

TABLE OF CONTENTS

CERTIFICATE	i
ABSTRACT	iii
ACKNOWLEDGMENT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vi
ABBREVIATIONS	1
1. INTRODUCTION	2
1.1 Background	
1.2 Introduction to the title of the project	
2. E R DIAGRAM	7
3. RELATIONAL SCHEMA DIAGRAM	8
4. SYSTEM DESIGN	9
4.1 Tables	
5. IMPLEMENTATION	11
5.1 Software's Used	
5.2 Snapshots	
5.3 Pseudo Code	
6. APPLICATIONS	18
7. CONCLUSIONS	19
8. FUTURE ENHANCEMENTS	20
REFERENCES	21

LIST OF FIGURES

Fig. No.	Descriptions	Page
Fig. 1.1	Entity Relationship Diagram	03
Fig. 1.2	Schema Diagram	04
Fig 5.1	Login Portal for LMS	10
Fig 5.2	Books Interface to add and delete books	10
Fig 5.3	Card Interface to add and remove Borrowers	11
Fig 5.4	Issue Interface: Book succesfully issued	11
Fig 5.5	Issue Interface: Book succesfully recieved1	12
Fig 5.6	Issue Interface: Recent books with return date	12
Fig 5.7	About Interface	13
Fig 5.8	Max number of books issued	13

ABBREVIATIONS

- RDBMS -Relational Database Management System
- SQL -Structured Query Language
- ER Diagram -Entity Relationship Diagram

Chapter 1

INTRODUCTION

1.1 Background

A relational database management system is a database management system based on the relational model invented by Edgar F. Codd at IBM's San Jose Research Laboratory. Most databases in widespread use today are based on his relational database model.

SQL is widely used in Storing and Retrieving Data. Data is stored in the form of a Data Base. A front end tool , like c# can be used to make an user interface which makes it easy for end users to manage the data.

This reduces the amount of Paper Work required in traditional storage . When compared with file Structures, DBMS makes the searching process easier.

Using suitable front end tools, end users need not write queries to retrieve data. But they can easily access the Data with the help of suitable front end tool. The design of back end and its connections to front end is the job of Data Base Engineers.

SQLite:

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress.

SYNTAX: (Examples)

Create Table

```
CREATE TABLE [dbo].[books] (  
[book_id] INT NOT NULL,  
[title] VARCHAR (15) NOT NULL,  
[author] VARCHAR (15) NULL,  
[pubname] VARCHAR (15) NULL,  
[isbn] VARCHAR (15) NULL,  
[price] INT NULL,  
[borrower] INT NULL,  
PRIMARY KEY CLUSTERED ([book_id] ASC),  
CONSTRAINT [FK_books_ToTable] FOREIGN KEY ([pubname]) REFERENCES  
[dbo].[publication] ([pub_name])  
);
```

Select Procedure

```
CREATE PROCEDURE [dbo].ShowAllBooksData_SP  
AS  
SELECT book_id, title, author, pubname, pub_id , isbn, price  
FROM books b , publication p  
WHERE b.pubname=p.pub_name  
RETURN 0
```

Insert Procedure

```
CREATE PROCEDURE [dbo].BooksAdd_SP  
@book_id int,  
@title varchar(15),  
@author varchar(15),  
@pubname varchar(15),  
@isbn varchar(15),  
@no_of_copies int  
AS  
Insert into books(book_id,title,author,pubname,isbn,price)  
values(@book_id,@title,@author,@pubname,@isbn,@no_of_copies) RETURN 0
```

Delete Procedure

```
CREATE PROCEDURE [dbo].booksDelete
@book_id int
AS
delete from books where book_id=@book_id
RETURN 0
```

C#:

C# is a simple, modern, general-purpose, object-oriented programming language developed by Microsoft within its .NET initiative led by Anders Hejlsberg.

A C# program consists of the following parts –

- Namespace declaration
- A class
- Class methods
- Class attributes
- A Main method
- Statements and Expressions
- Comments

Let us look at a simple code that prints the words "Hello World" –

```
using System;
namespace HelloWorldApplication {
class HelloWorld {
static void Main(string[] args) {
/* my first program in C# */
Console.WriteLine("Hello World");
Console.ReadKey();
}
}}
```

Let us look at the various parts of the given program –

- The first line of the program **using System;** - the **using** keyword is used to include the **System** namespace in the program. A program generally has multiple **using** statements.
- The next line has the **namespace** declaration. A **namespace** is a collection of classes. The *HelloWorldApplication* namespace contains the class *HelloWorld*.
- The next line has a **class** declaration, the class *HelloWorld* contains the data and method definitions that your program uses. Classes generally contain multiple methods. Methods define the behavior of the class. However, the *HelloWorld* class has only one method **Main**.
- The next line defines the **Main** method, which is the **entry point** for all C# programs. The **Main** method states what the class does when executed. The next line */*...*/* is ignored by the compiler and it is put to **addcomments** in the program. The **Main** method specifies its behavior with the statement **Console.WriteLine("Hello World");** *WriteLine* is a method of the *Console* class defined in the *System* namespace. This statement causes the message "Hello, World!" to be displayed on the screen. The last line **Console.ReadKey();** is for the VS.NET Users. This makes the program wait for a key press and it prevents the screen from running and closing quickly when the program is launched from Visual Studio .NET.

1.2 Introduction to Library Management System

The project titled Library Management System is a Library Management software for monitoring and controlling the transactions in a library .The project “Library Management System” is developed using Visual Studio, which mainly focuses on basic operations in a library like adding new books, and updating new information, searching books and members and return books.

This project on “LIBRARY MANAGEMENT” gives us the complete information about the library. We can enter the record of new books and retrieve the details of books available in the library. We can add and remove the Card holders and also issue books. Throughout the project the focus has been on presenting information and contents in an easy and intelligible manner. The project is very useful for Librarians to manage the books in a Library

Chapter 2

E-R Diagram

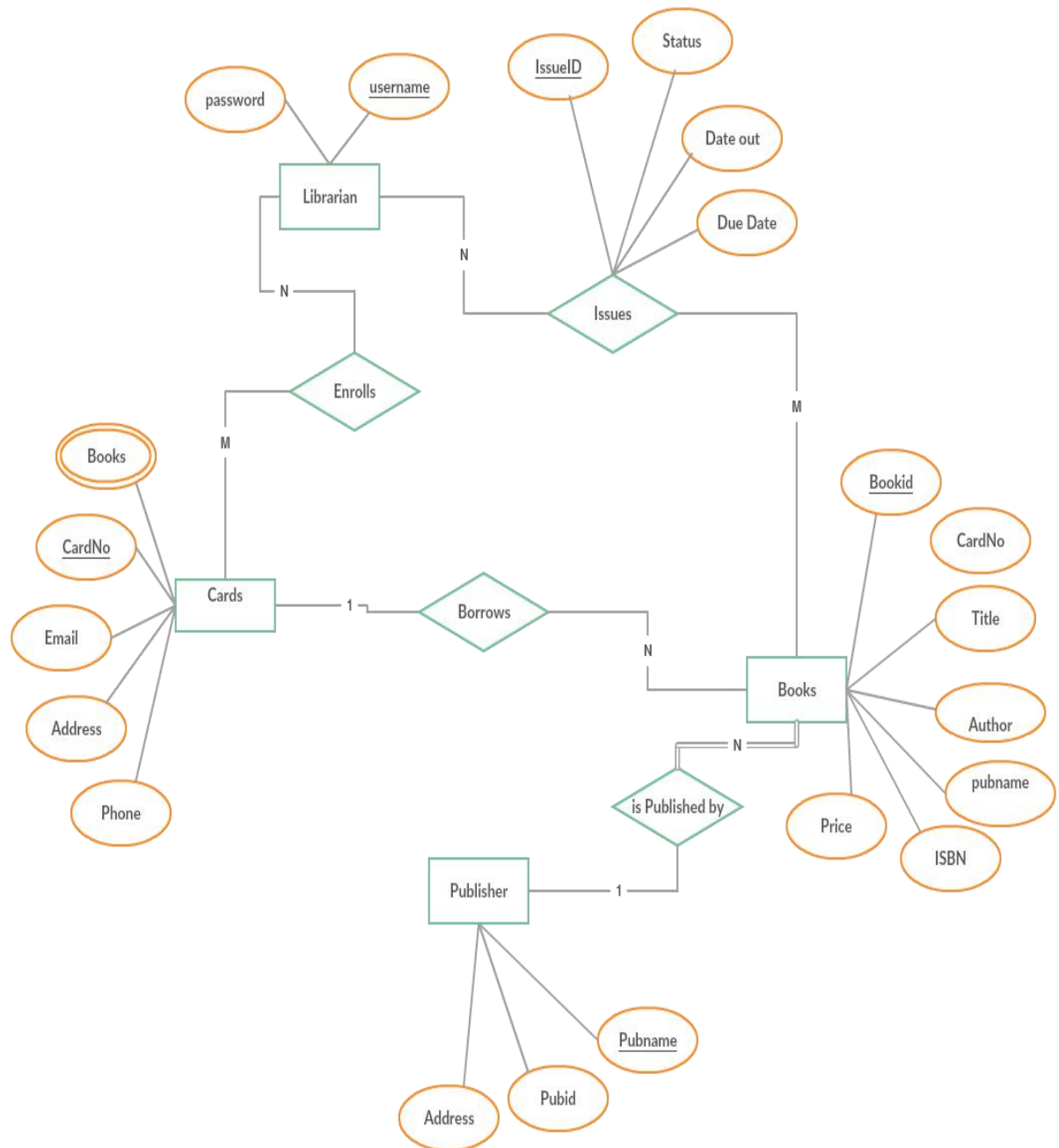


Fig 1.1 Entity Relationship Diagram

Chapter 3

Relational Schema Diagram

A Schema is a pictorial representation of the relationship between the database tables in the database that is created. The database schema of a database system is its structure described in a formal language supported by the database management system. The Schema used in Library Management System is as shown below :

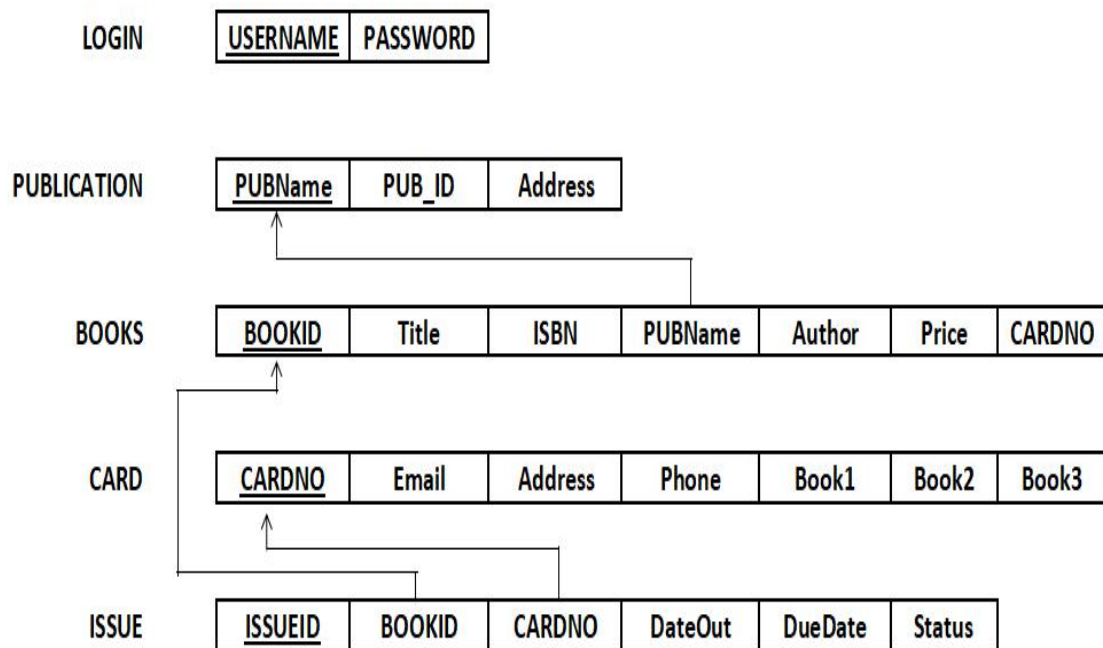



Fig 1.2 Schema Diagram

Chapter 4


System Tables

Login Table

	Name	Data Type	Allow Nulls	Default
	USERNAME	varchar(20)	<input type="checkbox"/>	
	PASSWORD	varchar(20)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	


Primary Key: USERNAME

Books Table

	Name	Data Type	Allow Nulls	Default
	book_id	int	<input type="checkbox"/>	
	title	varchar(15)	<input type="checkbox"/>	
	author	varchar(15)	<input checked="" type="checkbox"/>	
	pubname	varchar(15)	<input checked="" type="checkbox"/>	
	isbn	varchar(15)	<input checked="" type="checkbox"/>	
	price	int	<input checked="" type="checkbox"/>	
	borrower	int	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	


Primary Key: book_id

Cardno Table

	Name	Data Type	Allow Nulls	Default
	cid	int	<input type="checkbox"/>	
	email	varchar(25)	<input checked="" type="checkbox"/>	
	phone	char(10)	<input checked="" type="checkbox"/>	
	address	varchar(15)	<input checked="" type="checkbox"/>	
	book1	int	<input checked="" type="checkbox"/>	
	book2	int	<input checked="" type="checkbox"/>	
	book3	int	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	


Primary Key: cid

Issue Table

	Name	Data Type	Allow Nulls	Default
	issueid	int	<input type="checkbox"/>	
	cid	int	<input type="checkbox"/>	
	bookid	int	<input type="checkbox"/>	
	dateofissue	date	<input checked="" type="checkbox"/>	
	dateofreturn	date	<input checked="" type="checkbox"/>	
	status	varchar(15)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Primary Key: issueid

Publication Table

	Name	Data Type	Allow Nulls	Default
	pub_id	int	<input type="checkbox"/>	
	pub_name	varchar(15)	<input type="checkbox"/>	
	address	varchar(15)	<input checked="" type="checkbox"/>	
			<input type="checkbox"/>	

Primary Key: pub_name

Chapter 5

Implementation

5.1. Software and Hardware Requirements

Software Requirements:

Operating System -- Windows 2010.

Language --C#

Technologies --Form Based Application

Database -- SQLite

Software's -- Visual Studio 2017

Hardware Requirements:

Hardware components-- KeyBoard , Mouse , Monitor

Memory -- 20 GB Disk Storage

RAM -- 2 GB min

5.2. Snapshots of Project

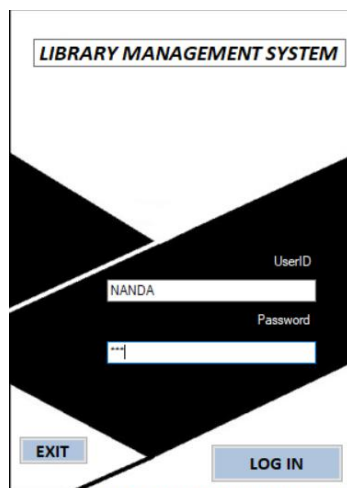


Fig 5.1 Login Portal for LMS

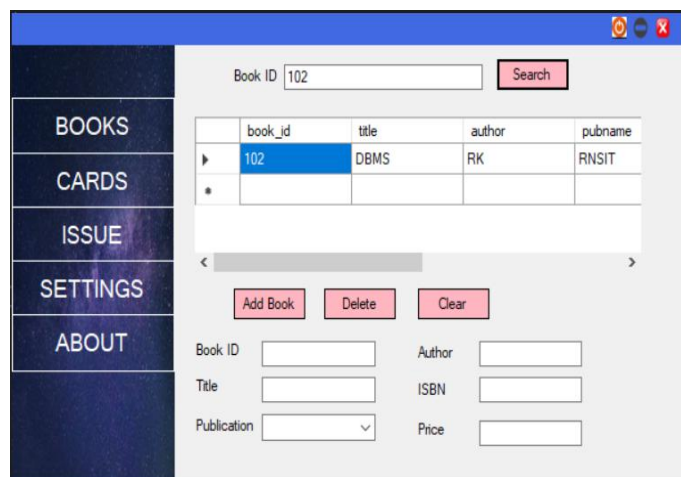


Fig 5.2 Books Interface to add and delete books

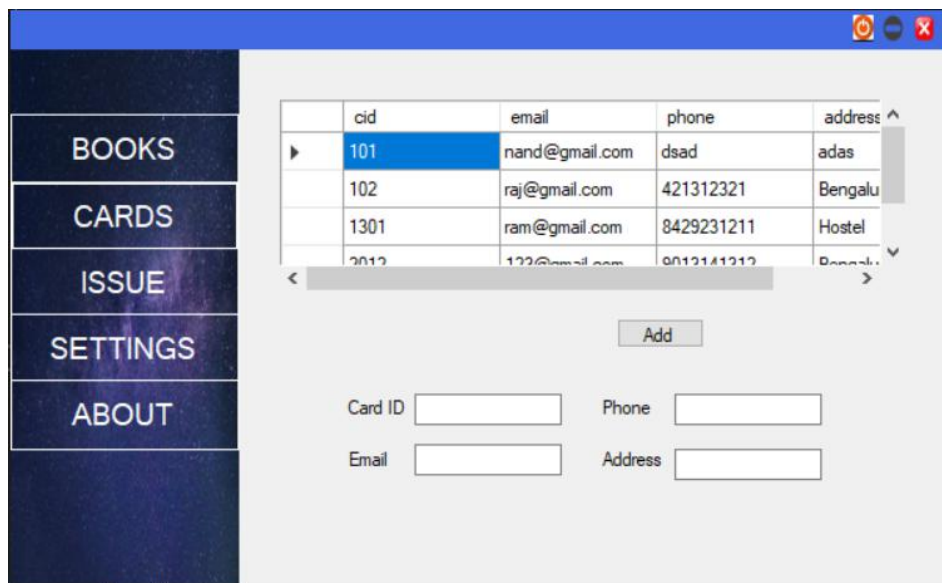


Fig 5.3 Card Interface to add and remove Borrowers

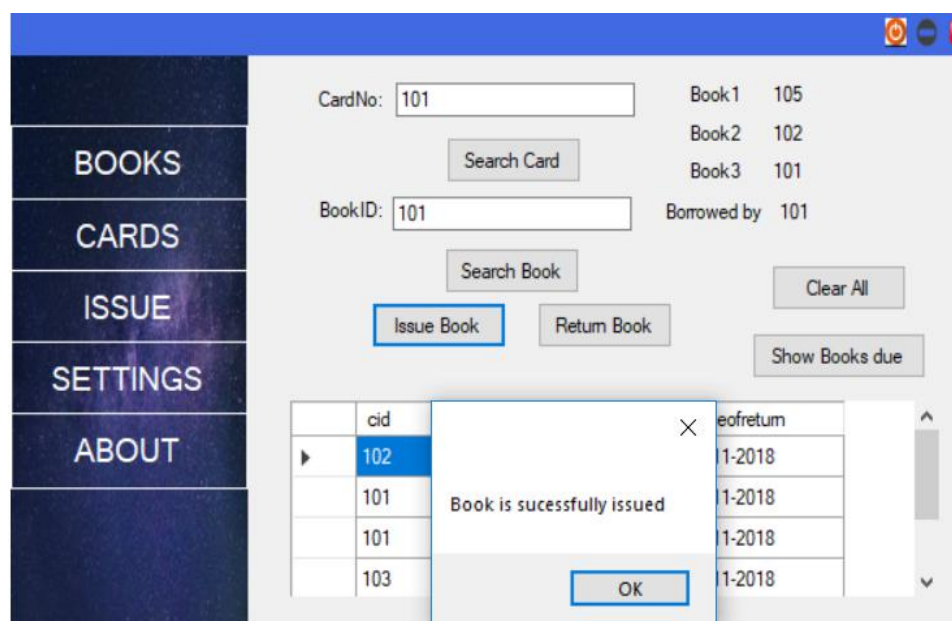


Fig 5.4 Issue Interface: Book succesfully issued

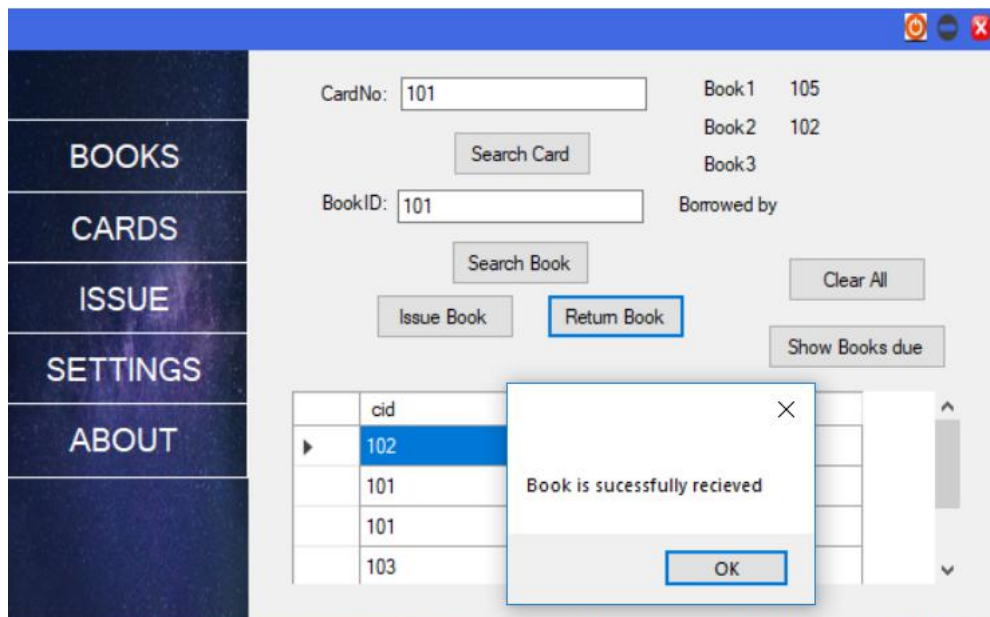


Fig 5.5 Issue Interface: Book sucessfully recieved

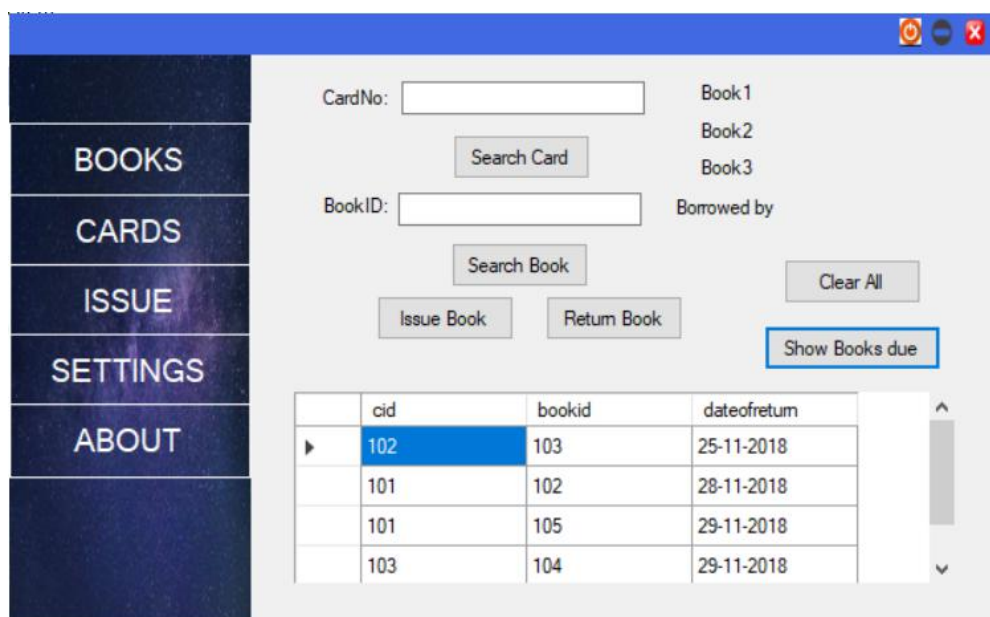


Fig 5.6 Issue Interface: Recent books with return date



Fig 5.7 About Interface

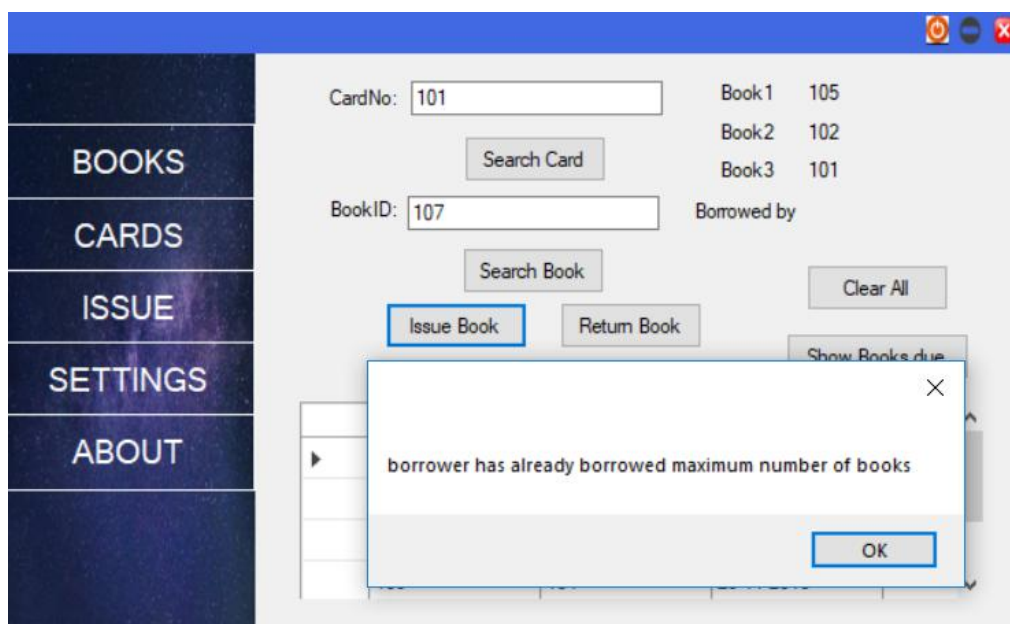


Fig 5.8 Max number of books issued

5.3 Sample Pseudo Code For Library Management System

The Sample code snippets for books user interface is as shown below

When BooksUserControl is Loaded onto the screen
refresh the DataGridView

Add new Book

```
{
SqlCommand cmd = new SqlCommand("BooksAdd_SP", con);
cmd.CommandType = CommandType.StoredProcedure;
cmd.Parameters.AddWithValue("@book_id", Bookid_textBox.Text);
cmd.Parameters.AddWithValue("@title", Title_textBox.Text);
cmd.Parameters.AddWithValue("@author", Author_textBox.Text);
cmd.Parameters.AddWithValue("@pubname", Pubname_comboBox.Text);
cmd.Parameters.AddWithValue("@isbn", ISBN_textBox.Text);
cmd.Parameters.AddWithValue("@no_of_copies", Copies_textBox.Text);
con.Open();
try
{
    cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    MessageBox.Show("Invalid SQL OPERATION:" + ex);
}
con.Close();
refresh_DataGridView();
}
```

refresh data grid view

```
{
try
{
    SqlCommand cmd = new SqlCommand("ShowAllBooksData_SP", con);
    cmd.CommandType = CommandType.StoredProcedure;
```



```

SqlDataAdapter DA = new SqlDataAdapter(cmd);
DataSet DS = new DataSet();
DA.Fill(DS);
con.Open();
try
{
    cmd.ExecuteNonQuery();
}
catch(Exception ex)
{
    MessageBox.Show("Invalid SQL OPERATION:" + ex);
}
con.Close();
dataGridView1.DataSource = DS.Tables[0];
}
catch(Exception e)
{
    MessageBox.Show("" + e);
}
}
Delete books data
{
    try
    {
        SqlCommand cmd = new SqlCommand("booksDelete", con);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@book_id", Bookid_textBox.Text);
        con.Open();
        try
        {
            cmd.ExecuteNonQuery();
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show("Invalid Sql Operation: " + ex);

        }
        con.Close();
        refresh_DataGridView();
    }
    catch(Exception ex)
    {
        MessageBox.Show("" + ex);
    }
}

```

Search books

```

{
    try
    {
        SqlCommand cmd = new SqlCommand("booksSearch_SP", con);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@book_id", searchTextBox.Text);
        SqlDataAdapter DA = new SqlDataAdapter(cmd);
        DataSet DS = new DataSet();
        DA.Fill(DS);
        con.Open();
        try
        {
            cmd.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Invalid SQL OPERATION:" + ex);
        }
        con.Close();
    }
}

```

```

        dataGridView1.DataSource = DS.Tables[0];
    }
    catch (Exception ex)
    {
        MessageBox.Show("" + ex);
    }
}

```

Clear all text fields

```

{
    searchTextBox.Text = "";
    Bookid_textBox.Text = "";
    Title_textBox.Text = "";
    Pubname_comboBox.Text = "";
    ISBN_textBox.Text = "";
    Author_textBox.Text = "";
    Copies_textBox.Text = "";
}

```

CREATE PROCEDURE [dbo].BooksAdd_SP

```

    @book_id int,
    @title varchar(15),
    @author varchar(15),
    @pubname varchar(15),
    @isbn varchar(15),
    @no_of_copies int

```

AS

```

    Insert into books (book_id,title,author,pubname,isbn,price)
    values(@book_id,@title,@author,@pubname,@isbn,@no_of_copies)

```

RETURN 0

```
CREATE PROCEDURE [dbo].booksDelete
```

```
    @book_id int
```

```
AS
```

```
    delete from books where book_id=@book_id
```

```
RETURN 0
```

```
CREATE PROCEDURE [dbo].booksSearch_SP
```

```
    @book_id int
```

```
AS
```

```
    SELECT book_id, title, author, pubname, pub_id , isbn, price
```

```
    FROM books b , publication p
```

```
    WHERE b.pubname=p.pub_name and b.book_id=@book_id
```

```
RETURN 0
```

```
CREATE PROCEDURE [dbo].ShowAllBooksData_SP
```

```
AS
```

```
    SELECT book_id, title, author, pubname, pub_id , isbn, price
```

```
    FROM books b , publication p
```

```
    WHERE b.pubname=p.pub_name
```

```
RETURN 0
```

All other user interfaces have been designed based on the same idea

Chapter 6

Conclusion

This application, Library Management System will replace the system of maintaining books in the Library. Searching of books, issuing it and maintaining records of Card holders in the library will be made easier using this project.

The main features of this project are providing user-friendly interface for the Librarian and it also avoids carrying out the work manually and reduces the wastage of paper as everything is done electronically .

This software can be used in Library for managing books, card holders , issuing and returning of books. It is provided with a secure LOGIN- only Librarians with valid user name and password can access and modify the data.

Chapter 7

Future Enhancements

This is a stand alone application. This can be made into a web based application where only a Librarian can login and access the data. A seperate interface can be made for card holders to Log into this user interface, Search the books, their availibity and Reserve Books.

This project is a simple demonstration of how library mangement System can be implemented for small libraries across the world. More Advanced Designs can be made depending on the number of Card holders, books , fine management and so on.

The Future Enhancements of this project is limitless.

References

[1] www.youtube.com

www.google.com

www.stackoverflow.com

www.quora.com

www.wikipeda.org