Introductory Project Document for Library Management System

Introduction

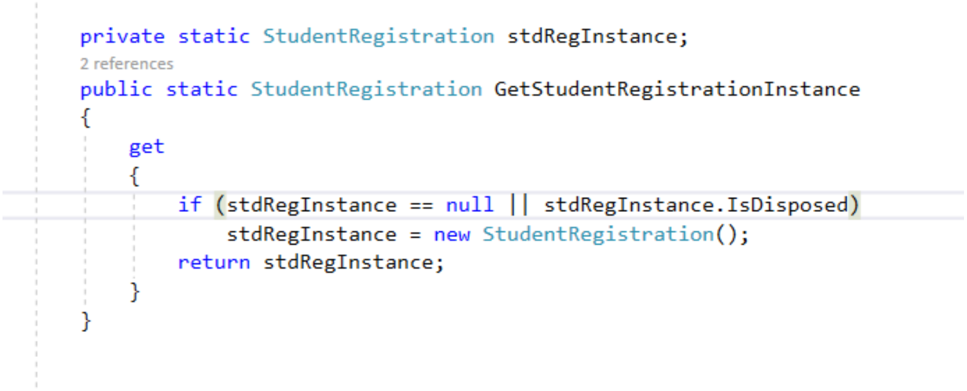
Library Management System is the second project I delivered during my internship. This document aims to introduce Library Management System to my responsibles in İzmir University of Economics and to my place of internship, Unipa. Entire work belongs to me and done for Unipa.

:

https://github.com/solidbaki/Unipa.LibraryManagementSystem.Project2/releases/tag/1.0.0

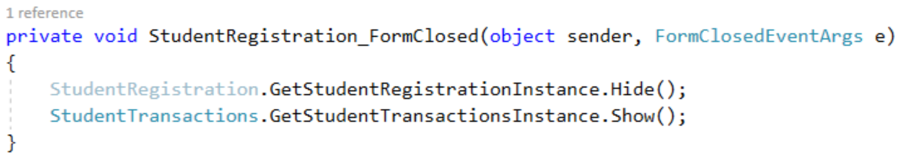
Implementation Details

Each WindowsForm created as a singleton object, which means they will be instantiated once and will be closed/opened.



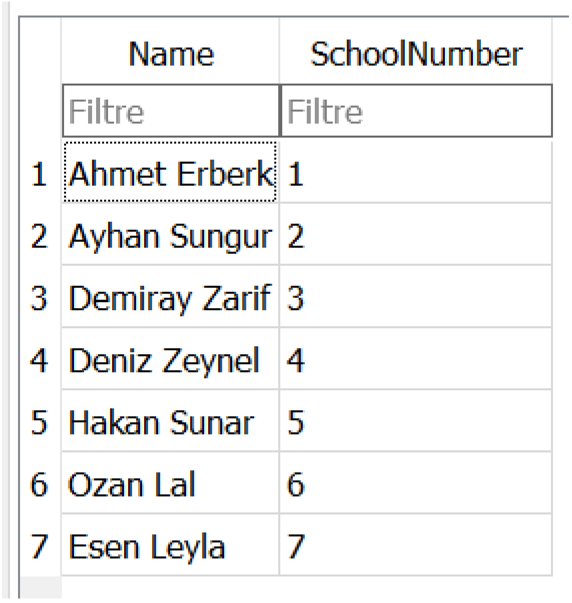
```
private static StudentRegistration stdRegInstance;
2 references
public static StudentRegistration GetStudentRegistrationInstance
{
    get
    {
        if (stdRegInstance == null || stdRegInstance.IsDisposed)
            stdRegInstance = new StudentRegistration();
        return stdRegInstance;
    }
}
```

Below code is executed when a form is closed. Using singleton also makes the code shorter.



```
1 reference
private void StudentRegistration_FormClosed(object sender, FormClosedEventArgs e)
{
    StudentRegistration.GetStudentRegistrationInstance.Hide();
    StudentTransactions.GetStudentTransactionsInstance.Show();
}
```

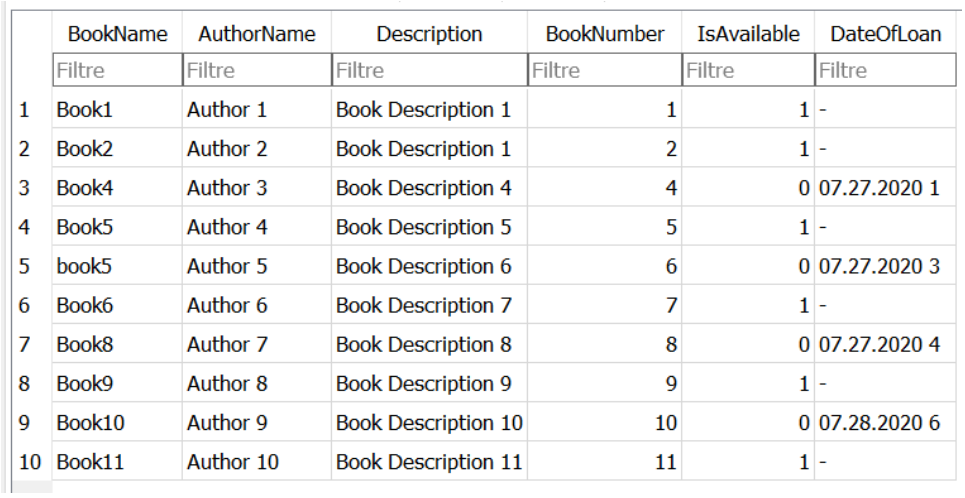
There are two database tables in Library Management System, and both of them are in their simplest forms.



	Name	SchoolNumber
	Filtre	Filtre
1	Ahmet Erberk	1
2	Ayhan Sungur	2
3	Demiray Zarif	3
4	Deniz Zeynel	4
5	Hakan Sunar	5
6	Ozan Lal	6
7	Esen Leyla	7

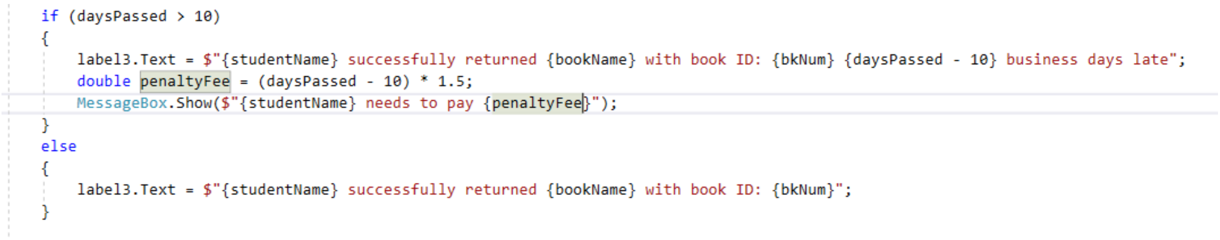
System can register and delete students. Also student names or school numbers can be modified using current school numbers. (Student number is a primarykey)

Also the system can register and delete books. When the system registers a new book, it assigns a BookNumber (ID) to the book. Book modifications can be done using this book number. Also a book is available by default. Which means it can be recieved by a student. When a student loans a book, IsAvailable becomes 0, and DateOfLoan is recorded with student number. In below table, Book4’s DateOfLoan is “07.27.2020 1”, in this example 07.27.2020 is date of loan, and 1 is the student number who loaned the book.



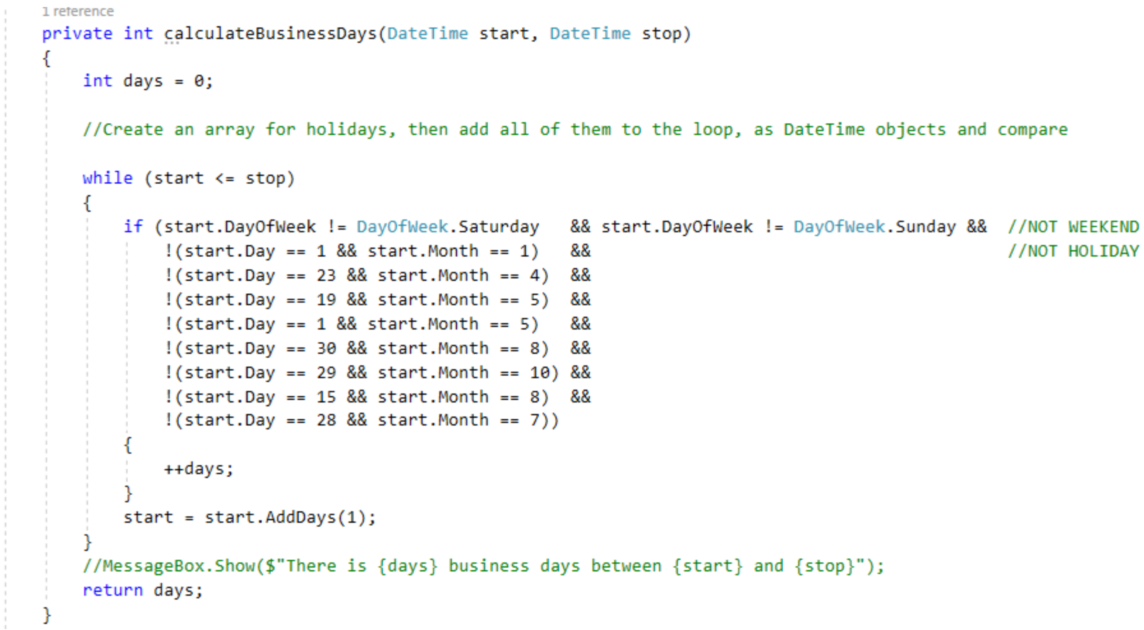
	BookName	AuthorName	Description	BookNumber	IsAvailable	DateOfLoan
	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	Book1	Author 1	Book Description 1	1	1	-
2	Book2	Author 2	Book Description 1	2	1	-
3	Book4	Author 3	Book Description 4	4	0	07.27.2020 1
4	Book5	Author 4	Book Description 5	5	1	-
5	book5	Author 5	Book Description 6	6	0	07.27.2020 3
6	Book6	Author 6	Book Description 7	7	1	-
7	Book8	Author 7	Book Description 8	8	0	07.27.2020 4
8	Book9	Author 8	Book Description 9	9	1	-
9	Book10	Author 9	Book Description 10	10	0	07.28.2020 6
10	Book11	Author 10	Book Description 11	11	1	-

When student returns the book, system splits date and student then calculates if there is a latency or not.



```
if (daysPassed > 10)
{
    label3.Text = $"{studentName} successfully returned {bookName} with book ID: {bkNum} {daysPassed - 10} business days late";
    double penaltyFee = (daysPassed - 10) * 1.5;
    MessageBox.Show($"{studentName} needs to pay {penaltyFee}");
}
else
{
    label3.Text = $"{studentName} successfully returned {bookName} with book ID: {bkNum}";
}
```

Below method shows how the system calculates business days passed, when a book is returned by a student.

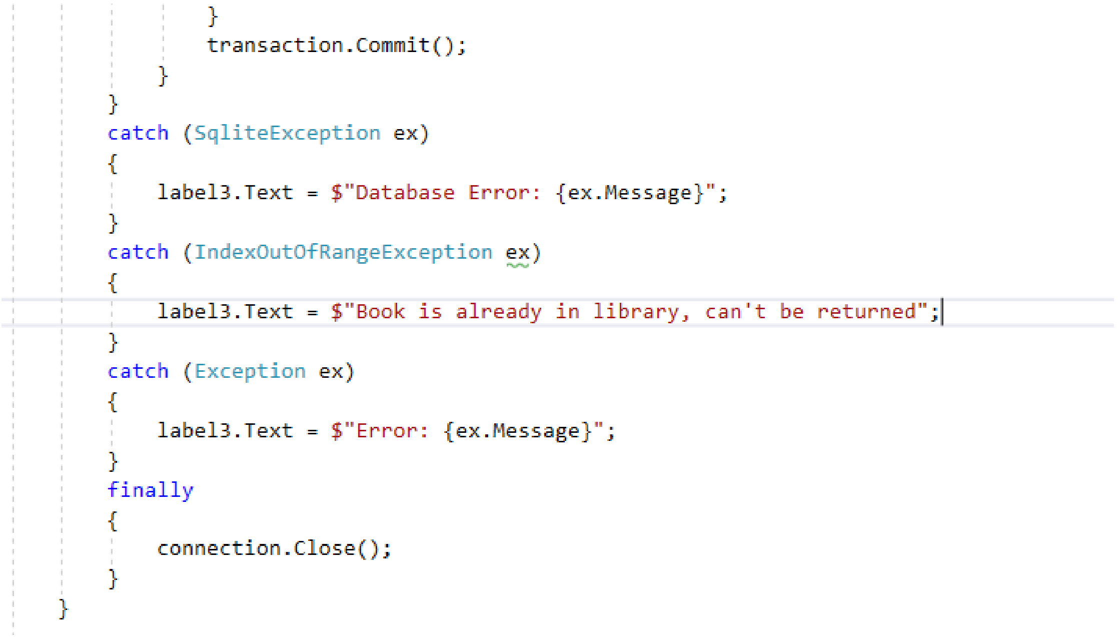


```
1 reference
private int calculateBusinessDays(DateTime start, DateTime stop)
{
    int days = 0;

    //Create an array for holidays, then add all of them to the loop, as DateTime objects and compare

    while (start <= stop)
    {
        if (start.DayOfWeek != DayOfWeek.Saturday   && start.DayOfWeek != DayOfWeek.Sunday &&  //NOT WEEKEND
            !(start.Day == 1 && start.Month == 1)   &&                                         //NOT HOLIDAY
            !(start.Day == 23 && start.Month == 4)  &&
            !(start.Day == 19 && start.Month == 5)  &&
            !(start.Day == 1 && start.Month == 5)   &&
            !(start.Day == 30 && start.Month == 8)  &&
            !(start.Day == 29 && start.Month == 10) &&
            !(start.Day == 15 && start.Month == 8)  &&
            !(start.Day == 28 && start.Month == 7))
        {
            ++days;
        }
        start = start.AddDays(1);
    }
    //MessageBox.Show($"There is {days} business days between {start} and {stop}");
    return days;
}
```

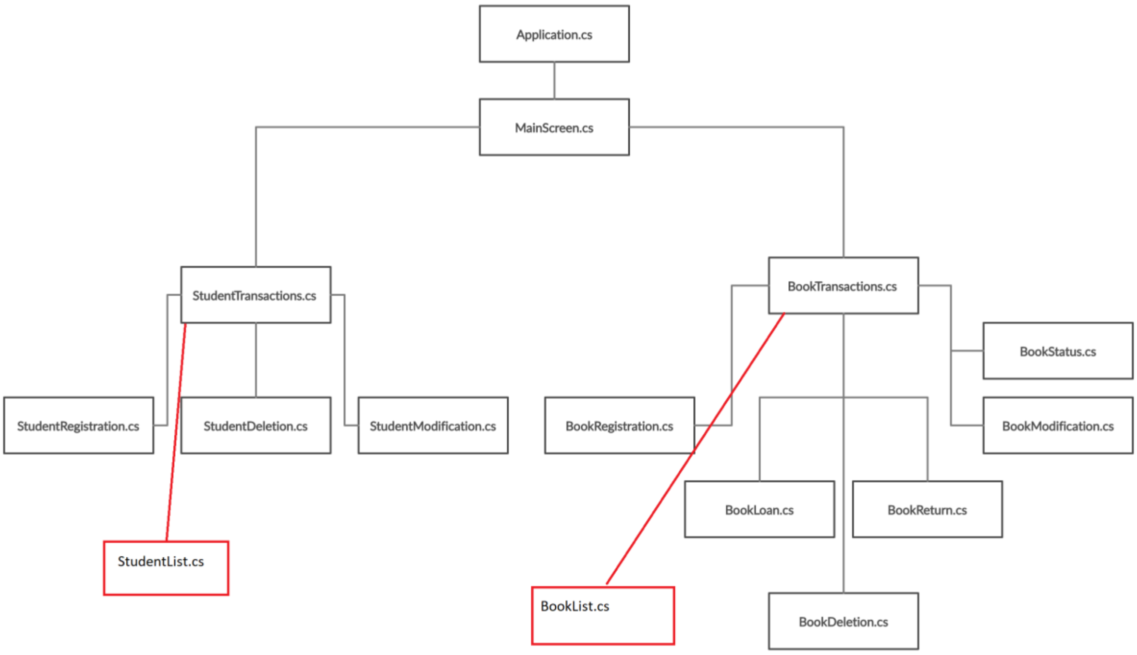
I tried to implement the program to make it safe to crash and all the WindowsForms are secured by try catch blocks, and forms show the reason of problems arised.

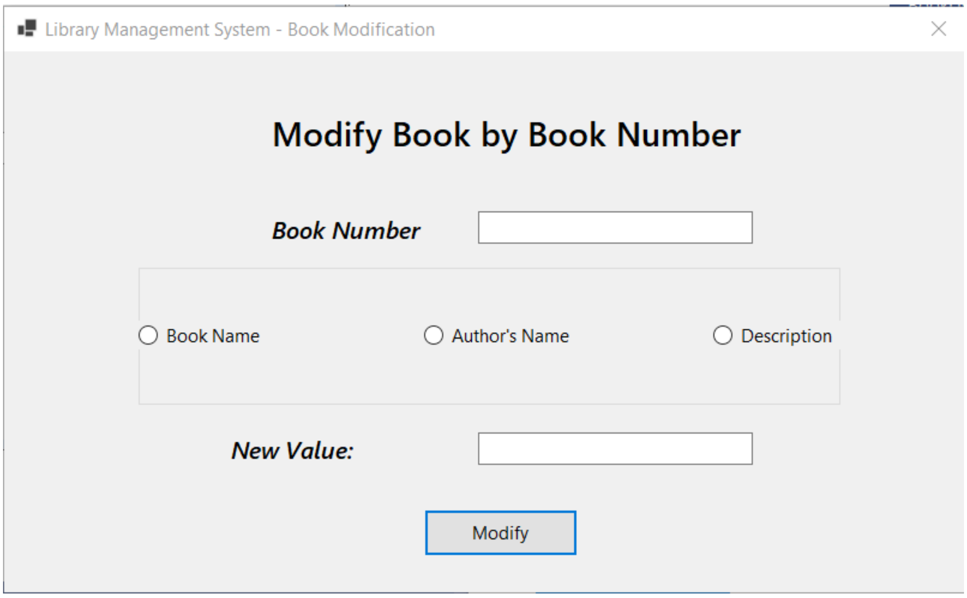


```
                }
                transaction.Commit();
            }
        }
        catch (SqliteException ex)
        {
            label3.Text = $"Database Error: {ex.Message}";
        }
        catch (IndexOutOfRangeException ex)
        {
            label3.Text = $"Book is already in library, can't be returned";
        }
        catch (Exception ex)
        {
            label3.Text = $"Error: {ex.Message}";
        }
        finally
        {
            connection.Close();
        }
    }
```

Modifications

Most of the modifications are about the program's GUI. The only functionality added is showing Book and Student lists.



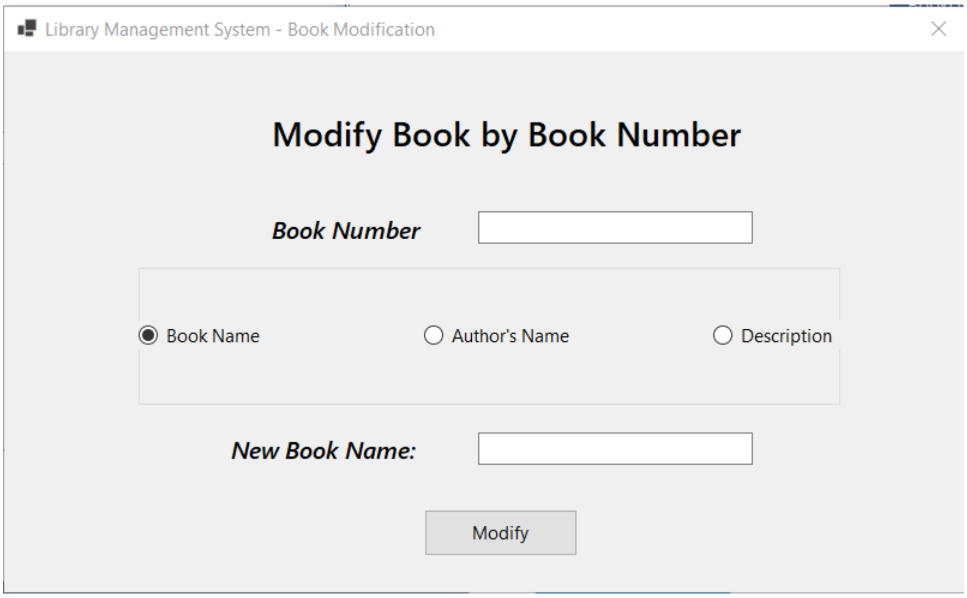

Library Management System - Book Modification

Modify Book by Book Number

Book Number

○ Book Name ○ Author's Name ○ Description

New Value:



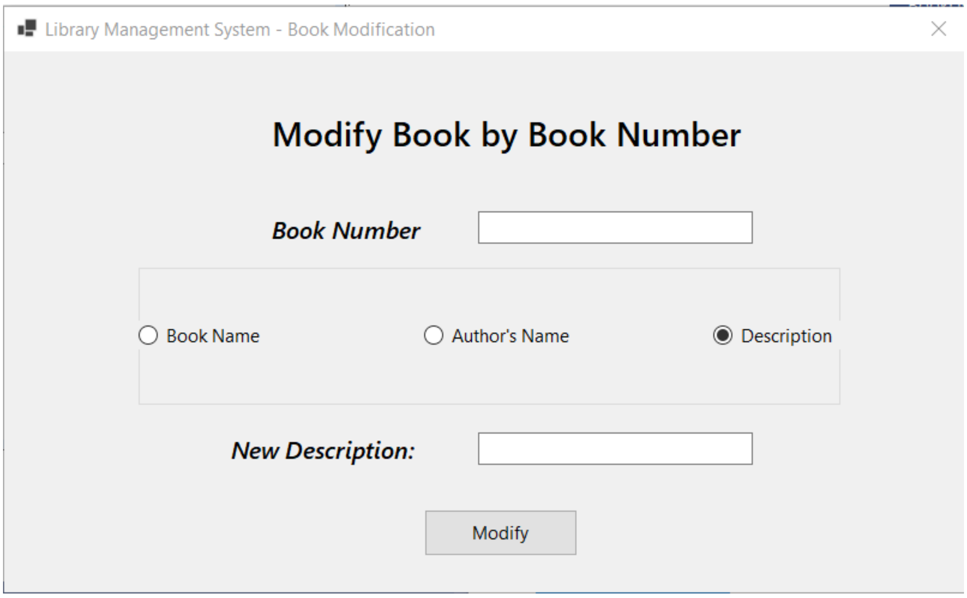
Library Management System - Book Modification

Modify Book by Book Number

Book Number

● Book Name ○ Author's Name ○ Description

New Book Name:



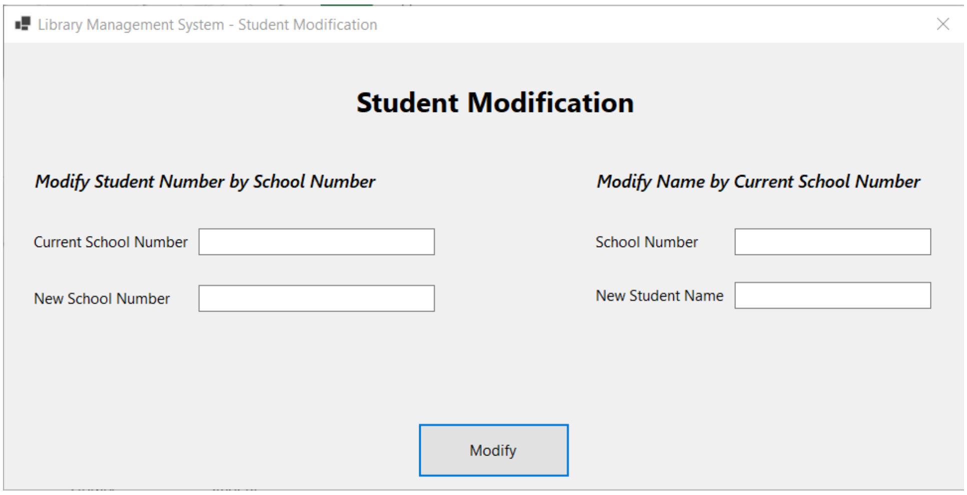
Library Management System - Book Modification

Modify Book by Book Number

Book Number

○ Book Name ○ Author's Name ● Description

New Description:



Library Management System - Student Modification

Student Modification

Modify Student Number by School Number

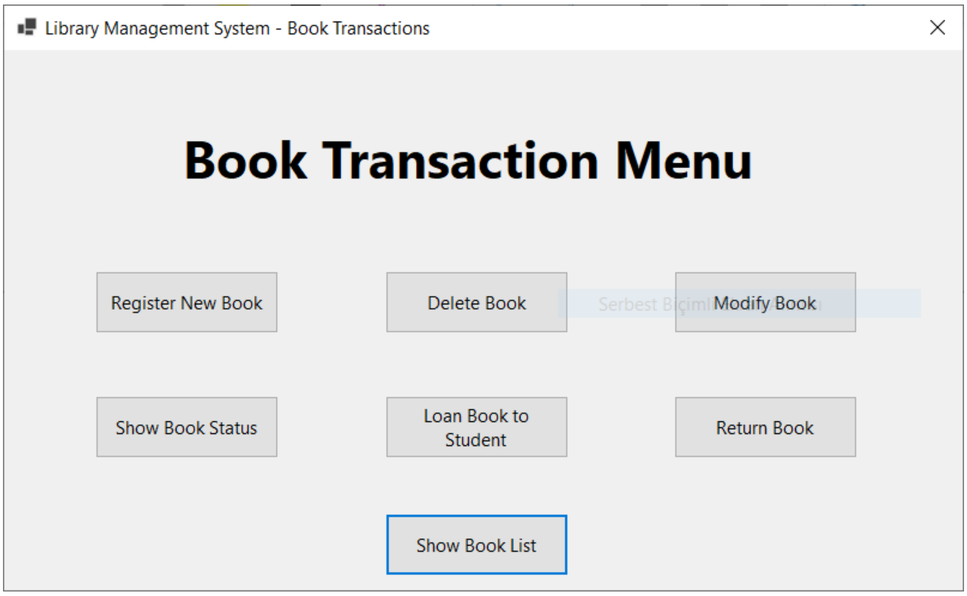
Current School Number

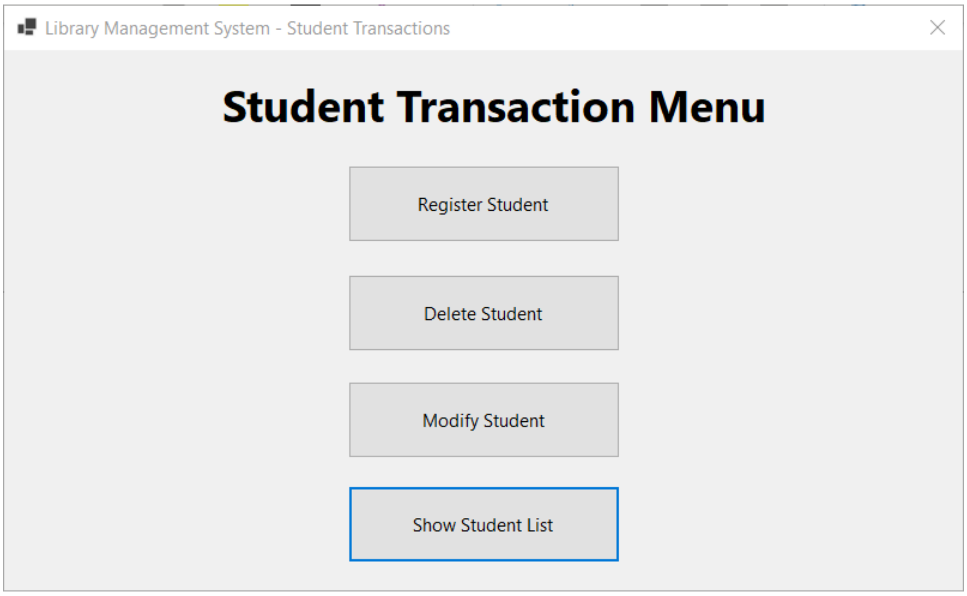
New School Number

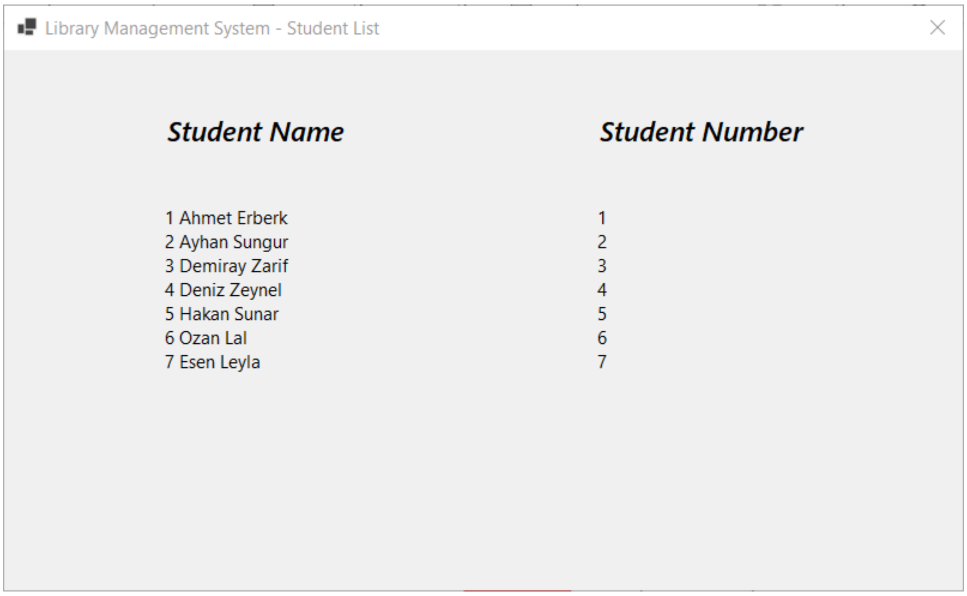
Modify Name by Current School Number

School Number

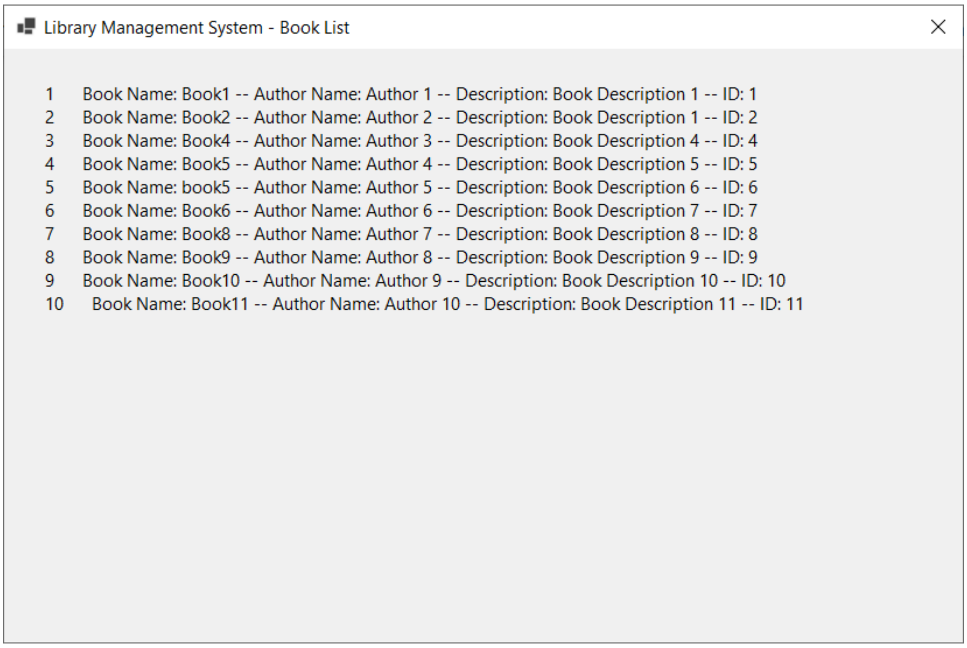
New Student Name


Library Management System - Book Transactions
Book Transaction Menu
Register New Book
Delete Book
Serbest Biçimli
Modify Book
Show Book Status
Loan Book to Student
Return Book
Show Book List


Library Management System - Student Transactions
Student Transaction Menu
Register Student
Delete Student
Modify Student
Show Student List



Student Name	Student Number
1 Ahmet Erberk	1
2 Ayhan Sungur	2
3 Demiray Zarif	3
4 Deniz Zeynel	4
5 Hakan Sunar	5
6 Ozan Lal	6
7 Esen Leyla	7



```
1     Book Name: Book1 -- Author Name: Author 1 -- Description: Book Description 1 -- ID: 1
2     Book Name: Book2 -- Author Name: Author 2 -- Description: Book Description 1 -- ID: 2
3     Book Name: Book4 -- Author Name: Author 3 -- Description: Book Description 4 -- ID: 4
4     Book Name: Book5 -- Author Name: Author 4 -- Description: Book Description 5 -- ID: 5
5     Book Name: book5 -- Author Name: Author 5 -- Description: Book Description 6 -- ID: 6
6     Book Name: Book6 -- Author Name: Author 6 -- Description: Book Description 7 -- ID: 7
7     Book Name: Book8 -- Author Name: Author 7 -- Description: Book Description 8 -- ID: 8
8     Book Name: Book9 -- Author Name: Author 8 -- Description: Book Description 9 -- ID: 9
9     Book Name: Book10 -- Author Name: Author 9 -- Description: Book Description 10 -- ID: 10
10     Book Name: Book11 -- Author Name: Author 10 -- Description: Book Description 11 -- ID: 11
```

What’s Next

1. There needs to be a refactoring process on the later stages. System could be more object oriented. Each Windows Forms has almost the same characteristics such as being non-resizable, and needs to be initialized center screen etc. But they are not inherited from a base class.
2. Since the singleton objects are static, they boot up once at the time when the program starts. It makes a quick booting when the program launches. Program can be updated to work with multi-threads on later stages
3. There might be some corrections on database, using different data types and fields can improve database management quality.
4. Unit testing would be helpful to detect current defects in the system.

Conclusion

To summarize, I successfully designed a library management system and developed the project as I stated in requirement analysis document. Although there are other things to do, I satisfied the requirements and delivered it with a simple GUI.