IUBAT–INTERNATIONAL UNIVERSITY OF BUSINESS AGRICULTURE AND TECHNOLOGY
Founded 1991 by Md.AlimullahMiyan
4 Embankment Drive Road, Sector 10, UttaraModelTown, Dhaka 1230, Bangladesh
Phone: 896 3523-27, 01714 014933, 892 3469-70, 891 8412, Fax: 892 2625, info@iubat.edu www.iubat.edu

Assignment

Course Details:

Course Name- Visual Programming

Course code- CSC439

Section- B

Submitted to:

Suhala Lamia

Senior lecturer, Dept. of Computer Science and Engineering, IUBAT

Submitted by:

Name- Md. Al-Amin

ID- 19303055

GitHub link

Date of submission- 19/ 04/ 2024

Assignment task: IUBAT has a well decorated library. Imagine you are assigned to create a online

platform for the library where user can view the list of books available on the

library. They can search the books by author, title or category.

They can check their history of borrowing the books and returning the books. They

get a notification if they do not return the book after the return date is passed and

fine will be also calculated.

Now write a C# program to implement the given scenario.

C# Code with section wise explanation of used features:

```
using System;
using System.Collections.Generic;
using System.Linq; // LINQ (System.Linq) for querying collections.

class Book
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Author { get; set; }
    public string Category { get; set; }
    public bool IsBorrowed { get; set; }
    public DateTime ReturnDate { get; set; }
} // This defines a Book class to represent individual books in the library.
Properties like Id, Title, Author, Category, IsBorrowed, and ReturnDate are
used to store book information. Using a class encapsulates related data and
behaviors, promoting code organization and reusability.

class User
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Book> BorrowedBooks { get; set; }

    public User()
    {
        BorrowedBooks = new List<Book>();
    }
} //The User class represents library users. It includes properties like Id,
Name, and BorrowedBooks to track books borrowed by the user. Using a
List<Book> for BorrowedBooks allows users to borrow multiple books.

class Library
{
    private List<Book> books;
    private List<User> users;

    public Library()
    {
        books = new List<Book>();
        users = new List<User>();
    } // The Library class encapsulates book and user management. Private
fields books and users store book and user data respectively. The constructor
initializes these lists when an instance of Library is created.
```

```
    public void AddBook(Book book)
    {
        book.Id = books.Count + 1;
        books.Add(book);
    } // AddBook method adds a book to the library. It assigns a unique ID to
the book based on the current count of books in the list. This ensures each
book has a distinct identifier.

    public List<Book> SearchBooks(string keyword)
    {
        return books.Where(b =>
            b.Title.Contains(keyword, StringComparison.OrdinalIgnoreCase) ||
            b.Author.Contains(keyword, StringComparison.OrdinalIgnoreCase) ||
            b.Category.Contains(keyword, StringComparison.OrdinalIgnoreCase))
            .ToList();
    } //SearchBooks method searches for books based on a given keyword
(title, author, or category). It utilizes LINQ's Where method to filter books
that match the keyword case-insensitively across title, author, or category.

    public void BorrowBook(User user, Book book, DateTime returnDate)
    {
        if (book.IsBorrowed)
        {
            Console.WriteLine("This book is already borrowed.");
            return;
        }

        book.IsBorrowed = true;
        book.ReturnDate = returnDate;
        user.BorrowedBooks.Add(book);
        Console.WriteLine($"Book '{book.Title}' borrowed by {user.Name}.");
    } // BorrowBook method handles book borrowing. It checks if the book is
already borrowed (IsBorrowed flag) and adds it to the user's BorrowedBooks
list with the return date specified.

    public void ReturnBook(User user, Book book)
    {
        if (!book.IsBorrowed || !user.BorrowedBooks.Contains(book))
        {
            Console.WriteLine("This book was not borrowed by the user.");
            return;
        }

        TimeSpan overdueDays = DateTime.Now - book.ReturnDate;
        if (overdueDays.TotalDays > 0)
        {
            double fineAmount = overdueDays.TotalDays * 0.5; // Assuming fine
is 50 cents per day
            Console.WriteLine($"Book returned late! Fine amount:
${fineAmount}");
        }

        book.IsBorrowed = false;
        user.BorrowedBooks.Remove(book);
        Console.WriteLine($"Book '{book.Title}' returned by {user.Name}.");
    } // ReturnBook method manages book returns. It checks if the book was
borrowed by the user, calculates fines for overdue returns, updates book
status, removes the book from the user's borrowed list, and displays relevant
messages.

    public void DisplayBooks()
    {
        Console.WriteLine("\nList of Available Books:");
```

```
        foreach (var book in books)
        {
            if (!book.IsBorrowed)
            {
                Console.WriteLine($"ID: {book.Id}, Title: {book.Title},
Author: {book.Author}, Category: {book.Category}");
            }
        }
    }

    public void DisplayUserHistory(User user)
    {
        Console.WriteLine($"\nBorrowing History for User: {user.Name}");
        foreach (var book in user.BorrowedBooks)
        {
            Console.WriteLine($"Title: {book.Title}, Borrowed on:
{book.ReturnDate.ToShortDateString()}");
        }
    }
} // The DisplayBooks and DisplayUserHistory methods are straightforward and
provide functionality to display available books and user borrowing history
respectively.

class Program
{
    static void Main()
    {
        Library library = new Library();

        // Adding sample books to the library
        library.AddBook(new Book { Title = "C# Programming", Author = "John
Doe", Category = "Programming" });
        library.AddBook(new Book { Title = "Harry Potter", Author = "J.K.
Rowling", Category = "Fantasy" });
        library.AddBook(new Book { Title = "To Kill a Mockingbird", Author =
"Harper Lee", Category = "Fiction" });

        // Creating a sample user
        User user1 = new User { Id = 1, Name = "Alice" };

        // Display available books
        library.DisplayBooks();

        // Searching for books
        List<Book> searchResults = library.SearchBooks("Harry Potter");
        foreach (var result in searchResults)
        {
            Console.WriteLine($"\nFound Book: {result.Title} by
{result.Author}");
        }

        // Borrowing a book
        if (searchResults.Count > 0)
        {
            Book bookToBorrow = searchResults[0];
            library.BorrowBook(user1, bookToBorrow,
DateTime.Now.AddDays(14)); // Return date after 2 weeks
        }

        // Display user borrowing history
        library.DisplayUserHistory(user1);

        // Returning a book
```

```
        if (user1.BorrowedBooks.Count > 0)
        {
            Book bookToReturn = user1.BorrowedBooks[0];
            library.ReturnBook(user1, bookToReturn);
        }
    }// The main program (Program.Main) creates a library instance, adds
sample books, demonstrates searching, borrowing, and returning books, and
displays user history.
}
```

External sources:

https://www.geeksforgeeks.org

https://learn.microsoft.com/en-us/dotnet/csharp

https://github.com/Husna-POYRAZ/library-management-system/tree/main/LMSPROJECT