan","sahan@gmail.com",0711234567);

            };
            payment(int pId,const char pMethod[], float pAmount)
```

```
		{
			payId=pId;
			strcpy(payMethod,pMethod);
			amount=pAmount;
		};
		getPayment(){

		};
		~payment(){
			cout<<" Payment Destructor Called!!"<<endl;
		};
};

int main(void)
{
	payment *pay1 = new payment();

	delete pay1;

	Registered_Member *reg1 = new
Registered_Member("001","Beesara","beesara@gmail.com",0771234567);
	Registered_Member *reg2 = new
Registered_Member("002","Sahan","sahan@gmail.com",0711234567);

	delete reg1;
	delete reg2;

	return 0;
```

```
}

//Registerd_Member Association Relationship to Borrow_Items
#include<iostream>
#include<cstring>

using namespace std;

class Registered_member;
class Borrow_items;

//borrow items class

class Borrow_items
{
	private:
		char date[20];
		char status[50];
		Registered_member *registered_member[2];

	public:
		Borrow_Items()
		{

		};
		Borrow_Items(const char bDate[],const char bStatus[])
```

```
		{
			strcpy(date, bDate);
			strcpy(status,bStatus);
		};
		addBorrowingDetails()
		{

		};

		~Borrow_items()
		{
			cout<<"Boorow Items destructer called!!"<<endl;
		};
};

class Registered_Member
{
	private:
		int rID;
		char rName[20];
		char rEmail[20];
		int rTellNo;
		Borrow_items *borrow_items;
	public:
		Registered_Member()
		{
```

```
        };
        editProfile()
        {

        };
        viewBooks()
        {

        };
        Registered_Member(int id,const char name[],const char email[] ,int
no,Borrow_items *pItrm)
        {
                rID = id;
                strcpy(rName,name);
                strcpy(rEmail,email);
                rTellNo = no;
                borrow_items = pItrm;
        };
        ~Registered_Member()
        {
                cout<<"Registered Member destructer called!!"<<endl;
        };
};

int main(void)
{
        Borrow_items *borrow_items1 = new Borrow_items("01/02/2001","Good");
        Borrow_items *borrow_items2 = new Borrow_items("10/15/2002","To Good");
```

```
        Registered_Member *registered_member1 = new
Registered_Member(001,"Sahan","sahan123@gmail.com",0701117153,borrow_items1);

        Registered_Member *registered_member2 = new
Registered_Member(002,"lahiranga","lahiranga123@gmail.com",0701117154,borrow_items2);


        delete borrow_items1;

        delete borrow_items2;


        delete registered_member1;

        delete registered_member2;

        return 0;




}


//Registerd_Member Aggregation Relationship to Books

#include<iostream>

#include<cstring>


using namespace std;


class Registered_member;

class Books;
```

```
//Books class


class Books
{
	private:
		char bookId[10];
		char bookName[20];
		Registered_member *registered_member[3];


	public:
		Books()
		{


		};
		Books(const char bkId[],const char bkName[])
		{
			strcpy(bookId,bkId);
			strcpy(bookName,bkName);
		};
		displayBooks()
		{


		};
		viewBooks()
		{


		};
```

```
                ~Books()
                {
                        cout<<"Books Destructer called!!"<<endl;
                };

};

class Registered_member
{
        private:
                char rID[10];
                char rName[20];
                char rEmail[30];
           char rTellNo[20];
           Books *books;
        public:
                Registered_Member()
                {

                };
                editProfile()
                {

                };
                viewBooks()
                {
```

```
        };
        Registered_Member(const char id[],const char name[], const char email[] ,const
char no[],Books *pBook)
        {
                strcpy(rID,id);
                strcpy(rName,name);
                strcpy(rEmail,email);
                strcpy(rTellNo,no);
                books = pBook;
    };
    ~Registered_member()
    {
        cout<<"Registered_Member Destructer called!!"<<endl;
        };
};

int main(void)
{
    Books *books1 = new Books("001","Madolduwa");
    Books *books2 = new Books("002","Laila");

    Registered_member *registered_member1 = new
Registered_member("003","vidura","vidura123@gmail.com","071821727",books1);
    Registered_member *registered_member2 = new
Registered_member("004","venura","venura321@gmail.com","071845727",books2);

    return 0;
}
//Books Aggregation Relationship to Borrow_Items
```

```
#include<iostream>
#include<cstring>

using namespace std;

//Implement Books class

class Books
{
	private:
		int bookId;
		char bookName[50];

	public:
		Books(int bkId,const char bkName[])
		{
			bookId=bkId;
			strcpy(bookName,bkName);
		};
		displayBooks()
		{

		};
		viewBooks()
		{

		};
```

```
		~Books()
		{
			cout<<"Books deleted:"<<bookId<<endl;
		};



};

class borrow_Items
{
	private:
		char date[20];
		char status[50];
		Books * books[3];

	public:
		borrow_Items()
		{
			books[0] = new Books(001,"A room without books is like a body without a
soul");
			books[1] = new Books(002,"Take a good book to bed with you—books do
not snore");
			books[2] = new Books(003,"Outside of a dog, a book is a man's best friend");
		};
		borrow_Items(const char bDate[],const char bStatus[])
		{
			strcpy(date,bDate);
			strcpy(status,bStatus);
```

```
        };
        addBorrowingDetails()
        {

        };
        ~borrow_Items()
        {
                cout<<"Delete Books Items"<<endl;
                        for(int x = 0; x < 3; x++){
                        delete books[x];
                }
        }



};

int main(void){

        borrow_Items * books1 = new borrow_Items();

        delete books1;

        return 0;
}

//Reference_Items Aggregation Relationship to Borrow_Items
```

```
#include<iostream>
#include<cstring>

using namespace std;

//Implement Reference Items class

class Reference_Items
{
	private:
		char rItemname[20];
		int rItemId;

	public:
		Reference_Items()
		{

		};
		Reference_Items(char rName[],int rId)
		{
			strcpy(rItemname,rName);
			rItemId=rId;
		};
		updateItem()
		{

		};
```

```
		addItem()
		{

		};
		~Reference_Items()
		{
			cout<<"Reference Items Deleted"<<rItemId<<endl;
		};
};

class borrow_Items
{
	private:
		char date[20];
		char status[50];
		Reference_Items * reference_Items[3];

	public:
		borrow_Items()
		{
			reference_Items[0] = new Reference_Items(" eBooks",001);
			reference_Items[1] = new Reference_Items("journal articles",002);
			reference_Items[2] = new Reference_Items("webpages",003);
		};
		borrow_Items(const char bDate[],const char bStatus[])
		{
			strcpy(date,bDate);
```

```
                strcpy(status,bStatus);
        };
        addBorrowingDetails()
        {

        };
        ~borrow_Items()
        {
                cout<<"Delete Reference Items"<<endl;
                        for(int x = 0; x < 3; x++){
                        delete reference_Items[x];
                }
        }



};

int main(void){

        Reference_Items *rf1 = new Reference_Items();

        delete rf1;

        return 0;
}
//Payment association Relationship to Borrow_Items
```

```
#include <iostream>
#include <cstring>

using namespace std;

class Borrow_items;
class Payment;

class Borrow_items{

    private :
        char date[20];
        char status[50];
        Payment *pay[2];

    public:
        Borrow_items(){

        };
        Borrow_items(const char bDate[],const char bStatus[]){

            strcpy(date,bDate);
            strcpy(status,bStatus);
        };
        void addBorrowingDetails(){

        };
```

```
        ~Borrow_items(){
            cout<<" Destructor called!!"<<endl;
        };
};

class Payment{

    private:
        int payId;
        char payMethod[10];
        float amount;
        Borrow_items *item1;

    public:
        Payment(){

        };
        Payment(int pID,const char pMethod[],float pAmount,Borrow_items *pItem){

            payId = pID;
            strcpy(payMethod,pMethod);
            amount = pAmount;
            item1 = pItem;
        };

        float getPayment(){
```

```
        };
        ~Payment(){
                cout<<" Destructor called!!"<<endl;
        };
};

int main(void)
{
        Borrow_items *item1 = new Borrow_items("2021.10.21","Good");
        Borrow_items *item2 = new Borrow_items("2021.10.30","Too Good");

        Payment *pay1 = new Payment(1001,"paypal",2300.00,item1);
        Payment *pay2 = new Payment(1002,"visa",5300.00,item2);

        delete item1;
        delete item2;

        delete pay1;
        delete pay2;

        return 0;
}

//Unregisterd_member association Relationship to Reference_Items
```

```
#include<iostream>
#include<cstring>

using namespace std;

class Unregistered_member;
class Reference_item;

//Reference_item class

class Reference_Items
{
	private:
		char rItemname[20];
		int rItemId;
		Unregistered_member *Items[3];


	public:
		Reference_Items()
		{

		};
		Reference_Items(const char rName[],int rId)
		{
			strcpy(rItemname,rName);
			rItemId=rId;
```

```
                };
                updateItem()
                {

                };
                addItem()
                {

                };
                ~Reference_Items()
                {
                        cout<<"Reference_Items Destructer called!!"<<endl;
                };
};

class Unregisterd_member
{
        private:
                char uName[20];
                int uTellNo;
                Reference_Items *Member;


        public:
                Unregisterd_member()
                {
```

```
        };

        Unregisterd_member(const char name[20],int no)
        {
                strcpy(uName,name);
                uTellNo=no;
        };

        view_Reference_Item()
        {

        };

        ~Unregisterd_member()
        {
                cout<<"Unregisterd_member Destructer called!!"<<endl;
        };

};

int main(void)
{
        Reference_Items *Items1 = new Reference_Items("Ebook",001);
        Reference_Items *Items2 = new Reference_Items("CD",002);
```

```
        Unregisterd_member *Member1 = new Unregisterd_member("Gimhani", 0701117153);
        Unregisterd_member *Member2 = new Unregisterd_member("shehan", 070117155);

        return 0;

}

//Admin association Relationship to Books
#include<iostream>
#include<cstring>

using namespace std;

class Books;
class Admin;

//Books class

#include<string>
#include<iostream>
#include<cstring>

using namespace std;

//Implement Books class

class Books
```

```
{
        private:
                int bookId;
                char bookName[50];
                Admin *admin[3];

        public:
                Books()
                {

                };
                Books(int bkId,char bkName[])
                {
                        bookId=bkId;
                        strcpy(bookName,bkName);
                };
                displayBooks()
                {

                };
                viewBooks()
                {

                };
                ~Books()
                {
                        cout<<"Books destructer called!"<<endl;
```

```
                };

};

class Admin
{
        private:
                int adminId;
                char adminName[20];
                char adminPassword[30];
                Books *books;

        public:
                Admin()
                {

                };
                Admin(int aId,char aName[],char aPassword[])
                {
                        adminId=aId;
                        strcpy(adminName,aName);
                        strcpy(adminPassword,aPassword);
                };
                addNewBooks()
                {

                };
```

```
                removeBooks()
                {

                };
                ~Admin()
                {
                        cout<<"ADmin Destructer Called!!"<<endl;
                };


};

int main(void)
{
        Books *book1 = new Books(001,"One Hundred Years of Solitude");
        Books *book2 = new Books(002,"One Thoused Years of Solitude");

        Admin *ad1 = new Admin(003,"sahan","sahan123");
        Admin *ad2 = new Admin(004,"bandara","bandara123");

        delete book1;
        delete book2;

        delete ad1;
        delete ad2;
        return 0;       }
```

Individual Contribution

Name	Student-ID	Identified classes	Coded part
A.M.B.S Bandara	**IT21229398**	registered member Payment	• Registered Member Aggregation Relationship to payment. • Admin association Relationship to Books.
Silva R.L.G	**IT21232954**	Borrow Items Admin	• Unregistered member association Relationship to Reference Items. • Books Aggregation Relationship to Borrow Items.
Hasanka G.S	**IT21233876**	Books	• Registered Member Association Relationship to Borrow Items. • Registered Member Aggregation Relationship to Books.
G.R.S Perera	**IT21228926**	unregister member	Payment association Relationship to Borrow Items.
Senevirathna .G.A	**IT21204548**	Reference Items	Reference Items Aggregation Relationship to Borrow Items.