Karunya UNIVERSITY

(Karunya Institute of Technology and Sciences)
(Declared as Deemed to be University under Sec. 3 of the UGC Act.1956)
Karunya Nagar, Coimbatore -641114

DEPARTMENT OF COMPUTER SCIENCES TECHNOLOGY

LABORATORY RECORD

2016-2017

Subject Code

14CS2055

Subject Name

C# and .NET Programming Lab

Register No. UR13CS043

It is hereby certified that this is the bonafide record of work done by Mr./Ms. GANDHAM SUHAAS SRINIVAS during the odd semester of the academic year 2016-2017 and submitted for the University Practical Examination held on 09/11/2016.

Staff-in-charge **HOD / Programme Coordinator**

Name:

ID:

Internal Examiner

List of Exercises

		Video URL : https://youtu.be/fnv35CrLJew?list=PLHLXpcLG4U7B99SaCH4ad_OFAgL7MRoJm	
9	19.10.2016	Web Application using ASP.NET Video URL : https://youtu.be/clJPlpbYwEo?list=PLHLXpcLG4U7B99SaCH4ad_OFAgL7MRoJm	117
10	02.11.2016	Advanced Web Design Video URL : https://youtu.be/E1X0z8wi_Dw?list=PLHLXpcLG4U7B99SaCH4ad_OFAgL7MRoJm	131

Ex. No. 0	BASIC C# PROGRAMS		
Date of Exercise	20.07.2016	Date of Upload	24.08.2016

1. Command line arguments

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{
    using System;

        class Program
        {
            static void Main(string[] args)
            {
            System.Console.WriteLine("This is line");
                for (int i = 0; i < args.Length; i++)
                    System.Console.WriteLine(args[i]);
            }
        }
}
```

OUTPUT:

```
C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program MYCOMMAND
This is line
MYCOMMAND

C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>
```

2. Multiple Main

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{

        class A
        {
            public static void Main()
            {
                System.Console.Write("I AM IN FUNCTION A");
                System.Console.ReadLine();
                B.Main();
```

```
            }
        }
        class B
        {
            public static void Main()
            {
                System.Console.Write("I am in FUNCTION B");
                System.Console.ReadLine();
            }
        }


}
```

OUTPUT:

```
:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program MYCOMMAND
his is line
YCOMMAND

:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>csc Program.cs
icrosoft (R) Visual C# Compiler version 1.2.0.60317
opyright (C) Microsoft Corporation. All rights reserved.


:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program.cs /main:A

:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program /main:A
his is line
main:A

:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>
```

3. Params keyword

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{

    class Test1
    {
        public void fun(params object[] list)
        {
            for (int i = 0; i < list.Length; i++)
                Console.Write(list[i] + " ");
            Console.WriteLine();
        }
        public static void Main()
        {
            Test1 t = new Test1();
            t.fun(1, 2, 3);
            t.fun(6, 7, 8, 9, 10);
            t.fun("hi", "how", "are","you");
            Console.ReadLine();
        }
```

```
    }


}
```

OUTPUT:

```
C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>csc Program.cs
Microsoft (R) Visual C# Compiler version 1.2.0.60317
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program.exe
1 2 3
6 7 8 9 10
hi how are you
```

4. Pass by value & Pass by reference (out, ref keyword and array)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{
    class Test
    {
        public void valuechange(int value)
        {
            value = value + 10;
        }
        public void refer(ref int value)
        {
            value = 100;
        }
        public void array(int[] value)
        {
            value[1] = 10;
        }
        public void outvalue(out int value)
        {
            value = 100;
        }
        public static void Main()
        {
            int value1 = 123;
            int value2 = 123;
            int z;
            Test obj = new Test();
            obj.valuechange(value1);
```

```
            System.Console.WriteLine("Passing 123 through VALUE  "+value1);
            obj.refer(ref value2);
            System.Console.WriteLine("Passing 123 through REF  " + value2);
            obj.outvalue(out z);
            System.Console.WriteLine("Passing through OUT  " + z);
            int[] arr = { 1, 2, 3, 4, 5 };
            obj.array(arr);
            System.Console.WriteLine("Passing ARRAY  " + arr[1]);
            Console.ReadLine();
        }
    }
}
```

OUTPUT:

```
C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>csc Program.cs
Microsoft (R) Visual C# Compiler version 1.2.0.60317
Copyright (C) Microsoft Corporation. All rights reserved.

C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>Program.exe
Passing 123 through VALUE  123
Passing 123 through REF  100
Passing through OUT  100
Passing ARRAY  10
```

5. Create dll file (/t:library) and refer it in main program (/r:filename.dll)

MathClient.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{
    class Client
    {
        public static void Main()
        {
            MathLib mathObj = new MathLib();
            Console.WriteLine(mathObj.Add(7, 8));
        }
    }
}
```

MathLibrary.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ZerothExperiment
{

        public class MathLib
        {
            public int Add(int x, int y)
            {
                return x + y;
            }
        }

}
```

OUTPUT:

```
C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>cs
c /t:library MathLibrary.cs
Microsoft (R) Visual C# Compiler version 1.2.0.60317
Copyright (C) Microsoft Corporation. All rights reserved.


C:\Users\chinnu\Documents\Visual Studio 2015\Projects\CSharpLab\ZerothExperiment\ZerothExperiment>cs
c MathClient.cs /r:MathLibrary.dll
Microsoft (R) Visual C# Compiler version 1.2.0.60317
Copyright (C) Microsoft Corporation. All rights reserved.
```

Ex. No. 1	Inheritance		
Date of Exercise	20.07.2016	Date of Upload	19.08.2016

Aim

To develop **Library Management System** using C# for various distinct keeping in my mind the necessary constraints and concepts.

Description

When creating a class, instead of writing completely new data members and member functions, the programmer can designate that the new class should inherit the members of an existing class. This existing class is called the **base class**, and the new class is referred to as the **derived class**. The idea of inheritance implements the **IS-A relationship.** For example, mammal IS A animal, dog IS-A mammal hence dog IS-A animal as well, and so on.

There are two distinct types of inheritance :

Implementation inheritance which is a derived type adopts the base type ' s implementation of each function.
Interface inheritance which inherits only the signatures of the functions and does not inherit any implementations.

Base and Derived Classes

A class can be derived from more than one class or interface, which means that it can inherit data and functions from multiple base classes or interfaces.

The syntax used in C# for creating derived classes is as follows:

```
<acess-specifier> class <base_class>
{
  ...
}
class <derived_class> : <base_class>
{
  ...
}
```

Abstract Classes and Functions

- An abstract class cannot be instantiated
- Abstract function does not have an implementation
- Must be overridden in any non - abstract derived class
- An abstract function is automatically virtual
- If any class contains any abstract functions, that class is also abstract

```
abstract class Building
{
private bool damaged = false; // field
public abstract decimal CalculateHeatingCost(); // abstract method
}
```

Sealed Classes and Methods

- Sealed class, can ’ t be inherit
- Sealed method, can ’ t be override
-

```
sealed class FinalClass {
// etc }
class DerivedClass : FinalClass
// wrong. Will give compilation error
{ // etc }
class MyClass { public sealed virtual void FinalMethod() { // etc. } }
class DerivedClass : MyClass {
public override void FinalMethod()
// wrong. Will give compilation error { }
 }
```

C# does not support multiple inheritance. However, you can use interfaces to implement multiple inheritance. The following program demonstrates this:

```
public interface Compute : Addition
{
void Sub();
}
public interface Addition
{
void Add();
}
public interface Subtraction
{
void Sub();
}
class Computation : Addition, Subtraction
{
public void Add(){ }
public void Sub(){ }
}
```

Program

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace LibraryManagementSystem
{
    //books structure
    public struct Books
    {
        public string title;
        public string author;
        public string subject;
        public  int book_id ;
        public string status;
        public int book_count;
        public string book_status;
    }

    //Abstract class as methods have no implementation
    abstract class Security {
        public static string ReadPassword() {
            return "for masking input";
        }

        public static void CheckPwd(Books[] arr) { }

    }

    //Inheriting Security class functions with same signature
    class Program:Security
    {
        public static int student_book_count = 3;
        public static int faculty_book_count = 5;

         public static  List<String>  book_issue_list = new List<String>();

        public static Hashtable BookIssueHash = new Hashtable();

        public static int count = 1;

        static void Main(string[] args)
        {

            Books[] bookarr = new Books[10];
            String[] users = {"admin","student","faculty"};
            CheckPwd(bookarr);

        }
        //Method Hiding
        public new static string ReadPassword()
```

```
        {
            string password = "";
            ConsoleKeyInfo info = Console.ReadKey(true);
            while (info.Key != ConsoleKey.Enter)
            {
                if (info.Key != ConsoleKey.Backspace)
                {
                    Console.Write("*");
                    password += info.KeyChar;
                }
                else if (info.Key == ConsoleKey.Backspace)
                {
                    if (!string.IsNullOrEmpty(password))
                    {
                        // remove one character from the list of password characters
                        password = password.Substring(0, password.Length - 1);
                        // get the location of the cursor
                        int pos = Console.CursorLeft;
                        // move the cursor to the left by one character
                        Console.SetCursorPosition(pos - 1, Console.CursorTop);
                        // replace it with space
                        Console.Write(" ");
                        // move the cursor to the left by one character again
                        Console.SetCursorPosition(pos - 1, Console.CursorTop);
                    }
                }
                info = Console.ReadKey(true);
            }
            // add a new line because user pressed enter at the end of their password
            Console.WriteLine();
            return password;
        }
        //Method Hiding
        public new static void  CheckPwd(Books[] arr) {
            Console.WriteLine("**---------------------------Welcome to Library Management
System---------------------------**");
            Console.WriteLine("\n");
            Console.Write("Username:");
            String user = Console.ReadLine();
            user = user.ToLowerInvariant();
            Console.Write("Password:");
            var password = ReadPassword();
                if (user.Equals("admin") && password.Equals("admin"))
                {
                //for admins
                AdminClass.AdminFun(arr);

                }
                else if (user.Equals("faculty") && password.Equals("faculty"))
                {
                //for faculty
                FacultyClass.FacultyFun(arr);

                }
                else if (user.Equals("student") && password.Equals("student"))
                {
                //for student
```

```
                StudentClass.StudentFun(arr);
                }

                else
                {
                    Console.WriteLine("\aBad Attempt!!! :( ");
                    CheckPwd(arr);
                }

        }

    }

    class FacultyClass {

        public static void FacultyFun(Books[] arr)
        {
            StudentClass stuobj = new StudentClass();
            AdminClass adminobj = new AdminClass();
            FacultyClass facobj = new FacultyClass();
            Console.WriteLine();
            Console.WriteLine("1.Search for Books\n2.Reserve Book\n3.Borrow
Book\n4.Return Book\n5.Renew a Book\n6.View Book Issue
Details\n7.Logout\n_______________________");
            int input = Convert.ToInt32(Console.ReadLine());
            switch (input)
            {
                case 1:
                    stuobj.SearchBooks(arr);
                    FacultyFun(arr);
                    break;

                case 2:
                    Console.WriteLine("1.Reserve via Search ");
                    Console.WriteLine("2.Reserve via Book_id
\n_______________________");
                    int cho = Convert.ToInt32(Console.ReadLine());
                    if (cho == 1) { stuobj.SearchBooks(arr);facobj.ReserveBooks(arr);  }
                    else { facobj.ReserveBooks(arr); }
                    FacultyFun(arr);

                    break;
                case 3:
                    Console.WriteLine("1.Borrow via Search ");
                    Console.WriteLine("2.Borrow via Book_id \n_______________________");
                    int ch = Convert.ToInt32(Console.ReadLine());
                    if (ch == 1) { stuobj.SearchBooks(arr); }
                    else { stuobj.BorrowBooks(arr, "faculty"); }
                    FacultyFun(arr);
                    break;

                case 4:
                    stuobj.ReturnBooks(arr,"faculty");
                    FacultyFun(arr);
                    break;

                case 6:
```

```
                    stuobj.BookIssueDeatils();
                    FacultyFun(arr);
                    break;

                case 5:
                    adminobj.ViewBooks(arr);
                    FacultyFun(arr);
                    break;
                case 7:
                    stuobj.LoginPage(arr);
                    break;

                default:
                    Console.WriteLine("_________________________Invalid Choice :(
_________________________");
                    FacultyFun(arr);
                    break;
            }
        }

        public void ReserveBooks(Books[] bookarr)
        {
            Console.WriteLine("_________________________Please Enter the book_id to be
Reserved ?_________________________");
            int id = Convert.ToInt32(Console.ReadLine());
            if (id != 0 && id > 0)
            {
                if ((bookarr[id].book_id).Equals(id))
                {

                    if (Program.student_book_count > 0 && Program.faculty_book_count > 0)
                    {
                        if ((bookarr[id].book_count) > 0)
                        {
                            String val = "reserved";
                            val = val.ToUpper();
                            bookarr[id].book_status = val;
                            Console.WriteLine("_________________________The book is
Reserved successfully :)_________________________");
                        }
                        else
                        { Console.WriteLine("_________________________Out of Stock :(
_________________________"); }
                    }
                    else { Console.WriteLine("_________________________Dear Staff, You
have currently issued  3 Books_________________________"); }
                }
                else { Console.WriteLine("_________________________Book_ID is not matching
:( _________________________"); ReserveBooks(bookarr); }
            }
            else
            {
                Console.WriteLine("_________________________Enter a Valid Book_Id
Please_________________________");
                ReserveBooks(bookarr);
            }
        }
```

```
    }

   interface CommonFunctions {
         void LoginPage(Books[] arr);
    }


    class AdminFunctions {
        public virtual void InsertBooks(Books[] bookarr) { }
        public virtual void ViewAccounts() { throw new NotImplementedException(); }
        public virtual void ViewBooks(Books[] arr) { }
    }

    class AdminClass:AdminFunctions,CommonFunctions {

        public static void AdminFun(Books[] arr)
        {
            AdminClass adminobj = new AdminClass();
            Console.WriteLine("1.Insert Books\n2.View Books\n3.Alter Books\n4.Alter
Accounts\n5.Logout\n_________________________");
            int input = Convert.ToInt32(Console.ReadLine());
            switch(input)
            {
                case 1:
                    adminobj.InsertBooks(arr);
                    AdminFun(arr);
                    break;

                case 2:
                    adminobj.ViewBooks(arr);
                    AdminFun(arr);
                    break;

                case 3:
                    break;

                case 4:
                    break;

                case 5:
                    adminobj.LoginPage(arr);
                    break;

                default:
                    Console.WriteLine("Invalid Choice :(");
                    AdminFun(arr);
                    break;
            }

        }
        //sealing the admin functions
        public override sealed  void InsertBooks(Books[] bookarr)
        {
            Console.WriteLine("How many you want to enter ?");
            int num_books = Convert.ToInt32(Console.ReadLine());
            int initial = Program.count;
            Console.WriteLine(initial);
```

```
            for (int i = initial; i <(num_books+initial); i++)
            {
                bookarr[i].book_id =(Program.count)++;
                Console.WriteLine("Enter the Title of the Book?");
                bookarr[i].title = (Console.ReadLine()).ToUpper();
                Console.WriteLine("Enter the Author of the Book?");
                bookarr[i].author = (Console.ReadLine()).ToUpper();
                Console.WriteLine("Enter the Subject of the Book?");
                bookarr[i].subject = (Console.ReadLine()).ToUpper();
                Console.WriteLine("Enter the Total Count of the Book?");
                bookarr[i].book_count = Convert.ToInt32(Console.ReadLine());
                bookarr[i].book_status = "unreserved";
            }
            Console.WriteLine("The Entered Books are :");
            ViewBooks(bookarr);
        }
        //Using the base function for throwing Exception
        public override void ViewAccounts()
        {
            base.ViewAccounts();
        }
        //sealing the admin functions
        public override sealed void ViewBooks(Books[] arr)
        {
            int length = arr.Length;

Console.WriteLine("Book_ID\t\tTITLE\t\tAUTHOR\t\tSUBJECT\t\tBOOK_COUNT\t\tStatus");
            for (int i = 0; i < length; i++)
            {
                if (arr[i].book_id != 0) {

Console.WriteLine("{0}\t\t{1}\t\t{2}\t\t{3}\t\t{4}\t\t{5}",arr[i].book_id, arr[i].title,
arr[i].author, arr[i].subject,arr[i].book_count,arr[i].book_status);
                }
            }

        }


        public void LoginPage(Books[] arr) {
            Console.WriteLine("Logging out of the System");
            Program.CheckPwd(arr);
        }
    }

    interface Bookfunctions {
        void SearchBooks(Books[] bookarr);
        void LocalSearch(Books[] bookarr, int choice, String title, String author, int
id);
        void BorrowBooks(Books[] bookarr, String username);
        void BookIssueDeatils();
        void ReturnBooks(Books[] bookarr, String username);
    }

    interface Displayfunction {
         void LocalView(Books[] arr, int[] matchval);
    }
```

```
    //Deriving Functions from Interface Bookfunctions
    //Deriving Functions from Interface Displayfunctions
    class StudentClass:CommonFunctions,Bookfunctions,Displayfunction {

        public static void StudentFun(Books[] arr)
        {
            StudentClass stuobj = new StudentClass();
            AdminClass adminobj = new AdminClass();
            Console.WriteLine();
            Console.WriteLine("\n1.Search for Books\n2.Borrow Book\n3.Return Book\n4.View
Book Issue Details\n5.View All Books\n6.Log Out\n_________________________");
            int input = Convert.ToInt32(Console.ReadLine());
                switch (input)
                {
                    case 1:
                        stuobj.SearchBooks(arr);
                        StudentFun(arr);
                        break;

                    case 2:
                        Console.WriteLine("1.Borrow via Search ");
                        Console.WriteLine("2.Borrow via Book_id
\n_________________________");
                        int ch = Convert.ToInt32(Console.ReadLine());
                    if (ch == 1){ stuobj.SearchBooks(arr); }
                    else { stuobj.BorrowBooks(arr,"student"); }
                    StudentFun(arr);
                        break;

                    case 3:
                        stuobj.ReturnBooks(arr,"student");
                        StudentFun(arr);
                        break;

                    case 4:
                    stuobj.BookIssueDeatils();
                        StudentFun(arr);
                        break;

                case 5:
                    adminobj.ViewBooks(arr);
                    StudentFun(arr);
                    break;
                case 6:
                    stuobj.LoginPage(arr);
                    break;

                    default:
                        Console.WriteLine("Invalid Choice :(");
                        StudentFun(arr);
                        break;
                }
        }

        public  void LoginPage(Books[] arr)
        {
```

```
            Console.WriteLine("Logging out of the System");
            Program.CheckPwd(arr);
        }

        public void SearchBooks(Books[] bookarr) {
            String title = "", author = "";
            Console.WriteLine("Please Input the Search type ");
            Console.WriteLine("1.Title\n2.Author\n3.Book_id\n4.Title & author");
            int cho = Convert.ToInt32(Console.ReadLine());
            switch (cho) {
                case 1:
                    Console.WriteLine("Enter the title of the Book ?");
                    title = Console.ReadLine();
                    title = title.ToUpper();
                    LocalSearch(bookarr, 1,title,author,0);
                    break;
                case 2:
                    Console.WriteLine("Enter the author of the Book ?");
                    author = Console.ReadLine();
                    author = author.ToUpper();
                    LocalSearch(bookarr, 2,title,author,0);
                    break;
                case 3:
                    Console.WriteLine("Enter the id of the Book ?");
                    int id = Convert.ToInt32(Console.ReadLine());
                    LocalSearch(bookarr, 3, title,author,id);
                    break;
                case 4:
                    Console.WriteLine("Enter the title of the Book ?");
                    title = Console.ReadLine();
                    title = title.ToUpper();
                    Console.WriteLine("Enter the author of the Book ?");
                    author = Console.ReadLine();
                    author = author.ToUpper();
                    LocalSearch(bookarr, 4, title, author,0);
                    break;
                default:
                    break;
            }

        }

        public void LocalSearch(Books[] bookarr,int choice,String title,String author,
int id) {
            int[] array =new int[bookarr.Length];
            int countval= 0;
            if (choice == 1)
            {
                for (int i = 1; i < bookarr.Length; i++)
                {
                    if ((bookarr[i].book_id) != 0) {
                        if ((bookarr[i].title).Equals(title))
                        { array[countval++] = bookarr[i].book_id; }
                    }
                }
                LocalView(bookarr, array);
            }
```

```
            else if (choice == 2)
            {
                for (int i = 1; i < bookarr.Length; i++)
                {
                    if ((bookarr[i].book_id) != 0)
                    {
                        if ((bookarr[i].author).Equals(author))
                        { array[countval++] = bookarr[i].book_id; }
                    }
                }
                LocalView(bookarr, array);
            }
            else if (choice == 3)
            {
                for (int i = 1; i < bookarr.Length; i++)
                {
                    if ((bookarr[i].book_id) != 0)
                    {
                        if ((bookarr[i].book_id).Equals(id))
                        { array[countval++] = bookarr[i].book_id; }
                    }
                }
                LocalView(bookarr, array);
            }
            else {
                for (int i = 1; i < bookarr.Length; i++)
                {
                    if ((bookarr[i].book_id) != 0)
                    {
                        if (((bookarr[i].title).Equals(title))||
((bookarr[i].author).Equals(author)))
                        { array[countval++] = bookarr[i].book_id; }
                    }
                }
                LocalView(bookarr, array);
            }
        }

        public void LocalView(Books[] arr,int[] matchval) {
            int length = arr.Length;
            if (matchval[0] == 0) { Console.WriteLine("\n_________________________Sorry :(
No records are found_________________________"); }
            else {
                Console.WriteLine("Book_ID\t\tTITLE\t\tAUTHOR\t\tSUBJECT");
                Console.WriteLine("_______\t\t_____\t\t______\t\t_______");
                foreach (int i in matchval)
                {
                    //Console.WriteLine("The val of i is " + i);
                    if (arr[i].book_id != 0)
                    {
                        Console.WriteLine();
                        Console.WriteLine("{0}\t\t{1}\t\t{2}\t\t{3}", arr[i].book_id,
arr[i].title, arr[i].author, arr[i].subject);
                    }

                }
            }
```

```
        }

        public void BorrowBooks(Books[] bookarr,String username) {
            Console.WriteLine("Please Enter the book_id to be borrowed ?");
            int id = Convert.ToInt32(Console.ReadLine());
            if (id != 0)
            {
                if ((bookarr[id].book_id).Equals(id))
                {
                    if (username.Equals("student"))
                    {
                        if (Program.student_book_count > 0)
                        {
                            if ((bookarr[id].book_count) > 0)
                            {
                                String val = "reserved";val = val.ToUpper();
                                if (bookarr[id].book_status.Equals(val)) {
Console.WriteLine("_________________________The book is Reserved by
faculty__________________________"); }
                                else {
                                    (bookarr[id].book_count)--;
(Program.student_book_count)--;
                                    String value = bookarr[id].book_id + "\t" + username
+ "\t" + DateTime.Now + "\t" + (DateTime.Now).AddDays(15);
                                    value = value.ToUpper();
                                    //Program.book_issue_list.Add(value);
                                    int IssueNo = new Random().Next(999, 99999);
                                    Program.BookIssueHash.Add(IssueNo, value);
                                    Console.WriteLine("\tDear" + username + ",your Issue
no is [{0}]", IssueNo);
                                    Console.WriteLine("\tBook Issued Successfully :) ");
                                }
                            }
                            else
                            { Console.WriteLine("_________________________Out of Stock :(
_________________________"); }
                        }
                        else { Console.WriteLine("Dear Student, You cannot issue more
than 3 Books"); }
                    }
                    else if (username.Equals("faculty"))
                    {
                        if (Program.faculty_book_count > 0)
                        {
                            if ((bookarr[id].book_count) > 0)
                            {
                                (bookarr[id].book_count)--; (Program.faculty_book_count)-
-;
                                String value = bookarr[id].book_id + "\t" + username +
"\t" + DateTime.Now + "\t" + (DateTime.Now).AddYears(1);
                                value = value.ToUpper();
                                Program.book_issue_list.Add(value);
                                int IssueNo = new Random().Next(999, 99999);
                                Program.BookIssueHash.Add(IssueNo, value);
                                Console.WriteLine("\tDear" + username + ",your Issue no
is [{0}]", IssueNo);
```

```
                                Console.WriteLine("_________________________Book Issued
Successfully :) _________________________");
                            }
                            else {Console.WriteLine("_________________________Out of Stock
:(_________________________");}
                        }
                        else { Console.WriteLine("_________________________Dear staff, You
cannot issue more than 5 Books_________________________"); }
                    }
                }
                else { Console.WriteLine("_________________________Book_ID is not matching
:(_________________________"); BorrowBooks(bookarr, username); }
            }
            else {
                Console.WriteLine("_________________________Enter a Valid Book_Id
Please_________________________");
                BorrowBooks(bookarr,username);
            }
        }

        public void BookIssueDeatils() {
            ICollection key = Program.BookIssueHash.Keys;
            if (key.Count != 0)
            {
                Console.WriteLine("\nBook_id\tHolder\tIssued On\t\tTo be Returned");
                Console.WriteLine("_______\t______\t_________\t\t_______________");
                // Get a collection of the keys.

                foreach (int k in key)
                {
                    Console.WriteLine(Program.BookIssueHash[k]);
                }
            }
            else {
                Console.WriteLine("\n\t---------------No Issue Records are found---------
------\t");
            }

        }

        public void ReturnBooks(Books[] bookarr,String username) {
            Console.WriteLine("Enter the Issue No to return your books ?");
            int Issno = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Your issue record is:
\n{0}",Program.BookIssueHash[Issno]);
            String text = Program.BookIssueHash[Issno].ToString();
            String[] sub = text.Split('\t');
            //Console.WriteLine("After Splitting");
            //foreach(String data in sub) { Console.WriteLine(data); }
            int id = Convert.ToInt32(sub[0]);
            Program.BookIssueHash.Remove(Issno);
            if (username.Equals("student")) { (Program.student_book_count)++; } else {
(Program.faculty_book_count)++; }
            (bookarr[id].book_count)++; bookarr[id].book_status="unreserved";
            Console.WriteLine("_________________________Book is returned Successfully
:)_________________________");
        }
```

```
    }

}
```

Output

```
**-------------------------Welcome to Library Management System-------------------------**


Username:admin
Password:*****
1.Insert Books
2.View Books
3.Alter Books
4.Alter Accounts
5.Logout
_____________________
1
How many you want to enter ?
2
1
Enter the Title of the Book?
html
Enter the Author of the Book?
thomas
Enter the Subject of the Book?
web
Enter the Total Count of the Book?
2
Enter the Title of the Book?
C Pr
Enter the Author of the Book?
yash
Enter the Subject of the Book?
Programng
Enter the Total Count of the Book?
5
The Entered Books are :
Book_ID         TITLE           AUTHOR          SUBJECT         BOOK_COUNT              Status
1               HTML            THOMAS          WEB             2               unreserved
2               C PR            YASH            PROGRAMNG               5               unreserved
1.Insert Books
```

```
1.Search for Books
2.Borrow Book
3.Return Book
4.View Book Issue Details
5.View All Books
6.Log Out
_________________________
2
1.Borrow via Search
2.Borrow via Book_id
_________________________
1
Please Input the Search type
1.Title
2.Author
3.Book_id
4.Title & author
1
Enter the title of the Book ?
C Pr
Book_ID          TITLE            AUTHOR           SUBJECT
_______          _____            ______           _______

2                C PR             YASH             PROGRAMNG


1.Search for Books
2.Borrow Book
3.Return Book
4.View Book Issue Details
5.View All Books
6.Log Out
_________________________
2
1.Borrow via Search
2.Borrow via Book_id
_________________________
2
Please Enter the book_id to be borrowed ?
2
        Dearstudent,your Issue no is [64167]
        Book Issued Successfully :)
```

```
1.Search for Books
2.Borrow Book
3.Return Book
4.View Book Issue Details
5.View All Books
6.Log Out
_________________________
3
Enter the Issue No to return your books ?
64167
Your issue record is:
2       STUDENT 19-AUG-16 21:07:14      03-SEP-16 21:07:14
_________________________Book is returned Successfully :)_______


1.Search for Books
2.Borrow Book
3.Return Book
4.View Book Issue Details
5.View All Books
6.Log Out
_________________________
```

If tried to borrowed more than 3 times

```
Please Enter the book_id to be borrowed ?
1
Dear Student, You cannot issue more than 3 Books
```

To reserve the book by faculty

```
Username:faculty
Password:*******

1.Search for Books
2.Reserve Book
3.Borrow Book
4.Return Book
5.Renew a Book
6.View Book Issue Details
7.Logout
_____________________
2
1.Reserve via Search
2.Reserve via Book_id
_____________________
2
_________________________Please Enter the book_id to be Reserved ?_________________________
1
_________________________The book is Reserved successfully :)_________________________

1.Search for Books
2.Reserve Book
3.Borrow Book
4.Return Book
5.Renew a Book
6.View Book Issue Details
7.Logout
_____________________
7
**--------------------------Welcome to Library Management System---------------------------**


Username:admin
Password:*****
1.Insert Books
2.View Books
3.Alter Books
4.Alter Accounts
5.Logout
_____________________
2
Book_ID         TITLE           AUTHOR          SUBJECT         BOOK_COUNT              Status
1               HTML            THOMAS          WEB DEV         2               RESERVED
1.Insert Books
2.View Books
3.Alter Books
4.Alter Accounts
5.Logout
_____________________
```

Issue Details

```
_________________________
2
Please Enter the book_id to be borrowed ?
1
        Dearstudent,your Issue no is [81823]
        Book Issued Successfully :)


1.Search for Books
2.Borrow Book
3.Return Book
4.View Book Issue Details
5.View All Books
6.Log Out
_________________________
4

Book_id Holder  Issued On                To be Returned
_______ ______  _________                ______________
1       STUDENT 19-AUG-16 21:16:45       03-SEP-16 21:16:45
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]

Ex. No. 2	OPERATOR OVERLOADING		
Date of Exercise	20.07.2016	Date of Upload	23.08.2016

Aim

To write a Program in C# to overload various operators such as arithmetic, comparison and further user defined casting for the **Matrix Application.**

Description

Syntax of operator overloading

We can redefine or overload most of the built-in operators available in C#. Thus a programmer can use operators with user-defined types as well. Overloaded operators are functions with special names the keyword **operator** followed by the symbol for the operator being defined. similar to any other function, an overloaded operator has a return type and a parameter list.

For example:

```
public static Box operator+ (Box b, Box c)
{
  Box box = new Box();
  box.length = b.length + c.length;
  box.breadth = b.breadth + c.breadth;
  box.height = b.height + c.height;
  return box;
}
```

The above function implements the addition operator (+) for a user-defined class Box. It adds the attributes of two Box objects and returns the resultant Box object.

Rules to overload comparison operator

Overloadable and Non-Overloadable Operators

The following table describes the overload ability of the operators in C#:

Operators	Description
+, -, !, ~, ++, --	These unary operators take one operand and can be overloaded.
+, -, *, /, %	These binary operators take one operand and can be overloaded.
==, !=, <, >, <=, >=	The comparison operators can be overloaded
&&, ||	The conditional logical operators cannot be overloaded directly.
+=, -=, *=, /=, %=	The assignment operators cannot be overloaded.
=, ., ?:, ->, new, is, sizeof, typeof	These operators cannot be overloaded.

User - Defined Casts

C# allows two different types of casts : **Implicit** and **Explicit**

```
int I = 3;
long l = I; // implicit
short s = (short)I; // explicit
```

Explicit Casts are required where there is a risk that the cast might fail or some data might be lost. The following are some examples:

1. When converting from an int to a short , the short might not be large enough to hold the value of the int .

2. When converting from signed to unsigned data types, incorrect results will be returned if the signed variable holds a negative value.
3. When converting from floating - point to integer data types, the fractional part of the number will be lost.
4. When converting from a nullable type to a non - nullable type, a value of null will cause an exception.

C# support casts to and from own data types (struct and class)

- define a cast as a member operator of one of the relevant classes
- cast operator must be marked as either implicit or explicit to indicate how you are intending it to be used
- If you know that the cast is always safe whatever the value held by the source variable, then you define it as implicit .
- If, however, you know there is a risk of something going wrong for certain values — perhaps some loss of data or an exception being thrown - then you should define the cast as explicit

```
public static implicit operator float (Currency value)
{
// processing
}
```

- The cast defined here allows to implicitly convert the value of a Currency into a float
- If a conversion has been declared as implicit , the compiler will permit its use either implicitly or explicitly.
- If it has been declared as explicit , the compiler will only permit it to be used explicitly

Program

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MatrixOperationsOverloading
{
    class Matrix
    {
        public int[,] Mat1 = new int[2, 2];

        public int[,] Mat2 = new int[2, 2];

        public float[,] Mat3 = new float[2, 2];

        public int[,] Matadd = new int[2, 2];
        public int[,] Matsub = new int[2, 2];
        public int[,] Matmul = new int[2, 2];
        public float[,] Matdiv = new float[2, 2];

        public static int[,] operator +(Matrix obj, int[,] Mat2)
        {
            int[,] Matx=new int[2,2];
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    Matx[i, j] = obj.Mat1[i,j]+Mat2[i,j];
                }
            }
            return Matx;
        }

        public static int[,] operator -(Matrix obj, int[,] Mat2)
        {
            int[,] Matx = new int[2, 2];
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    Matx[i, j] = obj.Mat1[i, j] - Mat2[i, j];
                }
            }
            return Matx;
        }

        public static int[,] operator *(Matrix obj, int[,] Mat2)
        {
            int[,] Matx = new int[2, 2];
            int c, d, k,sum=0;

            for (c = 0; c < 2; c++)
            {
                for (d = 0; d < 2; d++)
                {
                    for (k = 0; k < 2; k++)
                    {
                        sum = sum + obj.Mat1[c,k] * obj.Mat2[k,d];
```

```
                }

                Matx[c,d] = sum;
                sum = 0;
            }
        }
        return Matx;
    }

    public static float[,] operator /(Matrix obj, int x)
    {
        float[,] Matx = new float[2, 2];
        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 2; j++)
            {
                obj.Mat3[i,j] = (float)obj.Mat1[i,j];
                Matx[i, j] = obj.Mat3[i, j] / x;
            }
        }
        return Matx;
    }

    public static bool operator ==(Matrix obj, int[,] Mat2)
    {
        int count = 0;
        for (int i = 0; i < 2; i++)
        {
            for (int j = 0; j < 2; j++)
            {
                if (obj.Mat1[i, j] == Mat2[i, j]) { count++; }
            }
        }
        if (count == 4) { return true; } else { return false; }
    }

    public static bool operator !=(Matrix obj, int[,] Mat2)
    {
        return !(obj==Mat2);
    }

    public static implicit operator float(Matrix obj)
    {
        float f = 0;
        for(int i = 0;i< 2;i++)
        {
            for(int j=0;j<2;j++)
            {
                f = f + obj.Mat1[i, j];
            }
        }
        return f;
    }

    static void Main(string[] args)
    {
        Matrix matobj = new Matrix();
```

```
            matobj.Initialisation();
            matobj.SetValues(matobj.Mat1);
            matobj.SetValues(matobj.Mat2);
            Console.WriteLine("Ist Matrix is\n");
            matobj.GetValues(matobj.Mat1);
            Console.WriteLine("IInd Matrix is\n");
            matobj.GetValues(matobj.Mat2);
            matobj.Matadd = matobj + matobj.Mat2;
            Console.WriteLine("After Addition");
            matobj.GetValues(matobj.Matadd);
            matobj.Matadd = matobj - matobj.Mat2;
            Console.WriteLine("After Subtraction");
            matobj.GetValues(matobj.Matadd);
            matobj.Matadd = matobj * matobj.Mat2;
            Console.WriteLine("After Multiplication");
            matobj.GetValues(matobj.Matadd);
            matobj.Matdiv = matobj / 2;
            Console.WriteLine("After Division by 2");
            matobj.GetValues(matobj.Matdiv);
            bool check = matobj == matobj.Mat2;
            Console.WriteLine("After Checking Equality:");
            if (check) { Console.WriteLine("Matrices are equal"); } else {
Console.WriteLine("Matrices are not equal"); }
            float f = matobj;
            Console.WriteLine("The sum of matrix elements" + f);
            Console.WriteLine("_________________________________");
        }

        public void Initialisation()
        {
            Console.WriteLine("\n*___*___*___*___*___***__Welcome to the Matrix
Multiplication Program__***___*___*___*___*___*\n\n");
            Console.WriteLine("\t\t\t\t2*2 Matrix Operations");
            Console.WriteLine("\t\t\t_________________________________\n");
        }

        public void SetValues(int[,] Mat3)
        {
            Console.WriteLine("Enter the First Matrix Values?\n");
            for (int i = 0; i < 2; i++)
            {
                for (int j = 0; j < 2; j++)
                {
                    Console.WriteLine("Enter the [{0}][{1}]",i,j);
                    Mat3[i,j] = Convert.ToInt32(Console.ReadLine());
                }
            }
            Console.WriteLine("_________________________________");
        }

        public void GetValues(int[,] array) {
            for (int x = 0; x < array.GetLength(0); x += 1)
            {
                for (int y = 0; y < array.GetLength(1); y += 1)
                {
                    Console.Write(array[x, y]+"\t");
                }
```

```
                Console.WriteLine();
            }
            Console.WriteLine("_______________________________");
        }

        public void GetValues(float[,] array)
        {
            for (int x = 0; x < array.GetLength(0); x += 1)
            {
                for (int y = 0; y < array.GetLength(1); y += 1)
                {
                    Console.Write(array[x, y] + "\t");
                }
                Console.WriteLine();
            }
            Console.WriteLine("_______________________________");
        }

    }
}
```

Output

```
C:\WINDOWS\system32\cmd.exe

                    2*2 Matrix Operations
                    ----------------------------------

Enter the First Matrix Values?

Enter the [0][0]
1
Enter the [0][1]
2
Enter the [1][0]
3
Enter the [1][1]
4
--------------------------------------
Enter the First Matrix Values?

Enter the [0][0]
5
Enter the [0][1]
6
Enter the [1][0]
7
Enter the [1][1]
8
--------------------------------------
Ist Matrix is

1        2
3        4
--------------------------------------
IInd Matrix is

5        6
7        8
--------------------------------------
After Addition
6        8
10       12
--------------------------------------
After Subtraction
-4       -4
-4       -4
--------------------------------------
After Multiplication
19       22
43       50
--------------------------------------
After Division by 2
0.5      1
1.5      2
--------------------------------------
After Checking Equality:
Matrices are not equal
The sum of matrix elements10
--------------------------------------
```

Result

The above program for operating overloading is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**

Ex. No. 3	Delegates and Events		
Date of Exercise	03.08.2016	Date of Upload	19.10.2016

Aim

To develop **Subject Registration System** using C# by including the concept of Delegates and Events in order to perform various functions such as Subject selection, staff selection and time table generation.

Description

Delegates are nothing but the function pointers so as to:

- to pass methods around to other methods
- a delegate contains the address of a method

There **are 4 methods** associated with it:

1. **Delegate declaration**

- Delegate derived from System.Delegate

2. **Delegate methods definition**

- Any function whose signature matches the delegate signature

3. **Delegate instance creation**

- Hold reference to delegate method

4. **Delegate invocation**

- Invoke the method indirectly

Declaring and using delegates:

Syntax

```
[access specifier] delegate returntype delegatename(paramaters);
```

Example

```
delegate void SimpleDelegate();
public delegate void MathOperation(int x, inty);
```

Multicast delegates:

- Each method wrap just one single method call
- To call more methods, create more delegates explicitly
- It is possible for a delegate to wrap more than one method: multicast delegate
- Calling multicast delegate call successive methods wrapped on it
- Delegate signature is void or only get result of last method invoked
- A multicast delegate is a class derived from System.MulticastDelegate , which in turn is derived from System.Delegate

Example

```
class MathOperations
{
        public static void MultiplyByTwo(double value)
        {
        double result = value * 2;
        Console.WriteLine("Multiplying by 2: {0} gives {1}", value, result);
        }
        public static void Square(double value)
        {
        double result = value * value;
        Console.WriteLine("Squaring: {0} gives {1}", value, result);
        }
}
delegate void DoubleOp(double value);
class MainEntryPoint
{
static void Main()
{
DoubleOp operations = MathOperations.MultiplyByTwo;
operations += MathOperations.Square;
DoubleOp operation1 = MathOperations.MultiplyByTwo;
DoubleOp operation2 = MathOperations.Square;
```

```
DoubleOp operations = operation1 + operation2; //another way
operations(2.0); //inturn call MultiplyByTwo then Square method
operations(7.94);
operations(1.414);
}
}
```

Array of delegates:

The Delegate class defines the method **GetInvocationList()** that returns an array of Delegate objects. Using this we can invoke the methods associated with them directly, catch exceptions, and continue with the next iteration.

Example

```
static void Main()
{
DemoDelegate d1 = Program.One;
d1 += Two;
Delegate[] delegates = d1.GetInvocationList();
foreach(DemoDelegate d in delegates){
try{d();}catch (Exception){Console.WriteLine("Error in one");}
}
}
```

Events are user actions such as key press, clicks, mouse movements, etc., or some occurrence such as system generated notifications. Applications need to respond to events when they occur. For example, interrupts. Events are used for inter-process communication.

Steps in event:

- Event handler method (delegate method definition)
- Delegate declaration
- Event declaration
- Event instance creation (event binding)
- Event invocation

Event is a delegate type class member that is used by an object or a class to provide a notification to other objects that an event has occurred

```
[access modifier] event delegatetype event-name;
```

Define a method to handle the event and bind this to the event using += operator

```
event-name +=new delegatetype(method-name);
```

To remove a source of events, use -= operator

```
event-name -=new delegatetype(method-name);
```

Event invocation

```
event-name();
```

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;

namespace SubjectRegistrationSystem
{

    class SubjectRegistration {
        public string SubjectCode;
        public int SubjectCapacity;
        public string SubjectCategory;
        public int SubjectCredit;
        public string SubjectName;
        public string SubjectStatus;
        public string GroupCode;
    }
    class UniversityStudents{
        public string StudentName;
        public string StudentId;
    }
    class RegisteredSubject {
        public string SubjectCode;
        public string StudentId;
    }

    //delegate for initialisation all meta lists
    delegate void delegatemethodforinit(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects);

    //delegate for function having subject list
    delegate void delegatemethodforsubjects(List<SubjectRegistration> subjects);


    //delegate for function having student list
    delegate int delegatemethodforstudents(List<UniversityStudents> student);
```

```
    //Event-Delegate declaration
    delegate int ValueChangedEventHandler(List<UniversityStudents> student);//delegate
declaration

    class Program
    {
        public event ValueChangedEventHandler Changed;//event declaration

            public  void Handle() {
            Console.WriteLine("");
        }

        static void Main(string[] args)
        {

            int index;
            List<SubjectRegistration> subjects = new List<SubjectRegistration>();
            List<UniversityStudents> student = new List<UniversityStudents>();
            List<RegisteredSubject> regsubjects = new List<RegisteredSubject>();

            Program pro = new Program();
            //defining method to handle the event
            pro.Changed += pro.initialisestudent;
            //Event Invocation
            index = pro.Changed(student);

            delegatemethodforinit menu = pro.initialisemenu;

            delegatemethodforinit regsub = pro.registersubject;

            delegatemethodforinit deregsub = pro.deregistersubject;

            delegatemethodforinit gentime = pro.generatetimetable;


            delegatemethodforsubjects initsubject = pro.initialisevalue;

            initsubject(subjects);

            menu(subjects, index, student, regsubjects);

        }

        public  void initialisemenu(List<SubjectRegistration> subjects, int studentindex,
List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {
            Console.WriteLine("1.Register Subject");
            Console.WriteLine("2.Deregister Subject");
            Console.WriteLine("3.Faculty Selection");
            Console.WriteLine("4.Generate Timetable");
            Console.WriteLine("5.Logout the Program");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice) {
                case 1:
                    registersubject(subjects, studentindex,student,regsubjects);
                    break;
                case 2:
                    deregistersubject(subjects, studentindex, student, regsubjects);
```

```
                    break;
                case 3:
                    break;
                case 4:
                    generatetimetable(subjects, studentindex, student, regsubjects);

                    break;
                case 5:
                    studentindex= initialisestudent(student);
                    initialisemenu(subjects, studentindex, student,regsubjects);
                    break;
                default:
                    Console.WriteLine("Invalid Choice");
                    break;
            }

        }

        public  void registersubject(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student,List<RegisteredSubject> regsubjects) {
            displaysubject(subjects);
            Console.WriteLine("Enter the Subject Code to Register");
            string Subcode = Console.ReadLine();
            if (Subcode.Equals("N")) { initialisemenu(subjects, studentindex, student,
regsubjects); };
            Subcode = Subcode.ToUpper();

            string sid = student[studentindex].StudentId;
            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode==Subcode &&
s.StudentId==sid);
            if (checkval == -1) {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].StudentName);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity--;
                        Console.WriteLine("You are successfully registered for " +
subjects[index].SubjectName);
                        regsubjects.Add(new RegisteredSubject() { SubjectCode = Subcode,
StudentId = sid });
                        registersubject(subjects, studentindex, student, regsubjects);

                    }
                    else
                    {
                        Console.WriteLine("No seats are further available :( ");
                        Console.WriteLine("Please try again :) ");
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
                {
```

```
                    Console.WriteLine("Sorry!! :( The subcode is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Already registered");
                Console.WriteLine("Please try again :) ");
                initialisemenu(subjects, studentindex, student, regsubjects);
            }



        }

        public  void deregistersubject(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {

            string sid = student[studentindex].StudentId;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("You have registered for the following subjects");

            foreach (RegisteredSubject subval in subs)
            {
                String subcodeval = subval.SubjectCode;
                List<SubjectRegistration> subject1 =
subjects.FindAll(a=>a.SubjectCode==subcodeval);
                displaysubject(subject1);
            }

            Console.WriteLine("Enter the Subject Code to Deregister");
            string Subcode = Console.ReadLine();
            Subcode = Subcode.ToUpper();

            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval != -1)
            {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].StudentName);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity++;
                        Console.WriteLine("You are successfully \"DEREGISTERED\" for " +
subjects[index].SubjectName);
                        regsubjects.RemoveAll(a => a.SubjectCode == Subcode &&
a.StudentId == sid);
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
```

```
                {
                    Console.WriteLine("Sorry!! :( The subject code is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Not Found"); }

        }

        public  void generatetimetable(List<SubjectRegistration> subjects, int
studentindex, List<UniversityStudents> student, List<RegisteredSubject> regsubjects) {

            string sid = student[studentindex].StudentId;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("_________________Time Table_________________________");

Console.WriteLine("Hour|||\tMonday|||\tTuesday|||\tWednesday|||\tThursday|||\tFriday|||")
;
            foreach (RegisteredSubject subval in subs) {
                String subcodeval = subval.SubjectCode;
                int indexvalof = subjects.FindIndex(a => a.SubjectCode == subcodeval);
                String groupval=subjects[indexvalof].GroupCode;
                //Console.WriteLine("Group:" + groupval);
                if (groupval.Equals("A"))
                {
                    Console.WriteLine();
                    Console.Write("1\t");
                    for (int i = 0; i < 4; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }
                    Console.WriteLine();
                }
                else if (groupval.Equals("B")) {
                    Console.WriteLine();
                    Console.Write("2\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else if (groupval.Equals("C"))
                {
                    Console.WriteLine();
                    Console.Write("3\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
```

```
                else if (groupval.Equals("D"))
                {
                    Console.WriteLine();
                    Console.Write("4\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else if (groupval.Equals("E"))
                {
                    Console.WriteLine();
                    Console.Write("5\t");
                    for (int i = 0; i < 3; i++)
                    {
                        Console.Write(subcodeval + "\t");
                    }

                    Console.WriteLine();
                }
                else
                {
                    Console.WriteLine("_____________You are free___________");
                }

            }
            initialisemenu(subjects, studentindex, student, regsubjects);
        }


        public  int initialisestudent(List<UniversityStudents> student) {
            Console.WriteLine("Welcome to Subject Registration System  :) ");
            Console.WriteLine("Enter your name ?");
            string Name = Console.ReadLine().ToUpper();
            Console.WriteLine("Enter your Student ID ?");
            string Id = Console.ReadLine().ToUpper();
            int index = -1;
            index = student.FindIndex(a => a.StudentId == Id);
            if (index < 0)
            {
                student.Add(new UniversityStudents() { StudentName = Name, StudentId = Id
});
            }
            index = student.FindIndex(a => a.StudentId == Id);
            return index;

        }

        public  void initialisevalue(List<SubjectRegistration> subjects)
        {
            subjects.Add(new SubjectRegistration() { SubjectName = "C# PROGRM",
SubjectCode = "14CS2054", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 4, SubjectStatus = "Available", GroupCode = "A"});
```

```
            subjects.Add(new SubjectRegistration() { SubjectName = "STORAGEAN",
SubjectCode = "14CS2065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "B"});
            subjects.Add(new SubjectRegistration() { SubjectName = "BIG _DATA",
SubjectCode = "14CS3065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "C"});
            subjects.Add(new SubjectRegistration() { SubjectName = "INTEROFTH",
SubjectCode = "16EC2002", SubjectCategory = "FREEELEC", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "D"});
            subjects.Add(new SubjectRegistration() { SubjectName = "DATASTRUC",
SubjectCode = "14CS2009", SubjectCategory = "____CORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "E"});
        }

        public  void displaysubject(List<SubjectRegistration> subjects) {

Console.WriteLine("\n_____________________________________________________________________
_______________");
            Console.WriteLine("SUB_CODE\tNAME\t\tCREDIT\tCATEGORY\tCAPACITY\tSTATUS");
            Console.WriteLine("-------------------------------------------------------------
------------------------");
            foreach (SubjectRegistration subval in subjects)
            {
                Console.WriteLine("{0}\t{1}  \t[{2}]\t{3}\t{4}\t\t{5}",
subval.SubjectCode, subval.SubjectName, subval.SubjectCredit, subval.SubjectCategory,
subval.SubjectCapacity, subval.SubjectStatus);

            }
        }


    }
}
```

Output

- **Register Subject**

```
C:\WINDOWS\system32\cmd.exe
Welcome to Subject Registration System  :>
Enter your name ?
suhaas
Enter your Student ID ?
ur13cs043
1.Register Subject
2.Deregister Subject
3.Faculty Selection
4.Generate Timetable
5.Logout the Program
1

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        5               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        5               Available
Enter the Subject Code to Register
14cs2054
DearSUHAAS
You are successfully registered for C# PROGRM

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        5               Available
Enter the Subject Code to Register
14cs2009
DearSUHAAS
You are successfully registered for DATASTRUC

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available
14CS2065        STORAGEAN       [3]     SOFTCORE        5               Available
14CS3065        BIG _DATA       [3]     SOFTCORE        5               Available
16EC2002        INTEROFTH       [3]     FREEELEC        5               Available
14CS2009        DATASTRUC       [3]     ____CORE        4               Available
Enter the Subject Code to Register
```

- **Timetable Generation**

```
5.Logout the Program
4
_________________Time Table_____________________
Hour!!! Monday!!!       Tuesday!!!      Wednesday!!!    Thursday!!!     Friday!!!

1       14CS2054        14CS2054        14CS2054        14CS2054

5       14CS2009        14CS2009        14CS2009
```

- **Deregister Subjects**

```
2
You have registered for the following subjects

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
-----------------------------------------------------------------------------------
14CS2009        DATASTRUC       [3]     ____CORE        4               Available
Enter the Subject Code to Deregister
14cs2054
DearSUHAAS
You are successfully "DEREGISTERED" for C# PROGRM
```

- **Log Out**

```
5
Welcome to Subject Registration System  :)
Enter your name ?
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]

Ex. No. 4	String Manipulation and Regular Expression		
Date of Exercise	03.10.2016	Date of Upload	27.10.2016

Aim

To develop **Employee Information entry system** using C# by including the concept of various string functions and regular expressions.

Description

A **regular expression** is a pattern that could be matched against an input text. The .Net framework provides a regular expression engine that allows such matching. A pattern consists of one or more character literals, operators, or constructs.

The Regex Class

The Regex class is used for representing a regular expression. It has the following commonly used methods:

Sr.no	Methods
1	**public bool IsMatch(string input)** Indicates whether the regular expression specified in the Regex constructor finds a match in a specified input string.
2	**public bool IsMatch(string input, int startat)** Indicates whether the regular expression specified in the Regex constructor finds a match in the specified input string, beginning at the specified starting position in the string.
3	**public static bool IsMatch(string input, string pattern)** Indicates whether the specified regular expression finds a match in the specified input string.
4	**public MatchCollection Matches(string input)**

	Searches the specified input string for all occurrences of a regular expression.
5	**public string Replace(string input, string replacement)** In a specified input string, replaces all strings that match a regular expression pattern with a specified replacement string.
6	**public string[] Split(string input)** Splits an input string into an array of substrings at the positions defined by a regular expression pattern specified in the Regex constructor.

It contains two features:

- A set of escape codes for identifying specific types of characters.
- A system for grouping parts of substrings and intermediate results during a search operation

Instantiate a System.Text.RegularExpressions.RegEx object, pass it the string to be processed, and pass in a regular expression.

With regular expressions, perform quite sophisticated and high - level operations on strings. For example,

- Identify all repeated words in a string
- Convert all words to title case
- Convert all words longer than three characters to title case
- Ensure that sentences are properly capitalized
- Separate the various elements of a URI

A regular expression string looks at first sight rather like a regular string, but interspersed with escape sequences and other characters that have a special meaning.

- the sequence \b indicates the beginning or end of a word
- to search for all occurrences of th at the end of a word, you would write th\b

Example

```
String text = “Here is a text!”;
Regex regExp = new Regex(@”\b[a-z]+\b”);
```

- + for one or more information
- * for o or more information

```
MatchCollection matches = regExp.Matches(text);
```

- Returns the matches in text with RE

```
foreach(Match m in matches)
{
if(m.Length!=0)
{ Console.WriteLine(m); }
}
```

The following table lists some of the main special characters or escape sequences that you can use. It is not comprehensive, but a fuller list is available in the MSDN documentation.

SYMBOL	MEANING	EXAMPLE	MATCHES
^	Beginning of input text	^B	B, but only if first character in text
$	End of input text	X$	X, but only if last character in text
.	Any single character except the newline character (\ )	i.ation	isation, ization
*	Preceding character may be repeated zero or more times	ra*t	rt, rat, raat, raaat, and so on
+	Preceding character may be repeated one or more times	ra+t	rat, raat, raaat and so on, but not rt
?	Preceding character may be repeated zero or one time	ra?t	rt and rat only
\s	Any whitespace character	\sa	[space]a, \ta, \na (\t and \n have the same meanings as in C#)
\S	Any character that isn't whitespace	\SF	aF, rF, cF, but not \tf
\b	Word boundary	ion\b	Any word ending in ion
\B	Any position that isn't a word boundary	\BX\B	Any X in the middle of a word

An example of this is **http://www.wrox.com:4355**

```
\b(\S+)://(\S+)(?::(\S+))?\b
```

- The first group, (\S+):// , identifies one or more characters that don ’ t count as whitespace, and that are followed by :// i.e http://
- The subsequent (\S+) identifies the string www.wrox.com in the URI
- The next group identifies the port (:4355)
- ? indicates that this group is optional in the match

Program

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;

namespace Employee_Info_Entry
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("************************Welcome to Employee Information
Entry System********************");
            //  Readfromfile();
            Initialize_menu();
        }

        public static void Initialize_menu()
        {
            const string namepattern = @"^[A-Z][a-z]+$", numberpattern = @"\+\S+";
            const string mailpattern = @"(\S+)\@(\S+)\.(\S+)$", urlpattern =
@"\b(\S+)://([^:]+)(?:/(\S+))?(?::(\S+))?\b";
            const string userpattern = @"\$\S*\$", pwdpattern = @"\@\S*\@";
            while (true) {

Console.WriteLine("_____________________________________________________________________
_________");

Console.WriteLine("|_________________________________*MENU*_____________________________
_________|");
                Console.WriteLine("1.Get all Phone Numbers");
                Console.WriteLine("2.Get all Mail ID's");
                Console.WriteLine("3.Get all URL's");
                Console.WriteLine("4.Get all Usernames & Passwords");
                Console.WriteLine("5.Get all Names");
                Console.WriteLine("6.View Database");
                Console.WriteLine("7.Terminate the Program");
                int choice = Convert.ToInt32(Console.ReadLine());
                switch (choice)
                {
                    case 1:
                        Console.WriteLine("NUMBERS");
                        ReturnData(numberpattern);
                        break;
```

```
                    case 2:
                        Console.WriteLine("MAIL ID's");
                        ReturnData(mailpattern);
                        break;

                    case 3:
                        Console.WriteLine("URL'S");
                        ReturnData(urlpattern);
                        break;

                    case 4:
                        Console.WriteLine("Usernames\tPassword");
                        ReturnData(userpattern, pwdpattern);
                        break;

                    case 5:
                        Console.WriteLine("FNAME\tLASTNAME");
                        ReturnData(namepattern);
                        break;

                    case 6:
                        Console.WriteLine("FNAME\tLNAME\tMNO\t\tMail
Id's\t\tURL\t\t\tUsername    Pwd");
                        ReturnData(namepattern, numberpattern, mailpattern, urlpattern,
userpattern, pwdpattern);
                        break;

                    case 7:
                        Console.WriteLine("The program is terminated");
                        Environment.Exit(0);
                        break;
                    default:
                        Console.WriteLine("You have entered an Invalid Choice :( ");
                        Console.WriteLine("Please try again");
                        break;
                }
            }
        }

        public static void ReturnData(params string[] pattern) {
            try
            {
                // Create an instance of StreamReader to read from a file.
                // The using statement also closes the StreamReader.
                using (StreamReader sr = new
StreamReader("C:/Users/chinnu/Documents/Visual Studio
2015/Projects/CSharpLab/4.Strings_AND_RegEx[Employee info Entry
Sys]/Employee_Info_Entry/Employee.txt"))
                {
```

```
                    string line;

                    // Read and display lines from the file until
                    // the end of the file is reached.
                    while ((line = sr.ReadLine()) != null)
                    {
                        char tabdelem = '\t';
                        String[] splitfileds = line.Split(tabdelem);
                        foreach (string fieldval in splitfileds)
                        {
                            foreach (string currentpattern in pattern) {
                                if ((Regex.Match(fieldval, currentpattern)).Success)
                                {
                                    Console.Write(fieldval);
                                    if (pattern.Count() < 5)
                                    {
                                        Console.Write("\t");
                                    }
                                    else {
                                        Console.Write("  ");
                                    }
                                }

                            }
                        }
                        Console.Write("\n");
                     }
                }
            }
            catch (Exception e)
            {

                // Let the user know what went wrong.
                Console.WriteLine("The file could not be read:");
                Console.WriteLine(e.Message);
            }

        }

    }
}
```

Output

Phone Numbers

```
C:\WINDOWS\system32\cmd.exe
**************************Welcome to Employee Information Entry System**********************
¦____________________________________*MENU*____________________________________¦
1.Get all Phone Numbers
2.Get all Mail ID's
3.Get all URL's
4.Get all Usernames & Passwords
5.Get all Names
6.View Database
7.Terminate the Program
1
NUMBERS
+917708488989
+919004864898
```

Mail Id's

```
2
MAIL ID's
Michal99@gmail.com
Guijue@gmail.com
```

URL's

```
URL'S
https://git.com/broken-pot
https://www.karunya.edu:455
```

Usernames

```
4
Usernames        Password
$nuiasmoi$       @Mypwd@
$snebidhi$       @merapwd@
```

Username's and Password's

```
5
FNAME   LASTNAME
Michal  Bindhi
Guijue  Jeorge
```

First and Last Name's

```
6
FNAME   LNAME   MNO            Mail Id's              URL                      Username    Pwd
Michal  Bindhi  +917708488989  Michal99@gmail.com  https://git.com/broken-pot  $nuiasmoi$  @Mypwd@

Guijue  Jeorge  +919004864898  Guijue@gmail.com  https://www.karunya.edu:455  $snebidhi$  @merapwd@
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. J Anitha /Dr. S.P. Jeno Lovesum**]

Ex. No. 5	Exception Handling		
Date of Exercise	17.08.2016	Date of Upload	21.10.2016

Aim

To develop **Integer Stack Application** using C# by handling various catching various exceptions relating to size and format specification thereby handling them.

Description

An exception is a problem that arises during the execution of a program. A C# exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.

Exceptions provide a way to transfer control from one part of a program to another. C# exception handling is built upon four keywords: **try**, **catch**, **finally**, and **throw.**

- **try:** A try block identifies a block of code for which particular exceptions is activated. It is followed by one or more catch blocks.
- **catch:** A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
- **finally:** The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.
- **throw:** A program throws an exception when a problem shows up. This is done using a throw keyword.

Syntax:

```
try
{// statements causing exception}
catch( ExceptionName e1 )
{// error handling code}
catch( ExceptionName e2 )
{// error handling code}
```

```
catch( ExceptionName eN )
{// error handling code}
finally
{// statements to be executed}
```

Exception Classes in C#

C# exceptions are represented by classes. The exception classes in C# are mainly directly or indirectly derived from the **"System.Exception"** class. Some of the exception classes derived from the System.Exception class are the **"System.ApplicationException"** and **"System. SystemException"** classes. The "**System.ApplicationException**" class supports exceptions generated by application programs. Hence the exceptions defined by the programmers should derive from this class. The "**System.SystemException"** class is the base class for all predefined system exception.

The following table provides some of the predefined exception classes derived from the **Sytem.SystemException class**:

Exception Class	Description
System.IO.IOException	Handles I/O errors.
System.IndexOutOfRangeException	Handles errors generated when a method refers to an array index out of range.
System.ArrayTypeMismatchException	Handles errors generated when type is mismatched with the array type.
System.NullReferenceException	Handles errors generated from deferencing a null object.
System.DivideByZeroException	Handles errors generated from dividing a dividend with zero.

System.InvalidCastException	Handles errors generated during typecasting.
System.OutOfMemoryException	Handles errors generated from insufficient free memory.
System.StackOverflowException	Handles errors generated from stack overflow.

User - Defined Exception Classes

There are two types of exceptions:

- exceptions generated by an executing program
- exceptions generated by the CLR

The CLR throws **SystemException.**

The **ApplicationException** is thrown by a user program rather than the runtime where it is possible to create our own exception class. Exception must be the ultimate base class for all exceptions in C#. User-defined exception classes must inherit from either Exception class or one of its standard derived classes

Example

```
using System;
class MyException : ApplicationException
{ public MyException(string str) //constructor
{ Console.WriteLine ("User Defined Exception"); }
}
class MyClient
{
public static void Main()
{
try
{
throw new MyException ("Rajesh");
}
```

```
catch(MyException e)
{
Console.WriteLine ("Exception caught here" + e.ToString ());
}
Console.WriteLine("Last Statement");
}
}
```

Program

```
using System;
using System.Collections.Generic;

namespace Integer_Stack_Exception_Handling_5
{
    //To raise an Exception if the input in non-integer
    class NonInteger_Exception:ApplicationException {

        public NonInteger_Exception(String msg) {
            Console.WriteLine("\n--------------------EXCEPTION-------------------------
----");
            Console.WriteLine("|-->Main:The \'FORMAT\' of the input should be integer\n|-
->Msg: {0}",msg);
            Console.WriteLine("|-->Fix: Please try to insert integer value");
            Console.WriteLine("-------------------------------------------------------
----");
        }
    }
    //To raise an Exception if the input if Stack is empty
    class StackEmpty_Exception : ApplicationException {
        public StackEmpty_Exception(String msg)
        {
            Console.WriteLine("\n--------------------EXCEPTION-------------------------
----");
            Console.WriteLine("|-->Main:For \'POP\' or \'DISPLAY\' the \'SIZE\' of the
stack should be > 0\n|-->Msg: {0}", msg);
            Console.WriteLine("|-->Fix: Please try to \'PUSH\' values");
            Console.WriteLine("-------------------------------------------------------
----");
        }
    }

    class Program
    {
        //initialize the size of stack(default is 5)
        static int Stack_Size = 5;
        //Main Control Method
        static void Main(string[] args)
        {
            //Design
            Console.WriteLine("**********************WELCOME User to operate INTEGER
STACK [Deafult size->5]*******************");
            Console.WriteLine();
            //Initialising Starting default capacity to 5
            Stack<int> stackinteger = new Stack<int>(5);
            //Calling Menu Function
            Initialize_Menu(stackinteger);
```

```
        }
        //To Display Menu
        public static void Initialize_Menu(Stack<int> stackinteger)
        {

Console.WriteLine("_____________________________________________________________________
_________");

Console.WriteLine("|_________________________________*MENU*_____________________________
_________|");
            Console.WriteLine("1.Push\n2.Pop\n3.Peek\n4.Display\n5.IncrementStack
Size\n6.Clear\n7.Aggregate Functions\n8.Exit");

            try
            {
                string value = Console.ReadLine();
                int choice;
                //to check if the input is integer
                if (int.TryParse(value, out choice))
                {

                switch (choice)
                {
                    case 1:
                        Push_Integer(stackinteger);
                        break;
                    case 2:
                        Pop_Integer(stackinteger);
                        break;
                    case 3:
                        Peek_Integer(stackinteger);
                        break;
                    case 4:
                        Display_Integer(stackinteger);
                        break;
                    case 5:
                        Increment_Stack_Size(stackinteger);
                        break;
                    case 6:
                        Clear_Integer(stackinteger);
                        break;

                    case 8:
                        Console.WriteLine("The program is terminated");
                        Environment.Exit(0);
                        break;

                    default:
                        Console.WriteLine("Invalid Choice!\nPlease Try again :(");
                        Initialize_Menu(stackinteger);
```

```
                        break;
                    }

                }
                else
                {
                    throw new NonInteger_Exception("Enter a Valid Input");
                }
            }
            catch (NonInteger_Exception formatexception)
            {
                Console.WriteLine(formatexception.Message);
            }
            finally
            {
                Initialize_Menu(stackinteger);
            }
        }
        //To Increment Stack Size
        public static void Increment_Stack_Size(Stack<int> stackinteger)
        {
            Console.WriteLine("Current Size is {0}", Stack_Size);
            Console.WriteLine("New Size should be greater than the Current Size");
            Console.WriteLine("Enter the new Stack Size ?");
            try
            {
                string value = Console.ReadLine();
                int intvalue;
                //to check if the input is integer
                if (int.TryParse(value, out intvalue))
                {
                    //Exception if the user tries to decrement the stack size
                    if (intvalue <= Stack_Size) { throw new
InvalidOperationException("Lesser Size or equal Size is entered"); }
                    else { Stack_Size = intvalue; }
                }
                else
                {
                    throw new NonInteger_Exception("Not an Integer");
                }
            }
            catch (NonInteger_Exception formatexception)
            {
                Console.WriteLine(formatexception.Message);
            }
            catch (InvalidOperationException invalidsize) {
                Console.WriteLine("\n--------------------EXCEPTION---------------------
--------");
                Console.WriteLine("|-->Main:The \'SIZE\' of the stack entered should be
greater than {0}\n|-->Msg: {1}", Stack_Size, invalidsize.Message);
```

```
                Console.WriteLine("|-->Fix: Please Enter greater size to increment");
                Console.WriteLine("-------------------------------------------------------
--------");
            }
            finally
            {
                Initialize_Menu(stackinteger);
            }
        }
        //To Push an Integer
        public static void Push_Integer(Stack<int> stackinteger) {
            try
            {
                //To check if it reached the max size limit
                if (stackinteger.Count == Stack_Size)
                {
                    throw new StackOverflowException();
                }
                else
                {
                    Console.WriteLine("Enter the integer value ?");
                    //reading the value
                    string value = Console.ReadLine();
                    int intvalue;
                    //to check if the input is integer
                    if (int.TryParse(value, out intvalue))
                    {
                        stackinteger.Push(intvalue); Display_Integer(stackinteger);
                    }
                    else
                    {
                        throw new NonInteger_Exception("Not an Integer");
                    }
                }
            }
            catch (StackOverflowException sizeexception)
            {
                Console.WriteLine("\n---------------------EXCEPTION---------------------
--------");
                Console.WriteLine("|-->Main:The \'SIZE\' of the stack cannot be greater
than {0}\n|-->Msg: {1}", Stack_Size, sizeexception.Message);
                Console.WriteLine("|-->Fix: Please POP the elements");
                Console.WriteLine("-------------------------------------------------------
--------");
            }
            catch (NonInteger_Exception formatexception) {
                Console.WriteLine(formatexception.Message);
            }
            finally
            {
```

```
            Initialize_Menu(stackinteger);
        }

    }
    //To Pop an the top element
    public static void Pop_Integer(Stack<int> stackinteger) {
        try
        {
            //if the stack has no elements
            if (stackinteger.Count == 0)
            {
                throw new StackEmpty_Exception("Stack is empty");
            }
            else {
                Console.WriteLine("The Popped Value is : " + stackinteger.Pop());
            }
        }
        catch(StackEmpty_Exception stackempty) {
            Console.WriteLine(stackempty.Message);
        }
        finally
        {
            Initialize_Menu(stackinteger);
        }
    }
    //To Peek the top element
    public static void Peek_Integer(Stack<int> stackinteger)
    {
        Console.WriteLine("The Peeked Value is : " + stackinteger.Peek());
        Initialize_Menu(stackinteger);
    }
    //To Display the elements in a Stack
    public static void Display_Integer(Stack<int> stackinteger)
    {
        try
        {
            //if the stack has no elements
            if (stackinteger.Count == 0)
            {
                throw new StackEmpty_Exception("Stack is empty");
            }
            else
            {
                Console.WriteLine("The stack elements are : ");
                Console.Write("[ ");
                foreach (int val in stackinteger)
                {
                    Console.Write(val + ", ");
                }
                Console.WriteLine("]");
```

```
                }
            }
            catch (StackEmpty_Exception stackempty)
            {
                Console.WriteLine(stackempty.Message);
            }
            finally
            {
                Initialize_Menu(stackinteger);
            }
        }
        //To Clear the Stack elements
        public static void Clear_Integer(Stack<int> stackinteger) {
            Console.WriteLine("After clearing the Stack :");
            stackinteger.Clear();
            Display_Integer(stackinteger);
        }

    }

}
```

Output

Invalid Input

```
**********************WELCOME User to operate INTEGER STACK [Deafult size->5]*******************

_____________________________________________________________________________________
¦_____________________________*MENU*_____________________________________________¦
1.Push
2.Pop
3.Peek
4.Display
5.IncrementStack Size
6.Clear
7.Aggregate Functions
8.Exit
q

---------------------EXCEPTION-----------------------------
¦-->Main:The 'FORMAT' of the input should be integer
¦-->Msg: Enter a Valid Input
¦-->Fix: Please try to insert integer value
-----------------------------------------------------------
Error in the application.
```

Non-Integer Value

```
Enter the integer value ?
q

---------------------EXCEPTION-----------------------------
¦-->Main:The 'FORMAT' of the input should be integer
¦-->Msg: Not an Integer
¦-->Fix: Please try to insert integer value
-----------------------------------------------------------
Error in the application.
```

Max Stack Size limit

```
The stack elements are :
[ 5, 4, 3, 2, 1, ]
_____________________________________________________________________________________
¦_____________________________*MENU*_____________________________________________¦
1.Push
2.Pop
3.Peek
4.Display
5.IncrementStack Size
6.Clear
7.Aggregate Functions
8.Exit
1

---------------------EXCEPTION-----------------------------
¦-->Main:The 'SIZE' of the stack cannot be greater than 5
¦-->Msg: Operation caused a stack overflow.
¦-->Fix: Please POP the elements
-----------------------------------------------------------
```

Stack Empty

```
!_____________________________________*MENU*_____________________________________!
1.Push
2.Pop
3.Peek
4.Display
5.IncrementStack Size
6.Clear
7.Aggregate Functions
8.Exit
2

----------------------EXCEPTION------------------------------
!-->Main:For 'POP' or 'DISPLAY' the 'SIZE' of the stack should be > 0
!-->Msg: Stack is empty
!-->Fix: Please try to 'PUSH' values
-------------------------------------------------------------
Error in the application.
```

Invalid Operation

```
!_____________________________________*MENU*_____________________________________!
1.Push
2.Pop
3.Peek
4.Display
5.IncrementStack Size
6.Clear
7.Aggregate Functions
8.Exit
5
Current Size is 5
New Size should be greater than the Current Size
Enter the new Stack Size ?
3

----------------------EXCEPTION------------------------------
!-->Main:The 'SIZE' of the stack entered should be greater than 5
!-->Msg: Lesser Size or equal Size is entered
!-->Fix: Please Enter greater size to increment
-------------------------------------------------------------
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]

Ex. No. 6	Collections		
Date of Exercise	25.08.2016	Date of Upload	23.10.2016

Aim

To develop **Student Information System** using C# by implementing Lists for storing and retrieving values.

Description

Collection classes are specialized classes for data storage and retrieval. These classes provide support for stacks, queues, lists, and hash tables. Most collection classes implement the same interfaces.

Collection classes serve various purposes, such as allocating memory dynamically to elements and accessing a list of items on the basis of an index etc. These classes create collections of objects of the Object class, which is the base class for all data types in C#.

Various Collection Classes and Their Usage

The following are the various commonly used classes of the **System.Collection** namespace.

Class	Description and Usage
ArrayList	It represents ordered collection of an object that can be **indexed** individually. It is basically an alternative to an array. However, unlike array you can add and remove items from a list at a specified position using an **index** and the array resizes itself automatically. It also allows dynamic memory allocation, adding, searching and sorting items in the list.
Hashtable	It uses a **key** to access the elements in the collection.

	A hash table is used when you need to access elements by using key, and you can identify a useful key value. Each item in the hash table has a **key/value** pair. The key is used to access the items in the collection.
SortedList	It uses a **key** as well as an **index** to access the items in a list. A sorted list is a combination of an array and a hash table. It contains a list of items that can be accessed using a key or an index. If you access items using an index, it is an ArrayList, and if you access items using a key, it is a Hashtable. The collection of items is always sorted by the key value.
Stack	It represents a **last-in, first out** collection of object. It is used when you need a last-in, first-out access of items. When you add an item in the list, it is called **pushing** the item and when you remove it, it is called **popping** the item.
Queue	It represents a **first-in, first out** collection of object. It is used when you need a first-in, first-out access of items. When you add an item in the list, it is called **enqueue** and when you remove an item, it is called **deque**.
BitArray	It represents an array of the **binary representation** using the values 1 and 0. It is used when you need to store the bits but do not know the number of bits in advance. You can access items from the BitArray collection by using an **integer index**, which starts from zero.

- **Collection Initializers**

```
ArrayList objectList = new ArrayList() {1, 2}; //non-generic
List < int > intList = new List < int > () {1, 2}; //generic
```

- **Adding Elements**

```
objectList.Add("object1"); objectList.Add("object2"); //non- generic
intList.Add(1); intList.Add(2); //generic
```

AddRange() method of the List < T > class, add multiple elements to the collection at once

- **Inserting Elements**

```
objectList.Insert (2,"object3");
```

If the index set is larger than the number of elements in the collection, ArgumentOutOfRangeException is thrown.

InsertRange() offers the capability to insert a number of elements

- **Accessing Elements**

It access the elements by using an indexer and passing the item number.

e.g. The first item can be accessed with an index value 0

```
ArrayList a1 = objectList[2];
for (int i = 0; i < objectList.Count; i++) //count property
{
Console.WriteLine(objectList [i]);
}
foreach (ArrayList a in objectList)
{
Console.WriteLine(a);
}
```

- **Removing Elements**

```
objectList.RemoveAt(3);
objectList.Remove("object1");
```

RemoveAt() method remove the item in the given index, whereas Remove() method first searches in the collection to get the index of the item with the IndexOf() method, and then uses the index to remove the item. RemoveRange() removes a number of items from the collection where the first parameter specifies the index where the removal of items should begin and the second parameter specifies the number of items to be removed.

```
int index = 3; int count = 5;
objectList.RemoveRange(index, count);
```

- **Searching**

Various methods used for searching are IndexOf(), LastIndexOf(), FindIndex(), FindLastIndex(), Find(), and FindLast().

IndexOf() requires an object as parameter and returns the index of the item if it is found and return – 1 if the item is not found. It uses the IEquatable interface for comparing the elements

```
int index1 = objectList.IndexOf(object1);
```

FindIndex(), Find(), FindAll(), FindLast() requires a parameter of type Predicate :

```
int index3 = racers.FindIndex(r = > r.Country == "Finland");
```

- **Type Conversion**

List<T> method ConvertAll() , all types of a collection can be converted to a different type

e.g.

```
List < Person > persons = racers.ConvertAll < Person > (
r = > new Person(r.FirstName + " " +r.LastName));
```

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Student_Information_System_Collections
{
    class SubjectRegistration
    {
        public string SubjectCode;
        public int SubjectCapacity;
        public string SubjectCategory;
        public int SubjectCredit;
        public string SubjectName;
        public string SubjectStatus;
        public string GroupCode;
    }

    class RegisteredSubject
    {
        public string SubjectCode;
        public string StudentId;
    }

    class Students_Details {
        public string Student_Name;
        public string Student_Regno;
        public string Student_Branch;
        public string Student_Current_Sem;
        public string Student_Mailid;
        public string Student_Mobileno;
        public string Student_Mentor;
    }

    class Registered_Students
    {
        public string Reg_Student_Name;
        public string Reg_Student_Regno;
    }

    class Program
    {
        public static List<SubjectRegistration> subjects = new
List<SubjectRegistration>();
        public static List<Registered_Students> studentreg = new
List<Registered_Students>();
        public static List<Students_Details> studetails = new List<Students_Details>();
        public static List<RegisteredSubject> regsubjects = new
List<RegisteredSubject>();

        static void Main(string[] args)
        {
            initialisevalue(subjects);
```

```
            Student_Login(studentreg,studetails);
        }

        public static int initialisestudent(List<Registered_Students> student)
        {
            Console.Clear();

Console.WriteLine("|_____________________________________________________________
_________|");

Console.WriteLine("|_________________________________AUTHENTICATION_______________________
_________|");
            Console.WriteLine("|*Unique REG NO\t\t\t*CASE INSENSITIVE\t\t*Be PATIENT :)
|");
            Console.WriteLine("|-------------------------------------------------------
---------------------|");
            Console.WriteLine("Enter your NAME ?");
            string Name = Console.ReadLine().ToUpper();
            Console.WriteLine("Enter your REG NO?");
            string Id = Console.ReadLine().ToUpper();
            int index = -1;
            index = student.FindIndex(a => a.Reg_Student_Regno == Id);
            if (index < 0)
            {
                Console.WriteLine("New Student Registered");
                student.Add(new Registered_Students() { Reg_Student_Name = Name,
Reg_Student_Regno = Id });
            }
            index = student.FindIndex(a => a.Reg_Student_Regno == Id);
            return index;

        }

        public static void Student_Panel(List<Registered_Students> studentreg,
List<Students_Details> studetails,int index) {

Console.WriteLine("_____________________________________________________________
_________");

Console.WriteLine("|_________________________________|MENU|_____________________________
_________|");
            Console.WriteLine("1.Fill Preferences\n2.View Details\n3.Subject
Registration\n4.Generate Timetable\n5.Logout\n6.Terminate");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice)
            {
                case 1:
                    Student_Fill_Details(studentreg, studetails,index);
                    break;
                case 5:
                    Student_Login(studentreg, studetails);
                    break;
                case 6:
                    Environment.Exit(0);
                    break;
                case 2:
                    Student_Display_Details(studentreg, studetails, index);
```

```
                    break;
                case 3:initialisemenu(subjects,index,studentreg,regsubjects);
                    break;
                case 4:
                    generatetimetable(subjects, index, studentreg, regsubjects);
                    break;
                default:
                    break;
            }


        }

        public static void Student_Display_Details(List<Registered_Students> studentreg,
List<Students_Details> studetails, int index)
        {
            string regno = studentreg[index].Reg_Student_Regno, name =
studentreg[index].Reg_Student_Name;
            int detail_index = -1;
            detail_index = studetails.FindIndex(a => a.Student_Regno.Equals(regno));
            if (detail_index == -1)
            {
                Console.WriteLine("The details are stil Not filled!");
                Console.WriteLine("Plese fill out the details");
                Student_Fill_Details(studentreg, studetails, index);
            }
            else {

Console.WriteLine("|_____________________________________________________________
_________|");
                Console.WriteLine("|_____________________________STUDENT
INFORMATION_____________________________|");
                Console.WriteLine("|Student NAME:\t\t" + name );
                Console.WriteLine("|Student REG NO:\t\t" + regno);
                Console.WriteLine("|Mentor
Name:\t\t"+studetails[detail_index].Student_Mentor);
                Console.WriteLine("|-----------------------------------------------------
-------------------------|");
                Console.WriteLine("|\t\t\tAcademics");
                Console.WriteLine("|Branch:\t" + studetails[detail_index].Student_Branch
+ "\t\tCurrent Sem:\t" + studetails[detail_index].Student_Current_Sem);
                Console.WriteLine("|-----------------------------------------------------
-------------------------|");
                Console.WriteLine("|\t\t\tContact Details");
                Console.WriteLine("|Phone Number:\t" +
studetails[detail_index].Student_Mobileno+"\t\tMail Id:\t"+
studetails[detail_index].Student_Mailid);

Console.WriteLine("|_____________________________________________________________
_________|");
            }
            Student_Panel(studentreg, studetails, index);
        }

        public static void Student_Fill_Details(List<Registered_Students> studentreg,
List<Students_Details> studetails, int index)
        {
```

```
            string name = studentreg[index].Reg_Student_Name, regno =
studentreg[index].Reg_Student_Regno;

            Console.WriteLine("NAME:\t" + name + "\n" + "REG NO:\t" + regno);
            int detail_index = 0;
            detail_index = studetails.FindIndex(a => a.Student_Regno.Equals(regno));
            if (detail_index == -1)
            {
                Console.WriteLine("Enter Your Branch Name ?");
                String branch_name = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Current Sem ?");
                String current_sem = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Mail ID  ?");
                String mail_id = Console.ReadLine();
                Console.WriteLine("Enter Your Phone Number ?");
                String phone_num = Console.ReadLine().ToUpper();
                Console.WriteLine("Enter Your Mentor Name ?");
                String mentor = Console.ReadLine().ToUpper();
                studetails.Add(new Students_Details() { Student_Name = name,
Student_Regno = regno, Student_Branch = branch_name, Student_Current_Sem = current_sem,
Student_Mailid = mail_id, Student_Mentor = mentor, Student_Mobileno = phone_num });
                Console.WriteLine("Details are added successfully");
                Student_Panel(studentreg, studetails, index);
            }
            else
            {
                Console.WriteLine("Dear "+name+",you have filled your details already");
                Student_Display_Details(studentreg, studetails, index);
            }

            Student_Panel(studentreg, studetails, index);
        }

        public static void Student_Login(List<Registered_Students> studentreg,
List<Students_Details> studetails) {
            int index;
            index = initialisestudent(studentreg);
            Student_Panel(studentreg,studetails,index);
        }

        public static void registersubject(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {
            displaysubject(subjects);
            Console.WriteLine("Enter the Subject Code to Register or (N) to stop");
            string Subcode = Console.ReadLine();
            if (Subcode.Equals("N")) { subjectregistration(subjects, studentindex,
student, regsubjects); };
            Subcode = Subcode.ToUpper();

            string sid = student[studentindex].Reg_Student_Regno;
            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval == -1)
            {
                int index = -1;
```

```
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].Reg_Student_Name);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity--;
                        Console.WriteLine("You are successfully registered for " +
subjects[index].SubjectName);
                        regsubjects.Add(new RegisteredSubject() { SubjectCode = Subcode,
StudentId = sid });
                        registersubject(subjects, studentindex, student, regsubjects);

                    }
                    else
                    {
                        Console.WriteLine("No seats are further available :( ");
                        Console.WriteLine("Please try again :) ");
                        subjectregistration(subjects, studentindex, student,
regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subcode is not found");
                    Console.WriteLine("Please try again :) ");
                    subjectregistration(subjects, studentindex, student, regsubjects);
                }


            }
            else
            {
                Console.WriteLine("Already registered");
                Console.WriteLine("Please try again :) ");
                subjectregistration(subjects, studentindex, student, regsubjects);
            }



        }

        public static void deregistersubject(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {

            string sid = student[studentindex].Reg_Student_Regno;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            Console.WriteLine("You have registered for the following subjects");

            foreach (RegisteredSubject subval in subs)
            {
                String subcodeval = subval.SubjectCode;
                List<SubjectRegistration> subject1 = subjects.FindAll(a => a.SubjectCode
== subcodeval);
                displaysubject(subject1);
```

```
            }

            Console.WriteLine("Enter the Subject Code to Deregister");
            string Subcode = Console.ReadLine();
            Subcode = Subcode.ToUpper();

            //if the user already registered
            int checkval = regsubjects.FindIndex(s => s.SubjectCode == Subcode &&
s.StudentId == sid);
            if (checkval != -1)
            {
                int index = -1;
                //if the subject code is present or not
                index = subjects.FindIndex(a => a.SubjectCode == Subcode);
                if (index != -1)
                {
                    Console.WriteLine("Dear" + student[studentindex].Reg_Student_Name);
                    if (subjects[index].SubjectCapacity > 0)
                    {
                        subjects[index].SubjectCapacity++;
                        Console.WriteLine("You are successfully \"DEREGISTERED\" for " +
subjects[index].SubjectName);
                        regsubjects.RemoveAll(a => a.SubjectCode == Subcode &&
a.StudentId == sid);
                        initialisemenu(subjects, studentindex, student, regsubjects);
                    }
                }
                else
                {
                    Console.WriteLine("Sorry!! :( The subject code is not found");
                    Console.WriteLine("Please try again :) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                }


            }
            else { Console.WriteLine("Not Found"); }

        }

        private static void initialisemenu(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {
            subjectregistration(subjects, studentindex, student, regsubjects);
        }

        public static void generatetimetable(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects)
        {

            string sid = student[studentindex].Reg_Student_Regno;
            List<RegisteredSubject> subs = regsubjects.FindAll(s => s.StudentId == sid);
            if (subs.Count != 0)
            {

Console.WriteLine("|_____________________________________________________________
_________|");
```

```
                Console.WriteLine("|_________________________________|TIME
TABLE|_________________________________|");

Console.WriteLine("Hour|||\tMonday|||\tTuesday|||\tWednesday|||\tThursday|||\tFriday|||")
;
                foreach (RegisteredSubject subval in subs)
                {
                    String subcodeval = subval.SubjectCode;
                    int indexvalof = subjects.FindIndex(a => a.SubjectCode ==
subcodeval);
                    String groupval = subjects[indexvalof].GroupCode;
                    //Console.WriteLine("Group:" + groupval);
                    if (groupval.Equals("A"))
                    {
                        Console.WriteLine();
                        Console.Write("1\t");
                        for (int i = 0; i < 4; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }
                        Console.WriteLine();
                    }
                    else if (groupval.Equals("B"))
                    {
                        Console.WriteLine();
                        Console.Write("2\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("C"))
                    {
                        Console.WriteLine();
                        Console.Write("3\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("D"))
                    {
                        Console.WriteLine();
                        Console.Write("4\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else if (groupval.Equals("E"))
                    {
```

```
                        Console.WriteLine();
                        Console.Write("5\t");
                        for (int i = 0; i < 3; i++)
                        {
                            Console.Write(subcodeval + "\t");
                        }

                        Console.WriteLine();
                    }
                    else
                    {
                        Console.WriteLine("_____________You are free___________");
                    }

                }
                Student_Panel(studentreg, studetails, studentindex);
            }
            else {
                Console.WriteLine("No subjects to Display :( ");
                Console.WriteLine("You have still Not registered the subjects :( ");
                initialisemenu(subjects, studentindex, student, regsubjects);
            }
        }

        public static void subjectregistration(List<SubjectRegistration> subjects, int
studentindex, List<Registered_Students> student, List<RegisteredSubject> regsubjects) {

Console.WriteLine("_____________________________________________________________________
_________");
            Console.WriteLine("|__________________________|SUBJECT
REGISTRATION|_______________________________|");
            Console.WriteLine("1.Register Subject");
            Console.WriteLine("2.Deregister Subject");
            Console.WriteLine("3.Faculty Selection");
            Console.WriteLine("4.Generate Timetable");
            Console.WriteLine("5.Go Back To Main Menu");
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice)
            {
                case 1:
                    registersubject(subjects, studentindex, student, regsubjects);
                    break;
                case 2:
                    deregistersubject(subjects, studentindex, student, regsubjects);
                    break;
                case 3:
                    Console.WriteLine("The function is still under Construction ;) ");
                    initialisemenu(subjects, studentindex, student, regsubjects);
                    break;
                case 4:
                    generatetimetable(subjects, studentindex, student, regsubjects);

                    break;
                case 5:
                    Student_Login(studentreg, studetails);
                    break;
                default:
```

```
                    Console.WriteLine("Invalid Choice");
                    break;
            }
        }

        public static void initialisevalue(List<SubjectRegistration> subjects)
        {
            subjects.Add(new SubjectRegistration() { SubjectName = "C# PROGRM",
SubjectCode = "14CS2054", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 4, SubjectStatus = "Available", GroupCode = "A" });
            subjects.Add(new SubjectRegistration() { SubjectName = "STORAGEAN",
SubjectCode = "14CS2065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "B" });
            subjects.Add(new SubjectRegistration() { SubjectName = "BIG _DATA",
SubjectCode = "14CS3065", SubjectCategory = "SOFTCORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "C" });
            subjects.Add(new SubjectRegistration() { SubjectName = "INTEROFTH",
SubjectCode = "16EC2002", SubjectCategory = "FREEELEC", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "D" });
            subjects.Add(new SubjectRegistration() { SubjectName = "DATASTRUC",
SubjectCode = "14CS2009", SubjectCategory = "____CORE", SubjectCapacity = 5,
SubjectCredit = 3, SubjectStatus = "Available", GroupCode = "E" });
        }

        public static void displaysubject(List<SubjectRegistration> subjects)
        {

Console.WriteLine("\n_____________________________________________________________________
_______________");
            Console.WriteLine("SUB_CODE\tNAME\t\tCREDIT\tCATEGORY\tCAPACITY\tSTATUS");
            Console.WriteLine("-------------------------------------------------------------
-------------------------");
            foreach (SubjectRegistration subval in subjects)
            {
                Console.WriteLine("{0}\t{1}  \t[{2}]\t{3}\t{4}\t\t{5}",
subval.SubjectCode, subval.SubjectName, subval.SubjectCredit, subval.SubjectCategory,
subval.SubjectCapacity, subval.SubjectStatus);

            }
        }
    }


}
```

Output

Authentication

```
¦_____________________________________________________________________________¦
¦_____________________________AUTHENTICATION_____________________________¦
¦*Unique REG NO                 *CASE INSENSITIVE              *Be PATIENT :) ¦
¦-----------------------------------------------------------------------------¦
Enter your NAME ?
suhaas
Enter your REG NO?
ur13cs043
New Student Registered
_____________________________________________________________________________
¦_____________________________¦MENU¦_____________________________¦
1.Fill Preferences
2.View Details
3.Subject Registration
4.Generate Timetable
5.Logout
6.Terminate
```

Student Information

```
2
¦_____________________________________________________________________________¦
¦_____________________________STUDENT INFORMATION_____________________________¦
¦Student NAME:          SUHAAS
¦Student REG NO:                UR13CS043
¦Mentor Name:           A.P.JEYAKRISHNAN
¦-----------------------------------------------------------------------------¦
¦                       Academics
¦Branch:        CSE             Current Sem:    7
¦-----------------------------------------------------------------------------¦
¦                       Contact Details
¦Phone Number:  7708483438                      Mail Id:        suhaas95@gmail.com
¦_____________________________________________________________________________¦
_____________________________________________________________________________
¦_____________________________¦MENU¦_____________________________¦
```

Time table

```
4
¦_____________________________________________________________________________¦
¦_____________________________¦TIME TABLE¦_____________________________¦
Hour¦¦¦ Monday¦¦¦       Tuesday¦¦¦      Wednesday¦¦¦    Thursday¦¦¦     Friday¦¦¦

1       14CS2054        14CS2054        14CS2054        14CS2054

5       14CS2009        14CS2009        14CS2009
_____________________________________________________________________________
¦                               ¦MENU¦                                        ¦
```

Derigisteration

```
You have registered for the following subjects

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
14CS2054        C# PROGRM       [4]     SOFTCORE        4               Available

SUB_CODE        NAME            CREDIT  CATEGORY        CAPACITY        STATUS
14CS2009        DATASTRUC       [3]     ____CORE        4               Available
Enter the Subject Code to Deregister
14cs2054
DearSUHAAS
You are successfully "DEREGISTERED" for C# PROGRM
!___________________________!SUBJECT REGISTRATION!_____________________________!
1.Register Subject
2.Deregister Subject
```

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[**Dr. S.P. Jeno Lovesum**]

Ex. No. 7	Windows Form Application		
Date of Exercise	12.09.2016	Date of Upload	02.11.2016

Aim

To develop **Address Book Maintenance System** using C# by integrating Database with proper authentication

Description

Web - based applications have taken off over the past several years and are fast becoming the standard. The ability to have all of your application logic reside on a centralized server is very appealing from an administrator' s viewpoint. **Windows Forms** will seem familiar if you are a Visual Basic developer. You create new forms (also known as windows or dialogs) in much the same way that you drag and drop controls from a toolbox onto the Form Designer. However, if your background is in the classic C style of Windows programming, where you create the message pump and monitor messages, or if you are an MFC programmer, you will find that you are able to get to the lower - level internals if you need to. You can override the window procedure and catch messages, but you might be surprised that you really won't need to very often.

CREATING A WINDOWS FORMS APPLICATION

```
using System;

using System.Windows.Forms;

namespace NotepadForms

{
public class MyForm: System.Windows.Forms.Form

{

public MyForm()

{

}

[STAThread]

static void Main()
```

```
{ Application.Run(new MyForm());
}}}
```

PANEL

- simply a control that contains other controls
- Panel control is derived from ScrollableControl
- AutoScroll, scroll through all of the controls
- BorderStyle, use the Panel to visually group related controls using borders.
- Panel is the base class for the FlowLayoutPanel, TableLayoutPanel, TabPage, and SplitterPanel

SOME USEFUL SYNTAX

Syntax to create a **text box**

```
private System.Windows.Forms.TextBox textBox1;
```

Syntax to create a **button**

```
private System.Windows.Forms.Button button1;
```

Syntax to create a **label**

```
private System.Windows.Forms.Label label1;
```

Syntax to create a **check box**

```
private System.Windows.Forms.CheckBox checkBox1;
```

Syntax to create a **radio button**

```
private System.Windows.Forms.RadioButton radioButton1;
```

Syntax to create a **combo box**

```
private System.Windows.Forms.ComboBox comboBox1;
```

MDI:

•Multiple-document interface (MDI) applications enable you to display multiple documents at the same time, with each document displayed in its own window.

•MDI applications often have a Window menu item with submenus for switching between windows or documents.

To create an MDI parent form at design time

- Create a Windows Application project.
- In the **Properties** window, set the **IsMDIContainer** property to **true**. This designates the form as an MDI container for child windows. Background turns a dark gray color.
- From the **Toolbox**, drag a **MenuStrip** control to the form. Create a top-level menu item with the **Text** property set to **&File** with submenu items called **&New** and **&Close**. Also create a top-level menu item called **&Window**.
- The first menu will create and hide menu items at run time, and the second menu will keep track of the open MDI child windows. At this point, you have created an MDI parent window.
- Press **F5** to run the application. For information about creating MDI child windows that operate within the MDI parent form.

To create MDI child forms

- Create an MDI parent form
- In **Solution Explorer**, right-click the project, point to **Add**, and then select **Add New Item**.
- In the **Add New Item** dialog box, select **Windows Form**
- Add the code for the menu item called **&Window**

```
protected void MDIChildNew_Click(object sender, System.EventArgs e)
{
Form2 newMDIChild = new Form2();
// Set the Parent Form of the Child window. newMDIChild.MdiParent = this;
// Display the new form.
newMDIChild.Show();
}
```

Use the **ActiveMdiChild property**, which returns the child form that has the focus or that was most recently active.

```
Form activeChild = this.ActiveMdiChild;
```

The child forms can be arranged by calling **LayoutMdi** method. It takes **MdiLayout** enumeration with values of Cascade, TileHorizontal, TileVertical

```
this.LayoutMdi(System.Windows.Forms.MdiLayout.Cascade);
```

FORM INSTANTIATION AND DESTRUCTION

The events occur in the following order in the process of form creation

- Constructor
- Load
- Activated
- Closing
- Closed
- Deactivate

CUSTOM CONTROLS

- The ability to create your own controls, components, and user controls makes it even more productive.
- By creating controls, functionality can be encapsulated into packages that can be reused over and over.
- You can create a control in a number of ways.
- You can start from scratch, deriving your class from either Control , ScrollableControl , or ContainerControl
- Creating a customized textbox inherited from System.Windows.Forms.TextBox

USER CONTROL

- User controls are one of the more powerful features of Windows Forms.
- They enable you to encapsulate user interface designs into nice reusable packages that can be plugged into project after project.
- Create a simple address user control and use it in a form

PROGRAM

AUTHENTICATION[LOGIN]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace _7.Address_Book_Maintenance_System
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            //Validating the password
            String username, password;
            username = textBox1.Text;
            password = textBox2.Text;
            //NUll values
            if (username.Equals("") || password.Equals(""))
            {
                MessageBox.Show("Please fill the fields");
            }
            else {
                //Authentication
                if (username.Equals("admin") || password.Equals("admin")) {


                    Form2 f2 = new Form2();
```

```
                    this.Hide();
                    f2.ShowDialog();
                    this.Close();
                }
            }



        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void label3_Click(object sender, EventArgs e)
        {

        }
    }
}
```

FORM2.CS[ADDRESS BOOK]

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
namespace _7.Address_Book_Maintenance_System
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
            textBox1.ForeColor = Color.SlateGray;
            textBox2.ForeColor = Color.SlateGray;
            textBox3.ForeColor = Color.SlateGray;
            textBox4.ForeColor = Color.SlateGray;
            textBox5.ForeColor = Color.SlateGray;
            textBox6.ForeColor = Color.SlateGray;
            textBox7.ForeColor = Color.SlateGray;
```

```
            textBox8.ForeColor = Color.SlateGray;
            button5.Visible = false;
            button6.Visible = false;
            button7.Visible = false;
            button8.Visible = false;
            Hideall();
        }

        private void textBox1_Click(object sender, EventArgs e)
        {
            textBox1.ForeColor = Color.Black;

        }

        private void textBox3_Click(object sender, EventArgs e)
        {
            textBox3.ForeColor = Color.Black;

        }

        private void textBox4_Click(object sender, EventArgs e)
        {
            textBox4.ForeColor = Color.Black;

        }

        private void textBox5_Click(object sender, EventArgs e)
        {
            textBox5.ForeColor = Color.Black;

        }

        private void textBox6_Click(object sender, EventArgs e)
        {
            textBox6.ForeColor = Color.Black;

        }

        private void textBox2_Click(object sender, EventArgs e)
        {
            textBox2.ForeColor = Color.Black;

        }

        private void textBox7_Click(object sender, EventArgs e)
        {
            textBox7.ForeColor = Color.Black;

        }
```

```
        private void textBox8_Click(object sender, EventArgs e)
        {
            textBox8.ForeColor = Color.Black;

        }

        private void Form2_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the
'formsDbDataSet.citizendetails' table. You can move, or remove it, as needed.

        }
        //Insert
        private void button1_Click(object sender, EventArgs e)
        {
            Visibleall();
            bool val = true;
                val = IfNotNull();
            if (val==false)
            {
                MessageBox.Show("Please Fill out all the fields");
            }
            else
            {
                string source = @"Data Source= SUHAAS;Initial Catalog=FormsDb;Integrated
Security=True";
                SqlConnection con = new SqlConnection(source);
                con.Open();
                SqlCommand insertCommand = new SqlCommand("INSERT INTO [citizendetails]
(addressdb,pindb,citydb,statedb,countrydb,phonedb,maildb,aadharnum) VALUES
(@addr,@pin,@city,@state,@country,@phone,@mail,@aadhar)", con);
                insertCommand.Parameters.Add("@addr", SqlDbType.VarChar, 255,
"addressdb").Value = textBox2.Text;
                insertCommand.Parameters.Add("@pin", SqlDbType.VarChar, 255,
"pindb").Value = textBox6.Text;
                insertCommand.Parameters.Add("@city", SqlDbType.VarChar, 255,
"citydb").Value = textBox5.Text;
                insertCommand.Parameters.Add("@state", SqlDbType.VarChar, 255,
"statedb").Value = textBox4.Text;
                insertCommand.Parameters.Add("@country", SqlDbType.VarChar, 255,
"countrydb").Value = textBox3.Text;
                insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255,
"phonedb").Value = textBox7.Text;
                insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255,
"maildb").Value = textBox8.Text;
                insertCommand.Parameters.Add("@aadhar", SqlDbType.VarChar, 255,
"aadharnum").Value = textBox1.Text;
                int queryResult = insertCommand.ExecuteNonQuery();
                if (queryResult > 0)
                {
```

```
                    label1.Text = "Records Inserted Successfully";
                    label1.ForeColor = Color.Green;
                }
                else
                {
                    label1.Text = "Try Again";
                    label1.ForeColor = Color.Red;
                }
                SqlDataAdapter adp = new SqlDataAdapter("select * from citizendetails",
con);
                DataSet ds = new DataSet();
                adp.Fill(ds);
                dataGridView1.DataSource = ds.Tables[0];
                con.Close();
            }


        }

        public bool IfNotNull()
        {
            if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "" ||
textBox4.Text == ""
                || textBox5.Text == "" || textBox6.Text == "" ||
                textBox7.Text == "" || textBox8.Text == "" ||
textBox5.Text.Equals("City")) {
                return false;
            }
            else
            {
                return true;
            }
        }

        public void Hideall() {
            textBox1.Visible = false;
            textBox2.Visible = false;
            textBox3.Visible = false;
            textBox4.Visible = false;
            textBox5.Visible = false;
            textBox6.Visible = false;
            textBox7.Visible = false;
            textBox8.Visible = false;
            button5.Visible = false;
        }
        public void Visibleall()
        {
            textBox1.Visible = true;
            textBox2.Visible = true;
            textBox3.Visible = true;
```

```
            textBox4.Visible = true;
            textBox5.Visible = true;
            textBox6.Visible = true;
            textBox7.Visible = true;
            textBox8.Visible = true;
        }

        public void ButtonVisible() {
            button1.Visible = true;
            button2.Visible = true;
            button3.Visible = true;
            button4.Visible = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Hideall();
            textBox7.Visible = true;
            textBox8.Visible = true;
            button5.Visible = true;
            label1.Text = "Fill the Visible Boxes for verification";

        }

        private void panel1_Paint(object sender, PaintEventArgs e)
        {

        }

        private void button5_Click(object sender, EventArgs e)
        {
            string phonenum, mailid;
            phonenum = textBox7.Text;
            mailid = textBox8.Text;
            if (phonenum != null || mailid != null || phonenum != "Phone Number" ||
mailid != "Mail Id")
            {
                string source = @"Data Source= SUHAAS;Initial Catalog=FormsDb;Integrated
Security=True";
                SqlConnection con = new SqlConnection(source);
                con.Open();
                SqlCommand insertCommand = new SqlCommand("Select * from
[citizendetails] WHERE phonedb=@phone AND maildb=@mail", con);
                insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255,
"phonedb").Value = phonenum;
                insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255,
"maildb").Value = mailid;
                SqlDataReader queryResult = insertCommand.ExecuteReader();
                while (queryResult.Read())
                {
```

```
                    textBox2.Text = queryResult[0].ToString();
                    textBox6.Text = queryResult[1].ToString();
                    textBox3.Text = queryResult[2].ToString();
                    textBox4.Text = queryResult[3].ToString();
                    textBox5.Text = queryResult[4].ToString();
                    textBox1.Text = queryResult[7].ToString();
                }

                if (queryResult !=null)
                {
                    Visibleall();
                    button2.Visible = false;
                    textBox7.Visible = false;
                    textBox8.Visible = false;
                    label1.Text = "Click below to Update";
                    label1.ForeColor = Color.Green;
                    textBox7.Text = phonenum;
                    textBox8.Text = mailid;
                    button6.Visible = true;
                    button5.Visible = false;
                    button6.Text = "Update Please";
                }
                else
                {
                    label1.Text = "Invalid Credentials"+queryResult;
                    label1.ForeColor = Color.Red;
                }
                con.Close();
            }
            else { MessageBox.Show("Enter the fields "); }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            string phonenum, mailid;
            phonenum = textBox7.Text;
            mailid = textBox8.Text;

            bool val = true;
            val = IfNotNull();
            if (val == false)
            {
                MessageBox.Show("Please Fill out all the fields");
            }
            else
            {
                string source = @"Data Source= SUHAAS;Initial Catalog=FormsDb;Integrated
Security=True";
                SqlConnection con = new SqlConnection(source);
                con.Open();
```

```
                SqlCommand insertCommand = new SqlCommand("UPDATE [citizendetails] SET
addressdb=@addr,pindb=@pin,citydb=@city,statedb=@state,countrydb=@country,aadharnum=@aadh
ar WHERE phonedb=@phone AND maildb=@mail", con);
                insertCommand.Parameters.Add("@addr", SqlDbType.VarChar, 255,
"addressdb").Value = textBox2.Text;
                insertCommand.Parameters.Add("@pin", SqlDbType.VarChar, 255,
"pindb").Value = textBox6.Text;
                insertCommand.Parameters.Add("@city", SqlDbType.VarChar, 255,
"citydb").Value = textBox5.Text;
                insertCommand.Parameters.Add("@state", SqlDbType.VarChar, 255,
"statedb").Value = textBox4.Text;
                insertCommand.Parameters.Add("@country", SqlDbType.VarChar, 255,
"countrydb").Value = textBox3.Text;
                insertCommand.Parameters.Add("@aadhar", SqlDbType.VarChar, 255,
"aadharnum").Value = textBox1.Text;
                insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255,
"phonedb").Value = textBox7.Text;
                insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255,
"maildb").Value = textBox8.Text;
                int queryResult = insertCommand.ExecuteNonQuery();
                if (queryResult > 0)
                {
                    label1.Text = "Updated Successfully";
                    label1.ForeColor = Color.Green;
                }
                else
                {
                    label1.Text = "Try Again";
                    label1.ForeColor = Color.Red;
                }
                SqlDataAdapter adp = new SqlDataAdapter("select * from citizendetails ",
con);
                DataSet ds = new DataSet();
                adp.Fill(ds);
                dataGridView1.DataSource = ds.Tables[0];
                con.Close();
                button5.Visible = false;
                button6.Visible = false;
                button2.Visible = true;
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            button7.Visible = true;
            Hideall();
            textBox7.Visible = true;
            textBox8.Visible = true;
            label1.Text = "Press Confirm after Entering Details";
            label1.ForeColor = Color.Red;
```

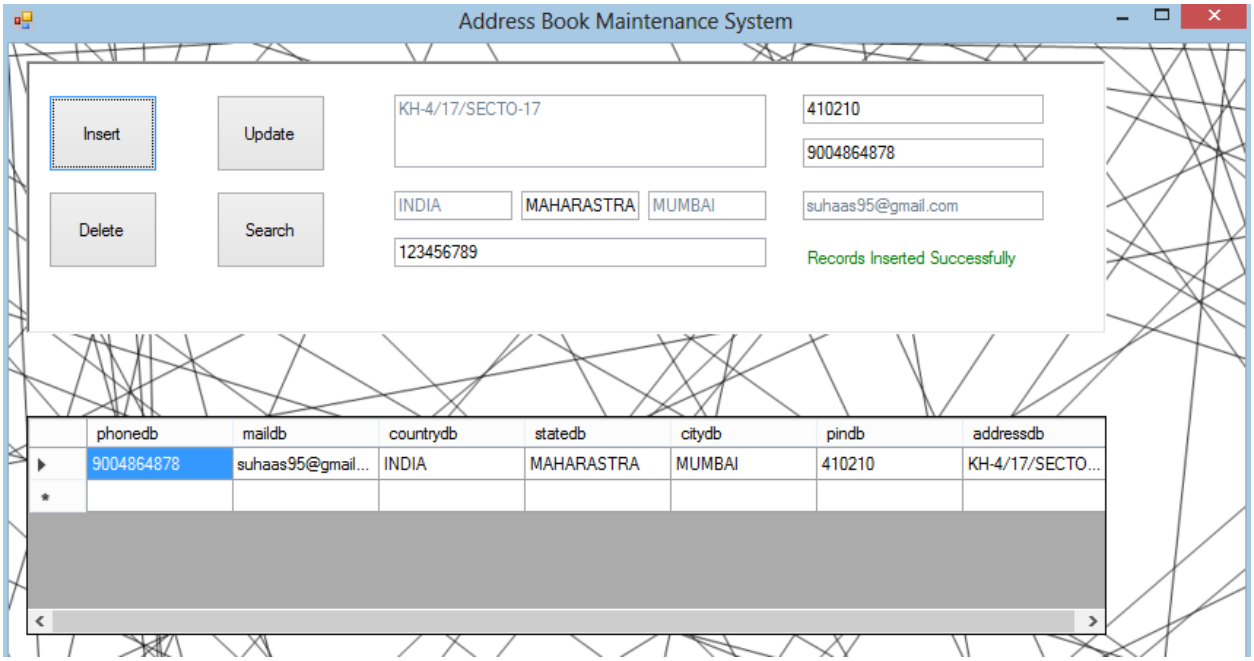
```
        }

        private void button7_Click(object sender, EventArgs e)
        {
            string source = @"Data Source= SUHAAS;Initial Catalog=FormsDb;Integrated
Security=True";
            SqlConnection con = new SqlConnection(source);
            con.Open();
            SqlCommand insertCommand = new SqlCommand("DELETE [citizendetails] FROM
[citizendetails]  WHERE phonedb=@phone AND maildb=@mail", con);
            insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255,
"phonedb").Value = textBox7.Text;
            insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255, "maildb").Value
= textBox8.Text;
            int queryResult = insertCommand.ExecuteNonQuery();
            if (queryResult > 0)
            {
                label1.Text = "Deleted Successfully";
                label1.ForeColor = Color.Green;
            }
            else
            {
                label1.Text = "Data Not Found";
                label1.ForeColor = Color.Red;
            }
            SqlDataAdapter adp = new SqlDataAdapter("select * from citizendetails ",
con);
            DataSet ds = new DataSet();
            adp.Fill(ds);
            dataGridView1.DataSource = ds.Tables[0];
            con.Close();
            button7.Visible = false;
        }

        private void button8_Click(object sender, EventArgs e)
        {
            string aadharnum;
            aadharnum = textBox1.Text;
            if (aadharnum != "" || aadharnum.Equals("Aadhar Card Number"))
            {
                string source = @"Data Source= SUHAAS;Initial Catalog=FormsDb;Integrated
Security=True";
                SqlConnection con = new SqlConnection(source);
                con.Open();
                SqlCommand insertCommand = new SqlCommand("Select * from
[citizendetails] WHERE aadharnum=@aadhar", con);
                insertCommand.Parameters.Add("@aadhar", SqlDbType.VarChar, 255,
"aadharnum").Value = aadharnum;
                SqlDataReader queryResult = insertCommand.ExecuteReader();
```

```
                if (!queryResult.HasRows)
                {
                    label1.Text = "Not Found";
                    label1.ForeColor = Color.Red;
                    ButtonVisible();
                    Visibleall();
                    MessageBox.Show("No Records are found", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                }
                else {
                    while (queryResult.Read())
                    {
                        textBox2.Text = queryResult[0].ToString();
                        textBox6.Text = queryResult[1].ToString();
                        textBox3.Text = queryResult[2].ToString();
                        textBox4.Text = queryResult[3].ToString();
                        textBox5.Text = queryResult[4].ToString();
                        textBox1.Text = queryResult[7].ToString();
                        textBox7.Text = queryResult[5].ToString();
                        textBox8.Text = queryResult[6].ToString();
                    }
                    Visibleall();
                    label1.Text = "Record Found";
                    label1.ForeColor = Color.Green;
                    Visibleall();
                    ButtonVisible();
                    button8.Visible = false;
                }

                con.Close();
            }
            else { MessageBox.Show("Enter the fields"); }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Hideall();
            textBox1.Visible = true;
            button8.Visible = true;
            button1.Visible = false;
            button2.Visible = false;
            button3.Visible = false;
            button4.Visible = false;
            button7.Visible = false;

            label1.Text = "Enter the Aadhar Card Number to Search";
            label1.ForeColor = Color.Red;
        }
    }
}
```

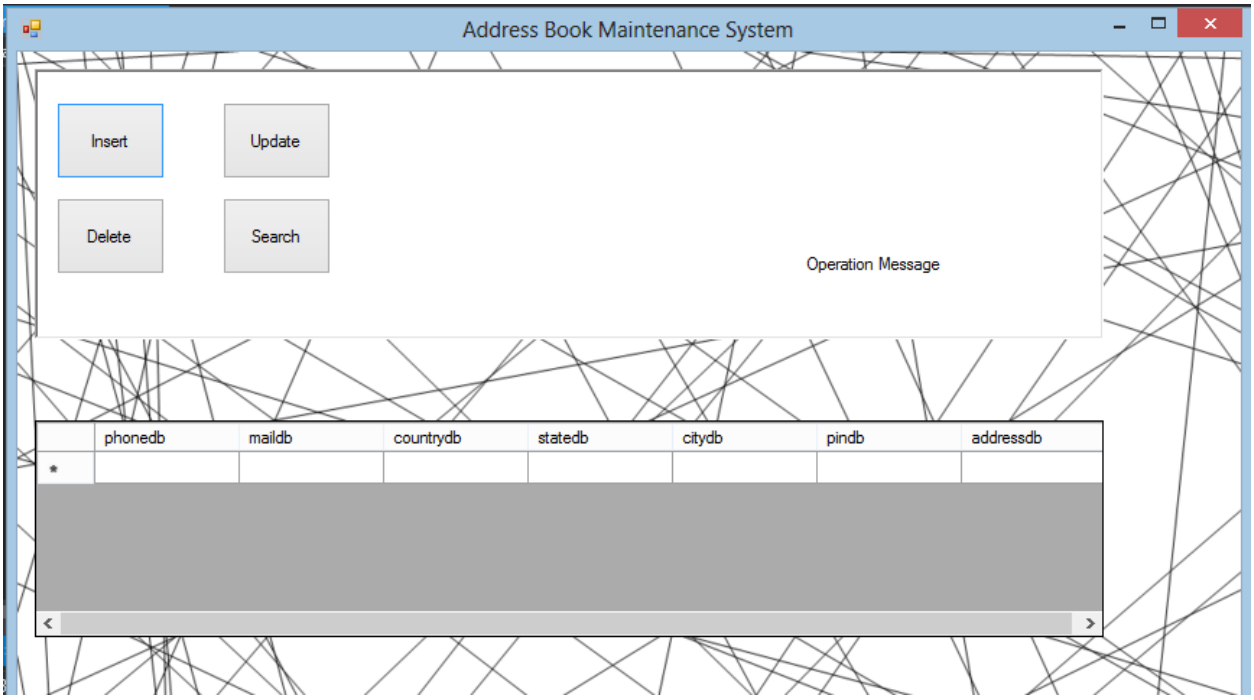
OUTPUT

AUTHENTICATION

INSERT

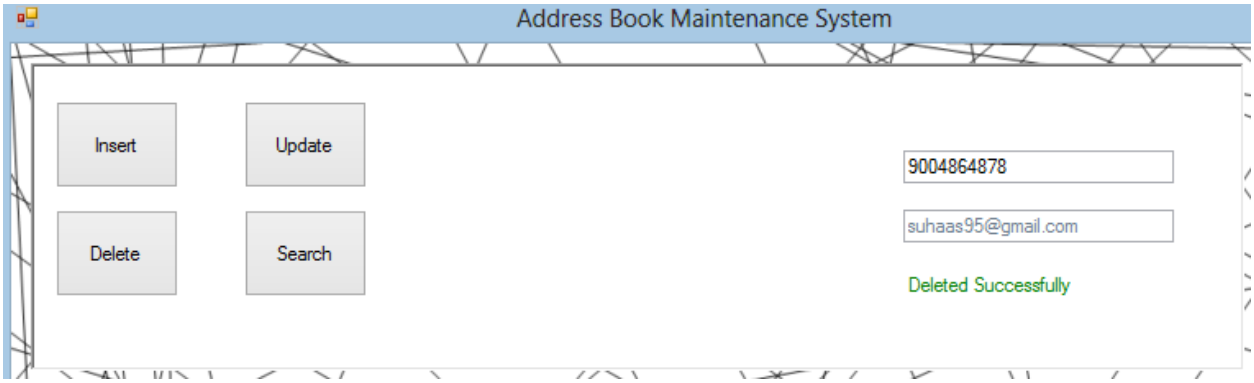

Address Book Maintenance System
Insert
Update
Delete
Search
KH-4/17/SECTO-17
410210
9004864878
INDIA
MAHARASTRA
MUMBAI
suhaas95@gmail.com
123456789
Records Inserted Successfully
phonedb
maildb
countrydb
statedb
citydb
pindb
addressdb
9004864878
suhaas95@gmail...
INDIA
MAHARASTRA
MUMBAI
410210
KH-4/17/SECTO...

MENU


Address Book Maintenance System
Insert
Update
Delete
Search
Operation Message
phonedb
maildb
countrydb
statedb
citydb
pindb
addressdb

UPDATE

DELETE



SEARCH

Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[Dr J Anitha/Dr. S.P. Jeno Lovesum]

Ex. No. 8	Threading and Synchronization		
Date of Exercise	05.10.2016	Date of Upload	06.11.2016

AIM

To perform **Banking Operations** using C# by using the concept of multithreading and to use synchronization method so as to avoid race conditions.

DESCRIPTION

A **thread** is defined as the execution path of a program. Each thread defines a unique flow of control. If your application involves complicated and time consuming operations, then it is often helpful to set different execution paths or threads, with each thread performing a particular job.

Threads are lightweight processes. One common example of use of thread is implementation of concurrent programming by modern operating systems. Use of threads saves wastage of CPU cycle and increase efficiency of an application.

Thread Life Cycle

The life cycle of a thread starts when an object of the System.Threading.Thread class is created and ends when the thread is terminated or completes execution.

Following are the various states in the life cycle of a thread:

- **The Unstarted State**: It is the situation when the instance of the thread is created but the Start method is not called.
- **The Ready State**: It is the situation when the thread is ready to run and waiting CPU cycle.
- **The Not Runnable State**: A thread is not executable, when:
  - Sleep method has been called
  - Wait method has been called
  - Blocked by I/O operations
- **The Dead State**: It is the situation when the thread completes execution or is aborted.

The following program demonstrates main thread execution:

```
using System;
using System.Threading;

namespace MultithreadingApplication
{
   class MainThreadProgram
   {
      static void Main(string[] args)
      {
         Thread th = Thread.CurrentThread;
         th.Name = "MainThread";
         Console.WriteLine("This is {0}", th.Name);
         Console.ReadKey();
      }
   }
}
```

PROPERTIES AND METHODS OF THE THREAD CLASS

Property	Description
CurrentContext	Gets the current context in which the thread is executing.
CurrentCulture	Gets or sets the culture for the current thread.

CurrentPrinciple	Gets or sets the thread's current principal (for role-based security).
CurrentThread	Gets the currently running thread.
CurrentUICulture	Gets or sets the current culture used by the Resource Manager to look up culture-specific resources at run-time.
ExecutionContext	Gets an ExecutionContext object that contains information about the various contexts of the current thread.
IsAlive	Gets a value indicating the execution status of the current thread.
IsBackground	Gets or sets a value indicating whether or not a thread is a background thread.
IsThreadPoolThread	Gets a value indicating whether or not a thread belongs to the managed thread pool.
ManagedThreadId	Gets a unique identifier for the current managed thread.
Name	Gets or sets the name of the thread.
Priority	Gets or sets a value indicating the scheduling priority of a thread.
ThreadState	Gets a value containing the states of the current thread.

THREADING ISSUES

Programming with multiple threads is not easy. When starting multiple threads that access the same data, you can get intermittent problems that are hard to find. This is the same if you use tasks, Parallel LINQ, or the Parallel class. To avoid getting into trouble, you must pay attention to synchronization issues and the problems that can happen with multiple threads.

RACE CONDITION

A race condition can occur if two or more threads access the same objects and access to the shared state is not synchronized.

```
private object sync = new object();

lock (sync)

{

//method

}
```

DEADLOCK

Too much locking can get you in trouble as well. In a deadlock, at least two threads halt and wait for each other to release a lock. As both threads wait for each other, a deadlock occurs and the threads wait endlessly.

SYNCHRONIZATION

It is best to avoid synchronization issues by not sharing data between threads. Of course, this is not always possible. If data sharing is necessary, you must use synchronization techniques so that only one thread at a time accesses and changes shared state.

Various synchronization technologies that you can use with multiple threads:

- lock statement
- Interlocked class
- Monitor class
- SpinLock struct
- WaitHandle class
- Mutex class
- Semaphore class
- Events classes
- Barrier class
- ReaderWriterLockSlim class

PROGRAM

Banking program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace _8.Banking_Operations
{
    public class BankAccount {
        public string name;
        public string accno;
        public decimal accbal;
    }

    public class TransactionLog {
        public string logname;
        public string acc_no;
    }

    class Program
    {
        static List<TransactionLog> log = new List<TransactionLog>();
        static List<BankAccount> account = new List<BankAccount>();
        static void Main(string[] args)
        {

Console.WriteLine("_____________________________________________________________________
_________");
            Console.WriteLine("|_____________________________|Banking
Operations|_____________________________|");


            InitializeAccounts(account);
            CheckPwd(account);
        }

        public static string ReadPassword()
        {
            string password = ""; ConsoleKeyInfo info = Console.ReadKey(true); while
(info.Key != ConsoleKey.Enter)
            {
                if (info.Key != ConsoleKey.Backspace) { Console.Write("*"); password +=
info.KeyChar; }
                else if (info.Key == ConsoleKey.Backspace)
```

```
                {
                    if (!string.IsNullOrEmpty(password))
                    {
                        // remove one character from the list of password characters
                        password = password.Substring(0, password.Length - 1); // get the
location of the cursor
                        int pos = Console.CursorLeft; // move the cursor to the left by
one character
                        Console.SetCursorPosition(pos - 1, Console.CursorTop); // replace
it with space
                        Console.Write(" "); // move the cursor to the left by one
character again
                        Console.SetCursorPosition(pos - 1, Console.CursorTop);
                    }
                }
                info = Console.ReadKey(true);
            } // add a new line because user pressed enter at the end of their password
            Console.WriteLine();
            return password;
        }

        public static void CheckPwd(List<BankAccount> account)
        {
            lOGIN:
            Console.WriteLine("NAME:");
            string name = Console.ReadLine().ToUpper();
            Console.WriteLine("Acc No:");
            string accno = ReadPassword();
            int index = -1;
            index = account.FindIndex(a => a.name.Equals(name) && a.accno.Equals(accno));
            if (index != -1)
            {
                //InitializeMenu(account, index, log);
                var t = new Thread(()=>InitializeMenu(account, index, log));
                t.Start();
            }
            else { Console.WriteLine("Bad Login"); goto lOGIN; }
        }

        public static void InitializeMenu(List<BankAccount> account,int index,
List<TransactionLog> log) {
            Console.WriteLine("Thread Id is"+Thread.CurrentThread.ManagedThreadId);
            Console.WriteLine("Welcome " + account[index].name
+"\t["+account[index].accno+"]");
            Console.WriteLine("Your Current Acc_Balanace is " + account[index].accbal);
            Console.WriteLine("1.Transfer Money");
            Console.WriteLine("2.Deposit Money");
            Console.WriteLine("3.Check Balance");
            Console.WriteLine("4.Transaction History");
            Console.WriteLine("5.Log Out");
```

```
            int choice = Convert.ToInt32(Console.ReadLine());
            switch (choice) {
                case 1:
                    Transfer_Money(account, index, log);
                    break;
                case 2:Deposit_Money(account, index, log);
                    break;
                case 3:CheckBalance(account, index, log);
                    break;
                case 4:Transaction_history(account, index, log);
                    break;
                case 5:
                    CheckPwd(account);
                    break;
                default:
                    Console.WriteLine("Invalid Choice");
                    InitializeMenu(account, index,log);
                    break;

            }
        }

        public static void CheckBalance(List<BankAccount> account, int index,
List<TransactionLog> log) {
            Console.WriteLine("|--------------------------Account Details---------------
------|");
            Console.WriteLine("|\tAccount Holder
Name\t{0}\t\t\t\t|",account[index].name);
            Console.WriteLine("|\tAccount Number\t\t{0}\t\t\t\t|" ,
account[index].accno);
            Console.WriteLine("|\tAccount Balance\t\t{0}\t\t\t\t|" ,
account[index].accbal);
            Console.WriteLine("|-------------------------------------------------------
------|");
            InitializeMenu(account, index, log);
        }

        public static void Transaction_history(List<BankAccount> account, int index,
List<TransactionLog> log) {
            string currnt_accno = account[index].accno;
            List<TransactionLog> Transactions = log.FindAll(a =>
a.acc_no.Equals(currnt_accno));
            for (int i=0; i < Transactions.Count;i++)
            {
                Console.WriteLine(Transactions[i].logname);
            }
            InitializeMenu(account, index, log);
        }

        private static  object lockMethod = new object();
```

```
        public static void Transfer_Money(List<BankAccount> account, int
index,List<TransactionLog> log) {

            lock (lockMethod)
            {
                Console.WriteLine("Enter the amount to be Transferred");
                decimal t_amt = Convert.ToDecimal(Console.ReadLine());
                decimal curr_amt = account[index].accbal;
                string currnt_accno = account[index].accno;
                if (t_amt < curr_amt)
                {
                    Console.WriteLine("Enter the Account No to be transferred : ");
                    string t_acc_no = Console.ReadLine();
                    int trans_account = account.FindIndex(a => a.accno.Equals(t_acc_no));
                    account[trans_account].accbal += t_amt;//Amount Transferred
                    account[index].accbal -= t_amt;//Amount Deducted
                    Why("Transferring");
                    Console.WriteLine("Transferred Successfully");
                    log.Add(new TransactionLog { logname = "" + System.DateTime.Now +
"\t" + "Transferred FROM " + currnt_accno + " TO " + t_acc_no, acc_no = currnt_accno });
                    InitializeMenu(account, index, log);
                }
                else {
                    Console.WriteLine("Insuffient Funds");
                    InitializeMenu(account, index, log);
                }
            }
        }

        public static void Deposit_Money(List<BankAccount> account, int index,
List<TransactionLog> log)
        {
            lock (lockMethod) {
                Console.WriteLine("Enter the amount to be Deposit");
                decimal t_amt = Convert.ToDecimal(Console.ReadLine());
                string currnt_accno = account[index].accno;
                if (t_amt > 0)
                {
                    account[index].accbal += t_amt;//Amount Debitted
                    Why("Debitting");
                    Console.WriteLine("Debitted Successfully");
                    log.Add(new TransactionLog { logname = "" + System.DateTime.Now +
"\t" + "Debitted to " + currnt_accno + " an amount of  " + account[index].accbal, acc_no
= currnt_accno });
                    InitializeMenu(account, index, log);
                }
                else
                {
                    Console.WriteLine("Incorrect Funds");
```

```
                    InitializeMenu(account, index, log);
                }
            }
        }

        public static void InitializeAccounts(List<BankAccount> account) {
            account.Add(new BankAccount { name="SUHAAS",accno="1001",accbal=5000});
            account.Add(new BankAccount { name = "SNEHAL", accno = "1002", accbal = 10000
});
            account.Add(new BankAccount { name = "ARUNA", accno = "1003", accbal = 15000
});
            account.Add(new BankAccount { name = "SRINIVAS", accno = "1004", accbal =
20000 });
        }

        static public void Why(string msg)
        {
            Console.Write(msg);
            using (var progress = new ProgressBar())
            {
                for (int i = 0; i <= 100; i++)
                {
                    progress.Report((double)i / 100);
                    Thread.Sleep(20);
                }
            }
            Console.WriteLine("Done.");
        }
    }
}
```

PROGRESSBAR.CS

```
using System;
using System.Text;
using System.Threading;

/// <summary>
/// An ASCII progress bar
/// </summary>
public class ProgressBar : IDisposable, IProgress<double>
{
    private const int blockCount = 10;
    private readonly TimeSpan animationInterval = TimeSpan.FromSeconds(1.0 / 8);
    private const string animation = @"|/-\";

    private readonly Timer timer;

    private double currentProgress = 0;
    private string currentText = string.Empty;
    private bool disposed = false;
    private int animationIndex = 0;

    public ProgressBar()
    {
        timer = new Timer(TimerHandler);

        // A progress bar is only for temporary display in a console window.
        // If the console output is redirected to a file, draw nothing.
        // Otherwise, we'll end up with a lot of garbage in the target file.
        if (!Console.IsOutputRedirected)
        {
            ResetTimer();
        }
    }

    public void Report(double value)
    {
        // Make sure value is in [0..1] range
        value = Math.Max(0, Math.Min(1, value));
        Interlocked.Exchange(ref currentProgress, value);
    }

    private void TimerHandler(object state)
    {
        lock (timer)
        {
            if (disposed) return;

            int progressBlockCount = (int)(currentProgress * blockCount);
```

```
            int percent = (int)(currentProgress * 100);
            string text = string.Format("[{0}{1}] {2,3}% {3}",
                new string('#', progressBlockCount), new string('-', blockCount -
progressBlockCount),
                percent,
                animation[animationIndex++ % animation.Length]);
            UpdateText(text);

            ResetTimer();
        }
    }

    private void UpdateText(string text)
    {
        // Get length of common portion
        int commonPrefixLength = 0;
        int commonLength = Math.Min(currentText.Length, text.Length);
        while (commonPrefixLength < commonLength && text[commonPrefixLength] ==
currentText[commonPrefixLength])
        {
            commonPrefixLength++;
        }

        // Backtrack to the first differing character
        StringBuilder outputBuilder = new StringBuilder();
        outputBuilder.Append('\b', currentText.Length - commonPrefixLength);

        // Output new suffix
        outputBuilder.Append(text.Substring(commonPrefixLength));

        // If the new text is shorter than the old one: delete overlapping characters
        int overlapCount = currentText.Length - text.Length;
        if (overlapCount > 0)
        {
            outputBuilder.Append(' ', overlapCount);
            outputBuilder.Append('\b', overlapCount);
        }

        Console.Write(outputBuilder);
        currentText = text;
    }

    private void ResetTimer()
    {
        timer.Change(animationInterval, TimeSpan.FromMilliseconds(-1));
    }

    public void Dispose()
    {
        lock (timer)
```

```
        {
            disposed = true;
            UpdateText(string.Empty);
        }
    }

}
```

OUTPUT

MENU

```
!_____________________________!Banking Operations!_____________________________!
NAME:
SUHAAS
Acc No:
****
Thread Id is3
Welcome SUHAAS  [1001]
Your Current Acc_Balanace is 5000
1.Transfer Money
2.Deposit Money
3.Check Balance
4.Transaction History
5.Log Out
```

TRANSFER

```
1
Enter the amount to be Transferred
220.98
Enter the Account No to be transferred :
1002
TransferringDone.
Transferred Successfully
Thread Id is3
Welcome SUHAAS  [1001]
Your Current Acc_Balanace is 4779.02
```

DEPOSIT

```
Enter the amount to be Deposit
100
DebittingDone.
Debitted Successfully
Thread Id is3
Welcome SUHAAS  [1001]
Your Current Acc_Balanace is 4879.02
```

HISTORY

```
4
06-Nov-16 15:32:32      Transferred FROM 1001 TO 1002
06-Nov-16 15:33:29      Debitted to 1001 an amount of  4879.02
Thread Id is3
Welcome SUHAAS  [1001]
Your Current Acc_Balanace is 4879.02
```

BALANCE

```
¦---------------------------Account Details----------------------¦
¦       Account Holder Name      SUHAAS                            ¦
¦       Account Number           1001                              ¦
¦       Account Balance          4879.02                           ¦
¦-----------------------------------------------------------------¦
Thread Id is3
Welcome SUHAAS  [1001]
Your Current Acc_Balanace is 4879.02
1.Transfer Money
2.Deposit Money
3.Check Balance
4.Transaction History
5.Log Out
```

Result

The above programme is compiled successfully and the screenshots are well described with successful outputs and constraints.

[Dr J Anitha/Dr. S.P. Jeno Lovesum]

Ex. No. 9	Web Application using ASP.NET		
Date of Exercise	19.10.2016	Date of Upload	06.11.2016

Aim

To develop **Online Job Portal System** using C#.NET by integrating Database with proper authentication and various operations in context with defined application.

Description

- ASP.NET is the part of the .NET Framework and is a technology that allows for the dynamic creation of documents on a Web server when they are requested via HTTP.
- ASP.NET, as its name suggests, has been designed to be fully integrated with the .NET Framework programming in this technology used scripting languages such as VBScript or JScript ASP.NET pages that generate HTML content are often called ***Web Forms***.

State Management in ASP.NET

- ASP.NET pages is effectively stateless.
- By default, no information is stored on the server between user requests.
- In short, information such as the state of controls on a Web is stored in a hidden viewstate field that is part of the page generated by the server and passed to the user.
- Triggering events that require server – side processing, like submitting form data, result in this information being sent back to the server; this is known as a postback operation

ASP.NET Web Forms

- Enclosing code in <script > elements, using two attributes on the opening < script > tag:

```
<script language=”c#” runat=”server” >
// Server-side code goes here.
</script >
```

- The runat= “ server ” attribute here is crucial because it instructs the ASP.NET engine to execute this code on the server rather than sending it to the client.
- If you omit the runat= “ server ” attribute, you are effectively providing client - side code,
- Use <script > elements to supply client - side script in languages such as JavaScript.

```
<script language=”JavaScript” type=”text/JavaScript” >
```

```
// Client-side code goes here.
< /script >
```

- It is possible to create ASP.NET files in Visual Studio
- The .aspx files can also include code in blocks enclosed by < % and % > tags.
- The code is compiled, not interpreted. This results in far better performance.

The ASP.NET Code Model

- In ASP.NET, a combination of layout (HTML) code, ASP.NET controls, and C# code is used to generate the HTML that users see.
- The layout and ASP.NET code are stored in an .aspx file.
- The C# code that you add to customize the behavior of the form is contained either in the .aspx file or, in a separate .aspx.cs file, which is usually referred to as the " code - behind " file.

PROGRAM

LOGIN PAGE .CS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class LoginPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Session.Abandon();
        Session.Clear();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        string mail = TextBox1.Text;
        string pwd = TextBox2.Text;

        if (mail.Equals("admin") && pwd.Equals("admin"))
        {

            Response.Redirect("AdminPage.aspx");
        }
        else {
            try
            {
```

```
                SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
                conn.Open();

                SqlCommand insertCommand = new SqlCommand("SELECT * FROM  [userdetails]
WHERE mail=@mail AND pwd=@pwd", conn);
                insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255,
"mail").Value = mail;
                insertCommand.Parameters.Add("@pwd", SqlDbType.VarChar, 255, "pwd").Value
= pwd;
                SqlDataReader queryResult = insertCommand.ExecuteReader();

                if (queryResult.HasRows)
                {

                    Response.Redirect("ViewJobs.aspx");
                }
                conn.Close();
            }
            catch (Exception ex)
            {
                Label1.Text = "Please Report to admin: " + ex.Message;
                Label1.ForeColor = System.Drawing.Color.Red;
            }
        }




    }
}
```

FORGOT PASSWORD.CS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class LoginPage : System.Web.UI.Page
{
    static string mail, phone;
    protected void Page_Load(object sender, EventArgs e)
    {


    }
```

```
    protected void ButtonEnterDetails_Click(object sender, EventArgs e)
    {
        try
        {

            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();

             mail = TextBoxMail.Text;
             phone = TextBoxPhone.Text;

            SqlCommand insertCommand = new SqlCommand("SELECT * FROM  [userdetails] WHERE
mail=@mail AND phone=@phone", conn);
            insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255, "mail").Value =
mail;
            insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255, "phone").Value
= phone;
            SqlDataReader queryResult = insertCommand.ExecuteReader();

            if (queryResult.HasRows)
            {
                Label1.Text = "Data Validated Successfully :) \n Please Enter the New
Password";
                Label1.ForeColor = System.Drawing.Color.Green;
                checkformmailphone.Visible = false;
                checkformpwd.Visible = true;
                ButtonEnterDetails.Visible = false;
                ButtonChangePassword.Visible = true;
            }
            conn.Close();
        }
        catch (Exception ex) {
            Label1.Text = "Please Report to admin: " + ex.Message;
            Label1.ForeColor = System.Drawing.Color.Red;
        }
    }

    protected void ButtonChangePassword_Click(object sender, EventArgs e)
    {

        try
        {

            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();

            string pwd = TextBoxPwd.Text;
            string repwd = TextBoxRePwd.Text;

            SqlCommand insertCommand = new SqlCommand("UPDATE [userdetails] SET pwd=@pwd
WHERE mail=@mail AND phone=@phone", conn);
            insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255, "mail").Value =
mail;
```

```
            insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255, "phone").Value
= phone;
            insertCommand.Parameters.Add("@pwd", SqlDbType.VarChar, 255, "pwd").Value =
pwd;
            int queryResult = insertCommand.ExecuteNonQuery();

            if (queryResult > 0)
            {
                WellFirst.Visible = false;
                panel.Visible = false;
                WellSecond.Visible = true;

            }
            else {
                Label1.Text = "Error Changing Password";
                Label1.ForeColor = System.Drawing.Color.Red;
            }
            conn.Close();
        }
        catch (Exception ex)
        {
            Label1.Text = "Please Report to admin: " + ex.Message;
            Label1.ForeColor = System.Drawing.Color.Red;
        }

    }
}
```

SIGN UP PAGES .CS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class LoginPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }




    protected void ButtonEnterDetails_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();
        string fname = TextBoxFname.Text;
        string lname = TextBoxLname.Text;
        string pwd = TextBoxPwd.Text;
        string repwd = TextBoxRePwd.Text;
        string prof = DropDownListProfession.SelectedValue;
        string mail = TextBoxMail.Text;
        string phone = TextBoxPhone.Text;
        string dob = TextBoxDob.Text;
        string gender = RadioButtonListGender.SelectedValue;

            SqlCommand insertCommand = new SqlCommand("INSERT INTO [userdetails]
(fname,lname,pwd,prof,mail,phone,dob,gender) VALUES
(@fname,@lname,@pwd,@prof,@mail,@phone,@dob,@gender)", conn);
            insertCommand.Parameters.Add("@fname", SqlDbType.VarChar, 255, "fname").Value
= fname;
            insertCommand.Parameters.Add("@lname", SqlDbType.VarChar, 255, "lname").Value
= lname;
            insertCommand.Parameters.Add("@pwd", SqlDbType.VarChar, 255, "pwd").Value =
pwd;
            insertCommand.Parameters.Add("@prof", SqlDbType.VarChar, 255, "prof").Value =
prof;
            insertCommand.Parameters.Add("@mail", SqlDbType.VarChar, 255, "mail").Value =
mail;
            insertCommand.Parameters.Add("@phone", SqlDbType.VarChar, 255, "phone").Value
= phone;
            insertCommand.Parameters.Add("@dob", SqlDbType.VarChar, 255, "dob").Value =
dob;
            insertCommand.Parameters.Add("@gender", SqlDbType.VarChar, 255,
"gender").Value = gender;
            int queryResult = insertCommand.ExecuteNonQuery();
            conn.Close();
            if (queryResult > 0) {
                Label1.Text = "Data Inserted Successfully :) ";
                Label1.ForeColor = System.Drawing.Color.Green;
            }
        }
        catch(Exception ex)
        {
            Label1.Text = "Please Refer to Error: " + ex.Message;
            Label1.ForeColor = System.Drawing.Color.Red;
        }



    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("LoginPage.aspx");
    }
}
```

EDIT PAGE .CS

```
using System;
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class EditPage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["ExampleConnectionString"].Connectio
nString);
            conn.Open();
        }
        catch
        {
        }
    }
    protected void Button2_Click(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();
            SqlCommand insertCommand = new SqlCommand("select * from [userdetails]",
conn);
            SqlDataReader queryResult = insertCommand.ExecuteReader();


            GridView1.DataBind();


            conn.Close();
        }
        catch
        {
            Label3.Text = "Event not added due to DB access problem.";
        }
    }
```

```
}
```

WEB USER CONTROL.ASCX

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs"
Inherits="WebUserControl" %>
<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Caption.png" />
<br />
```

VIEWJOBS.CS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class ViewJobs : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();

            SqlCommand insertCommand = new SqlCommand("SELECT * FROM  [jobdetails]",
conn);
            SqlDataReader queryResult = insertCommand.ExecuteReader();

            StringBuilder html = new StringBuilder();
            html.Append("<div class='col-sm-9 col-sm-offset-3 col-lg-10 col-lg-offset-2
main'>");
            html.Append("<div runat='server' id='jobpostpanel2' class='row'>");
            html.Append("<div class='col-lg-12'>");
            html.Append("<h2>Job Posts</h2>");
            html.Append("</div>");

            if (queryResult.HasRows)
            {
                Label1.Text = "Data Fetched Successfully :)";
                Label1.ForeColor = System.Drawing.Color.Green;
                while (queryResult.Read())
                {
                    html.Append("<div class='col-md-4'>");
```

```
                    html.Append("<div class='panel panel-default'>");
                    html.Append("<div class='panel-heading text-center'>");
                    html.Append("<i class='fa fa-quote-left fa-1x fa-pull-left ' aria-
hidden='true'></i>");
                    html.Append(queryResult[0].ToString());
                    html.Append("<i class='fa fa-quote-right fa-1x fa-pull-right ' aria-
hidden='true'></i>");
                    html.Append("</div>");
                    html.Append("<div class='panel-body'>");
                    html.Append("<i class='fa fa-building fa-border' aria-
hidden='true'></i>"); html.Append(" " + queryResult[3].ToString() + "  ");
                    html.Append("<i class='fa fa-compass fa-border' aria-
hidden='true'></i>"); html.Append(" " + queryResult[4].ToString() + "  ");
                    html.Append("<i class='fa fa-rupee fa-border' aria-
hidden='true'></i>"); html.Append(" " + queryResult[5].ToString() +
"  <hr/>");
                    html.Append("<i class='fa fa-paperclip fa-1x fa-pull-left ' aria-
hidden='true'></i>");
                    html.Append("<p>"); html.Append(queryResult[1].ToString());
html.Append("</p><hr/>");
                    html.Append("<i class='fa fa-hand-o-right fa-1x fa-pull-left ' aria-
hidden='true'></i>"); html.Append(" Experience Required of " +
queryResult[2].ToString() + "  <hr/>");
                    html.Append("</div>");
                    html.Append("</div>");
                    html.Append("</div>");
                }
            }
            else
            {
                html.Append("<div class='row'><div class='col-lg-12'>");
                html.Append("<div class='alert bg-danger' role='alert'><svg class='glyph
stroked cancel'><use xlink:href='#stroked-cancel'></use></svg> No Jobs are posted <a
href= '#' class='pull-right'><span class='glyphicon glyphicon-
remove'></span></a></div>");
                html.Append("</div></div>");
            }

            html.Append("</div></div>");
            PlaceHolder1.Controls.Add(new Literal { Text = html.ToString() });

            conn.Close();
        }
        catch (Exception ex)
        {
            Label1.Text = "Please Report to admin: " + ex.Message;
            Label1.ForeColor = System.Drawing.Color.Red;
        }
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
    }
}
```

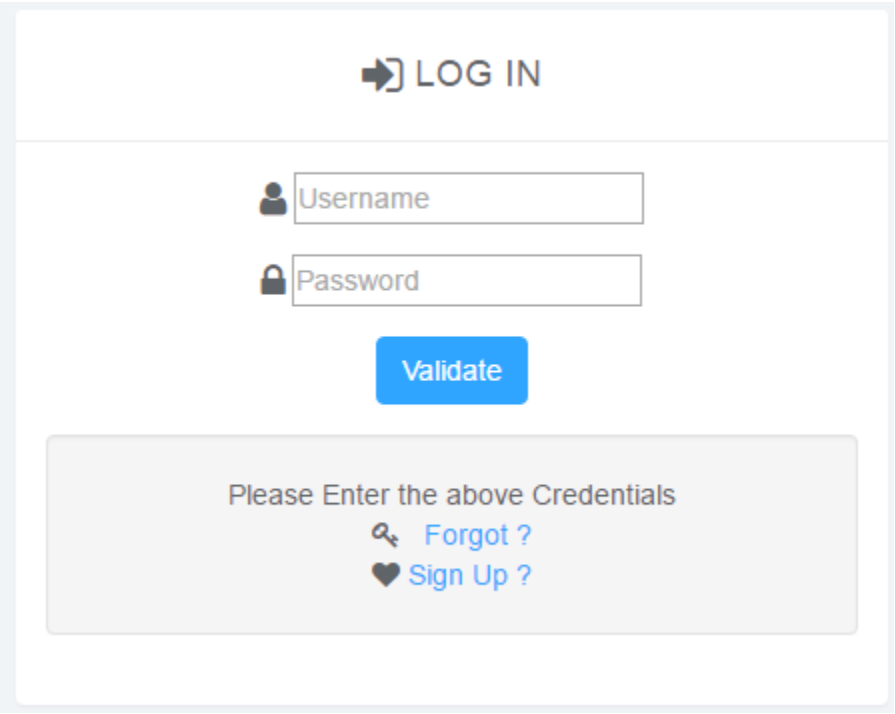
JOBPOST.CS

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            SqlConnection conn = new
SqlConnection(ConfigurationManager.ConnectionStrings["jobportalConnectionString"].Connect
ionString);
            conn.Open();
            WellSecond.Visible = true;
            Label1.Text = "Hey";
            string jdesgn = TextBoxDesg.Text;
            string jdesc = TextAreaDesc.InnerText;
            string jexp = TextBoxExp.Text;
            string jcomp = TextBoxCompa.Text;
            string jloc = TextBoxLoca.Text;
            string jsal = TextBoxSalary.Text;

            SqlCommand insertCommand = new SqlCommand("INSERT INTO [jobdetails]
(jdesgn,jdesc,jexp,jcomp,jloc,jsal) VALUES (@jdesgn,@jdesc,@jexp,@jcomp,@jloc,@jsal)",
conn);
            insertCommand.Parameters.Add("@jdesgn", SqlDbType.VarChar, 255,
"jdesgn").Value = jdesgn;
            insertCommand.Parameters.Add("@jdesc", SqlDbType.VarChar, 255, "jdesc").Value
= jdesc;
            insertCommand.Parameters.Add("@jexp", SqlDbType.VarChar, 255, "jexp").Value =
jexp;
            insertCommand.Parameters.Add("@jcomp", SqlDbType.VarChar, 255, "jcomp").Value
= jcomp;
            insertCommand.Parameters.Add("@jloc", SqlDbType.VarChar, 255, "jloc").Value =
jloc;
            insertCommand.Parameters.Add("@jsal", SqlDbType.VarChar, 255, "jsal").Value =
jsal;

            int queryResult = insertCommand.ExecuteNonQuery();

            if (queryResult > 0)
            {
```

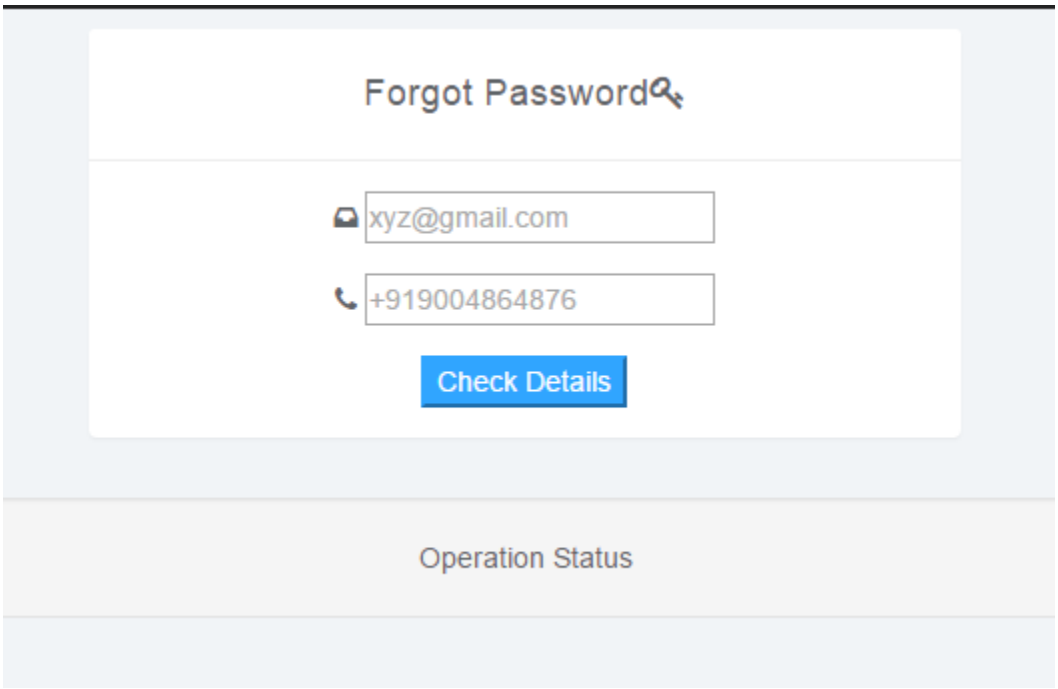
```
                WellSecond.Visible = true;
                Label1.Text = "Data Inserted Successfully :) ";
                Label1.ForeColor = System.Drawing.Color.Green;
            }
            conn.Close();
        }
        catch (Exception ex)
        {
            WellSecond.Visible = true;
            Label1.Text = "Please Refer to Error: " + ex.Message;
            Label1.ForeColor = System.Drawing.Color.Red;
        }

    }


}
```

OUTPUT

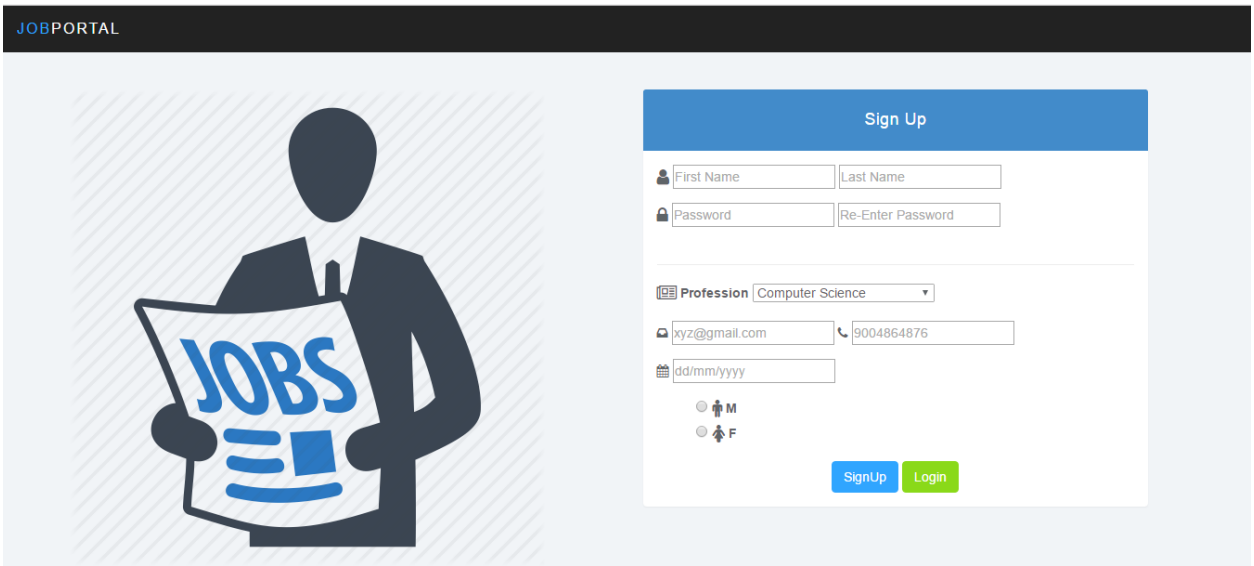
LOGIN



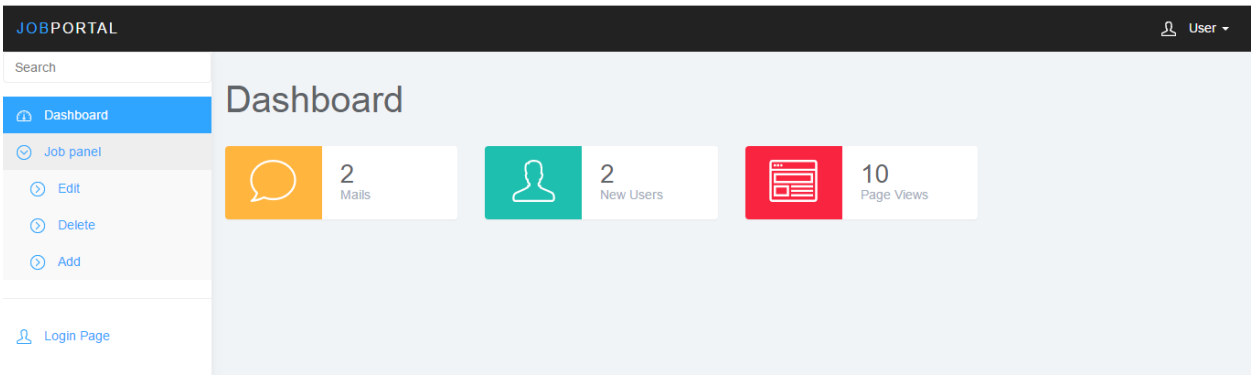
FORGOT PASSWORD (UPDATE)



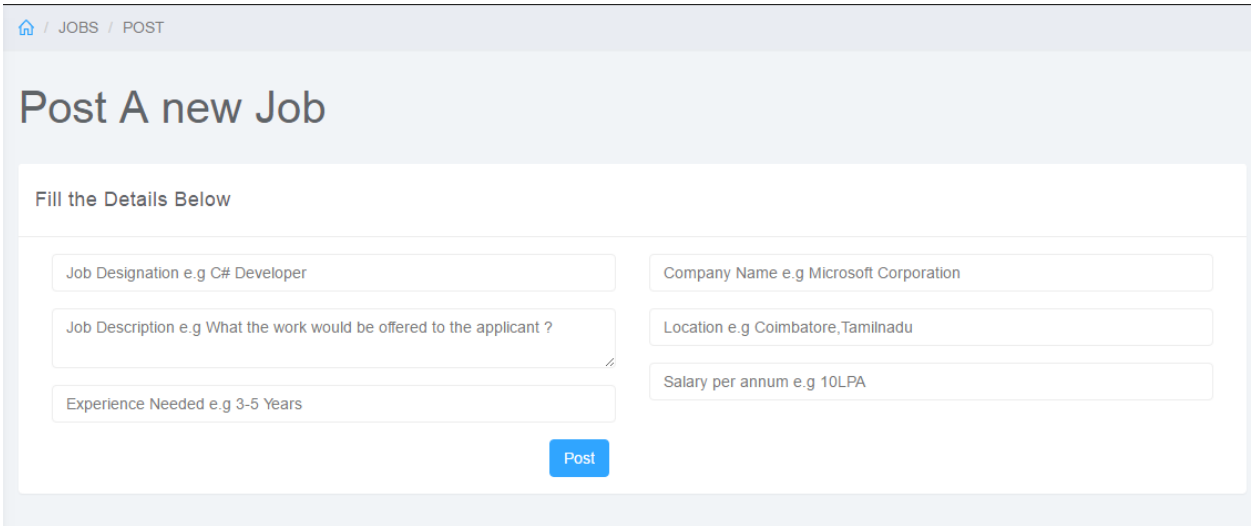
SIGN UP(INSERT)



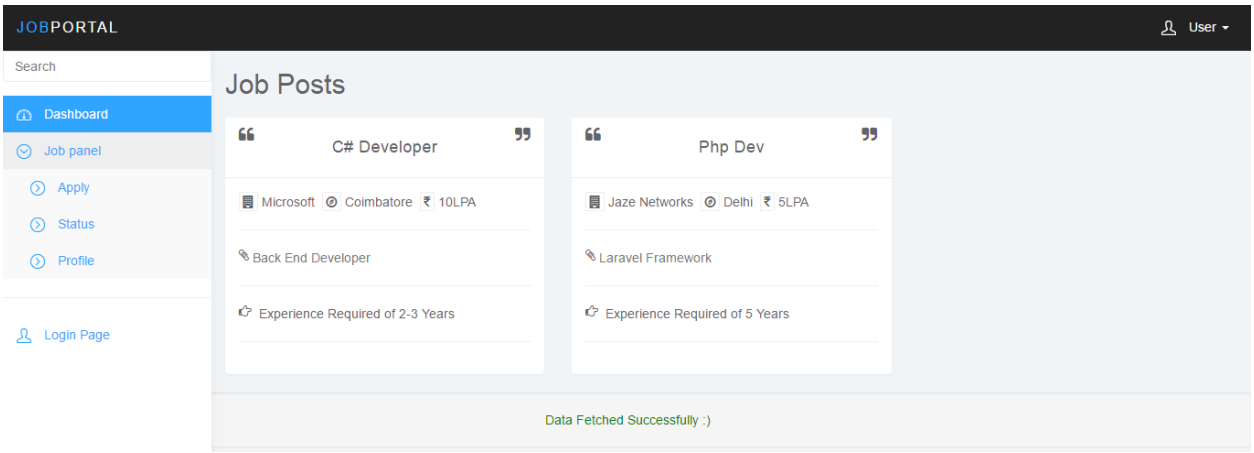
ADMIN PANEL



POST A JOB -ADMIN PANEL



VIEW JOB – USER PANEL



Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[Dr J Anitha/Dr. S.P. Jeno Lovesum]

Ex. No. 10	Advanced Web Design		
Date of Exercise	02.11.2016	Date of Upload	06.11.2016

Aim

To develop **Address Book Maintenance System** using C# by integrating Database with proper authentication

Description

Two different kinds of controls:

- User controls
- Custom controls

To make use of the controls you have created, you simply copy the assemblies containing those controls along with the rest of the code. You can even place frequently used controls in an assembly located in the **global assembly cache (GAC)** on the Web server, so that all Web applications on the server have access to them

User Controls

- Create using ASP.NET code
- After you have created a user control you can reuse it in multiple ASP.NET pages.
- Then define properties and methods for user controls.

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs" Inherits="WebUserControl" %>
```

- There is no HTML code present, and in particular no < form > element. This is because user controls are inserted inside ASP.NET forms in other files and so don ’ t need a < form > tag of their own. Place the code in <% @ Control %> directive generated in .ascx.
- The CodeFile attribute specifies the code -behind file and Inherits specifies the class defined in the code -behind file from which the page inherits. To do this, you use the < % @ Register % > directive at the top of the code in Default.aspx , as follows:

```
<%@ Register TagPrefix=”pcs” TagName=”UserC1” Src=”WebUserControl.ascx” %>
```

- The TagPrefix and TagName attributes specify the tag name to use (in the form < TagPrefix:TagName >), and you use the Src attribute to point to the file containing your user control.

- Now use the control by adding the following element in Default.aspx :

```
<pcs:UserC1 Runat=”server” ID=”myUserControl”/>
```

Custom Controls

Entirely self -contained in C# assemblies, don’t need to go through the process of assembling a user interface (UI) in an .ascx file.

To get the most customizable behavior for your custom controls,

- Derive a class from System .Web.UI.WebControls.WebControl
- Extend the functionality of an existing control, creating a derived custom control.
- Group existing controls together, create a composite custom control.

Master Pages

Provides an excellent way to make your Web sites easier to design. Master pages are created in files with the extension .master , and can be added via the Web site -> Add New Item

The differences are:

- A <% @ Master % > directive is used instead of a <% @ Page % > directive, although the attributes are the same.
- A ContentPlaceHoldercontrol with an ID of head is placed in the page header.
- A ContentPlaceHoldercontrol with an ID of ContentPlaceHolder1 is placed in the page body.

For an .aspx page to use a master page, you need to modify the < % @ Page % > directive as follows:

```
< %@ Page Language=”C#” AutoEventWireup=”true” CodeFile=”Default.aspx.cs” Inherits=”_Default” MasterPageFile=”~/MasterPage.master” Title=”Page Title” % >
```

MasterPage used to create a consistent layout for the pages in your application:

- A single master page defines the look and feel and standard behavior that you want for all of the pages
- Then create individual content pages that contain the content you want to display.
- When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.

Accessing Master Page Content from Web Pages

To access the master page from code in your Web page

- Use the Page.Master property, which will return a reference to the master page in the form of a MasterPage object.
- Use the MasterPage.FindControl() method to locate controls on the master page by their identifier.
- This enables you to manipulate content on the master page that is outside of content placeholders

VALIDATION CONTROLS

To check the **required field**

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="Not to be empty" ControlToValidate="TextBox1"> </asp:RequiredFieldValidator>
```

To check the **comparison**

```
 <asp:CompareValidator ID="CompareValidator1" runat="server" ErrorMessage="Password and re-type password must be same" ControlToCompare="TextBox2" ControlToValidate="TextBox3">  </asp:CompareValidator>
```

To check the **range**

```
<asp:RangeValidator ID="RangeValidator1" runat="server" ErrorMessage="Between 25 to 50" ControlToValidate="TextBox4" MaximumValue="50" MinimumValue="25">
</asp:RangeValidator>
```

To check the **regular expression**

```
<asp:RegularExpressionValidator runat="server" ErrorMessage="10-digit" ControlToValidate="TextBox5" ValidationExpression="[0-9]{10}">
</asp:RegularExpressionValidator>
```

PROGRAM

SIGNUP PAGE VALIDATION CONTROLS AND USER CONTROLS

```
<%@ Page Title="Login" Language="C#" MasterPageFile="~/MasterPage2.master"
AutoEventWireup="true" CodeFile="SignUpPage.aspx.cs" Inherits="LoginPage" %>
<%@ Register TagPrefix="pcs" TagName="UserC1" Src="~/WebUserControl.ascx"%>




<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
    <title>Login</title>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <br /><br />
    <div class="row ">
         <div class="col-lg-6  text-center">
            <pcs:UserC1 runat="server" ID="myusercontrol"/>
        </div>
              <div class="col-lg-5 pull-left">
                     <div class="login-panel panel panel-primary">
                            <div class="panel-heading text-center">Sign Up</div>
                            <div class="panel-body">
                                          <fieldset >
                                                 <div class="form-group">
                                 <i class="fa fa-user fa-lg"></i>
                                 <asp:TextBox ID="TextBoxFname" runat="server"
Required="true" placeholder="First Name" ></asp:TextBox>
                                 <asp:TextBox ID="TextBoxLname" runat="server"
Required="true" placeholder="Last Name" ></asp:TextBox>

                                 <asp:RequiredFieldValidator ID="RequiredValidator1"
runat="server" ErrorMessage="Please Fill the fields" ControlToValidate="TextBoxLname">
</asp:RequiredFieldValidator>
                                                 </div>

                                                 <div class="form-group">
                                 <i class="fa fa-lock fa-lg"></i>
                                 <asp:TextBox ID="TextBoxPwd" runat="server"
Required="true" placeholder="Password" ></asp:TextBox>
                                 <asp:TextBox ID="TextBoxRePwd" runat="server"
Required="true" placeholder="Re-Enter Password" ></asp:TextBox>
                                 <asp:CompareValidator ID="CompareValidator1"
runat="server" ErrorMessage="Password and re-type password must be same"
ControlToCompare="TextBoxPwd" ControlToValidate="TextBoxRePwd"> </asp:CompareValidator>
                                                         <hr />
                            </div>

                            <div class="form-group">
                                 <i class="fa fa-newspaper-o fa-lg"></i>
                                 <label>Profession</label>
```

```
                                <asp:DropDownList ID="DropDownListProfession"
runat="server">
                                    <asp:ListItem>Computer Science</asp:ListItem>
                                    <asp:ListItem>Electronics
Communication</asp:ListItem>
                                    <asp:ListItem>Electrical Engineering</asp:ListItem>
                                    <asp:ListItem>Biotechnology</asp:ListItem>
                                    <asp:ListItem>Media Technology</asp:ListItem>
                                </asp:DropDownList>
                            </div>
                            <div class="form-group">
                                <i class="fa fa-inbox"></i>
                                <asp:TextBox ID="TextBoxMail" runat="server"
Required="true" placeholder="xyz@gmail.com" ></asp:TextBox>
                                <i class="fa fa-phone"></i>
                                <asp:TextBox ID="TextBoxPhone" runat="server"
Required="true" placeholder="9004864876" ></asp:TextBox>
                                <asp:RegularExpressionValidator runat="server"
ErrorMessage="10-digit" ControlToValidate="TextBoxPhone" ValidationExpression="[0-
9]{10}"> </asp:RegularExpressionValidator>

                            </div>
                            <div class="form-group">
                                <i class="fa fa-calendar"></i>
                                <asp:TextBox ID="TextBoxDob" runat="server"
Required="true" placeholder="dd/mm/yyyy" ></asp:TextBox>
                        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ErrorMessage="Not to be empty" ControlToValidate="TextBoxDob">
</asp:RequiredFieldValidator>
                            </div>
                            <div class="form-group col-lg-offset-1">
                            <asp:RadioButtonList ID="RadioButtonListGender"
runat="server">
                                   <asp:ListItem Value="M"> <i class="fa fa-male fa-
lg"></i> M</asp:ListItem>
                                   <asp:ListItem Value="F"> <i class="fa fa-female
fa-lg"></i> F</asp:ListItem>
                                </asp:RadioButtonList>
                                </div>
                            <div class="row text-center">
                             <asp:Button ID="ButtonEnterDetails" CssClass="btn btn-
primary" runat="server" Text="SignUp" OnClick="ButtonEnterDetails_Click"  />
                            <asp:Button ID="Button1" CssClass="btn btn-success"
runat="server" Text="Login" OnClick="Button1_Click"   />
                            </div>
                                        </fieldset>

                           </div>
                       </div>

               </div><!-- /.col-->

       </div><!-- /.row -->


     <p></p>
    <div class="well text-center">
```

```
        <asp:Label ID="Label1" runat="server" Text="Operation Status"></asp:Label>
    </div>
    <p> </p>
    <p> </p>

</asp:Content>
```

WEB USER CONTROL.ASCX

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs"
Inherits="WebUserControl" %>
<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Caption.png" />
<br />
```

MASTER PAGE .MASTER

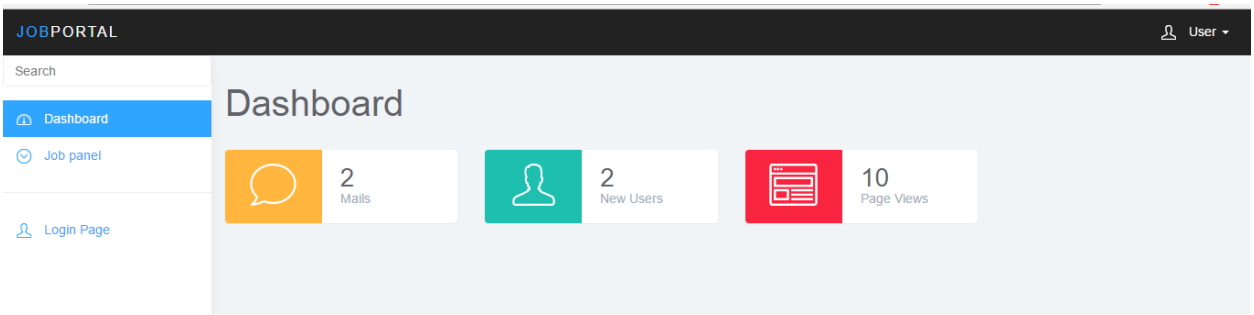
```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
Inherits="MasterPage" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">

<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Lumino - Dashboard</title>

<link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/datepicker3.css" rel="stylesheet">
<link href="css/styles.css" rel="stylesheet">
    <link href="font-awesome-4.7.0/font-awesome-4.7.0/css/font-awesome.min.css"
rel="stylesheet" />

<!--Icons-->
<script src="js/lumino.glyphs.js"></script>

<!--[if lt IE 9]>
<script src="js/html5shiv.js"></script>
<script src="js/respond.min.js"></script>
<![endif]-->

    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
              <div class="container-fluid">
                     <div class="navbar-header">
                            <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#sidebar-collapse">
                                   <span class="sr-only">Toggle navigation</span>
                                   <span class="icon-bar"></span>
```

```
                                <span class="icon-bar"></span>
                                <span class="icon-bar"></span>
                        </button>
                        <a class="navbar-brand" href="#"><span>JoB</span>PORtal</a>
                        <ul class="user-menu">
                                <li class="dropdown pull-right">
                                        <a href="#" class="dropdown-toggle" data-
toggle="dropdown"><svg class="glyph stroked male-user"><use xlink:href="#stroked-male-
user"></use></svg> User <span class="caret"></span></a>
                                        <ul class="dropdown-menu" role="menu">
                                                <li><a href="#"><svg class="glyph stroked
male-user"><use xlink:href="#stroked-male-user"></use></svg> Profile</a></li>
                                                <li><a href="#"><svg class="glyph stroked
gear"><use xlink:href="#stroked-gear"></use></svg> Settings</a></li>
                                                <li><a href="#"><svg class="glyph stroked
cancel"><use xlink:href="#stroked-cancel"></use></svg> Logout</a></li>
                                        </ul>
                                </li>
                        </ul>
                </div>

        </div><!-- /.container-fluid -->
    </nav>



    <div id="sidebar-collapse" class="col-sm-3 col-lg-2 sidebar">
        <div class="form-group">
                        <input type="text" class="form-control" placeholder="Search">
                </div>
        <ul class="nav menu">
                <li class="active"><a href="ViewJobs.aspx"><svg class="glyph stroked
dashboard-dial"><use xlink:href="#stroked-dashboard-dial"></use></svg> Dashboard</a></li>
                <li class="parent ">
                        <a href="#">
                                <span data-toggle="collapse" href="#sub-item-1"><svg
class="glyph stroked chevron-down"><use xlink:href="#stroked-chevron-
down"></use></svg></span> Job panel
                        </a>
                        <ul class="children collapse" id="sub-item-1">
                                <li>
                                        <a class="" href="#">
                                                <svg class="glyph stroked chevron-
right"><use xlink:href="#stroked-chevron-right"></use></svg> Apply
                                        </a>
                                </li>
                                <li>
                                        <a class="" href="#">
                                                <svg class="glyph stroked chevron-
right"><use xlink:href="#stroked-chevron-right"></use></svg> Status
                                        </a>
                                </li>
                                <li>
                                        <a class="" href="#">
                                                <svg class="glyph stroked chevron-
right"><use xlink:href="#stroked-chevron-right"></use></svg> Profile
                                        </a>
```

```
                                    </li>
                                </ul>
                            </li>
                            <li role="presentation" class="divider"></li>
                            <li><a href="LoginPage.aspx"><svg class="glyph stroked male-
user"><use xlink:href="#stroked-male-user"></use></svg> Login Page</a></li>
                        </ul>

                    </div><!--/.sidebar-->

                     </div>

                     <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

                     </asp:ContentPlaceHolder>
                </form>
                <script src="js/jquery-1.11.1.min.js"></script>
                <script src="js/bootstrap.min.js"></script>
                <script src="js/chart.min.js"></script>
                <script src="js/chart-data.js"></script>
                <script src="js/easypiechart.js"></script>
                <script src="js/easypiechart-data.js"></script>
                <script src="js/bootstrap-datepicker.js"></script>
                <script>
                          $('#calendar').datepicker({
                          });

                          !function ($) {
                              $(document).on("click","ul.nav li.parent > a > span.icon", function(){
                                  $(this).find('em:first').toggleClass("glyphicon-minus");
                              });
                              $(".sidebar span.icon").find('em:first').addClass("glyphicon-plus");
                          }(window.jQuery);

                          $(window).on('resize', function () {
                            if ($(window).width() > 768) $('#sidebar-collapse').collapse('show')
                          })
                          $(window).on('resize', function () {
                            if ($(window).width() <= 767) $('#sidebar-collapse').collapse('hide')
                          })
                </script>
            </body>
            </html>
```

OUTPUT

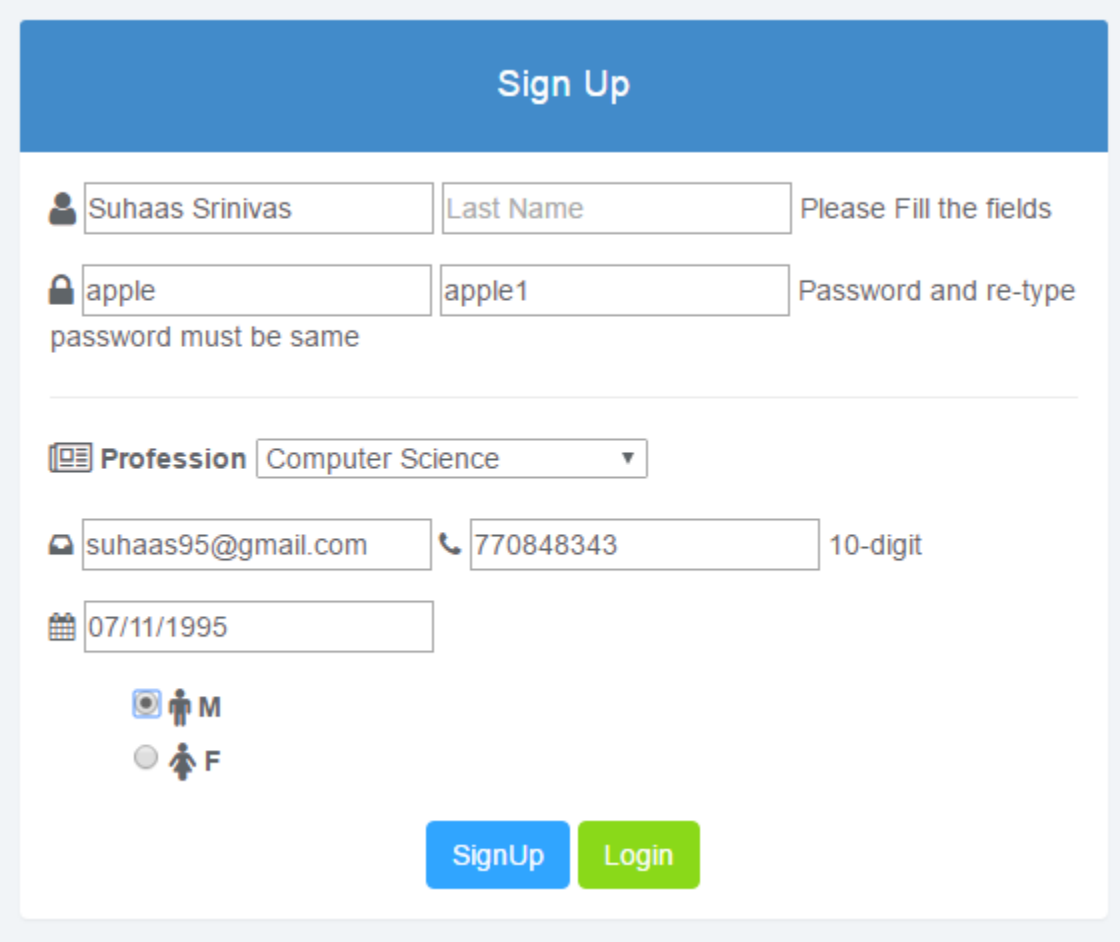
MASTER PAGE WITH TREE VIEW



CUSTOM CONTROLS -IMAGE

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="WebUserControl.ascx.cs" Inherits="WebUserControl" %>
<asp:Image ID="Image1" runat="server" ImageUrl="~/Images/Caption.png" />
<br />
```

SIGN UP PAGE -VALIDATION CONTROLS



Result

The above programmed is compiled successfully and the screenshots are well described with successful outputs and constraints.

[Dr J Anitha/Dr. S.P. Jeno Lovesum]