Table of Contents

Task 1

Prepare a report discussing the principles; characteristics and features of object oriented programming. Consider the following features like data encapsulation, data abstraction, inheritance, constructor, etc. [P 1.1]

1. Introduction

Programming paradigm is approach or model of programming according to which the process of the programming drives (Jana, 2005). Different programming paradigm has different programming pattern and techniques. There are basically four main programming paradigms exist; procedural, functional, logic and Object oriented. In this report, I have discussed about principles, features and characteristics of object oriented programming paradigm.

Object Oriented Programming i.e. OOP is a software development approach in which software structures are based on the interaction between objects. According to Rouse (2008), OOP is programming pattern in which programming is organized around data and object rather than action or logic. Objects in software development are similar to real life object. Like in real life, we (object) interact with other objects like car (object), TV (object), books (object) to achieve different tasks. Similar interaction between different objects occurs in object oriented programming approach to achieve required task. Additionally, in OOP approach data is critical element and are not allowed to run around the programming freely (Balagurusamy, 2006).

According to Virginia Tech College of engineering (n.d.) OOP is evolution for software engineering field. Implementation of OOP concept in the project leads to short development time scale and increased productivity. It offers great flexibility and manageability to the software. Unlike older programming paradigm like procedural programming, OOP is less complex, reliable and easy to understand. It supports programming languages like Ruby, C++, VB.net, C#, Java, etc.

2. Terminologies in Object Oriented Programming

Method

Method is group of statements written to perform a task. According to Jones (1991), a method is blocks of codes that performs a task and return the control to its caller. There are two steps required to use a method; define method and call method.

```
    public void Welcome(string Name){
      Console.WriteLine("WELCOME TO OUR COLLEGE\n" + Name);
      Console.ReadLine();
    }
```

EXAMPLE I: METHOD in C# Console

Object

Objects in OOP are instance of class. Object is item that self-contained data members (properties) and methods in which properties denotes what that object understand and method defines what that object can do.

```
using System;
namespace project1
{
  class example2
  {      string FirstName="Sachin";
    string LastName="Tendulkar";
    public void FullName(){                    //Method Defined
      Console.WriteLine("FullName:\t"+FirstName+" "+LastName);
      Console.ReadLine();
    }  }
  class program
  {      static void Main(string[] args)
    {        example2 obj = new example2();   // object of class example2 is created
      obj.FullName();       //method called using (.) operator and object
   }  }
}
After Code Executed:
Full Name:       Sachin Tendulkar
```

EXAMPLE II in C#

In Example II, Object named obj which is instance of class example2 is created inside main function. This object demonstrates the characteristics and behavior as set by its class; example2.Now to call method inside the example2 class (.) operator with object obj is used. A program may have many

different types of objects, each object derived from definite class of that type.

Class

When developing objects oriented programming, class is building block of code, which describes the behavior and feature of object of that class (Blackwasp, 2007). A program may have more than 1 object with same properties and behaviors. Class provides blueprint for objects. Each object is created from class.

Class is advanced and extended concept of data structure with additional capability of containing data member and methods (CPLUSPLUS, n.d.). In OOP class is template for creating objects. There can also be subclass. These subclasses can also have their own data member and methods. And the structure between class and subclass is known as class hierarchy. In Example II, example2 is a class which provides blueprint for its object which is created inside main function to call the methods of example2. Unlike data structures, class example2 contains method name FullName ().

Constructor

Constructor is a method which is used for initializing the object. Constructor has same name as their class and has no return type, not even void. Constructor is automatically called when object is created.

```
class example2        // class named example2
{
   string Name;
   public void Welcome(string SetName){
      Name = SetName;
     Console.WriteLine("WELCOME TO OUR COLLEGE\n" + Name);
     Console.ReadLine();  }
   public example2()            //Constructor is created
   {          Name = "TEST_NAME";     // initialize the value of Name to TEST_NAME
   }   }
```

EXAMPLE III CONSTRUCTOR in C#

In Example III, constructor of class example2 is created to initialize the value of Name. Whenever the object of this class will be created, value of string Name will be initialized to “TEST_NAME”.

3. Principles of Object Oriented Programming

There are four major principles that make a program object oriented. Encapsulation, Abstraction, Inheritance and polymorphism are core principles or feature of OOP. Below in this report, each principle has been explained.

Encapsulation

Encapsulation is providing the objects members protective custody or in simple words, information hiding. In C#, encapsulation is bundling data and method inside class to protected them from being modified from outside. A class has 3 parts, public, protected and private. Only public part can be accessed using object and (.) operator. Private part of the class can only be accessed by methods inside the class. Hence, it protects its member from being exposed to other parts of the program and only allows access to public part.

```
#include <iostream>
using namespace std;
class multiply{
  public:          Multiply(int x = 1) // constructor      // interface to outside world
    {       total = x;      }
    void getMultiply(int number)
    {         total *= number;      }
    int getTotal()
    {         return total;      };
  private:      // hidden data from outside world
    int total;};
int main( )
{   multiply object1;        //object of class Adder
    object1.getMultiply(10);
  object1.getMultiply (20);
  object1.getMultiply (30);
  cout << "Total: " << object1.getTotal() <<endl;
  return 0;}
[When Code is compiled and executed:
Total: 6000
```

EXAMPLE IV ENCAPSULATION in C++

In above Example IV, two different specifier, are used in class multiply to make its data member public and private. Keyword public: is used to make data member public and outside program can access them using class's object. But private data members which is declared after keyword private: is protected from outside program and only accessible from methods inside the class multiply.

Abstraction

Abstraction is one of the core features of OOP which provides only necessary information to user and hides its internal structure (TUTORIALSPOINT, n.d.). This feature is closely linked with encapsulation in which un-necessary data are hidden from the user. Like in real life scenario, we can use mobile phone for different purposes like making call, SMS, gaming or listening music but we do not know internal details of that mobile phone. Similarly, in OOP user are allowed to use public method of class without knowing internal details of that class.

```
#include <iostream>
using namespace std;
int main( )
{   cout << " Hi, Welcome to C++ programming " <<endl;
  return 0;
}
[Code when compiled and executed]
Hi, Welcome to C++ programming
```

EXAMPLE V ABSTRACTION in C++

Above in the Example V, programmer does not know the internal structure of cout but he/she can use it anywhere in the program. Similarly, another example of abstraction can be use public method of a class outside that class without knowing their internal details. Abstraction helps to protect data from un-necessary modification and provide easiness in programming process.

Inheritance

Inheritance is one of the core principles of OOP that allows creating a class with ability to inherit instant variables and method from existing class. 4inv (n.d.) states, the derived class is known as subclass and other class is called base class. Here subclass inherit the methods and instant variables of base class and may add more methods and instant variables. According to INTROPROGRAMMING (n.d.), this feature improves code readability and enables the reuse of functionality.

```
using System;
namespace project1
{   namespace InheritanceExample
  {     class Calculation   //Base Class
    {         public void Length(int l)
      {             length = l;
      }         public void Height(int h)
      {             height = h;
      }         protected int length;
      protected int height;     }
    class AreaCalculation : Calculation      // Derived class
    {         public int Area()
      {             return (length * height);
      }     }
    class Program
    {         static void Main(string[] args)
      {             AreaCalculation result = new AreaCalculation();
        result.Length(10);
        result.Height(20);
        // Print the area of the object.
        Console.WriteLine ("Total area: " + result. Area());
        Console.ReadLine();
      }     }
  }}
```

[When compiled and Executed:]

Total Area: 200

EXAMPLE VI INHERITANCE, C# CONSOLE

In above Example VI, AreaCalculation is derived class (Subclass) from base class Calculation. AreaCalculation inherit the data members of its base class and add new method named Area (). Now from main function only Area () can be called using object of AreaCalculation class to calculate the area. This program also demonstrated the abstraction and encapsulation principle of OOP properly.

Some impacts of Inheritance are follows (ERPBASIC, 2012):

- Code Reusability

  Public members of base class are inherited in derived class and can be used hence we do not need to rewrite the code again.
- Data Hiding

  Only public and protected data are available for the derived class that ensures data hiding of the data in private section.
- Overriding

  Methods of the base class can be overridden if necessary by the use of method in subclass.

Polymorphism

Polymorphism is that principle of Object Oriented Programming that allows single name of method but multiple functionalities. It is derived from the combination of two words, Poly means many and morph mean forms. In more simple word it is creating more than one method with same name but giving different behaviors.

```
using System;
namespace project1
{    namespace PolymorphismExample
   {       class A{           public int Sum(int a, int b)
          {              return a + b;          }
          public double Sum(int z, int x, int c)
          {              return z + x + c;          }       }
      class RectangleTester
      {           static void Main(string[] args)
          {              A Obj1 = new A();
             int x=Obj1.Sum(5, 10);
             double y = Obj1.Sum(5, 10, 15);
             Console.WriteLine("RESULT 1: "+x+"\nResult 2: "+ y);
             Console.ReadKey();          }       }      }}
[After Code Compiled And Executed]
Result 1: 15
Result 2: 30
```

EXAMPLE VII POLYMORPHISM, C# CONSOLE

Example VII demonstrates the polymorphism feature of OO programming. Two methods with same name Sum is created but with different parameter passing also demonstrating method overloading. To call the different methods using object different parameter is passed and different result is achieved which perfectly demonstration the principle of polymorphism where method Sum has same name but different functionalities.

Task 2[P2.1, M1]

Identify the various objects and the data and the file structures required to implement the solution for the given problem. Justify the use of these objects, data and the file structure you are going to implement for developing the design and the program. Consider the various classes, objects and methods, etc.

Introduction

To implement the solution for the given problem of ISMT College, suggested application "ISMT FINANCE" should be designed on three tier architecture. Three tiers architecture contains Business layer, Data Access layer and Presentation Layer. There are several advantages of implementing three tiers architecture in the application. Advantages of three tiers architecture can be listed as below (Hagos, 2008):

- Migration to new application type is faster (to mobile, web, tab etc.)
- Application Code Management is Better
- It is possible and easier to make changes in presentation layer without affecting other two (business and Data Access Layer)
- Data is more secure
- In one tier, if application fails chances of data loss is greatly higher than in three tier application where database is not directly accessed.

Three tiers architecture in ISMT Finance application is shown in table 1:

SN	Tier Type	Name in the Application	Purpose of This Tier
1.	Business Layer	BusinessLogicLayer	This layer will be for applying business logics in the system, it acts as the medium between presentation layer and DataAccessLayer. Database activities will be verified via this layer.
2.	Data Access Layer	DataAccessLayer	This layer will be used for Accessing the Database and perform activities like save, update or delete.

3.	Presentation Layer	ISMT_MANAGEMENT	This layer will be available of front end. User will be able to interact with software via this layer. This layer is group of different forms like, login, student management etc.

Table 1

Class Library

Class library is nothing but a library of classes. For example, data class for faculty and user can be put together in a single class library for the efficiency of the program structure. In ISMT Finance Application there are three class libraries. In table 2, all three class libraries and their purposes are described.

Class Libraries in this Solution		
SN.	**Name of Class Library**	**Reason for building the Class Library**
1.	DataAccessLayer	This library contains the classes that contain connection string to connect with database, maintain database, and perform queries like insert, update or delete. For example, Class Batch in the DataAccessLayer is coded for performing queries related to batch management. This layer is able to perform actions like save, update or delete as per user's instruction.
2.	BusinessLogicLayer	This Library is for classes that contain necessary Code for validating database operation in DataAccessLayer. This layer is used for checking if operations like insert, delete or update actually taken place or not. If yes then it returns value true else false. For example, BusinesLogicClass has method called manageClass, which validate the database operation took place in Class Batch and returns value true of falls.

3.	ISMTR_MANAGEMENT	This library contains different forms to enable user interaction with the application and contains various controls. This library sends the commands of save, edit or delete to logic or data access layer. For example, through BatchForm, user instruct software to perform various task like save, delete or update, if value return from business logic layer is true is shows success message, else shows error message.

Table 2

Namespace

Namespace is logical grouping of names used within program (WOWWIKI, n.d.). It is container that contains identifiers (knows as name, symbols). Namespace encapsulates the codes that to gain access to that code, we need to first reference the namespace to which we are referring. The inbuilt namespaces used in this solution is shown below in table 3 including the reasons to include those namespaces.

Namespaces in this solution		
SN.	**Name of Namespace**	**Reason for using this Namespace**
1.	System	Contains essential classes and base classes that defines commonly used data types, attributes and event handlers etc.
2.	System.Configuration	Provides classes and interfaces that allows to configuration settings and handle errors in configuration files
3.	System.Data;	Mostly consists of the classes that used for ADO.NET. Used for Database management.
4.	System.Data.Sqlclient	Contains classes used for SQL server connection management
5.	System.Text	This namespace contains classes representing Unicode, UTF-7, ASCII and UTF-8 character encodings.
6.	System.Drawing	This namespace is used for GDI+ basic graphics functionality.
7.	System.Collection.Generic	This namespace contains interfaces and classes that define generic collections.

8.	System.Windows.Form	This namespace contains classes for creating Windows-based applications that enables us to take advantage of the rich user interface features available in Microsoft Windows operating system.
9.	System.Threading.Tasks	System.Threading provides classes and interfaces that enable multithreaded programming. This namespace provides types that simplify the work of writing concurrent and asynchronous code.
10.	System.Linq	This namespace includes classes and interfaces that allow queries that use Language-Integrated Query (LINQ).
11.	System.componentModel	Includes classes that are used for implementing the run-time and design-time behavior of controls.

Table 3

Inbuilt classes and methods:

Inbuilt Classes in Program		
S.N.	**Name of class**	Use of Class
1.	SqlCommand	Represents a Transact-SQL statement used for SQL database store procedure
2.	SqlConnection	Used for SQL server connection
3.	SqlDataAdapter	Is set of data command and data connection, used for filling DataSet
4.	ExecuteNonQuery	This method is used for checking the number of rows effected in database
5.	ExecuteReader	This method is used for building SqlDataReader
6.	DataTable	DataTable is used for storing data from database. And display in grid view.
7.	DataSet	This is collection of DataTable. So we can store so many data table in single collection.
8.	MessageBox	MessageBox creates a small popup form with some information like warning or error.
9.	DialogResult	This is used for creating message box confirmation. Action depends on user choice
10.	TryCatch	This is used for exceptional handling in our system.
11.	SqlDataReader	It is used for reading database rows one-by-one

Table 4

Task 3 [P2.2, M2]

Design the object oriented programming solution to the given problem. Your design should include pseudo code, algorithm and flowchart to address the following problem:

a. **Maintain Student's information.**
b. **Maintain the Financial status of the student.**
c. **Managing the inquiry system.**

---

1. Pseudo Code

Pseudo code is not a programming language, but a comfortable method of relating a sequence of program development. It is used for creating an outline or draft for the program to be developed and cannot itself be compiled into an executable program. It helps the programmer to stay on track and concentrate on what is needed to be done in program.

The pseudo code for the ISMT Finance Management system is written below.

1. Start
2. Open up the Software
3. Show the Login form
4. Read Role, Username and password
5. Check the supplied details Role, Username and Password in database
6. If all parameter details are correct as in database show up main form
    - If Role is Administrator or (Admin), show Admin Main Form
    - For other roles show normal/Reception Main Form
7. Allow all ability and controls to the admin User
    - If user choose Fee Management, show the fee management Form
    - If user choose Student Management, show the Student management Form
    - If user choose Batch Management, show the Batch management Form
    - If user choose Transaction Management, show the Transaction management Form
    - If user choose Role Management, show the Role management Form
    - If user choose User Management, show the User management Form
    - If user choose Manage Faculty, show the Faculty management Form
    - If user choose Awarding Body Management, show the Awarding Body Form

- If user choose Transaction Management, Show Transaction form
- If user choose Inquiry, show Inquiry form

8. Allow only Inquiry and Student Management Control to other User
    - If user choose Student Manage, show student Form
    - If user choose Inquiry Manage, show Inquiry Form
9. if User Click log off, close the main form and show up login form asking credential
10. If user clicks on exit, exit the application.
11. End.

2. Algorithm

Algorithm is step-by-step problem-solving procedure. The algorithm for the ISMT Finance Management system is written below.

1. Start
2. Open up the software
12. Display login form asking three login parameter
    - Role
    - Username
    - Password
3. Read user supplied values form text boxes txtUsername ,txtUserPassword and combo box cmbUserRole
4. Check credentials, if (user exists)
   {If (cmbUserRole.Text==”Admin”) {
   Show admin main form}
   else if (cmbUserRole.Text==”Account”) {show account main form}
   else {show reception account}
5. (if user click on Admin Panel){
   Show 4 menus
    - Manage Role
    - Manage User
    - Manage Awarding Body
    - Manage Faculty }
6. If (user click on manage Role){

Show the Role Management Form{ Display data in grid view dgvRole

- If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
- If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
- If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

7. If (user click on manage User){

   Show the User Management Form{ Display data in grid view dgvUser

   - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
   - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
   - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

8. If (user click on manage Faculty){

   Show the Role Management Form{ Display data in grid view dgvFaculty

   - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
   - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
   - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

9. If (user click on manage AwardingBody){

   Show the User Management Form{ Display data in grid view dgvAwardingBody

   - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){

show success message} if (no){ show error message and ask for required information}

- If (user click on Update){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
- If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

10. If user click on student show 2 menus
    - Manage Batch
    - Manage Student
11. If (user click on manage Batch){

    Show the Role Management Form{ Display data in grid view dgvBatch
    - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}
12. If (user click on manage Student){

    Show the User Management Form{ Display data in grid view dgvStudent
    - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}
13. If (user clicked on Manage Fee){ show three menu
    - Manage First Year Fee

- Manage Second Year Fee
- Manage Third year fee

14. If (user click on manage first Year){

    Show the User Management Form{ Display first year fee data in grid view dgvFirstYear

    - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

15. If (user click on manage Second Year){

    Show the Role Management Form{ Display second year fee data in grid view dgvSecondYear

    - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}

16. If (user click on manage Third Year){

    Show the User Management Form{ Display Second Year fee data in grid view dgvThirdYear

    - If (user click on save){ check if (all approperiete fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success

message} if (no){ cancel the process}

17. If(user click on Inquiry){show inquiry Form}
    - If (user click on save){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}
18. If (user click on Transaction Form){Show Transaction Form)
    - If (clicked on calculate) {Calculate the discounts, total amounts and due}
    - If (user click on save){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Update){ check if (all appropriate fields has been supplied){ if (yes){ show success message} if (no){ show error message and ask for required information}
    - If (user click on Delete){ ask if (really want to delete){ if (yes){ show success message} if (no){ cancel the process}
19. If (user click on help){ show up help form}
20. If (user Click on log off){ ask user if (really want to log off) if (yes)
    {Close the for and open up login form and ask credentials} if (no) { cancel the process}
21. If(user click on exit)){ ask user if (really want to exit) if (yes)
    {Exit the application} if (no) {cancel the process}
22. End

3. Flowchart

In programming flowchart is diagram of sequence on movement done or need to be done in order to make system work. It is geometric representation of the program logic. The Flowchart for the suggested Solution is drawn below in diagram 1.

4. Organization Requirement and Program Design

In this section, I have discussed about the organizational requirement against the features of the designed software. Software techniques/method can be justified with help of table 1:

SN.	Organizational Requirement	Feature in software to satisfy the requirements
1.	Maintain Student Information	To maintain the student information system of the college, this software has student management form that can save, update and delete student information from database.
2.	Maintain Financial Information	For maintaining the Financial information regarding fee structures, student fee transactions, this program has individual forms for both fee management and transaction management.
3.	Maintain Inquiry Details	To record the information of the student who comes to colleges to get information about courses, this software has separate inquiry form where user will be able to save, update and delete the inquiry details.
4.	Maintain Faculties	ISMT offers different academic faculties like, BBA, BHM, BIT etc. admin users will have facility to maintain the faculties the college offer.
5.	Have Admin/User Login Feature	This software startup with a login form where it is decided whether user is admin or a normal user, admin is given full control over the software whereas user has access to inquiry and student management system.

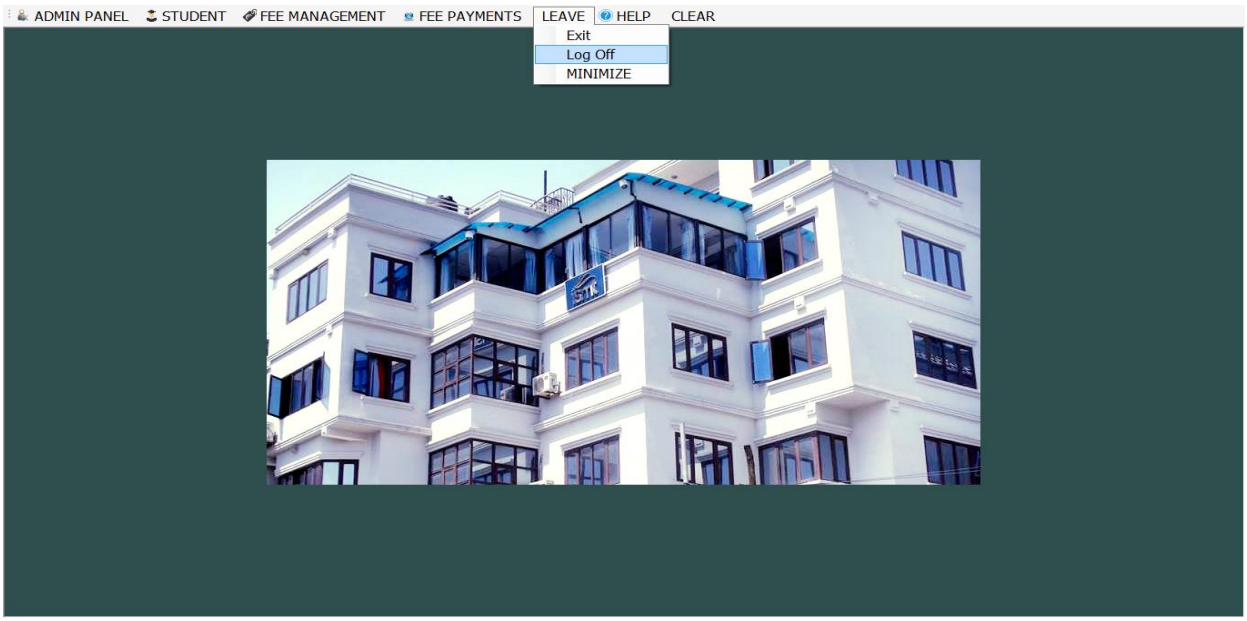
Task4

Implement an object oriented solution based on the prepared design. [P3.1, P3.2, P3.3, P3.4, D2]

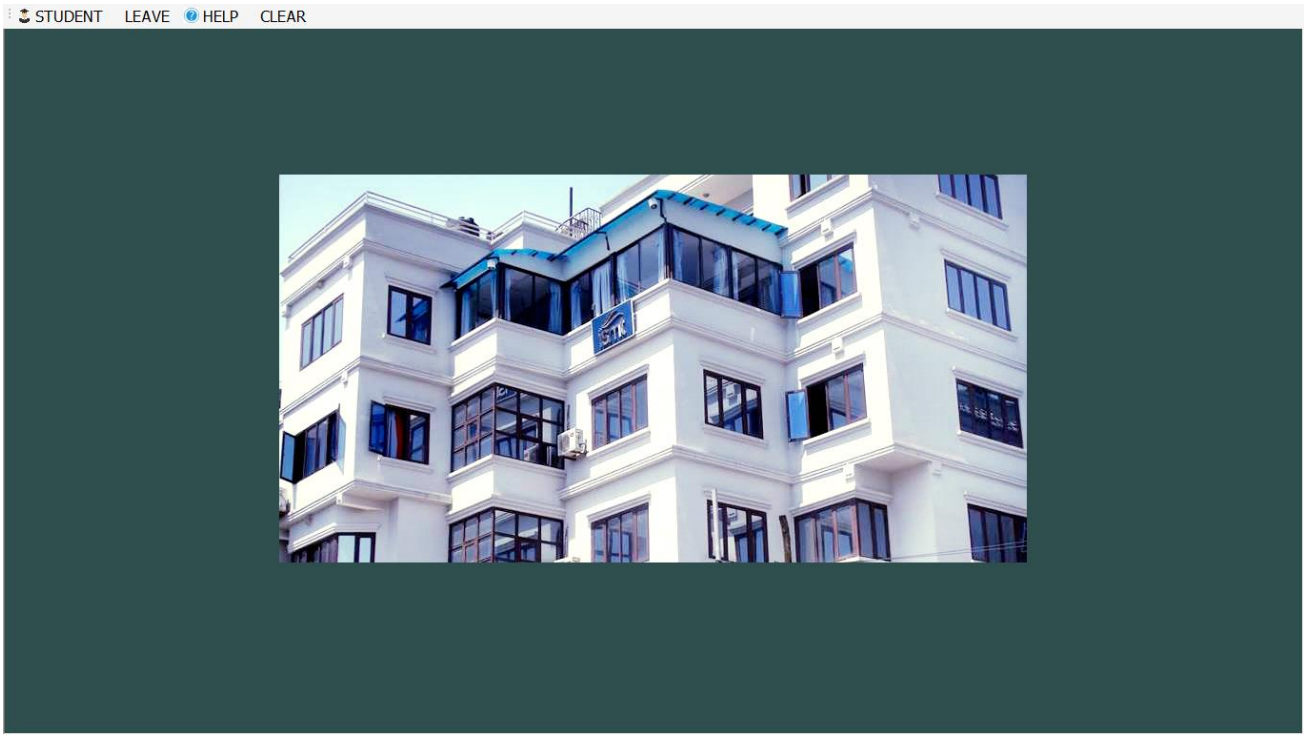
The implemented solution's programming codes and screenshots of each form is shown below.

Screenshots

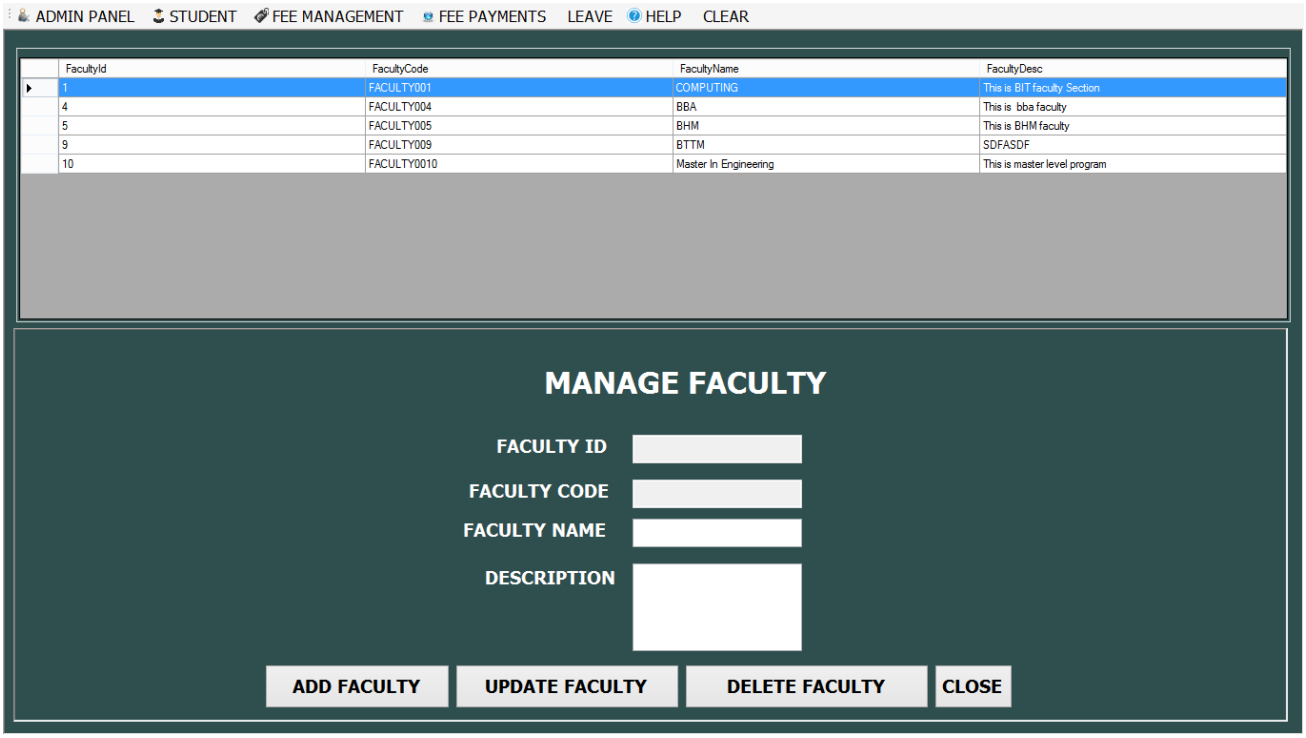
Screenshot 1Login Form



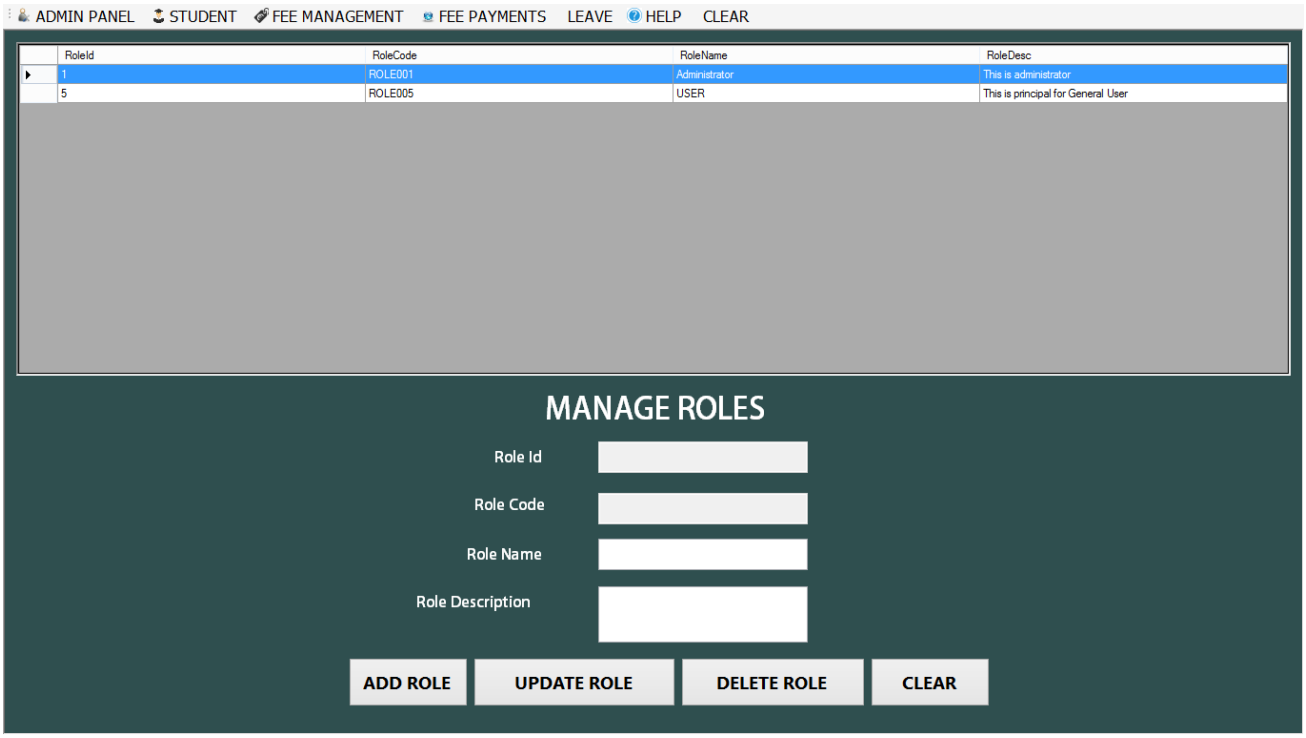
Screenshot 2 Main Screen Form (Admin)



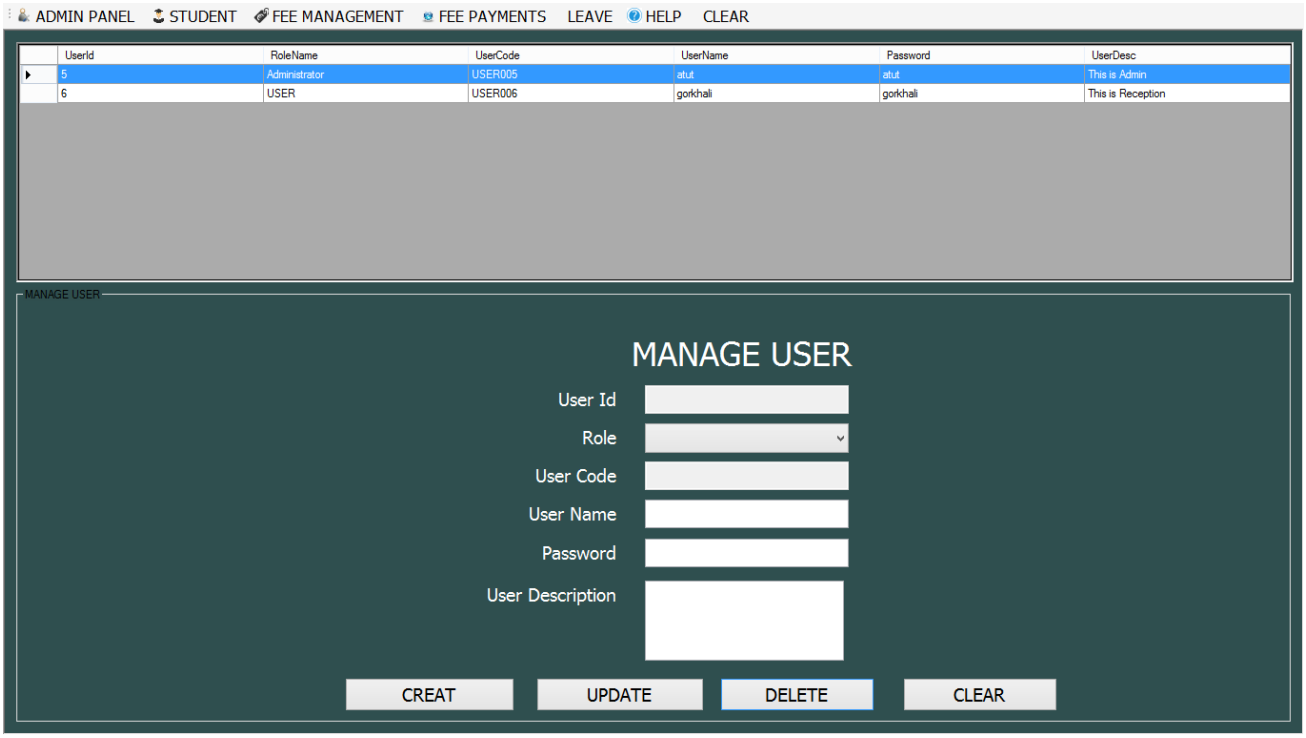
Screenshot 3 Main Scree Form (Users)



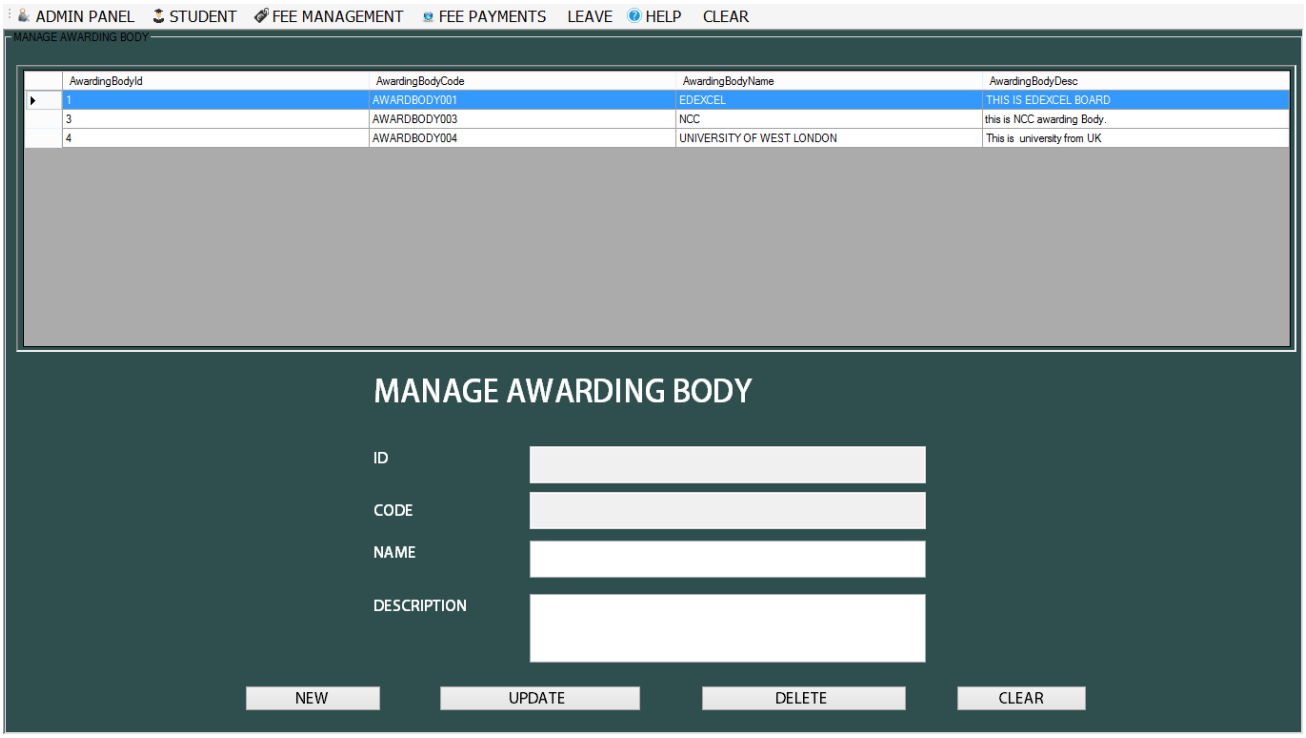
Screenshot 4 Faculty Form



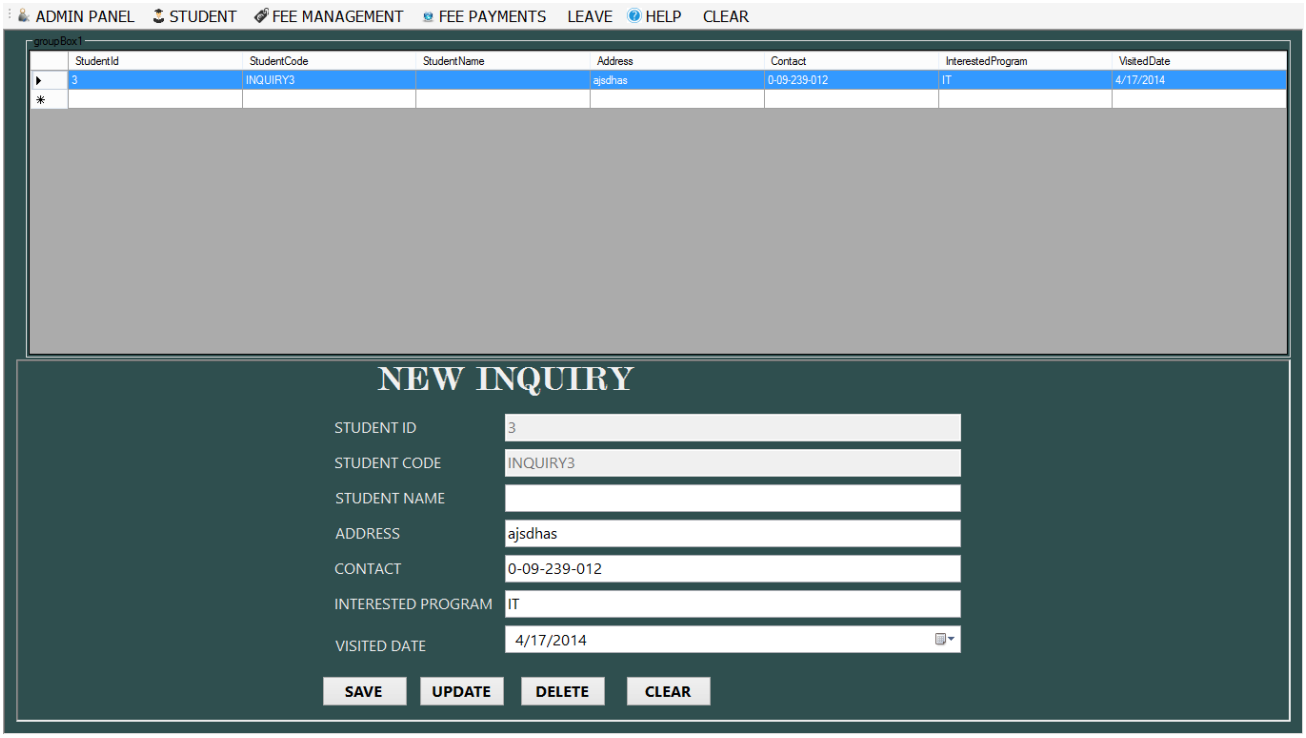
Screenshot 5 Manage Role Form



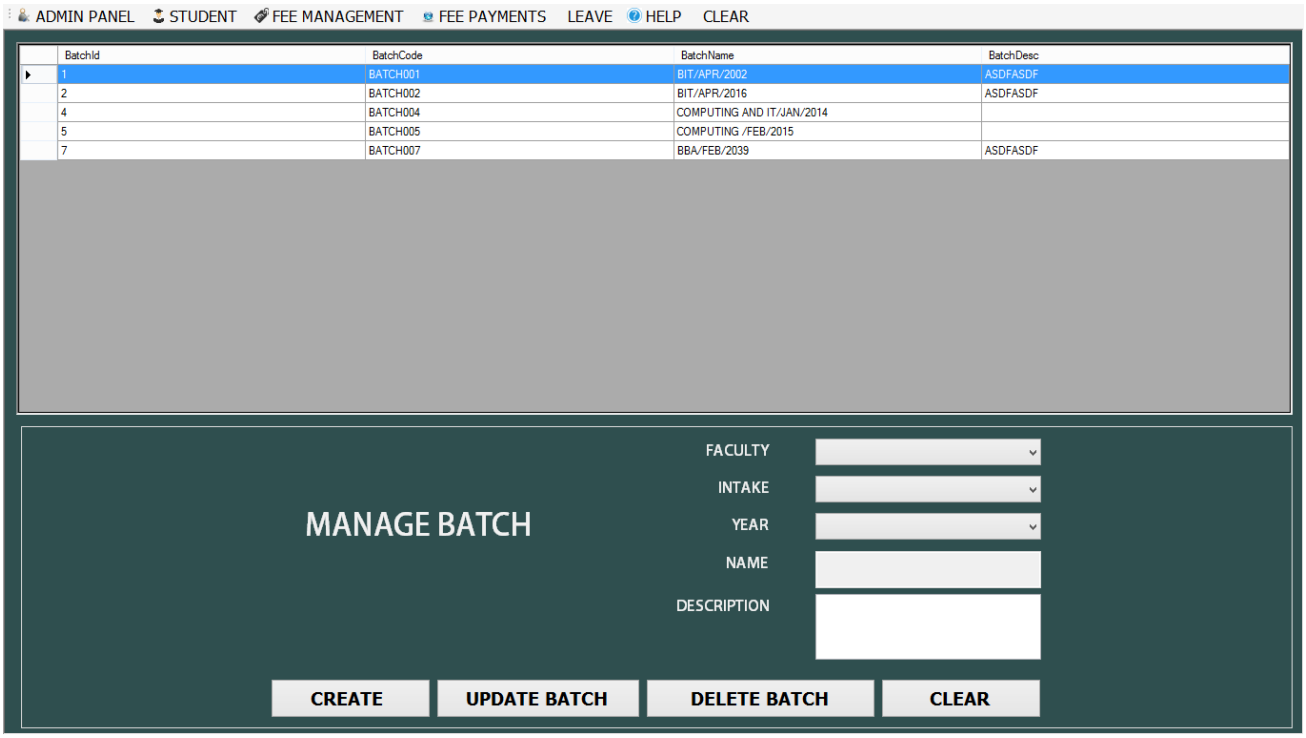
Screenshot 6 Manage User Form



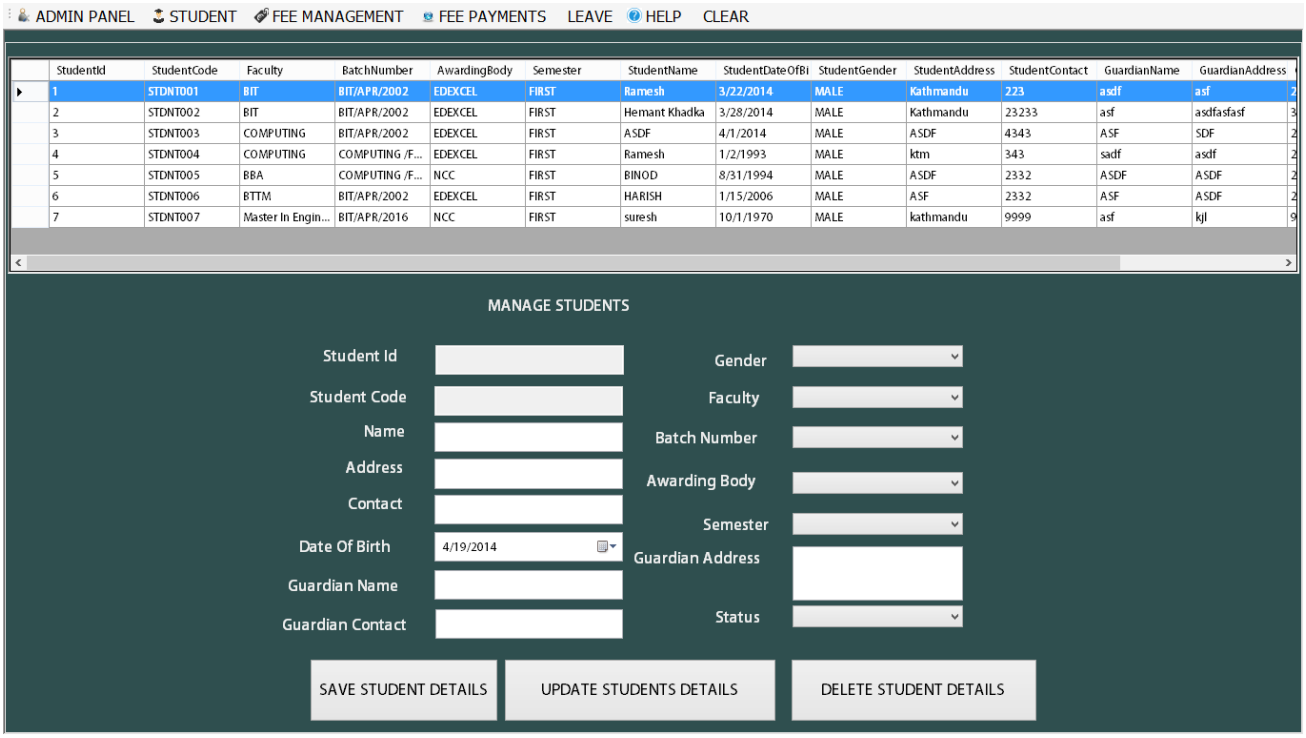
Screenshot 7 Manage Awarding Body Form



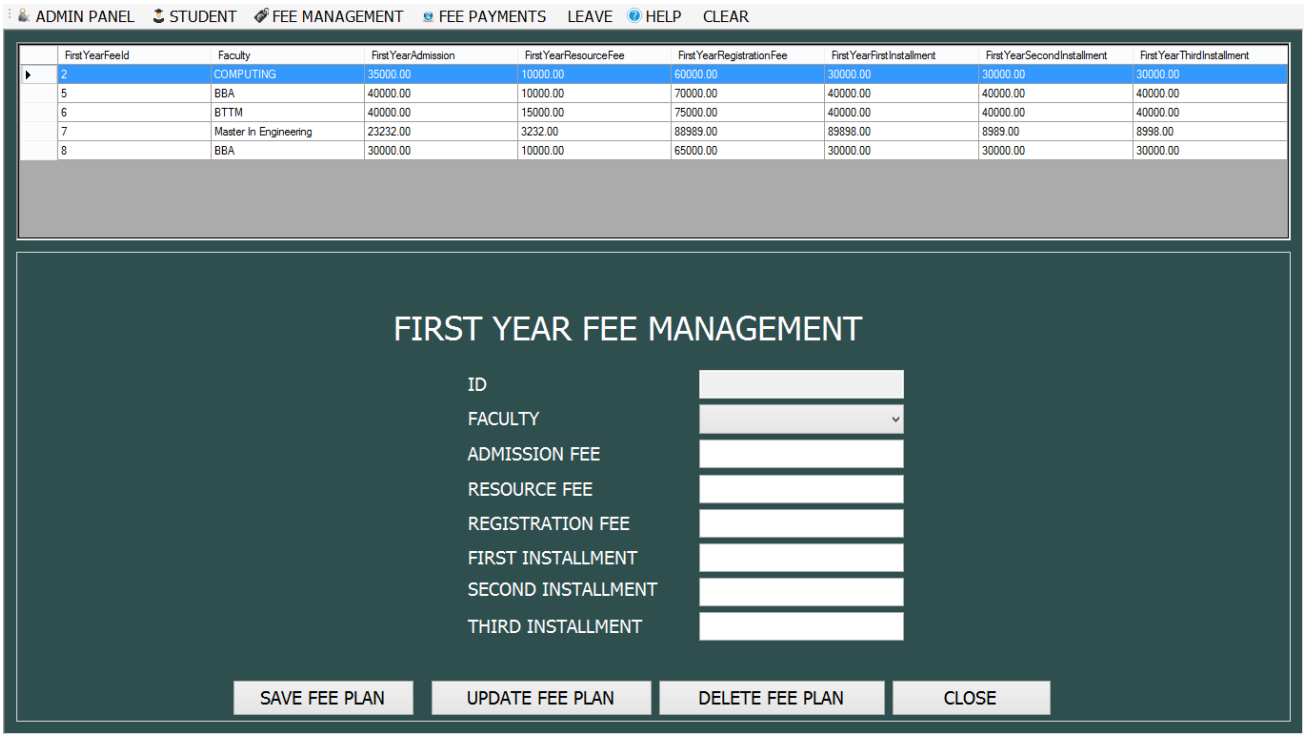
Screenshot 8 Inquiry Form



Screenshot 9 Manage Batch Form

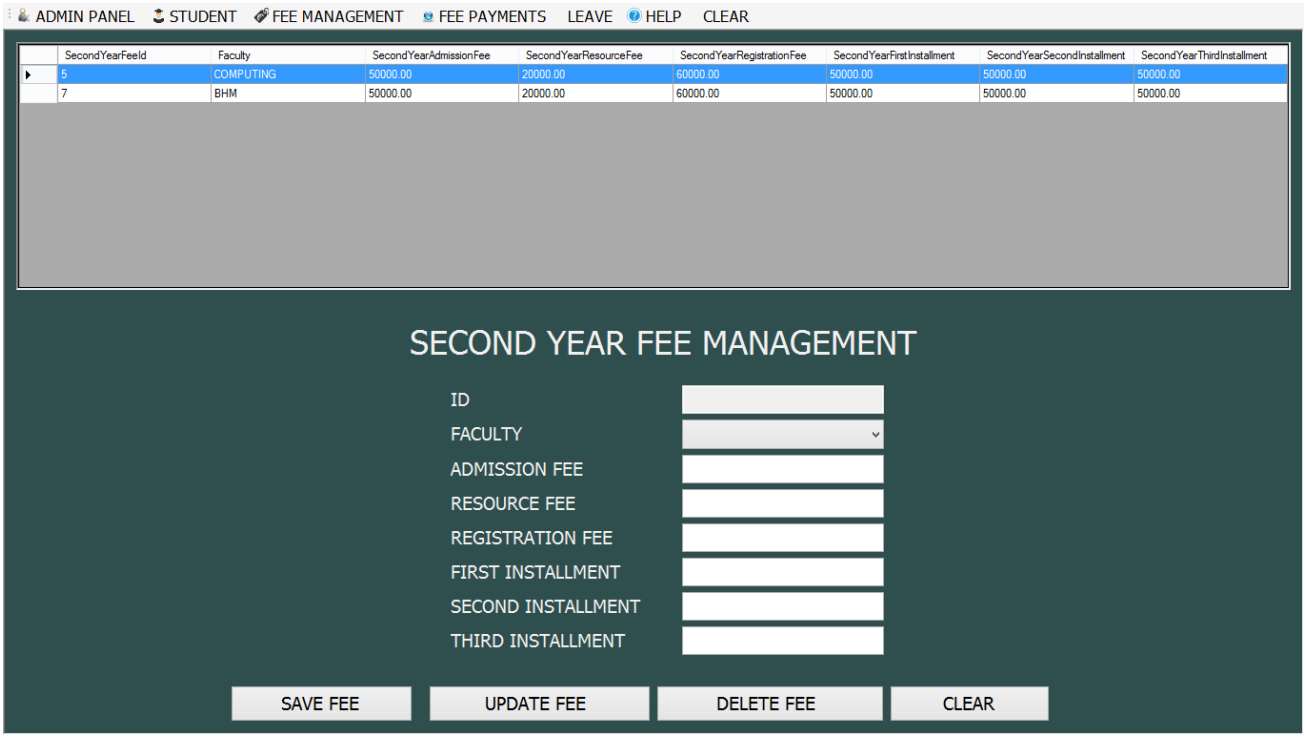


Screenshot 10 Manage Student



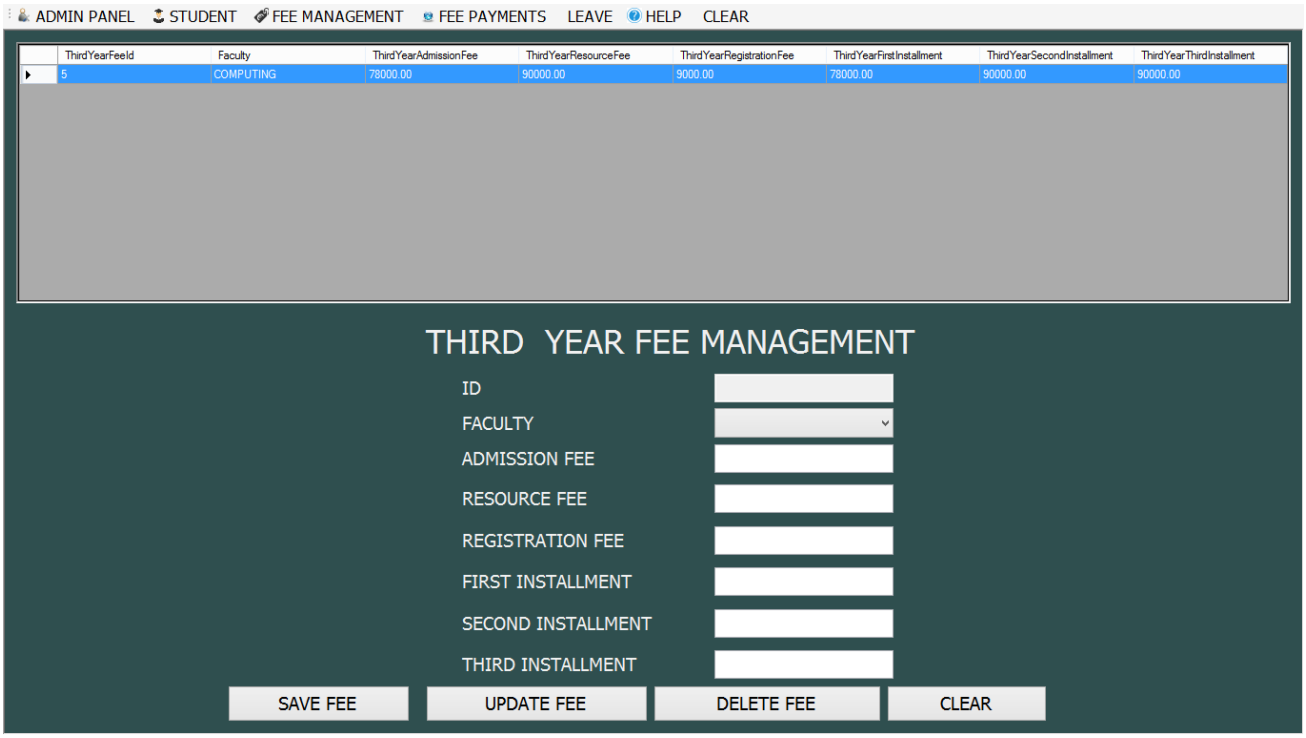
FirstYearFeeId	Faculty	FirstYearAdmission	FirstYearResourceFee	FirstYearRegistrationFee	FirstYearFirstInstallment	FirstYearSecondInstallment	FirstYearThirdInstallment
2	COMPUTING	35000.00	10000.00	60000.00	30000.00	30000.00	30000.00
5	BBA	40000.00	10000.00	70000.00	40000.00	40000.00	40000.00
6	BTTM	40000.00	15000.00	75000.00	40000.00	40000.00	40000.00
7	Master In Engineering	23232.00	3232.00	88989.00	89898.00	8989.00	8998.00
8	BBA	30000.00	10000.00	65000.00	30000.00	30000.00	30000.00

Screenshot 11 First Year Fee Management Form

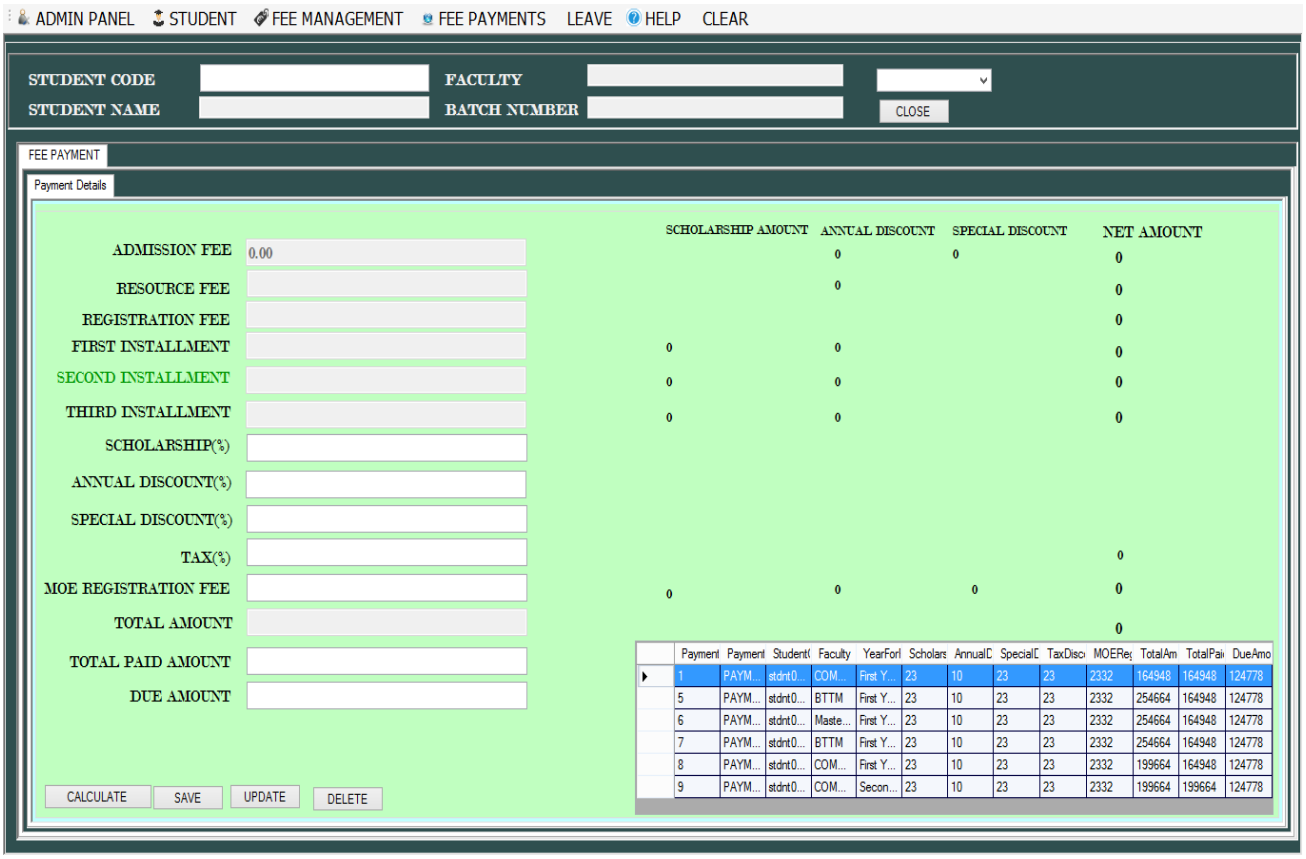


SecondYearFeeId	Faculty	SecondYearAdmissionFee	SecondYearResourceFee	SecondYearRegistrationFee	SecondYearFirstInstallment	SecondYearSecondInstallment	SecondYearThirdInstallment
5	COMPUTING	50000.00	20000.00	60000.00	50000.00	50000.00	50000.00
7	BHM	50000.00	20000.00	60000.00	50000.00	50000.00	50000.00

Screenshot 12 Second Year Fee Management Form



Screenshot 13 Third Year Fee Management Form



Screenshot 14 Fee Payment (Transaction) Form

Solution's Programming Code

BusinessLogicClass

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using IsmtDataAccessLayer;
namespace IsmtBusinessLogicLayer
{
   public class BusinessLogicClass
   {
      ClassDataAccess idac = new ClassDataAccess();
      ClassBatch bc = new ClassBatch();
```

```
ClassRole rc = new ClassRole();
ClassUserManagement uc = new ClassUserManagement();
ClassAwardingBody awbc = new ClassAwardingBody();
ClassStudentManagement sc = new ClassStudentManagement();
ClassFeeManagement fmc = new ClassFeeManagement();
ClassFeeTransaction fpc = new ClassFeeTransaction();
ClassInquiry ic = new ClassInquiry();
public bool FacultyManagement(int FacultyId, string FacultyName, string FacultyDesc, int
Mode)
{
   int rs = 0;
   bool result = false;
   if (Mode == 1)
   {
      rs = idac.FacultyManagement(FacultyId, FacultyName, FacultyDesc, Mode);
      if (rs > 0)
      {
         result = true;
      }
      else
      {
         result = false;
      }
   }
      else if(Mode==2)
      {
          rs = idac.FacultyManagement(FacultyId, FacultyName, FacultyDesc, Mode);
      if (rs > 0)
      {
         result = true;
      }
      else
      {
```

```
            result = false;
        }


        }
    else if (Mode == 3)
    {
        rs = idac.FacultyManagement(FacultyId, FacultyName, FacultyDesc, Mode);
        if (rs > 0)
        {
            result = true;
        }
        else
        {
            result = false;
        }
    }
    return result;

}
public bool ManageBatch(int BatchId, String BatchName, String BatchDesc, int Mode)
{
    bool result = false;
    int rs = 0;
    rs = bc.ManageBatch(BatchId, BatchName, BatchDesc, Mode);
    if (Mode == 1)
    {
        if (rs > 0)
            result = true;
        else
            result = false;
    }
    else if (Mode == 2)
    {
```

```
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        return result;
    }
    public bool ManageRole(int RoleId, String RoleName, String RoleDesc, int Mode)
    {
        bool result = false;
        int rs = 0;
        rs = rc.ManageRole(RoleId, RoleName, RoleDesc, Mode);
        if (Mode == 1)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
```

```
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        return result;
    }
    public bool ManageUser(int UserId, String RoleName, String UserName, String Password,
String UserDesc, int Mode)
    {
        bool result = false;
        int rs = 0;
        rs = uc.ManageUser(UserId, RoleName, UserName, Password, UserDesc, Mode);
        if (Mode == 1)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
```

```
            else
                result = false;
        }
        return result;
    }
    public bool ManageAwardingBody(int AwardingBodyId, String AwardingBodyName, String
AwardingBodyDescription, int Mode)
    {
        bool result = false;
        int rs = 0;
        rs = awbc.ManageAwardingBody(AwardingBodyId, AwardingBodyName,
AwardingBodyDescription, Mode);
        if (Mode == 1)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
```

```
    return result;
}
public bool ManageStudent(
                    int StudentId,
                    String Faculty,
                    String BatchNumber,
                    String AwardingBody,
                    String Semester,
                    String StudentName,
                    String DateOfBirth,
                    String Gender,
                    String Address,
                    String Contact,
                    String GuardianName,
                    String GuardianAddress,
                    String GuardianContact,
                    String Status,
                   int Mode
                  )
{
    bool result = false;
    int rs = 0;
    rs = sc.ManageStudent(StudentId,
                     Faculty,
                      BatchNumber,
                      AwardingBody,
                      Semester,
                      StudentName,
                      DateOfBirth,
                      Gender,
                      Address,
                      Contact,
                      GuardianName,
```

```
                    GuardianAddress,
                    GuardianContact,
                    Status,
                    Mode);
        if (Mode == 1)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
            else
                result = false;
        }
        return result;
    }
    public bool FirstYearFeeManagement(int FeeYearId,
      string FacultyName,
      double FirstYearAdmission,
      double FirstYearResource,
      double FirstYearRegistration,
      double FirstYearFirstInstallment,
```

```
    double FirstYearSecondInstallment,
    double FirstYearThirdInstallment,
int Mode)
 {
        int rs = fmc.FirstYearFeeManagement(FeeYearId,
        FacultyName,
        FirstYearAdmission,
        FirstYearResource,
        FirstYearRegistration,
        FirstYearFirstInstallment,
        FirstYearSecondInstallment, FirstYearThirdInstallment, Mode);
     bool result=false;
     if (Mode == 1)
     {
        if (rs > 0)
            result = true;
        else
            result = false;
     }
     else if (Mode == 2)
     {
        if (rs > 0)
            result = true;
        else
            result = false;
     }
     else if (Mode == 3)
     {
        if (rs > 0)
            result = true;
        else
            result = false;
     }
```

```
    return result;

}
public bool SecondYearFeeManagement(int SecondYearFeeId,
 string Faculty,
 double SecondYearAdmissionFee,
 double SecondYearResourceFee,
 double SecondYearRegistrationFee,
 double SecondYearFirstInstallmentFee,
 double SecondYearSecondInstallmentFee,
 double SecondYearThirdInstallmentFee,
 int Mode)
{
    try
    {
        bool result = false;
        int rs = fmc.SecondYearFeeManagement(SecondYearFeeId,
            Faculty,
            SecondYearAdmissionFee,
            SecondYearResourceFee,
            SecondYearRegistrationFee,
            SecondYearFirstInstallmentFee,
            SecondYearSecondInstallmentFee,
            SecondYearThirdInstallmentFee, Mode);
        if (Mode == 1)
        {
            if (rs > 0)
                result = true;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
```

```
            }
            else if (Mode == 3)
            {
                if (rs > 0)
                    result = true;
            }
            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
     public bool ThirdYearFeeManagement(int ThirdYearFeeId,
    string Faculty,
    double ThirdYearAdmissionFee,
    double ThirdYearResourceFee,
    double ThirdYearRegistrationFee,
    double ThirdYearFirstInstallmentFee,
    double ThirdYearSecondInstallmentFee,
    double ThirdYearThirdInstallmentFee,
    int Mode)
     {
        bool result = false;
            int rs = fmc.ThirdYearFeeManagement(ThirdYearFeeId,
                Faculty,
                ThirdYearAdmissionFee,
                ThirdYearResourceFee,
               ThirdYearRegistrationFee,
                ThirdYearFirstInstallmentFee,
                ThirdYearSecondInstallmentFee,
                ThirdYearThirdInstallmentFee, Mode);
```

```
        if (Mode == 1)
            {
                if (rs > 0)
                    result = true;
            }
            else if (Mode == 2)
            {
                if (rs > 0)
                    result = true;
            }
            else if (Mode == 3)
            {
                if (rs > 0)
                    result = true;
            }
            return result;
        }
    public bool FeePayment(int PaymentId,
     String StudentCode,
     String Faculty,
     String YearForPayment,
     double ScholarshipPercentage,
     double AnnualDiscount,
     double SpecialDiscount,
     double TaxDiscount,
     double MOERegistration,
     double TotalAmount,
     double TotalPaidAmount,
     double DueAmount,
     int Mode)
    {
        try
        {
```

```
        bool result = false;
        int rs = fpc.FeePayment(PaymentId,
            StudentCode,
            Faculty,
            YearForPayment,
            ScholarshipPercentage,
            AnnualDiscount,
            SpecialDiscount,
            TaxDiscount,
            MOERegistration,
            TotalAmount,
            TotalPaidAmount,
            DueAmount,
            Mode);
    if (Mode == 1)
        {
            if (rs > 0)
                result = true;
        }
        else if (Mode == 2)
        {
            if (rs > 0)
                result = true;
        }
        else if (Mode == 3)
        {
            if (rs > 0)
                result = true;
        }
        return result;
    }
    catch (Exception ex)
    {
```

```
            throw ex;
        }
    }
    public bool NewInquiry(int StudentId, String StudentName, String Address, String Contact,
String InterestedProgram, DateTime VisitedDate, int Mode)
    {
        try
        {
            bool result = false;
            int rs = ic.NewInquiry(StudentId, StudentName, Address, Contact, InterestedProgram,
VisitedDate, Mode);

        if (Mode == 1)
            {
                if (rs > 0)
                    result = true;
            }
            else if (Mode == 2)
            {
                if (rs > 0)
                    result = true;
            }
            else if (Mode == 3)
            {
                if (rs > 0)
                    result = true;
            }
            return result;

        }
        catch (Exception ex)
        {
```

```
            throw ex;
        }      }

    }   }
```

ClassAwardingBody

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
   public class ClassAwardingBody
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int ManageAwardingBody(int AwardingBodyId, String AwardingBodyName, String
AwardingBodyDescription, int Mode)
      {
         try
         {
            int result = 0;
            string Query = "";
            if (Mode == 1)
               Query = "Insert into AwardingBody values('" + AwardingBodyName + "','" +
AwardingBodyDescription + "')";
            if(Mode==2)
               Query="Update AwardingBody set AwardingBodyName='" + AwardingBodyName +
"',AwardingBodyDesc='" + AwardingBodyDescription + "' where AwardingBodyId='" +
```

```
AwardingBodyId + """;
            if(Mode==3)
               Query="Delete from AwardingBody where AwardingBodyId='" + AwardingBodyId +
"'";
            SqlCommand cmd=new SqlCommand(Query,conn);
            conn.Open();
            result=cmd.ExecuteNonQuery();
            conn.Close();
            return result;
         }
         catch (Exception ex)
         {

            throw ex;
         }
      }
      public DataTable SelectAllAwardingBody()
      {
         try
         {
            SqlCommand cmd = new SqlCommand("Select * from AwardingBody", conn);
            DataTable dt = new DataTable();
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
         }
         catch (Exception ex)
         {

            throw ex;
         }     }    }}
```

ClassBatch

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
   public class ClassBatch
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int ManageBatch(int BatchId, String BatchName, String BatchDesc, int Mode)
      {
         try
         {
            int result = 0;
            string Batch = "";
            if (Mode == 1)

               Batch = "insert into Batch values('" + BatchName + "','" + BatchDesc + "')";
            else if (Mode == 2)
               Batch = "Update Batch set BatchName='" + BatchName + "',BatchDesc='" + BatchDesc
+ "' where BatchId='" + BatchId + "'";
            else if (Mode == 3)
               Batch = "Delete from Batch where BatchId='" + BatchId + "'";
            SqlCommand cmd = new SqlCommand(Batch, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
```

```
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable SelectAllBatch()
    {
        try
        {
            string selectBatch = "Select * from Batch";
            SqlCommand cmd = new SqlCommand(selectBatch, conn);
            DataTable dt = new DataTable();
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }     }   }}
```

ClassDataAccess

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

```
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{

   public class ClassDataAccess
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int FacultyManagement(int FacultyId,string FacultyName, string FacultyDesc,int Mode)
      {
         int result = 0;
         string Faculty = "";
         if (Mode == 1)
         {
           Faculty = "insert into FacultyTable values('" + FacultyName + "','" + FacultyDesc + "')";

         }
         else if (Mode == 2)
         {
            Faculty = "Update FacultyTable set FacultyName='" + FacultyName + "',FacultyDesc='" +
FacultyDesc + "' where FacultyId='" + FacultyId + "'";
         }
         else if (Mode == 3)
         {
            Faculty = "Delete from FacultyTable where FacultyId='" + FacultyId + "'";
         }
         SqlCommand cmd = new SqlCommand(Faculty, conn);
         conn.Open();
         result = cmd.ExecuteNonQuery();
         conn.Close();
         return result;
      }
      public DataTable SelectFaculties()
      {
```

```
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from FacultyTable", conn);
        DataTable dt = new DataTable();
           conn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        dt.Load(dr);
        conn.Close();
       return dt;
        }
        catch (Exception ex)
        {

           throw ex;
        }

    }
  }
}
```

ClassDataBaseConnection

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer

{
  public class ClassDataBaseConnection
```

```
    {
        public static string DbConnection
        {
            get { return
System.Configuration.ConfigurationManager.ConnectionStrings["dbConn"].ConnectionString; }
        }
    }
}
```

ClassFeeManagement

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
  public class ClassFeeManagement
    {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int FirstYearFeeManagement(int FeeYearId,
      string FacultyName,
          double FirstYearAdmission,
          double FirstYearResource,
          double FirstYearRegistration,
          double FirstYearFirstInstallment,
          double FirstYearSecondInstallment,
          double FirstYearThirdInstallment,
      int Mode)
      {
          try
```

```
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into FirstYearFee values('" + FacultyName + "','" + FirstYearAdmission
+ "','" + FirstYearResource + "','" + FirstYearRegistration + "','" + FirstYearFirstInstallment + "','" +
FirstYearSecondInstallment + "','" + FirstYearThirdInstallment + "')";
            else if (Mode == 2)
                query = "Update FirstYearFee set Faculty='" + FacultyName + "',FirstYearAdmission='"
+ FirstYearAdmission + "',FirstYearResourceFee='" + FirstYearResource +
"',FirstYearRegistrationFee='" + FirstYearRegistration + "',FirstYearFirstInstallment='" +
FirstYearFirstInstallment + "',FirstYearSecondInstallment='" + FirstYearSecondInstallment +
"',FirstYearThirdInstallment='" + FirstYearThirdInstallment + "' where FirstYearFeeId='" +
FeeYearId + "'";
            else if (Mode == 3)
                query = "Delete from FirstYearFee where FirstYearFeeId='" + FeeYearId + "'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetAllFirstYearFeeDetails()
    {
        SqlCommand cmd = new SqlCommand("Select * from FirstYearFee", conn);
        DataTable dt = new DataTable();
        conn.Open();
```

```
        SqlDataReader dr = cmd.ExecuteReader();
        dt.Load(dr);
        conn.Close();
        return dt;
    }
    public int SecondYearFeeManagement(int SecondYearFeeId,
        string Faculty,
        double SecondYearAdmissionFee,
        double SecondYearResourceFee,
        double SecondYearRegistrationFee,
        double SecondYearFirstInstallmentFee,
        double SecondYearSecondInstallmentFee,
        double SecondYearThirdInstallmentFee,
        int Mode)
    {
        try
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into SecondYearFeeTable values('" + Faculty + "','" +
SecondYearAdmissionFee + "','" + SecondYearResourceFee + "','" + SecondYearRegistrationFee +
"','" + SecondYearFirstInstallmentFee + "','" + SecondYearSecondInstallmentFee + "','" +
SecondYearThirdInstallmentFee + "')";
            else if (Mode == 2)
                query = "Update SecondYearFeeTable set Faculty='" + Faculty +
"',SecondYearAdmissionFee='" + SecondYearAdmissionFee + "',SecondYearResourceFee='" +
SecondYearResourceFee + "',SecondYearRegistrationFee='" + SecondYearRegistrationFee +
"',SecondYearFirstInstallment='" + SecondYearFirstInstallmentFee +
"',SecondYearSecondInstallment='" + SecondYearSecondInstallmentFee +
"',SecondYearThirdInstallment='" + SecondYearThirdInstallmentFee + "'where SecondYearFeeId='"
+ SecondYearFeeId + "'";
            else if (Mode == 3)
```

```
            query = "Delete from SecondYearFeeTable where SecondYearFeeId='" +
SecondYearFeeId + "'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetAllSecondYearFeeDetails()
    {
        SqlCommand cmd = new SqlCommand("Select * from SecondYearFeeTable", conn);
        DataTable dt = new DataTable();
        conn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        dt.Load(dr);
        conn.Close();
        return dt;
    }
    public int ThirdYearFeeManagement(int ThirdYearFeeId,
      string Faculty,
      double ThirdYearAdmissionFee,
      double ThirdYearResourceFee,
      double ThirdYearRegistrationFee,
      double ThirdYearFirstInstallmentFee,
      double ThirdYearSecondInstallmentFee,
      double ThirdYearThirdInstallmentFee,
      int Mode)
```

```
    {
        try
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into ThirdYearFeeTable values('" + Faculty + "','" +
ThirdYearAdmissionFee + "','" + ThirdYearResourceFee + "','" + ThirdYearRegistrationFee + "','" +
ThirdYearFirstInstallmentFee + "','" + ThirdYearSecondInstallmentFee + "','" +
ThirdYearThirdInstallmentFee + "')";
            else if (Mode == 2)
                query = "Update ThirdYearFeeTable set Faculty='" + Faculty +
"',ThirdYearAdmissionFee='" + ThirdYearAdmissionFee + "',ThirdYearResourceFee='" +
ThirdYearResourceFee + "',ThirdYearRegistrationFee='" + ThirdYearRegistrationFee +
"',ThirdYearFirstInstallment='" + ThirdYearFirstInstallmentFee + "',ThirdYearSecondInstallment='"
+ ThirdYearSecondInstallmentFee + "',ThirdYearThirdInstallment='" +
ThirdYearThirdInstallmentFee + "'where ThirdYearFeeId='" + ThirdYearFeeId + "'";
            else if (Mode == 3)
                query = "Delete from ThirdYearFeeTable where ThirdYearFeeId='" + ThirdYearFeeId +
"'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetAllThirdYearFeeDetails()
```

```
    {
        SqlCommand cmd = new SqlCommand("Select * from ThirdYearFeeTable", conn);
        DataTable dt = new DataTable();
        conn.Open();
        SqlDataReader dr = cmd.ExecuteReader();
        dt.Load(dr);
        conn.Close();
        return dt;
    }
    public DataTable GetFirstYearFeeDetailsByFaculty(String Faculty)
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from FirstYearFee where Faculty='" +
Faculty + "'", conn);
            DataTable dt = new DataTable();
            conn.Open();
           SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

           throw ex;
        }

    }
    public DataTable GetSecondYearFeeDetailsByFaculty(String Faculty)
    {
        try
        {
```

```
            SqlCommand cmd = new SqlCommand("Select * from SecondYearFeeTable where
Faculty='" + Faculty + "'", conn);
            DataTable dt = new DataTable();
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetThirdYearFeeDetailsByFaculty(string Faculty)
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from ThirdYearFeeTable where
Faculty='" + Faculty + "'", conn);
            DataTable dt = new DataTable();
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }
```

```
    }

  }

}
```

ClassFeeTransaction

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
  public class ClassFeeTransaction
  {
    SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
    public int FeePayment(int PaymentId,
      String StudentCode,
      String Faculty,
      String YearForPayment,
      double ScholarshipPercentage,
      double AnnualDiscount,
      double SpecialDiscount,
      double TaxDiscount,
      double MOERegistration,
      double TotalAmount,
      double TotalPaidAmount,
      double DueAmount,
      int Mode)
    {
```

```
        try
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into FeePaymentTable values('" + StudentCode + "','" + Faculty + "','" +
YearForPayment + "','" + ScholarshipPercentage + "','" + AnnualDiscount + "','" + SpecialDiscount +
"','" + TaxDiscount + "','" + MOERegistration + "','" + TotalAmount + "','" + TotalPaidAmount + "','"
+ DueAmount + "')";
            else if (Mode == 2)
                query = "Update FeePaymentTable set YearForPayment='" + YearForPayment +
"',ScholarshipPercent='" + ScholarshipPercentage + "',AnnualDiscount='" + AnnualDiscount +
"',SpecialDiscount='" + SpecialDiscount + "',TaxDiscount='" + TaxDiscount +
"',MOERegistration='" + MOERegistration + "',TotalAmount='" + TotalAmount +
"',TotalPaidAmount='" + TotalAmount + "',DueAmount='" + DueAmount + "' where StudentCode='"
+ StudentCode + "' and YearForPayment='" + YearForPayment + "'";
            else if (Mode == 3)
                query = "Delete From FeePaymentTable where StudentCode='" + StudentCode + "' and
YearForPayment ='" + YearForPayment + "'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetAllFeePaidDetails()
    {
```

```
        try
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = new SqlCommand("Select * from FeePaymentTable", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
  }
}
```

ClassInquiry

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
  public class ClassInquiry
  {
    SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
    public int NewInquiry(int StudentId, String StudentName, String Address, String Contact,
```

```
String InterestedProgram, DateTime VisitedDate, int Mode)
    {
        try
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into InquiryTable values('" + StudentName + "','" + Address + "','" +
Contact + "','" + InterestedProgram + "','" + VisitedDate + "')";
            else if (Mode == 2)
                query = "Update InquiryTable set StudentName='" + StudentName + "',Address='" +
Address + "',Contact='" + Contact + "',InterestedProgram='" + InterestedProgram + "',VisitedDate='"
+ VisitedDate + "'";
            else if (Mode == 3)
                query = "Delete from InquiryTable where StudentId='" + StudentId + "'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
            return result;
        }
        catch (Exception ex)
        {


            throw ex;
        }
    }
    public DataTable GetAllInquiry()
    {
        try
        {
            DataTable dt = new DataTable();
            SqlCommand cmd = new SqlCommand("Select * from InquiryTable", conn);
```

```
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
  }
}
```

ClassRole

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
   public class ClassRole
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int ManageRole(int RoleId, String RoleName, String RoleDesc, int Mode)
      {
         try
         {
          int result = 0;
```

```
        string ManageRole = "";
        if (Mode == 1)
            ManageRole = "insert into RoleTable values('" + RoleName + "','" + RoleDesc + "')";
        else if (Mode == 2)
            ManageRole = "Update RoleTable set RoleName='" + RoleName + "',RoleDesc='" +
RoleDesc + "' where RoleId='" + RoleId + "'";
        else if (Mode == 3)
            ManageRole = "Delete from RoleTable where RoleId ='" + RoleId + "'";
        SqlCommand cmd = new SqlCommand(ManageRole, conn);
        conn.Open();
        result = cmd.ExecuteNonQuery();
        conn.Close();
        return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable GetAllRoles()
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from RoleTable", conn);
           conn.Open();
             DataTable dt = new DataTable();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();

            return dt;
        }
```

```
        catch (Exception ex)
        {

            throw ex;
        }
      }
   }
}
```

ClassStudentManagement

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
namespace IsmtDataAccessLayer
{
   public class ClassStudentManagement
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int ManageStudent(
                        int StudentId,
                        String Faculty,
                        String BatchNumber,
                        String AwardingBody,
                        String Semester,
                        String StudentName,
                        String DateOfBirth,
                        String Gender,
                        String Address,
                        String Contact,
```

```
                    String GuardianName,
                    String GuardianAddress,
                    String GuardianContact,
                    String Status,
                   int Mode
                  )
    {
        try
        {
            int result = 0;
            string query = "";
            if (Mode == 1)
                query = "insert into StudentTable values('" + Faculty + "','" + BatchNumber + "','" +
AwardingBody + "','" + Semester + "','" + StudentName + "','" + DateOfBirth + "','" + Gender + "','"
+ Address + "','" + Contact + "','" + GuardianName + "','" + GuardianAddress + "' ,'" +
GuardianContact + "','" + Status + "')";
            else if (Mode == 2)
                query = "Update StudentTable set Faculty='" + Faculty + "',BatchNumber='" +
BatchNumber + "',AwardingBody='" + AwardingBody + "',Semester='" + Semester +
"',StudentName='" + StudentName + "',StudentDateOfBirth='" + DateOfBirth + "',StudentGender='"
+ Gender + "',StudentAddress='" + Address + "',StudentContact='" + Contact + "',GuardianName='"
+ GuardianName + "',GuardianAddress='" + GuardianAddress + "',GuardianContact='" +
GuardianContact + "',StudentStatus='" + Status + "' where StudentId='" + StudentId + "'";
            else if (Mode == 3)
                query = "Delete from StudentTable where StudentId='" + StudentId + "'";
            SqlCommand cmd = new SqlCommand(query, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();
           return result;
        }
        catch (Exception ex)
        {
```

```
            throw ex;
        }
    }
    public DataTable SelectAllStudents()
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from StudentTable", conn);
            DataTable dt = new DataTable();
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataSet  SelectStudentByCode(String StudentCode)
    {
        try
        {
            DataSet ds = new DataSet();
            SqlDataAdapter da = new SqlDataAdapter("Select StudentName,Faculty,BatchNumber
from StudentTable where StudentCode='" + StudentCode + "'", conn);
            da.Fill(ds);

            return ds;
        }
```

```
        catch (Exception ex)
        {

            throw ex;
        }
      }
   }
}
```

ClassUserManagement

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.Data;
namespace IsmtDataAccessLayer
{
   public class ClassUserManagement
   {
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      public int ManageUser(int UserId, String RoleName, String UserName, String Password, String
UserDesc, int Mode)
      {
         try
         {
            int result = 0;
            String  User = "";
            if (Mode == 1)
               User = "insert into UserTable values('" + RoleName + "','" + UserName + "','" +
Password + "','" + UserDesc + "')";
            else if (Mode == 2)
```

```
            User = "Update UserTable set RoleName='" + RoleName + "',UserName='" +
UserName + "',Password='" + Password + "',UserDesc= '" + UserDesc + "' where UserId='" + UserId
+ "'";
            else if(Mode==3)
                User="Delete from UserTable where UserId='" + UserId + "'";
            SqlCommand cmd = new SqlCommand(User, conn);
            conn.Open();
            result = cmd.ExecuteNonQuery();
            conn.Close();

            return result;
        }
        catch (Exception ex)
        {

            throw ex;
        }
    }
    public DataTable SelectAllUsers()
    {
        try
        {
            SqlCommand cmd = new SqlCommand("Select * from UserTable", conn);
            DataTable dt = new DataTable();

            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            dt.Load(dr);
            conn.Close();
            return dt;
        }
        catch (Exception ex)
        {
```

```
            throw ex;
          }
        }
    }
}
```

AllValues

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

namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public class AllValues
    {
        public double FirstInstallmentScholarship;
        public double SecondInstallmentScholarship;
        public double ThirdInstallmentScholarship;
        public double FirstInstallmentAmount;
        public double SecondInstallmentAmount;
        public double ThirdInstallmentAmount;
        public double TotalScholarshipAmount;
        public double AdmissionAmount;
        public double YearlyScholarship;
        public double YearlyScholarshipPercentage;
        public double YearlyResourceScholarship;
        public double YearlyResourceAmount;
        public double SpecialDiscountPercent;
        public double SpecialDiscountAmount;
```

```
    }
}
```

FeeTranaction Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class FeeTransaction : Form
    {

        public FeeTransaction()
        {
            InitializeComponent();

        }
        ClassDataAccess idac = new ClassDataAccess();
        ClassStudentManagement sc = new ClassStudentManagement();
        ClassFeeManagement fmc = new ClassFeeManagement();
        AllValues gv = new AllValues();
        ClassFeeTransaction fpc = new ClassFeeTransaction();
        BusinessLogicClass blc = new BusinessLogicClass();
        private void txtStudentCode_TextChanged(object sender, EventArgs e)
        {
```

```
        try
        {
            DataSet ds = new DataSet();
            ds = sc.SelectStudentByCode(txtStudentCode.Text);
            txtStudentName.Text = ds.Tables[0].Rows[0][0].ToString();
            txtFaculty.Text = ds.Tables[0].Rows[0][1].ToString();
            txtBatchNumber.Text = ds.Tables[0].Rows[0][2].ToString();
        }
        catch (Exception)
        {



        }
    }
    private void FeePaymentFrm_Load(object sender, EventArgs e)
    {
        try
        {
            dgvFeeDetails.DataSource = fpc.GetAllFeePaidDetails();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void txtFaculty_TextChanged(object sender, EventArgs e)
    {
        DataTable dt = new DataTable();
        if (cmbYear.Text == "First Year")
        {
            dt = fmc.GetFirstYearFeeDetailsByFaculty(txtFaculty.Text);
```

```
    }
    else if (cmbYear.Text == "Second Year")
    {
        dt = fmc.GetSecondYearFeeDetailsByFaculty(txtFaculty.Text);
    }
    else if (cmbYear.Text == "Third Year")
    {
        dt = fmc.GetThirdYearFeeDetailsByFaculty(txtFaculty.Text);
    }
    try
    {

        txtFirstYearAdmission.Text = dt.Rows[0][2].ToString();
        txtFirstYearResourceFee.Text = dt.Rows[0][3].ToString();
        txtFirstYearRegistration.Text = dt.Rows[0][4].ToString();
        txtFirstYearFirstInstallment.Text = dt.Rows[0][5].ToString();
        txtFirstYearSecondInstallment.Text = dt.Rows[0][6].ToString();
        txtFirstYearThirdInstallment.Text = dt.Rows[0][7].ToString();
    }
    catch (Exception)
    {



    }


}

private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

```
public void btnCalculate_Click(object sender, EventArgs e)
{
   if (txtFaculty.Text == "")
   {
      MessageBox.Show("Please Set a Student First");
      return;
   }
   else
   {
      try
      {
         double admission = Convert.ToDouble(txtFirstYearAdmission.Text);
         double Resource = Convert.ToDouble(txtFirstYearResourceFee.Text);
         double YearlyAdmissionScholarship =
Convert.ToDouble(lblYearlyAdmissionScholarhip.Text);
         double ResourceDiscount = Convert.ToDouble(lblResourceScholarship.Text);
         lblNetAdmission.Text = (admission - YearlyAdmissionScholarship).ToString();
         lblNetResource.Text = (Resource - ResourceDiscount).ToString();
         double FirstInstallment = Convert.ToDouble(txtFirstYearFirstInstallment.Text);
         double FirstInstallmentScholarship = Convert.ToDouble(lblFirstScholarship.Text);
         double FirstInstallmentAnnualDiscount = Convert.ToDouble(lblAnnualDiscount.Text);
         lblNetFirstInstallment.Text = (FirstInstallment - (FirstInstallmentScholarship +
FirstInstallmentAnnualDiscount)).ToString();
         double secondInstallment = Convert.ToDouble(txtFirstYearSecondInstallment.Text);
         double SecondScholarship = Convert.ToDouble(lblSecondScholarship.Text);
         double SecondDiscount = Convert.ToDouble(lblAnnualSecondDiscount.Text);
         lblNetSecondInstallment.Text = (secondInstallment - SecondScholarship -
SecondDiscount).ToString();
         double thirdInstallment = Convert.ToDouble(txtFirstYearThirdInstallment.Text);
         double thirdDiscount = Convert.ToDouble(lblSecondScholarship.Text);
         double thirdAnnual = Convert.ToDouble(lblThirdAnnualDiscount.Text);
         lblNetThirdInstallment.Text = (thirdInstallment - thirdDiscount -
thirdAnnual).ToString();
```

```
            lblNetRegistration.Text = txtFirstYearRegistration.Text;
            double NetRegistration = Convert.ToDouble(lblNetRegistration.Text);
            double TotalAmountToBePaid = Convert.ToDouble(lblNetAdmission.Text) +
Convert.ToDouble(lblNetResource.Text) + Convert.ToDouble(lblNetFirstInstallment.Text)
                + Convert.ToDouble(lblNetSecondInstallment.Text) +
Convert.ToDouble(lblNetThirdInstallment.Text)
                + Convert.ToDouble(lbltaxAmount.Text) +
Convert.ToDouble(lblMOERegistration.Text) + Convert.ToDouble(txtFirstYearRegistration.Text) +
Convert.ToDouble(lblMOERegistration.Text);
            lblTotalAmount.Text = TotalAmountToBePaid.ToString();
            txtTotalAmount.Text = lblTotalAmount.Text;
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }
}

    private void txtScholarship_TextChanged(object sender, EventArgs e)
    {
        try
        {
            double ScholarShipPercentage = Convert.ToDouble(txtScholarship.Text);
            gv.FirstInstallmentAmount = Convert.ToDouble(txtFirstYearFirstInstallment.Text);
            gv.SecondInstallmentAmount = Convert.ToDouble(txtFirstYearSecondInstallment.Text);
            gv.ThirdInstallmentAmount = Convert.ToDouble(txtFirstYearThirdInstallment.Text);
            gv.FirstInstallmentScholarship = ScholarShipPercentage * gv.FirstInstallmentAmount /
100;
            gv.SecondInstallmentScholarship = ScholarShipPercentage *
gv.SecondInstallmentAmount / 100;
            gv.ThirdInstallmentScholarship = ScholarShipPercentage * gv.ThirdInstallmentAmount /
```

```
100;
            lblFirstScholarship.Text = gv.FirstInstallmentScholarship.ToString();
            lblSecondScholarship.Text = gv.FirstInstallmentScholarship.ToString();
            lblThirdScholarship.Text = gv.FirstInstallmentScholarship.ToString();
            lblScholarshipTotal.Text = (gv.FirstInstallmentScholarship +
gv.SecondInstallmentScholarship + gv.ThirdInstallmentScholarship).ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void txtAnnualDiscount_TextChanged(object sender, EventArgs e)
    {
        try
        {
            gv.YearlyScholarshipPercentage = Convert.ToDouble(txtAnnualDiscount.Text);
            gv.AdmissionAmount = Convert.ToDouble(txtFirstYearAdmission.Text);
            gv.YearlyScholarship = gv.YearlyScholarshipPercentage / 100 * gv.AdmissionAmount;
            lblYearlyAdmissionScholarhip.Text = gv.YearlyScholarship.ToString();
            gv.YearlyResourceAmount = Convert.ToDouble(txtFirstYearResourceFee.Text);
            gv.YearlyScholarship = gv.YearlyScholarshipPercentage * gv.YearlyResourceAmount /
100;
            lblResourceScholarship.Text = gv.YearlyScholarship.ToString();
            double a = Convert.ToDouble(lblFirstScholarship.Text);
            double b = Convert.ToDouble(lblSecondScholarship.Text);
            double c = Convert.ToDouble(lblThirdScholarship.Text);
            double FirstRemain = gv.FirstInstallmentAmount - a;
            double SecondRemain = gv.SecondInstallmentAmount - b;
            double ThirdRemain = gv.ThirdInstallmentAmount - c;
            gv.YearlyScholarship = gv.YearlyScholarshipPercentage / 100 * FirstRemain;
```

```
            lblAnnualDiscount.Text = gv.YearlyScholarship.ToString();
            gv.YearlyScholarship = gv.YearlyScholarshipPercentage / 100 * SecondRemain;
            lblAnnualSecondDiscount.Text = gv.YearlyScholarship.ToString();
            gv.YearlyScholarship = gv.YearlyScholarshipPercentage / 100 * ThirdRemain;
            lblThirdAnnualDiscount.Text = gv.YearlyScholarship.ToString();
            double a1, b1, c1, d1, e1, total1;
            a1 = Convert.ToDouble(lblYearlyAdmissionScholarhip.Text);
            b1 = Convert.ToDouble(lblResourceScholarship.Text);
            c1 = Convert.ToDouble(lblAnnualDiscount.Text);
            d1 = Convert.ToDouble(lblAnnualSecondDiscount.Text);
            e1 = Convert.ToDouble(lblThirdAnnualDiscount.Text);
            total1 = (a1 + b1 + c1 + d1 + e1);
            lblTotalAnnualDiscount.Text = total1.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }

    }

    private void txtSpecialDiscount_TextChanged(object sender, EventArgs e)
    {
        try
        {
            gv.SpecialDiscountPercent = Convert.ToDouble(txtSpecialDiscount.Text);
            double AdmissionRemain = gv.AdmissionAmount -
Convert.ToDouble(lblYearlyAdmissionScholarhip.Text);
            gv.SpecialDiscountAmount = gv.SpecialDiscountPercent / 100 * AdmissionRemain;
            lblSpecialDiscount.Text = gv.SpecialDiscountAmount.ToString();
            lblTotalAnnual.Text = lblSpecialDiscount.Text;
        }
```

```
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void txtTax_TextChanged(object sender, EventArgs e)
    {
        double totalTaxableAmount = Convert.ToDouble(lblNetFirstInstallment.Text) +
Convert.ToDouble(lblNetSecondInstallment.Text) +
Convert.ToDouble(lblNetThirdInstallment.Text);
        double taxPercent = Convert.ToDouble(txtTax.Text);
        double taxAmount = taxPercent / 100 * totalTaxableAmount;
        lbltaxAmount.Text = taxAmount.ToString();
    }

    private void txtMoeRegistrationFee_TextChanged(object sender, EventArgs e)
    {
        lblMOERegistration.Text = txtMoeRegistrationFee.Text;
    }

    private void txtTotalPaidAmount_TextChanged(object sender, EventArgs e)
    {
        try
        {
            double totalPaid = Convert.ToDouble(txtTotalPaidAmount.Text);
            double total = Convert.ToDouble(txtTotalAmount.Text);
            txtDueAmount.Text = (total - totalPaid).ToString();
        }
        catch (Exception ex)
        {
```

```
            MessageBox.Show(ex.Message);
        }
    }

    private void btnSave_Click(object sender, EventArgs e)
    {
        try
        {
            double ScholarshipPercentage = Convert.ToDouble(txtScholarship.Text);
            double AnnualDiscount = Convert.ToDouble(txtAnnualDiscount.Text);
            double SpecialDiscount = Convert.ToDouble(txtSpecialDiscount.Text);
            double taxPercentage = Convert.ToDouble(txtTax.Text);
            double MOERegistration = Convert.ToDouble(txtMoeRegistrationFee.Text);
            double TotalAmount = Convert.ToDouble(txtTotalPaidAmount.Text);
            double TotalPaidAmount = Convert.ToDouble(txtTotalPaidAmount.Text);
            double DueAmount = Convert.ToDouble(txtDueAmount.Text);
            bool result = blc.FeePayment(0, txtStudentCode.Text, txtFaculty.Text, cmbYear.Text,
ScholarshipPercentage, AnnualDiscount, SpecialDiscount, taxPercentage, MOERegistration,
TotalAmount,TotalPaidAmount, DueAmount, 1);
            if (result == true)
            {
                MessageBox.Show("Fee information saved");
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }

        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
```

```
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        if (txtFaculty.Text == "")
        {
            MessageBox.Show("Please Set a Student First");
            return;
        }
        else
        {
            SaveTransaction();
        }

    }

    private void SaveTransaction()
    {
        try
        {
            int ScholarshipPercentage = Convert.ToInt32(txtScholarship.Text);
            int AnnualDiscount = Convert.ToInt32(txtAnnualDiscount.Text);
            int SpecialDiscount = Convert.ToInt32(txtSpecialDiscount.Text);
            int taxPercentage = Convert.ToInt32(txtTax.Text);
            double MOERegistration = Convert.ToDouble(txtMoeRegistrationFee.Text);
            double TotalAmount = Convert.ToDouble(txtTotalAmount.Text);
            double TotalPaidAmount = Convert.ToDouble(txtTotalPaidAmount.Text);
            double DueAmount = Convert.ToDouble(txtDueAmount.Text);
            bool result = blc.FeePayment(0, txtStudentCode.Text, txtFaculty.Text, cmbYear.Text,
ScholarshipPercentage, AnnualDiscount, SpecialDiscount, taxPercentage, MOERegistration,
TotalAmount, TotalPaidAmount, DueAmount, 1);
            if (result == true)
```

```
            {
                MessageBox.Show("Fee information saved");
                dgvFeeDetails.DataSource = fpc.GetAllFeePaidDetails();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }

        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        if (txtFaculty.Text == "")
        {
            MessageBox.Show("Please Set a Student First");
            return;
        }
        else
        {
            DialogResult dr = new DialogResult();
            dr = MessageBox.Show("UPDATE???", "CONFIRMATION",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
            if (dr == DialogResult.OK)
            {
              UpdateTransaction();
            }
```

```
            else
            {
                return;
            }
        }

    }

    private void UpdateTransaction()
    {
        try
        {
            int ScholarshipPercentage = Convert.ToInt32(txtScholarship.Text);
            int AnnualDiscount = Convert.ToInt32(txtAnnualDiscount.Text);
            int SpecialDiscount = Convert.ToInt32(txtSpecialDiscount.Text);
            int taxPercentage = Convert.ToInt32(txtTax.Text);
            double MOERegistration = Convert.ToDouble(txtMoeRegistrationFee.Text);
            double TotalAmount = Convert.ToDouble(txtTotalAmount.Text);
            double TotalPaidAmount = Convert.ToDouble(txtTotalPaidAmount.Text);
            double DueAmount = Convert.ToDouble(txtDueAmount.Text);
            bool result = blc.FeePayment(0, txtStudentCode.Text, txtFaculty.Text, cmbYear.Text,
ScholarshipPercentage, AnnualDiscount, SpecialDiscount, taxPercentage, MOERegistration,
TotalAmount, TotalPaidAmount, DueAmount, 2);
            if (result == true)
            {
                MessageBox.Show("Fee information updated");
                dgvFeeDetails.DataSource = fpc.GetAllFeePaidDetails();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
```

```
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void dgvFeeDetails_Click(object sender, EventArgs e)
{
    try
    {
        txtStudentCode.Text = dgvFeeDetails.SelectedRows[0].Cells[2].Value.ToString();
        cmbYear.Text = dgvFeeDetails.SelectedRows[0].Cells[4].Value.ToString();
        txtScholarship.Text = dgvFeeDetails.SelectedRows[0].Cells[5].Value.ToString();
        txtAnnualDiscount.Text = dgvFeeDetails.SelectedRows[0].Cells[6].Value.ToString();
        txtSpecialDiscount.Text = dgvFeeDetails.SelectedRows[0].Cells[7].Value.ToString();
        txtTax.Text = dgvFeeDetails.SelectedRows[0].Cells[8].Value.ToString();
         txtMoeRegistrationFee.Text= dgvFeeDetails.SelectedRows[0].Cells[9].Value.ToString();
        txtTotalAmount.Text = dgvFeeDetails.SelectedRows[0].Cells[10].Value.ToString();
        txtTotalPaidAmount.Text = dgvFeeDetails.SelectedRows[0].Cells[11].Value.ToString();
        txtDueAmount.Text = dgvFeeDetails.SelectedRows[0].Cells[12].Value.ToString();

    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void button6_Click(object sender, EventArgs e)
{
```

```
        if (txtFaculty.Text == "")
        {
            MessageBox.Show("Please Set a Student First");
            return;
        }
        else
        {
            DialogResult dr = new DialogResult();
            dr = MessageBox.Show("Delete???", "CONFIRMATION",
MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
            if (dr == DialogResult.OK)
            {
                DeleteTransaction();
            }
            else
            {
                return;
            }
        }

    }

    private void DeleteTransaction()
    {
        try
        {
            int ScholarshipPercentage = Convert.ToInt32(txtScholarship.Text);
            int AnnualDiscount = Convert.ToInt32(txtAnnualDiscount.Text);
            int SpecialDiscount = Convert.ToInt32(txtSpecialDiscount.Text);
            int taxPercentage = Convert.ToInt32(txtTax.Text);
            double MOERegistration = Convert.ToDouble(txtMoeRegistrationFee.Text);
            double TotalAmount = Convert.ToDouble(txtTotalAmount.Text);
            double TotalPaidAmount = Convert.ToDouble(txtTotalPaidAmount.Text);
```

```
            double DueAmount = Convert.ToDouble(txtDueAmount.Text);
            bool result = blc.FeePayment(0, txtStudentCode.Text, txtFaculty.Text, cmbYear.Text,
ScholarshipPercentage, AnnualDiscount, SpecialDiscount, taxPercentage, MOERegistration,
TotalAmount, TotalPaidAmount, DueAmount, 3);
            if (result == true)
            {
               MessageBox.Show("Fee information deleted");
               dgvFeeDetails.DataSource = fpc.GetAllFeePaidDetails();
            }
            else
            {
               MessageBox.Show("SOME ERRORS OCCURED");
            }

         }
         catch (Exception ex)
         {

            MessageBox.Show(ex.Message);
         }
      }

      private void groupBox1_Enter(object sender, EventArgs e)
      {

      }
   }
}
```

Login Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
using System.Data.SqlClient;

namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
   public partial class LoginForm : Form
   {
      public LoginForm()
      {
         InitializeComponent();
      }
      ClassRole RoleClass = new ClassRole();
      SqlConnection conn = new SqlConnection(ClassDataBaseConnection.DbConnection);
      private void LoginForm_Load(object sender, EventArgs e)
      {

         cmbRoleLogin.DataSource = RoleClass.GetAllRoles();
         cmbRoleLogin.DisplayMember = "RoleName";
         cmbRoleLogin.ValueMember = "RoleName";
         cmbRoleLogin.SelectedIndex = -1;
         txtPassword.Clear();
         txtUserName.Clear();
      }

      private void btnCancel_Click(object sender, EventArgs e)
      {
```

```
        ResetLoginForm();
    }

    private void ResetLoginForm()
    {
        txtPassword.Clear();
        txtUserName.Clear();
        cmbRoleLogin.SelectedIndex = -1;
        lblError.Visible = false;
    }

    private void btnLogin_Click(object sender, EventArgs e)
    {
        if (txtPassword.Text == "" || txtUserName.Text == "" || cmbRoleLogin.SelectedIndex < 0)
        {
            MessageBox.Show("PLEASE FILL ALL FIELDS FIRST", "ACTION REQUIRED");
            return;
        }
        else
        {
            try
            {

                SqlDataAdapter dAdapter = new SqlDataAdapter("select * from UserTable where
RoleName='" + cmbRoleLogin.Text + "' and UserName='" + txtUserName.Text + "' and
Password='" + txtPassword.Text + "'", conn);
                DataSet dSet = new DataSet();
                bool restult = Convert.ToBoolean(dAdapter.Fill(dSet));
                if (restult == false)
                {
                    lblError.Visible = true;
                    cmbRoleLogin.SelectedIndex = -1;
                    return;
```

```
            }
            else if (restult == true)
            {
                if (cmbRoleLogin.Text == "Administrator")
                {
                    this.Hide();
                    ProjectMainForm FormMain = new ProjectMainForm();
                    FormMain.Show();
                }
                else
                {
                    this.Hide();
                    ProjectMainFormUser FormMainUser = new ProjectMainFormUser();
                    FormMainUser.Show();
                }
            }
        }

        catch (Exception exp)
        {

            MessageBox.Show(exp.Message);
        }
    }
}

private void btnExit_Click(object sender, EventArgs e)
{
    Application.Exit();
}
}
}
```

ManageAwardingBody Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
   public partial class ManageAwardingBody : Form
   {
      public ManageAwardingBody()
      {
         InitializeComponent();
      }
      ClassAwardingBody awbc = new ClassAwardingBody();
      BusinessLogicClass iblc = new BusinessLogicClass();
      private void btnClose_Click(object sender, EventArgs e)
      {
         ClearAwardingBodyFields();
      }

      private void ClearAwardingBodyFields()
      {
         txtCode.Clear();
         txtDescription.Clear();
         txtId.Clear();
```

```
    txtName.Clear();
}

private void btnAddAwardingBody_Click(object sender, EventArgs e)
{
    if (txtName.Text == "")
    {
        MessageBox.Show("please Provide Awarding Body Name", "ACTION REQUIRED");
        txtName.BackColor = Color.Red;
    }
    else
    {
        NewAwardingBody();
    }

}

private void NewAwardingBody()
{
    try
    {
        bool result = iblc.ManageAwardingBody(0, txtName.Text, txtDescription.Text, 1);
        if (result == true)
        {
            MessageBox.Show("NEW AWARDING BODY CREATED");
            dgvAwardingDetails.DataSource = awbc.SelectAllAwardingBody();
        }
        else
        {
            MessageBox.Show("SOME ERRORS OCCURED");
        }
    }
    catch (Exception ex)
```

```
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void AwardingBodyFrm_Load(object sender, EventArgs e)
    {
        dgvAwardingDetails.DataSource = awbc.SelectAllAwardingBody();
    }

    private void dgvAwardingDetails_Click(object sender, EventArgs e)
    {
        try
        {
            txtId.Text = dgvAwardingDetails.SelectedRows[0].Cells[0].Value.ToString();
            txtCode.Text = dgvAwardingDetails.SelectedRows[0].Cells[1].Value.ToString();
            txtName.Text = dgvAwardingDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtDescription.Text = dgvAwardingDetails.SelectedRows[0].Cells[3].Value.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnUpdateAwardingBody_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
```

```
        {
            if (txtId.Text == "")
            {
                MessageBox.Show("Please Select a AwardingBody First");
                return;
            }
           else if (txtName.Text == "")
            {
                MessageBox.Show("Please Provide AwardingBody Name First");
                txtName.BackColor = Color.Red;
                return;
            }
            else
            {
                  UpdateAwardingBody();
            }

        }
        else
        {
            ClearAwardingBodyFields();
            return;
        }

    }

    private void UpdateAwardingBody()
    {
        try
        {
            bool result = iblc.ManageAwardingBody(Convert.ToInt16(txtId.Text), txtName.Text,
txtDescription.Text, 2);
            if (result == true)
```

```
            {
                MessageBox.Show("AWARDING BODY UPDATED");
                dgvAwardingDetails.DataSource = awbc.SelectAllAwardingBody();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnDeleteAwardingBody_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO DELETE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            if (txtId.Text == "")
            {
                MessageBox.Show("Please Select a AwardingBody First");
                return;
            }
            else if (txtName.Text == "")
            {
                MessageBox.Show("Please Provide AwardingBody Name First");
                txtName.BackColor = Color.Red;
                return;
```

```
            }
            else
            {
                DeleteAwardingBody();
            }
        }
        else
        {
            ClearAwardingBodyFields();
            return;
        }
    }

    private void DeleteAwardingBody()
    {
        try
        {
            bool result = iblc.ManageAwardingBody(Convert.ToInt16(txtId.Text), txtName.Text,
txtDescription.Text, 3);
            if (result == true)
            {
                MessageBox.Show("AWARDING BODY DELETED");
                dgvAwardingDetails.DataSource = awbc.SelectAllAwardingBody();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
```

```
        }
    }

    private void txtName_TextChanged(object sender, EventArgs e)
    {
        txtName.BackColor = Color.White;
    }
  }
}
```

ManageBatch Form

```
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 IsmtDataAccessLayer;
using IsmtBusinessLogicLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
  public partial class ManageBatch : Form
  {
    public ManageBatch()
    {
      InitializeComponent();
    }
    ClassDataAccess idac = new ClassDataAccess();
    BusinessLogicClass iblc = new BusinessLogicClass();
    ClassBatch bc = new ClassBatch();
```

```
private void BatchFrm_Load(object sender, EventArgs e)
{
    for (int i = 2001; i < 3000; i++)
    {
        cmbYear.Items.Add(i.ToString());
    }
    cmbFaculty.DataSource = idac.SelectFaculties();
    cmbFaculty.DisplayMember = "FacultyName";
    cmbFaculty.ValueMember = "FacultyName";
    cmbFaculty.SelectedIndex = -1;
    cmbMonth.SelectedIndex = -1;
    cmbYear.SelectedIndex = -1;
    dgvBatchDetails.DataSource = bc.SelectAllBatch();

}



private void btnCreateNewBatch_Click(object sender, EventArgs e)
{
    if (txtBatchName.Text == "")
    {
        MessageBox.Show("Please Generate Batch number first");
        return;
    }
    else
    {
        AddNewbatch();
    }
}

private void AddNewbatch()
{
```

```
        try
        {
            bool result = iblc.ManageBatch(0, txtBatchName.Text, txtBatchDescription.Text, 1);
            if (result == true)
            {
                MessageBox.Show("NEW BATCH SUCCESSFULLY CREATED");
                dgvBatchDetails.DataSource = bc.SelectAllBatch();
            }
            else
            {
                MessageBox.Show("ERRORS IN CREATING BATCH");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void dgvBatchDetails_Click(object sender, EventArgs e)
    {
        try
        {
             txtBatchName.Text = dgvBatchDetails.SelectedRows[0].Cells[2].Value.ToString();
        txtBatchDescription.Text = dgvBatchDetails.SelectedRows[0].Cells[3].Value.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }
```

```
    private void btnUpdateBatch_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {

            if (txtBatchName.Text=="")
            {
                MessageBox.Show("Please Provide Batch Name First");
                        return;
            }
            else
            {
                UpdateBatch();
            }

        }

    }

    private void UpdateBatch()
    {
        try
        {
            int BatchId =
Convert.ToInt32(dgvBatchDetails.SelectedRows[0].Cells[0].Value.ToString());
            bool result = iblc.ManageBatch(BatchId, txtBatchName.Text, txtBatchDescription.Text,
2);
            if (result == true)
            {
```

```
            MessageBox.Show("BATCH SUCCESSFULLY UPDATED");
            dgvBatchDetails.DataSource = bc.SelectAllBatch();
        }
        else
        {
            MessageBox.Show("ERRORS IN UPDATING BATCH");
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void btnDeleteBatch_Click(object sender, EventArgs e)
{
    DialogResult dR = new DialogResult();
    dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
    if (dR == DialogResult.OK)
    {
         Deletebatch();
    }
    else
    {
        return;
    }
}

private void Deletebatch()
{
    try
```

```
        {
            int BatchId =
Convert.ToInt32(dgvBatchDetails.SelectedRows[0].Cells[0].Value.ToString());
            bool result = iblc.ManageBatch(BatchId, txtBatchName.Text, txtBatchDescription.Text,
3);
            if (result == true)
            {
                MessageBox.Show("BATCH SUCCESSFULLY DELETED");
                dgvBatchDetails.DataSource = bc.SelectAllBatch();
            }
            else
            {
                MessageBox.Show("ERRORS IN DELETING BATCH");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        txtBatchName.Clear();
        txtBatchDescription.Clear();
        cmbFaculty.SelectedIndex = -1;
        cmbMonth.SelectedIndex = -1;
        cmbYear.SelectedIndex = -1;
    }

    private void dgvBatchDetails_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
```

```
    }

    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void cmbYear_SelectedIndexChanged_1(object sender, EventArgs e)
    {
      if (cmbFaculty.SelectedIndex > 0 || cmbMonth.SelectedIndex >0 || cmbYear.SelectedIndex >
0)
      {
        try
        {
          if (cmbFaculty.SelectedIndex > -1 && cmbMonth.SelectedIndex > -1 &&
cmbYear.SelectedIndex > -1)
          {
            string batch = cmbFaculty.Text + "/" + cmbMonth.Text + "/" + cmbYear.Text;
            txtBatchName.Text = batch;

          }
        }
        catch (Exception ex)
        {

          MessageBox.Show(ex.Message);
        }
      }
    }

    private void cmbMonth_SelectedIndexChanged(object sender, EventArgs e)
    {
```

```
        if (cmbFaculty.SelectedIndex > 0 || cmbMonth.SelectedIndex > 0 || cmbYear.SelectedIndex >
0)
        {
            try
            {
                if (cmbFaculty.SelectedIndex > -1 && cmbMonth.SelectedIndex > -1 &&
cmbYear.SelectedIndex > -1)
                {
                    string batch = cmbFaculty.Text + "/" + cmbMonth.Text + "/" + cmbYear.Text;
                    txtBatchName.Text = batch;

                }
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
        }
    }

    private void cmbFaculty_SelectedIndexChanged(object sender, EventArgs e)
    {
        if (cmbFaculty.SelectedIndex > 0 || cmbMonth.SelectedIndex > 0 || cmbYear.SelectedIndex >
0)
        {
            try
            {
                if (cmbFaculty.SelectedIndex > -1 && cmbMonth.SelectedIndex > -1 &&
cmbYear.SelectedIndex > -1)
                {
                    string batch = cmbFaculty.Text + "/" + cmbMonth.Text + "/" + cmbYear.Text;
                    txtBatchName.Text = batch;
```

```
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }
  }
}
```

FirstYearFeeManagement Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
  public partial class ManageFirstYearFeePlan : Form
  {
    public ManageFirstYearFeePlan()
    {
      InitializeComponent();
    }
```

```
ClassDataAccess idac = new ClassDataAccess();
BusinessLogicClass iblc = new BusinessLogicClass();
ClassFeeManagement fmc = new ClassFeeManagement();
private void FirstYearFeeFrm_Load(object sender, EventArgs e)
{
   cmbFaculty.DataSource = idac.SelectFaculties();
   cmbFaculty.DisplayMember = "FacultyName";
   cmbFaculty.ValueMember = "FacultyName";
   dgvFirstYearFeeDetails.DataSource = fmc.GetAllFirstYearFeeDetails();
   cmbFaculty.SelectedIndex = -1;
}

private void btnSave_Click(object sender, EventArgs e)
{
   if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text ==
"" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text == "" ||
cmbFaculty.SelectedIndex < 0)
      {
         MessageBox.Show("Please Provide All details First");
         return;
      }
      else
      {
        SaveFirstYearFeeDetails();
      }
}

private void SaveFirstYearFeeDetails()
{
   try
   {
      string faculty = cmbFaculty.Text;
      double FirstYearAdmission = Convert.ToDouble(txtAdmissionFee.Text);
```

```
            double FirstYearResource = Convert.ToDouble(txtResourceFee.Text);
            double FirstYearRegistration = Convert.ToDouble(txtRegistrationFee.Text);
            double FirstYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double FirstYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double FirstYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            bool result = iblc.FirstYearFeeManagement(0, faculty, FirstYearAdmission,
FirstYearResource, FirstYearRegistration, FirstYearFirstInstallment, FirstYearSecondInstallment,
FirstYearThirdInstallment, 1);
            if (result == true)
            {
               MessageBox.Show("FEE PLAN SUCCESSFULLY SAVED");
               dgvFirstYearFeeDetails.DataSource = fmc.GetAllFirstYearFeeDetails();
            }
            else
            {
               MessageBox.Show("SOME ERRORS OCCURED WHILE PERFORMING THE
OPERATION");
            }
         }
         catch (Exception ex)
         {

            MessageBox.Show(ex.Message);
         }
      }

      private void btnUpdate_Click(object sender, EventArgs e)
      {
         DialogResult dr = new DialogResult();
         dr = MessageBox.Show("ARE YOU SURE YOU WANT TO UPDATE THIS???",
"CONFIRM DELETE", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
         if (dr == DialogResult.Yes)
         {
```

```
            if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text
== "" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text ==
"" || cmbFaculty.SelectedIndex < 0)
            {
                MessageBox.Show("Please Provide All details First");
                return;
            }
            else
            {
                UpdateFirstYearFeeDetails();
            }
        }
        else
        {
            return;
        }
    }
    private void UpdateFirstYearFeeDetails()
    {
        try
        {
            int FeeId = Convert.ToInt16(txtFeeId.Text);
            string faculty = cmbFaculty.Text;
            double FirstYearAdmission = Convert.ToDouble(txtAdmissionFee.Text);
            double FirstYearResource = Convert.ToDouble(txtResourceFee.Text);
            double FirstYearRegistration = Convert.ToDouble(txtRegistrationFee.Text);
            double FirstYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double FirstYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double FirstYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            bool result = iblc.FirstYearFeeManagement(FeeId, faculty, FirstYearAdmission,
FirstYearResource, FirstYearRegistration, FirstYearFirstInstallment, FirstYearSecondInstallment,
FirstYearThirdInstallment, 2);
            if (result == true)
```

```
            {
                MessageBox.Show("FEE PLAN SUCCESSFULLY UPDATED");
                dgvFirstYearFeeDetails.DataSource = fmc.GetAllFirstYearFeeDetails();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED WHILE PERFORMING THE
OPERATION");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        txtAdmissionFee.Clear();
        txtFeeId.Clear();
        txtFirstInstallment.Clear();
        txtRegistrationFee.Clear();
        txtResourceFee.Clear();
        txtSecondInstallment.Clear();
        txtThirdInstallment.Clear();
    }

    private void dgvFirstYearFeeDetails_Click(object sender, EventArgs e)
    {
        GridViewToTextBoxBinding();
    }
```

```
    private void GridViewToTextBoxBinding()
    {
        try
        {
            txtFeeId.Text = dgvFirstYearFeeDetails.SelectedRows[0].Cells[0].Value.ToString();
            txtAdmissionFee.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtResourceFee.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[3].Value.ToString();
            txtRegistrationFee.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[4].Value.ToString();
            txtFirstInstallment.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[5].Value.ToString();
            txtSecondInstallment.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[6].Value.ToString();
            txtThirdInstallment.Text =
dgvFirstYearFeeDetails.SelectedRows[0].Cells[7].Value.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnDelete_Click(object sender, EventArgs e)
    {
        DeleteFirstYearFeeDetails();
    }

    private void DeleteFirstYearFeeDetails()
    {
        try
```

```
        {
            int FeeId = Convert.ToInt16(txtFeeId.Text);
            string faculty = cmbFaculty.Text;
            double FirstYearAdmission = Convert.ToDouble(txtAdmissionFee.Text);
            double FirstYearResource = Convert.ToDouble(txtResourceFee.Text);
            double FirstYearRegistration = Convert.ToDouble(txtRegistrationFee.Text);
            double FirstYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double FirstYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double FirstYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            bool result = iblc.FirstYearFeeManagement(FeeId, faculty, FirstYearAdmission,
FirstYearResource, FirstYearRegistration, FirstYearFirstInstallment, FirstYearSecondInstallment,
FirstYearThirdInstallment, 3);
            if (result == true)
            {
                MessageBox.Show("FEE PLAN SUCCESSFULLY DELTED");
                dgvFirstYearFeeDetails.DataSource = fmc.GetAllFirstYearFeeDetails();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED WHILE PERFORMING THE
OPERATION");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void txtAdmissionFee_TextChanged(object sender, EventArgs e)
    {
        txtAdmissionFee.BackColor = Color.White;
```

```
        }

        private void txtThirdInstallment_MouseClick(object sender, MouseEventArgs e)
        {
            lblThirdInstallment.Text = "";
        }
    }
}
```

SecondYearFeeManagement Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class ManageSecondYearFeePlan : Form
    {
        public ManageSecondYearFeePlan()
        {
            InitializeComponent();
        }
        BusinessLogicClass iblc = new BusinessLogicClass();
        ClassFeeManagement fmc = new ClassFeeManagement();
        ClassDataAccess idac = new ClassDataAccess();
        private void btnSave_Click(object sender, EventArgs e)
```

```
    {
        if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text == "" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text == "" || cmbFaculty.SelectedIndex < 0)
        {
            MessageBox.Show("Please Provide All details First");
            return;
        }
        else
        {
            NewSecondYearFee();
        }
    }

    private void NewSecondYearFee()
    {
        try
        {
            bool result = false;
            string faculty = cmbFaculty.Text;
            double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
            double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
            double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
            double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            int Mode = 1;
            result = iblc.SecondYearFeeManagement(0, faculty,
                AdmissionFee,
                ResourceFee,
                RegistrationFee,
                SecondYearFirstInstallment,
                SecondYearSecondInstallment,
```

```
            SecondYearThirdInstallment,
            Mode);
        if (result == true)
        {
            MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY SAVED");
            dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
        }
        else
        {
            MessageBox.Show("Some errors occured");
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void SecondYearFeeFrm_Load(object sender, EventArgs e)
{
    dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
    cmbFaculty.DataSource = idac.SelectFaculties();
    cmbFaculty.DisplayMember = "FacultyName";
    cmbFaculty.ValueMember = "FacultyName";
    cmbFaculty.SelectedIndex = -1;
}

private void dgvSecondYearFeeDetails_Click(object sender, EventArgs e)
{
    try
    {
        txtFeeId.Text = dgvSecondYearFeeDetails.SelectedRows[0].Cells[0].Value.ToString();
```

```
            txtAdmissionFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtResourceFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[3].Value.ToString();
            txtRegistrationFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[4].Value.ToString();
            txtFirstInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[5].Value.ToString();
            txtSecondInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[6].Value.ToString();
            txtThirdInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[7].Value.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnUpdate_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {

            if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text
== "" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text ==
"" || cmbFaculty.SelectedIndex < 0)
            {
                MessageBox.Show("Please Provide Batch Name First");
```

```
            return;
        }
        else
        {
            UpdateSecondYearFee();
            ClearSecondYearFeeFields();
        }

    }
}

private void UpdateSecondYearFee()
{
    try
    {
        bool result = false;
        int SecondYearFeeId = Convert.ToInt32(txtFeeId.Text);
        string faculty = cmbFaculty.Text;
        double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
        double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
        double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
        double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
        double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
        double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
        int Mode = 2;
        result = iblc.SecondYearFeeManagement(SecondYearFeeId, faculty,
            AdmissionFee,
            ResourceFee,
            RegistrationFee,
            SecondYearFirstInstallment,
            SecondYearSecondInstallment,
            SecondYearThirdInstallment,
            Mode);
```

```
            if (result == true)
            {
                MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY UPDATED");
                dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
            }
            else
            {
                MessageBox.Show("Some errors occured");
            }
        }
        catch (Exception EX)
        {

            MessageBox.Show(EX.Message);
        }
    }

    private void btnDelete_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO DELETE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            DeleteSecondYearFee();
            ClearSecondYearFeeFields();
        }
        else
        {
            ClearSecondYearFeeFields();
            return;
        }
    }
```

```
private void DeleteSecondYearFee()
{
    try
    {
        bool result = false;
        int SecondYearFeeId = Convert.ToInt32(txtFeeId.Text);
        string faculty = cmbFaculty.Text;
        double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
        double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
        double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
        double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
        double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
        double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
        int Mode = 3;
        result = iblc.SecondYearFeeManagement(SecondYearFeeId, faculty,
            AdmissionFee,
            ResourceFee,
            RegistrationFee,
            SecondYearFirstInstallment,
            SecondYearSecondInstallment,
            SecondYearThirdInstallment,
            Mode);
        if (result == true)
        {
            MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY DELETED ");
            dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
        }
        else
        {
            MessageBox.Show("Some errors occured");
        }
    }
```

```
        catch (Exception EX)
        {

            MessageBox.Show(EX.Message);
        }
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        ClearSecondYearFeeFields();
    }

    private void ClearSecondYearFeeFields()
    {
        txtAdmissionFee.Clear();
        txtFeeId.Clear();
        txtFirstInstallment.Clear();
        txtRegistrationFee.Clear();
        txtResourceFee.Clear();
        txtSecondInstallment.Clear();
        txtThirdInstallment.Clear();
    }
    }
}
```

ThirdYearFeeManagement Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
   public partial class ManageSecondYearFeePlan : Form
   {
      public ManageSecondYearFeePlan()
      {
         InitializeComponent();
      }
      BusinessLogicClass iblc = new BusinessLogicClass();
      ClassFeeManagement fmc = new ClassFeeManagement();
      ClassDataAccess idac = new ClassDataAccess();
      private void btnSave_Click(object sender, EventArgs e)
      {
         if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text ==
"" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text == "" ||
cmbFaculty.SelectedIndex < 0)
         {
            MessageBox.Show("Please Provide All details First");
            return;
         }
         else
         {
            NewSecondYearFee();
         }
      }

      private void NewSecondYearFee()
      {
```

```
try
{
    bool result = false;
    string faculty = cmbFaculty.Text;
    double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
    double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
    double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
    double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
    double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
    double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
    int Mode = 1;
    result = iblc.SecondYearFeeManagement(0, faculty,
        AdmissionFee,
        ResourceFee,
        RegistrationFee,
        SecondYearFirstInstallment,
        SecondYearSecondInstallment,
        SecondYearThirdInstallment,
        Mode);
    if (result == true)
    {
        MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY SAVED");
        dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
    }
    else
    {
        MessageBox.Show("Some errors occured");
    }
}
catch (Exception ex)
{

    MessageBox.Show(ex.Message);
```

```
        }
    }

    private void SecondYearFeeFrm_Load(object sender, EventArgs e)
    {
        dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
        cmbFaculty.DataSource = idac.SelectFaculties();
        cmbFaculty.DisplayMember = "FacultyName";
        cmbFaculty.ValueMember = "FacultyName";
        cmbFaculty.SelectedIndex = -1;
    }

    private void dgvSecondYearFeeDetails_Click(object sender, EventArgs e)
    {
        try
        {
            txtFeeId.Text = dgvSecondYearFeeDetails.SelectedRows[0].Cells[0].Value.ToString();
            txtAdmissionFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtResourceFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[3].Value.ToString();
            txtRegistrationFee.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[4].Value.ToString();
            txtFirstInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[5].Value.ToString();
            txtSecondInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[6].Value.ToString();
            txtThirdInstallment.Text =
dgvSecondYearFeeDetails.SelectedRows[0].Cells[7].Value.ToString();
        }
        catch (Exception ex)
        {
```

```
            MessageBox.Show(ex.Message);
        }
    }

    private void btnUpdate_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {

            if (txtAdmissionFee.Text == "" || txtFirstInstallment.Text == "" || txtRegistrationFee.Text
== "" || txtResourceFee.Text == "" || txtSecondInstallment.Text == "" || txtThirdInstallment.Text ==
"" || cmbFaculty.SelectedIndex < 0)
            {
                MessageBox.Show("Please Provide Batch Name First");
                return;
            }
            else
            {
                UpdateSecondYearFee();
                ClearSecondYearFeeFields();
            }

        }
    }

    private void UpdateSecondYearFee()
    {
        try
        {
            bool result = false;
```

```
            int SecondYearFeeId = Convert.ToInt32(txtFeeId.Text);
            string faculty = cmbFaculty.Text;
            double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
            double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
            double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
            double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            int Mode = 2;
            result = iblc.SecondYearFeeManagement(SecondYearFeeId, faculty,
                AdmissionFee,
                ResourceFee,
                RegistrationFee,
                SecondYearFirstInstallment,
                SecondYearSecondInstallment,
                SecondYearThirdInstallment,
                Mode);
            if (result == true)
            {
                MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY UPDATED");
                dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
            }
            else
            {
                MessageBox.Show("Some errors occured");
            }
        }
        catch (Exception EX)
        {

            MessageBox.Show(EX.Message);
        }
    }
```

```
    private void btnDelete_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO DELETE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            DeleteSecondYearFee();
            ClearSecondYearFeeFields();
        }
        else
        {
            ClearSecondYearFeeFields();
            return;
        }
    }

    private void DeleteSecondYearFee()
    {
        try
        {
            bool result = false;
            int SecondYearFeeId = Convert.ToInt32(txtFeeId.Text);
            string faculty = cmbFaculty.Text;
            double AdmissionFee = Convert.ToDouble(txtAdmissionFee.Text);
            double ResourceFee = Convert.ToDouble(txtResourceFee.Text);
            double RegistrationFee = Convert.ToDouble(txtRegistrationFee.Text);
            double SecondYearFirstInstallment = Convert.ToDouble(txtFirstInstallment.Text);
            double SecondYearSecondInstallment = Convert.ToDouble(txtSecondInstallment.Text);
            double SecondYearThirdInstallment = Convert.ToDouble(txtThirdInstallment.Text);
            int Mode = 3;
            result = iblc.SecondYearFeeManagement(SecondYearFeeId, faculty,
```

```
            AdmissionFee,
            ResourceFee,
            RegistrationFee,
            SecondYearFirstInstallment,
            SecondYearSecondInstallment,
            SecondYearThirdInstallment,
            Mode);
        if (result == true)
        {
            MessageBox.Show("SECOND YEAR FEE PLAN SUCCESSFULLY DELETED ");
            dgvSecondYearFeeDetails.DataSource = fmc.GetAllSecondYearFeeDetails();
        }
        else
        {
            MessageBox.Show("Some errors occured");
        }
    }
    catch (Exception EX)
    {

        MessageBox.Show(EX.Message);
    }
}

private void btnClose_Click(object sender, EventArgs e)
{
    ClearSecondYearFeeFields();
}

private void ClearSecondYearFeeFields()
{
    txtAdmissionFee.Clear();
    txtFeeId.Clear();
```

```
            txtFirstInstallment.Clear();
            txtRegistrationFee.Clear();
            txtResourceFee.Clear();
            txtSecondInstallment.Clear();
            txtThirdInstallment.Clear();
        }
        }
    }
```

Inquiry Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{

    public partial class ManageInquiry : Form
    {
        public ManageInquiry()
        {
            InitializeComponent();
        }
        BusinessLogicClass iblc = new BusinessLogicClass();
        ClassInquiry ic = new ClassInquiry();
```

```
private void SaveInquiry()
{
   try
   {
      bool result = iblc.NewInquiry(0, txtStudentName.Text, txtAddress.Text, txtContact.Text,
txtInterestedProgram.Text, Convert.ToDateTime(dtpVisitedDate.Text), 1);
      if (result == true)
      {
         MessageBox.Show("NEW INQUIRY HAS BEEN SAVED");
         dgvInquiryDetails.DataSource = ic.GetAllInquiry();
      }
      else
      {
         MessageBox.Show("SOME ERROR HAS OCCURED");
      }
   }
   catch (Exception ex)
   {

      MessageBox.Show(ex.Message);
   }
}

private void NewInquiry_Load(object sender, EventArgs e)
{
   dgvInquiryDetails.DataSource = ic.GetAllInquiry();
}

private void dgvInquiryDetails_Click(object sender, EventArgs e)
{
   txtStudentId.Text = dgvInquiryDetails.SelectedRows[0].Cells[0].Value.ToString();
```

```
        txtStudentCode.Text = dgvInquiryDetails.SelectedRows[0].Cells[1].Value.ToString();
        txtStudentName.Text = dgvInquiryDetails.SelectedRows[0].Cells[2].Value.ToString();
        txtAddress.Text = dgvInquiryDetails.SelectedRows[0].Cells[3].Value.ToString();
        txtContact.Text = dgvInquiryDetails.SelectedRows[0].Cells[4].Value.ToString();
        txtInterestedProgram.Text = dgvInquiryDetails.SelectedRows[0].Cells[5].Value.ToString();
        dtpVisitedDate.Text = dgvInquiryDetails.SelectedRows[0].Cells[6].Value.ToString();
    }




    private void DeleteInquiry()
    {
        try
        {
            bool result = iblc.NewInquiry(Convert.ToInt32(txtStudentId.Text), txtStudentName.Text,
txtAddress.Text, txtContact.Text, txtInterestedProgram.Text,
Convert.ToDateTime(dtpVisitedDate.Text), 3);
            if (result == true)
            {
                MessageBox.Show("INQUIRY HAS BEEN DELETED");
                dgvInquiryDetails.DataSource = ic.GetAllInquiry();
            }
            else
            {
                MessageBox.Show("SOME ERROR HAS OCCURED");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
```

```
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        txtAddress.Clear();
        txtContact.Clear();
        txtInterestedProgram.Clear();
        txtStudentCode.Clear();
        txtStudentId.Clear();
        txtStudentName.Clear();
    }

    private void button1_Click_1(object sender, EventArgs e)
    {
        if (txtStudentName.Text == "" || txtInterestedProgram.Text == "" || txtContact.Text == "" ||
txtAddress.Text == "")
        {
            MessageBox.Show("Please Provide All Details First");
            return;
        }
        else
        {
            SaveInquiry();
        }
    }

    private void button2_Click_1(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
```

```
        {
            if (txtStudentName.Text == "" || txtInterestedProgram.Text == "" || txtContact.Text == "" ||
txtAddress.Text == "")
            {
                MessageBox.Show("Please Provide All Details First");
                return;
            }
            else
            {
                UpdateInquiry();
            }
        }
        else
        {
            return;
        }
    }

    private void UpdateInquiry()
    {
        try
        {
            bool result = iblc.NewInquiry(Convert.ToInt32(txtStudentId.Text), txtStudentName.Text,
txtAddress.Text, txtContact.Text, txtInterestedProgram.Text,
Convert.ToDateTime(dtpVisitedDate.Text), 2);
            if (result == true)
            {
                MessageBox.Show("INQUIRY HAS BEEN UPDATED");
                dgvInquiryDetails.DataSource = ic.GetAllInquiry();
            }
            else
            {
                MessageBox.Show("SOME ERROR HAS OCCURED");
```

```
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void button3_Click_1(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            DeleteInquiry();
        }
        else
        {
            return;
        }
    }
  }
}
```

ManageRole Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
   public partial class ManageRole : Form
   {
      public ManageRole()
      {
         InitializeComponent();
      }
      BusinessLogicClass iblc = new BusinessLogicClass();
      ClassDataAccess idac = new ClassDataAccess();
      ClassRole rc = new ClassRole();
      private void btnAddRole_Click(object sender, EventArgs e)
      {
         if (txtRoleName.Text == "")
         {
            MessageBox.Show("Role cannot be empty");
         }
         else if (txtRoleDesc.Text == "")
         {
            MessageBox.Show("The description field cannot be blank");
         }
         else
         {
            try
            {
               bool result = iblc.ManageRole(0, txtRoleName.Text, txtRoleDesc.Text, 1);
               if (result == true)
               {
```

```
            MessageBox.Show("NEW ROLE SUCCESSFULLY CREATED");
            dgvRoleDetails.DataSource = rc.GetAllRoles();

        }
        else
        {
            MessageBox.Show("ERRORS IN CREATING ROLE");
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
  }
}

private void RoleFrm_Load(object sender, EventArgs e)
{
  dgvRoleDetails.DataSource = rc.GetAllRoles();
}

private void dgvRoleDetails_Click(object sender, EventArgs e)
{
  try
  {

    txtRoleId.Text = dgvRoleDetails.SelectedRows[0].Cells[0].Value.ToString();
    txtRoleCode.Text = dgvRoleDetails.SelectedRows[0].Cells[1].Value.ToString();
    txtRoleName.Text = dgvRoleDetails.SelectedRows[0].Cells[2].Value.ToString();
    txtRoleDesc.Text = dgvRoleDetails.SelectedRows[0].Cells[3].Value.ToString();
  }
  catch (Exception ex)
```

```
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnClose_Click(object sender, EventArgs e)
    {
        txtRoleCode.Clear();
        txtRoleDesc.Clear();
        txtRoleId.Clear();
        txtRoleName.Clear();
    }

    private void btnUpdateRole_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {

            if (txtRoleName.Text == "")
            {
                MessageBox.Show("Please Provide Batch Name First");
                return;
            }
            else
            {
                UpdateRole();
            }

        }
```

```
    }

    private void UpdateRole()
    {
        try
        {
            bool result = iblc.ManageRole(Convert.ToInt32(txtRoleId.Text), txtRoleName.Text,
txtRoleDesc.Text, 2);
            if (result == true)
            {
                MessageBox.Show("ROLE SUCCESSFULLY UPDATED");
                dgvRoleDetails.DataSource = rc.GetAllRoles();

            }
            else
            {
                MessageBox.Show("ERRORS IN UPDATING ROLE");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnDeleteRole_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
```

```
            DeleteRole();
        }
        else
        {
            return;
        }
    }

    private void DeleteRole()
    {
        try
        {
            bool result = iblc.ManageRole(Convert.ToInt32(txtRoleId.Text), txtRoleName.Text,
txtRoleDesc.Text, 3);
            if (result == true)
            {
                MessageBox.Show("ROLE SUCCESSFULLY DELETED");
                dgvRoleDetails.DataSource = rc.GetAllRoles();

            }
            else
            {
                MessageBox.Show("ERRORS IN DELETING ROLE");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }
  }
}
```

Faculty Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
   public partial class Faculty : Form
   {
      public Faculty()
      {
         InitializeComponent();
      }
      BusinessLogicClass iblc = new BusinessLogicClass();
      ClassDataAccess idac = new ClassDataAccess();
      private void button2_Click(object sender, EventArgs e)
      {
```

```
        if (txtFacultyName.Text == "")
        {
            MessageBox.Show("FACULTY NAME CANNOT BE BLANK");
        }
        else if (txtDescription.Text == "")
        {
            MessageBox.Show("THE DESCRIPTION FIELD CANNOT BE BLANK");
        }
        else
        CreateNewFaculty();
    }

    private void CreateNewFaculty()
    {
        bool result = iblc.FacultyManagement(0, txtFacultyName.Text, txtDescription.Text, 1);
        if (result == true)
        {
            MessageBox.Show("NEW FACULTY SUCCESSFULLY CREATED");
            dgvFacultyDetails.DataSource = idac.SelectFaculties();
        }
        else
        {
            MessageBox.Show("SOME ERRORS OCCURED");
        }
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        dgvFacultyDetails.DataSource = idac.SelectFaculties();
    }

    private void dgvFacultyDetails_Click(object sender, EventArgs e)
    {
```

```
        try
        {
            txtFacultyId.Text = dgvFacultyDetails.SelectedRows[0].Cells[0].Value.ToString();
            txtFacultyCode.Text = dgvFacultyDetails.SelectedRows[0].Cells[1].Value.ToString();
            txtFacultyName.Text = dgvFacultyDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtDescription.Text = dgvFacultyDetails.SelectedRows[0].Cells[3].Value.ToString();
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        DialogResult ConfirmUpdate = new DialogResult();
        ConfirmUpdate = MessageBox.Show("ARE YOU SURE YOU WANT TO UPDATE???",
"CONFIRM UPDATE", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (ConfirmUpdate == DialogResult.Yes)
        {
            bool result = iblc.FacultyManagement(Convert.ToInt32(txtFacultyId.Text),
txtFacultyName.Text, txtDescription.Text, 2);
            if (result == true)
            {
                MessageBox.Show("NEW FACULTY SUCCESSFULLY UPDATED");
                dgvFacultyDetails.DataSource = idac.SelectFaculties();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
```

```
    }

    private void button4_Click(object sender, EventArgs e)
    {
        DialogResult ConfirmDelete = new DialogResult();
        ConfirmDelete = MessageBox.Show("ARE YOU SURE YOU WANT TO DELETE???",
"CONFIRM DELETE",MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (ConfirmDelete == DialogResult.Yes)
        {
            bool result = iblc.FacultyManagement(Convert.ToInt32(txtFacultyId.Text),
txtFacultyName.Text, txtDescription.Text, 3);
            if (result == true)
            {
                MessageBox.Show("NEW FACULTY SUCCESSFULLY DELETED");
                dgvFacultyDetails.DataSource = idac.SelectFaculties();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
        else
        {
            return;
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void button1_Click(object sender, EventArgs e)
```

```
        {


        }


    }
}
```

ManageStudnets Form

```
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 IsmtBusinessLogicLayer;
using IsmtDataAccessLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class ManageStudents : Form
    {
        public ManageStudents()
        {
            InitializeComponent();
        }
        ClassBatch bc = new ClassBatch();
        ClassDataAccess idac = new ClassDataAccess();
        ClassAwardingBody awbc = new ClassAwardingBody();
        ClassStudentManagement sc = new ClassStudentManagement();
        BusinessLogicClass iblc = new BusinessLogicClass();
        private void ManageStudents_Load(object sender, EventArgs e)
```

```
{
    try
    {
    cmbBatchNumber.DataSource = bc.SelectAllBatch();
    cmbBatchNumber.ValueMember = "BatchName";
    cmbBatchNumber.DisplayMember = "BatchName";
    cmbBatchNumber.SelectedIndex = -1;
    cmbAwardingBody.DataSource = awbc.SelectAllAwardingBody();
    cmbAwardingBody.DisplayMember = "AwardingBodyName";
    cmbAwardingBody.ValueMember = "AwardingBodyName";
    cmbAwardingBody.SelectedIndex = -1;
    cmbFaculty.DataSource = idac.SelectFaculties();
    cmbFaculty.DisplayMember = "FacultyName";
    cmbFaculty.ValueMember = "FacultyName";
    cmbFaculty.SelectedIndex = -1;
    dgvStudentDetails.DataSource = sc.SelectAllStudents();
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void button4_Click(object sender, EventArgs e)
{
    this.Close();
}

private void btnSaveDetails_Click(object sender, EventArgs e)
{
    if (cmbAwardingBody.SelectedIndex < 0)
    {
```

```
        MessageBox.Show("Provide AwardingBody First");
        return;
    }
    else if (cmbBatchNumber.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Batch First");
        return;
    }

        else if(cmbFaculty.SelectedIndex<0){
            MessageBox.Show("Provide Faculty First");
            return;
    }
    else if (cmbGender.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Gender First");
        return;
    }
    else if (cmbSemester.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Semester First");
        return;
    }
    else if (cmbStatus.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Student Status First");
        return;
    }
    else if (txtStudentName.Text == "")
    {
        MessageBox.Show("Provide Student Name First");
        return;
    }
```

```
    else if (txtStudentContact.Text == "")
    {
        MessageBox.Show("Provide Student Contact First");
        return;
    }
    else
    {
            NewStudent();
    }

}

private void NewStudent()
{
    try
    {
        bool result = iblc.ManageStudent(0,
        cmbFaculty.Text,
        cmbBatchNumber.Text,
        cmbAwardingBody.Text,
        cmbSemester.Text,
        txtStudentName.Text,
        dtpDateOfBirth.Text,
        cmbGender.Text,
        txtStudentAddress.Text,
        txtStudentContact.Text,
        txtGuardianName.Text,
        txtGuardianAddress.Text,
        txtGuardianContact.Text,
        cmbStatus.Text, 1);
        if (result == true)
        {
            MessageBox.Show("STUDENT INFORMATION HAS BEEN SAVED");
```

```
            dgvStudentDetails.DataSource = sc.SelectAllStudents();
        }
        else
        {
            MessageBox.Show("SOME ERROR HAS OCCURED");
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show(ex.Message);
    }
}

private void btnUpdateDetails_Click(object sender, EventArgs e)
{
    if (cmbAwardingBody.SelectedIndex < 0)
    {
        MessageBox.Show("Provide AwardingBody First");
        return;
    }
    else if (cmbBatchNumber.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Batch First");
        return;
    }

    else if (cmbFaculty.SelectedIndex < 0)
    {
        MessageBox.Show("Provide Faculty First");
        return;
    }
    else if (cmbGender.SelectedIndex < 0)
```

```
        {
            MessageBox.Show("Provide Gender First");
            return;
        }
        else if (cmbSemester.SelectedIndex < 0)
        {
            MessageBox.Show("Provide Semester First");
            return;
        }
        else if (cmbStatus.SelectedIndex < 0)
        {
            MessageBox.Show("Provide Student Status First");
            return;
        }
        else if (txtStudentName.Text == "")
        {
            MessageBox.Show("Provide Student Name First");
            return;
        }
        else if (txtStudentContact.Text == "")
        {
            MessageBox.Show("Provide Student Contact First");
            return;
        }
        else
        {
            DialogResult dR = new DialogResult();
            dR = MessageBox.Show("ARE YOU SURE TO Update???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
            if (dR == DialogResult.OK)
            {
              UpdateStudent();
            }
```

```
            else
            {
                return;
            }
        }
    }

    private void UpdateStudent()
    {
        try
        {
            bool result = iblc.ManageStudent(Convert.ToInt32(txtStudentId.Text),
            cmbFaculty.Text,
            cmbBatchNumber.Text,
            cmbAwardingBody.Text,
            cmbSemester.Text,
            txtStudentName.Text,
            dtpDateOfBirth.Text,
            cmbGender.Text,
            txtStudentAddress.Text,
            txtStudentContact.Text,
            txtGuardianName.Text,
            txtGuardianAddress.Text,
            txtGuardianContact.Text,
            cmbStatus.Text, 2);
            if (result == true)
            {
                MessageBox.Show("STUDENT INFORMATION HAS BEEN UPDATED");
                dgvStudentDetails.DataSource = sc.SelectAllStudents();
            }
            else
            {
                MessageBox.Show("SOME ERROR HAS OCCURED");
```

```
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnDeleteDetails_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO DELETE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            DeleteStudent();
        }
        else
        {
            return;
        }
    }

    private void DeleteStudent()
    {
        try
        {
            bool result = iblc.ManageStudent(Convert.ToInt32(txtStudentId.Text),
            cmbFaculty.Text,
            cmbBatchNumber.Text,
            cmbAwardingBody.Text,
            cmbSemester.Text,
```

```
            txtStudentName.Text,
            dtpDateOfBirth.Text,
            cmbGender.Text,
            txtStudentAddress.Text,
            txtStudentContact.Text,
            txtGuardianName.Text,
            txtGuardianAddress.Text,
            txtGuardianContact.Text,
            cmbStatus.Text, 3);
             if (result == true)
             {
                MessageBox.Show("STUDENT INFORMATION HAS BEEN DELETED");
                dgvStudentDetails.DataSource = sc.SelectAllStudents();
             }
             else
             {
                MessageBox.Show("SOME ERROR HAS OCCURED");
             }
          }
          catch (Exception ex)
          {

             MessageBox.Show(ex.Message);
          }
       }

       private void dgvStudentDetails_Click(object sender, EventArgs e)
       {
          txtStudentId.Text = dgvStudentDetails.SelectedRows[0].Cells[0].Value.ToString();
          txtStudentCode.Text = dgvStudentDetails.SelectedRows[0].Cells[1].Value.ToString();
       }

       private void dgvStudentDetails_CellContentClick(object sender, DataGridViewCellEventArgs
```

```
e)
        {


        }


    }
}
```

ManageUser Form

```
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 IsmtDataAccessLayer;
using IsmtBusinessLogicLayer;
namespace ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class ManageUser : Form
    {
        public ManageUser()
        {
            InitializeComponent();
        }
        ClassRole rc = new ClassRole();
        ClassUserManagement uc = new ClassUserManagement();
        BusinessLogicClass iblc = new BusinessLogicClass();
        private void UserFrm_Load(object sender, EventArgs e)
        {
```

```
    cmbRole.DataSource = rc.GetAllRoles();
    cmbRole.DisplayMember = "RoleName";
    cmbRole.ValueMember = "RoleName";
    cmbRole.SelectedIndex = -1;
    dgvUserDetails.DataSource = uc.SelectAllUsers();
}

private void btnCreateUser_Click(object sender, EventArgs e)
{
    if (txtUserName.Text == "")
    {
        MessageBox.Show("PLEASE PROVIDE USERNAME FIRST");
        txtUserName.BackColor = Color.Red;
        return;
    }
    else if (txtPassword.Text == "")
    {
        MessageBox.Show("PLEASE PROVIDE PASSWORD FIRST");
        txtPassword.BackColor = Color.Red;
        return;
    }
    else
    {
          NewUser();
    }

}

private void NewUser()
{
    try
    {
        bool rs = iblc.ManageUser(0, cmbRole.Text, txtUserName.Text, txtPassword.Text,
```

```
txtUserDesc.Text, 1);
            if (rs == true)
            {
                MessageBox.Show("NEW USER SUCCESSFULLY CREATED");
                dgvUserDetails.DataSource = uc.SelectAllUsers();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void dgvUserDetails_Click(object sender, EventArgs e)
    {
        try
        {
            txtUserId.Text = dgvUserDetails.SelectedRows[0].Cells[0].Value.ToString();
            txtUserName.Text = dgvUserDetails.SelectedRows[0].Cells[2].Value.ToString();
            txtUserCode.Text =  dgvUserDetails.SelectedRows[0].Cells[3].Value.ToString();
            txtPassword.Text = dgvUserDetails.SelectedRows[0].Cells[4].Value.ToString();
            txtUserDesc.Text = dgvUserDetails.SelectedRows[0].Cells[5].Value.ToString();

        }
        catch (Exception)
        {

            throw;
```

```
        }
    }

    private void btnUpdateUser_Click(object sender, EventArgs e)
    {

        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO UPDATE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {

            if (txtPassword.Text==""||txtUserName.Text=="")
            {
                MessageBox.Show("User Name or Password Field Cannot be Blank");
                return;
            }
            else
            {
                UpdateUser();
                ClearUserFields();
            }

        }
    }

    private void UpdateUser()
    {
        try
        {
            bool rs = iblc.ManageUser(Convert.ToInt32(txtUserId.Text), cmbRole.Text,
txtUserName.Text, txtPassword.Text, txtUserDesc.Text, 2);
            if (rs == true)
```

```
            {
                MessageBox.Show("NEW USER SUCCESSFULLY UPDATED");
                dgvUserDetails.DataSource = uc.SelectAllUsers();
            }
            else
            {
                MessageBox.Show("SOME ERRORS OCCURED");
            }
        }
        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);
        }
    }

    private void btnDeleteUser_Click(object sender, EventArgs e)
    {
        DialogResult dR = new DialogResult();
        dR = MessageBox.Show("ARE YOU SURE TO DELETE???", "CONFIRMATION
REQUIRED", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
        if (dR == DialogResult.OK)
        {
            DeleteUser();
            ClearUserFields();
        }
        else
        {
            ClearUserFields();
            return;
        }
    }
```

```
        private void DeleteUser()
        {
            try
            {
                bool rs = iblc.ManageUser(Convert.ToInt32(txtUserId.Text), cmbRole.Text,
txtUserName.Text, txtPassword.Text, txtUserDesc.Text, 3);
                if (rs == true)
                {
                    MessageBox.Show("NEW USER SUCCESSFULLY DELETED");
                    dgvUserDetails.DataSource = uc.SelectAllUsers();
                }
                else
                {
                    MessageBox.Show("SOME ERRORS OCCURED");
                }
            }
            catch (Exception ex)
            {

                MessageBox.Show(ex.Message);
            }
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            ClearUserFields();
        }

        private void ClearUserFields()
        {
            txtPassword.Clear();
            txtUserCode.Clear();
            txtUserDesc.Clear();
```

```
            txtUserId.Clear();
            txtUserName.Clear();
        }


    }
}
```

ProjectMain Form

```
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 ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class ProjectMainForm : Form
    {
        public ProjectMainForm()
        {
            InitializeComponent();
        }

        private void mANAGEFACULTYToolStripMenuItem_Click(object sender, EventArgs e)
        {
            pnlContainer.Controls.Clear();
            Faculty Faculty= new Faculty();
   Faculty.TopLevel = false;
```

```
pnlContainer.Controls.Add(Faculty);
Faculty.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
 Faculty.Dock = DockStyle.Fill;
 Faculty.Show();
    }

    private void mANAGEBATCHToolStripMenuItem_Click(object sender, EventArgs e)
    {
        pnlContainer.Controls.Clear();
        ManageBatch batch = new ManageBatch();
        batch.TopLevel = false;
        pnlContainer.Controls.Add(batch);
        batch.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
        batch.Dock = DockStyle.Fill;
        batch.Show();
    }

    private void mANAGEROLESToolStripMenuItem_Click(object sender, EventArgs e)
    {
        pnlContainer.Controls.Clear();
        ManageRole role = new ManageRole();
        role.TopLevel = false;
        pnlContainer.Controls.Add(role);
        role.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
        role.Dock = DockStyle.Fill;
        role.Show();
    }

    private void mANAGEUSERSToolStripMenuItem_Click(object sender, EventArgs e)
    {
        pnlContainer.Controls.Clear();
        ManageUser frm = new ManageUser();
        frm.TopLevel = false;
```

```
            pnlContainer.Controls.Add(frm);
           frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            frm.Dock = DockStyle.Fill;
            frm.Show();
        }

        private void mANAGEAWARDINGBODYToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            pnlContainer.Controls.Clear();
            ManageAwardingBody frm = new ManageAwardingBody();
            frm.TopLevel = false;
            pnlContainer.Controls.Add(frm);
            frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            frm.Dock = DockStyle.Fill;
            frm.Show();
        }

        private void mANAGESTUDENTSToolStripMenuItem_Click(object sender, EventArgs e)
        {
            pnlContainer.Controls.Clear();
            ManageStudents frm = new ManageStudents();
            frm.TopLevel = false;
            pnlContainer.Controls.Add(frm);
            frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
            frm.Dock = DockStyle.Fill;
            frm.Show();
        }

        private void fIRSTYEARToolStripMenuItem_Click(object sender, EventArgs e)
        {
            pnlContainer.Controls.Clear();
            ManageFirstYearFeePlan frm = new ManageFirstYearFeePlan();
```

```
    frm.TopLevel = false;
    pnlContainer.Controls.Add(frm);
    frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    frm.Dock = DockStyle.Fill;
    frm.Show();
}

private void sECONDYEARToolStripMenuItem_Click(object sender, EventArgs e)
{
    pnlContainer.Controls.Clear();
    ManageSecondYearFeePlan frm = new ManageSecondYearFeePlan();
    frm.TopLevel = false;
    pnlContainer.Controls.Add(frm);
    frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    frm.Dock = DockStyle.Fill;
    frm.Show();
}

private void tHIRDYEARToolStripMenuItem_Click(object sender, EventArgs e)
{
    pnlContainer.Controls.Clear();
    ManageThirdYearFeePlan frm = new ManageThirdYearFeePlan();
    frm.TopLevel = false;
    pnlContainer.Controls.Add(frm);
    frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    frm.Dock = DockStyle.Fill;
    frm.Show();
}


private void sEARCHToolStripMenuItem_Click(object sender, EventArgs e)
{
    pnlContainer.Controls.Clear();
```

```
    FeeTransaction frm = new FeeTransaction();
    frm.TopLevel = false;
    pnlContainer.Controls.Add(frm);
    frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    frm.Dock = DockStyle.Fill;
    frm.Show();
}

private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
    pnlContainer.Controls.Clear();
    ManageInquiry frm = new ManageInquiry();
    frm.TopLevel = false;
    pnlContainer.Controls.Add(frm);
    frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    frm.Dock = DockStyle.Fill;
    frm.Show();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void mINIMIZEToolStripMenuItem_Click(object sender, EventArgs e)
{
    ProjectMainForm frm = new ProjectMainForm();
    this.WindowState = FormWindowState.Minimized;

}

private void cLEARToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
            pnlContainer.Controls.Clear();
        }

        private void aDMINPANELToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void logOffToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LoginForm FormLogin = new LoginForm();
            this.Hide();
            FormLogin.Show();
        }


    }
}
```

ProjectMain Form (User)

```
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 ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class ProjectMainFormUser : Form
```

```
{
    public ProjectMainFormUser()
    {
        InitializeComponent();
    }

    private void toolStripMenuItem1_Click(object sender, EventArgs e)
    {
        pnlContainer.Controls.Clear();
        ManageInquiry frm = new ManageInquiry();
        frm.TopLevel = false;
        pnlContainer.Controls.Add(frm);
        frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
        frm.Dock = DockStyle.Fill;
        frm.Show();
    }

    private void mANAGESTUDENTSToolStripMenuItem_Click(object sender, EventArgs e)
    {
        pnlContainer.Controls.Clear();
        ManageStudents frm = new ManageStudents();
        frm.TopLevel = false;
        pnlContainer.Controls.Add(frm);
        frm.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
        frm.Dock = DockStyle.Fill;
        frm.Show();
    }

    private void exitToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
```

```
        private void logOffToolStripMenuItem_Click(object sender, EventArgs e)
        {
            LoginForm FormLogin = new LoginForm();
            this.Hide();
            FormLogin.Show();
        }

        private void mINIMIZEToolStripMenuItem_Click(object sender, EventArgs e)
        {
            ProjectMainForm frm = new ProjectMainForm();
            this.WindowState = FormWindowState.Minimized;
        }

        private void cLEARToolStripMenuItem_Click(object sender, EventArgs e)
        {
            pnlContainer.Controls.Clear();
        }
    }
}
```

Task 5

Critically review and test object oriented programming solution and analyses actual test results against expected result to identify discrepancies.

You may consider the following for writing the critical review of your program:

1.1.Introduction

1.2.Body

1.3.Conclusion

2. Perform Following types of test operation of you program:

- **Unit Testing**
- **Integrated Testing**
- **Stress Testing**

Introduction: ISMT MANAGEMENT SYSTEM

ISMT (International School of Management and Technology) is international college that offers various international educations to their student. Due to large student numbers, financial activities and inquiry numbers, it had decided to develop a software solution which can solve various problems they are facing due to the lack of proper management software.

ISMT MANAGEMENT SYSTEM is an OOP based college management software designed by Atut Gorkhali (student of ISMT, IT first semester) using C# .net technology. This software is GUI based application which has lots of features specifically developed to address the issues of ISMT. This software is designed to deal tasks like finance management, student information management and inquiry management etc.

Developer of this software has tried to give easy control and look to software and avoid being complicated. The controls in the program like saving information, updating and deleting is quite simple. This review paper will analyses and evaluate the software's features, strengths and weaknesses.

BODY

As described above, the main purpose of this application is to solve issues of ISMT like finance information management, student information management, faculty information management and inquiry information management. In this part, I have described about the key features of the software.

a. Maintain Student's Information:

Maintaining the information of the student, studying and have studied in this college is one of the core feature of the system. User is able to supply student's details and can save them into the database. There are also features like updating or deleting student. Student Management Form is simple and can save all necessary details like name, batch, faculty, status, guardian's details etc.

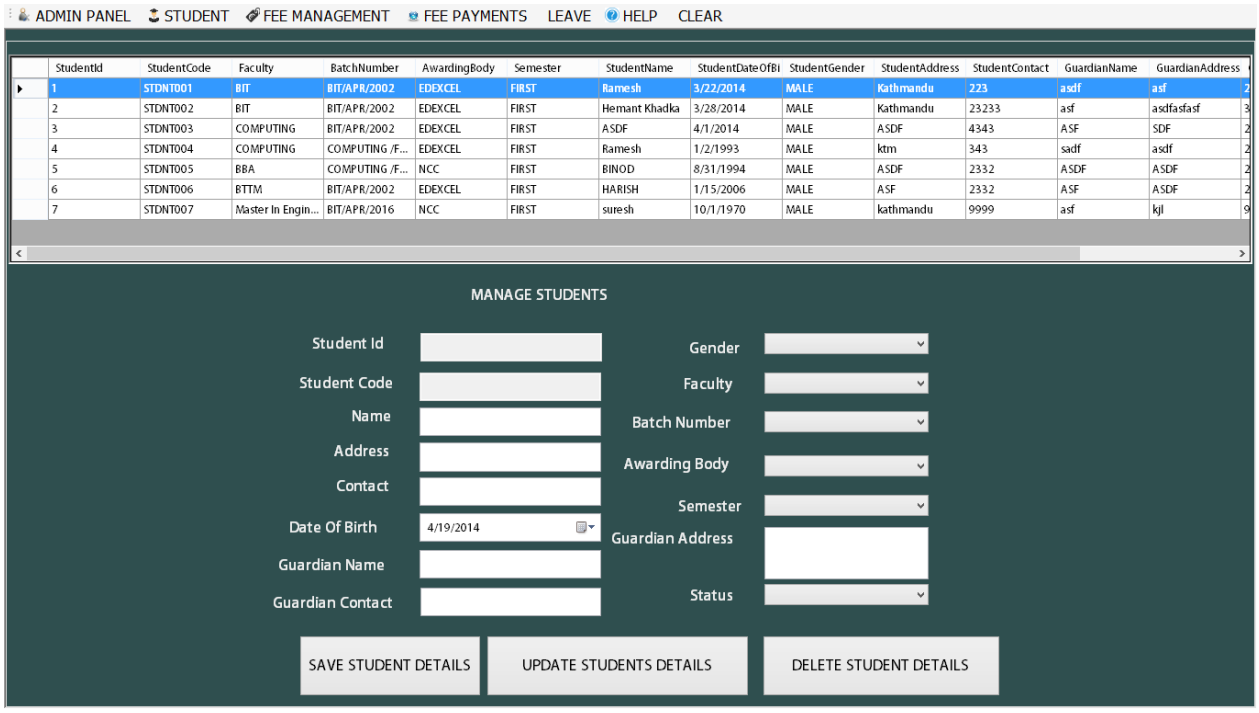


Figure 1 Student Management Form

b. Finance Management

Users with admin privilege have access to the finance management section of the system. They are able to update, delete and add new fee plan for each first year, second year and third year. Separated transaction management form allows performing payment procedure and saving the details in database. There is also features of updating, deleting the transaction history. Figure 2 and 3 shows how this application allows maintaining fee plans for each year and maintaining transaction history.

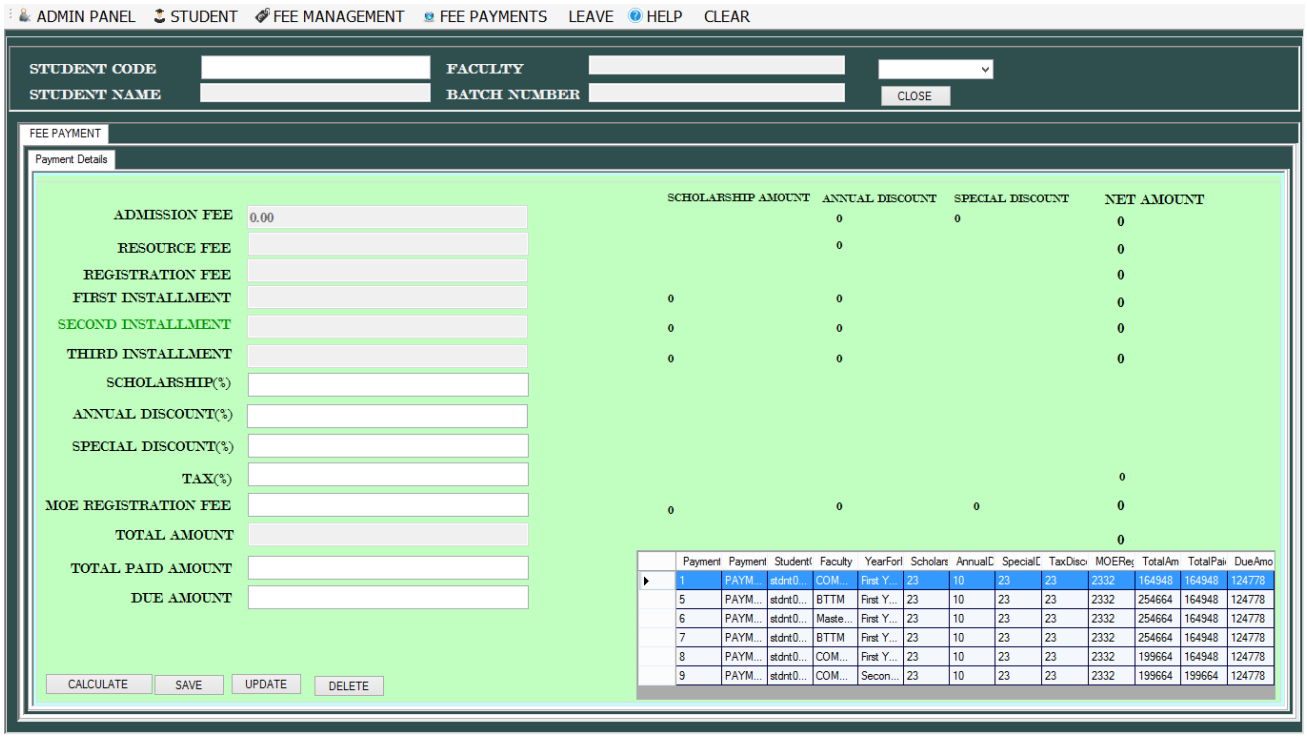


Figure 2 Fee Transaction Form

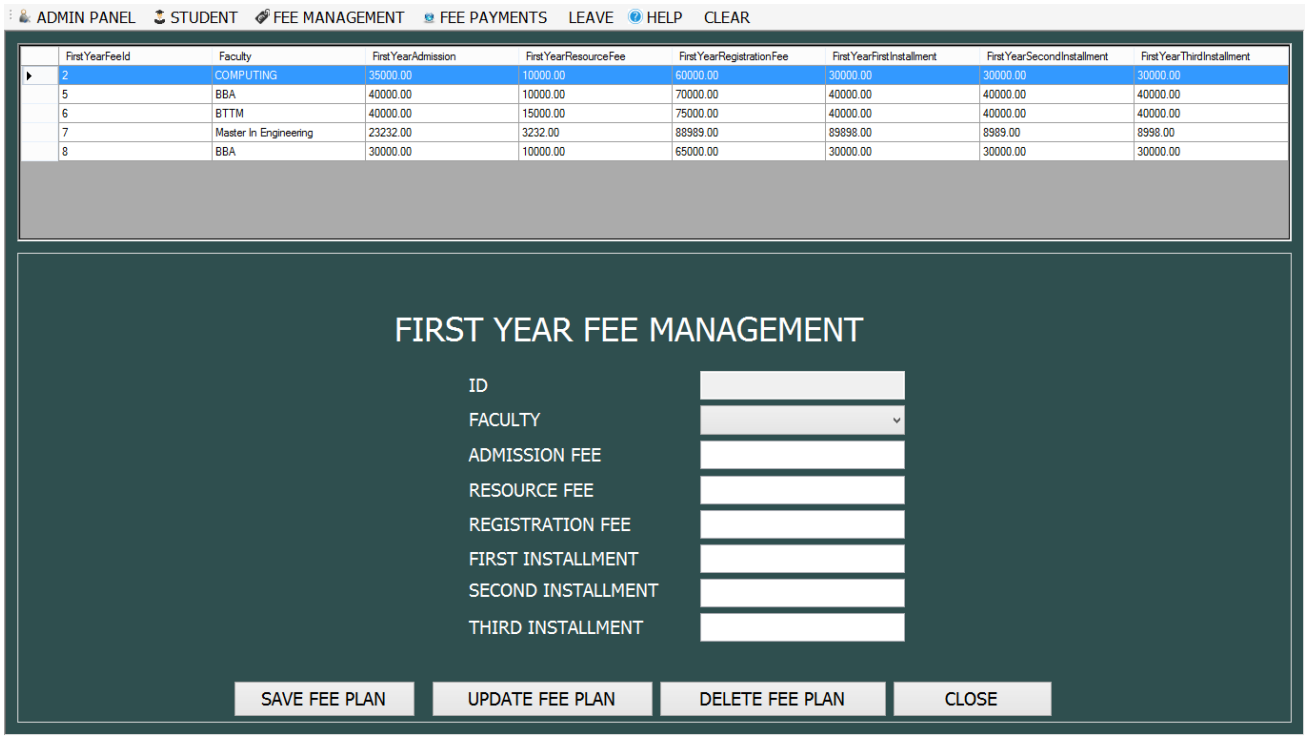


Figure 3 Fee Plan Form (First Year)

c. Inquiry Management

This application also allows maintaining the information of the student who visits college for different inquiry. Inquiry management form allows user to save student's name, their interested subject, contact details and inquiry date. This feature allows maintaining inquiry database easily and properly.

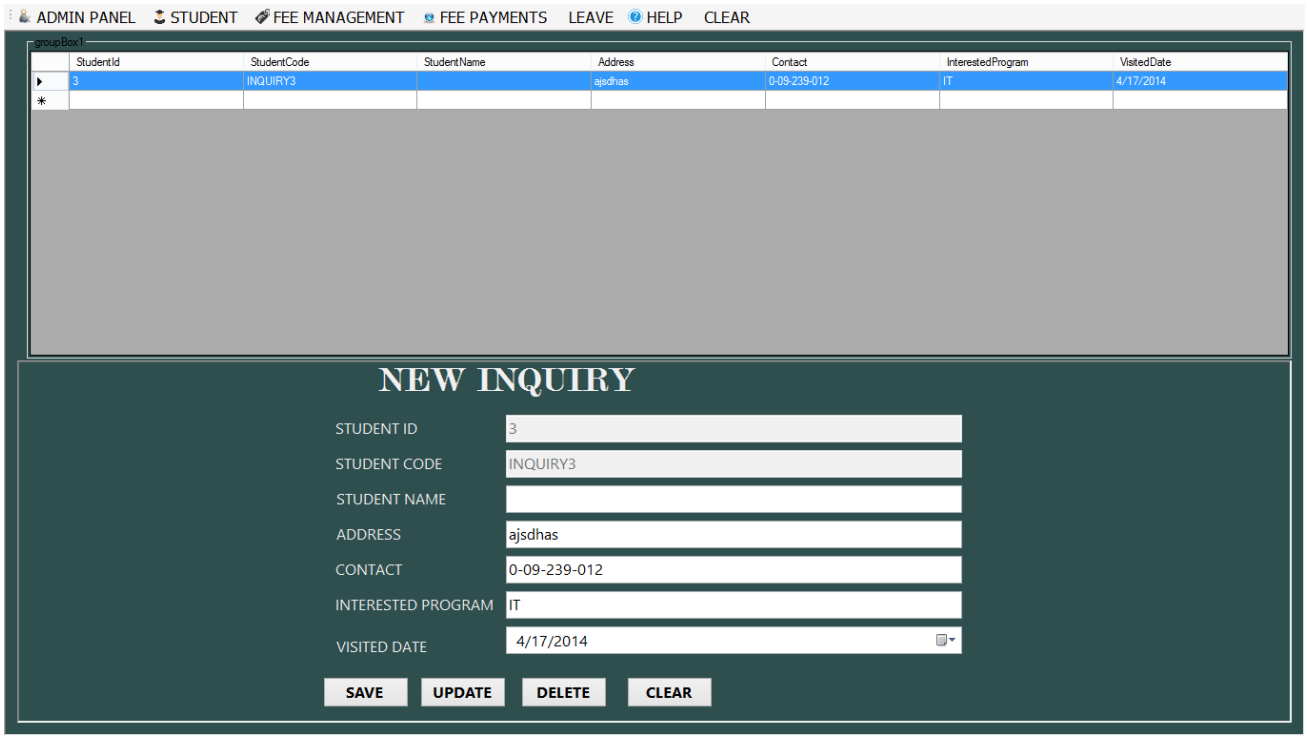


Figure 4 Inquiry Form

d. Easy Control and simple design:

The software surely has simple and easy controls. Menus are simple, changing forms, logging off or leaving the application is quite simple. Fonts in the form are large enough to understand easily, buttons are smooth and user can open desired form using single click on the menu.

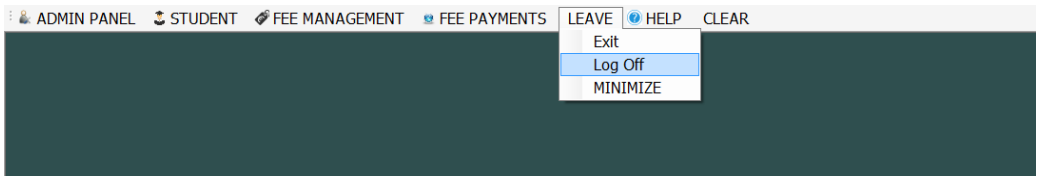


Figure 5 Simple Design and Easy Controls

e. Login Features:

ISMT MANAGEMENT system has login feature, which ask for user credentials in order to be able to use the system. The main form open after the login form varies depends on the role of the user. Admin role user has given full authority while other user has limited control on the system. As shown in fig7 and fig8, main menu after login using admin and user credentials respectively.

Figure 6 Login Form

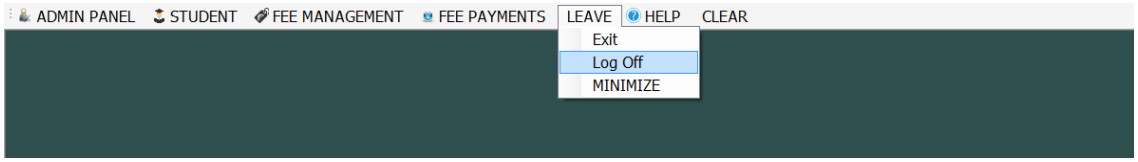


Figure 7 Login with (Admin)

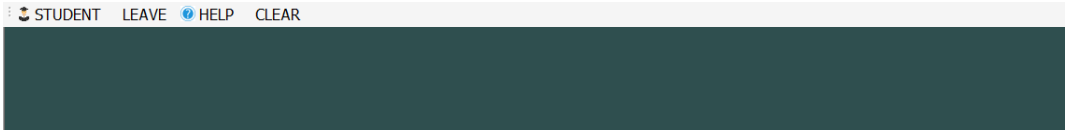


Figure 8 Login with (USER)

Weaknesses and Limitation

After critically reviewing this software, I found there are some weaknesses in the system. Basically I found there are 3 main weaknesses in the system. In this part of the document I have discussed about these three weaknesses.

Firstly, this system lacks the search ability to search students properly or search transaction based on date or student. With no search ability user have to manually search the student through grid view which is kind of time consuming and not user friendly.

Secondly, this system does not have ability to generate invoice and bill while after transaction process. User need to prepare paper bill manually due to the lack of feature in software to auto generate the bill.

Finally, I have found one of the main weaknesses of the system is that, it does not have any mechanism to trace if student have paid their bills off or not. There is no warning massage mechanism for the students who have not paid their bills.

Conclusion

This critical review has evaluated the ISMT MANAGEMENT SYSTEM by Atut Gorkhali (ISMT, First Semester IT). The design and features of software are very interesting and does a good job to address the issues that ISMT has been facing. However, software is weakened by the presence of weaknesses likes of inabilities to search, generate bill and give information regarding student's financial status. Though considering this software was developed with in single semester, it is really good software that can address large number of the issues that college is facing.

Task 6

Evaluate independent feedback on a developed object oriented programming solution and make necessary recommendations for improvement. You can consider the following for the purpose of independence feedback:

- **Interface Design**
- **Coding Architecture**
- **Security**
- **Error tracing mechanisms**

Interface Design:

ISMT MANAGEMENT SYSTEM is developed in C# Win Forms UI technology. This technology uses Windows GDI+ libraries to develop user interface. It has simple and easy interface and easy really easy for user to use.

After the interface design evaluation of ISMT MANAGEMENT SYSTEM the goods and not good of the program (interface design perspective) can be sum up with table below.

S.N.	The Good	The Drawbacks
1.	The system is designed in C#.net environment which uses windows library to create interface. This gives program simple and nice look.	It is developed using windows library hence it is not cross platform supported.
2.	Modification of design of software is easy.	Despite having simple and user friendly design, this interface is not suitable for other platforms like mobile or web.
3.	Uses grid view to display data which makes users easier to look data.	Loading large data into grid view can make application run slow and cause laggings.
4.	Program contains help icon from where user can find help and on screen assistance.	Lacks separated help menu for each forms, user need to go through single help file to find solution.

5.	Control Menus are simple; moving from one form to another form is just one click away. User will find this easier to use.	

Table 5Good and Drawbacks of the system (interface Design)

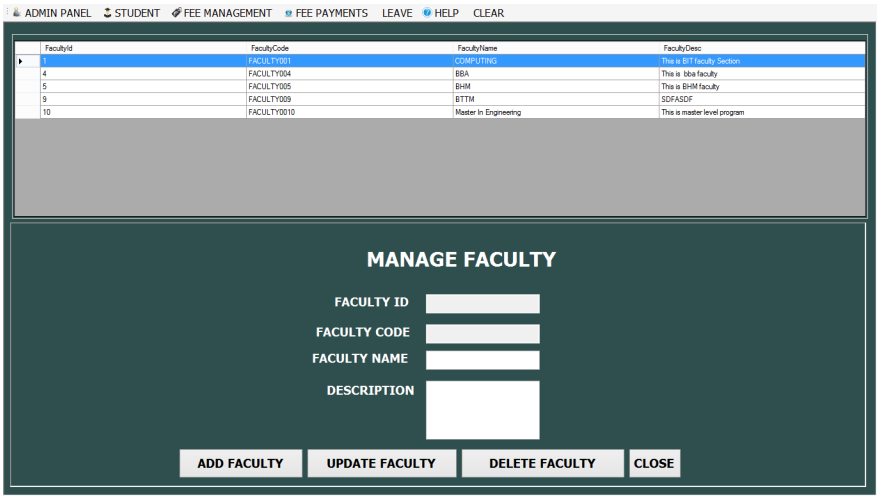


Figure 8Simple interface design

Feedback: The interface design of the software is simple and easy for user to use. But it still lacks rich user interface that new technology have brought in the market. Using WPF for developing this software can really be a good choice. UI virtualization enables to maintain performance of the system when loading large database. It offer rich control tools and make developing better looking software so easy. Fig.3 demonstrates a sample rich user interfaced apps developed using WPF.

Figure 9 Cool Interface Design using WPF

Coding Architecture:

This developed software is coded using three tier architecture. In this architecture, presentation, data connection and logic of the software are separated from each other. Flexibility, Maintainability, Reusability, Scalability, Reliability are some of the advantages of using 3 tiers architecture.

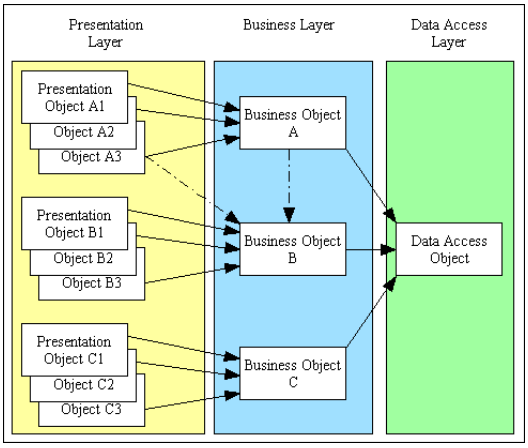


Figure 10 graphical representation of 3 tiers architecture

After evaluating the coding architecture of the system there are some good and some drawbacks in the system caused by using three tier architectures. Table 2 demonstrates some of the key drawbacks and good points existed in the system from coding architecture.

S.N.	Good thing about the system	Drawback in the system
1.	Migration to new application type is faster (to mobile, web, tab etc.)	It is more complex structure
2.	Application Code Management is Better	More difficult to set up and maintain
3.	It is possible and easier to make changes in presentation layer without affecting other two (business and Data Access Layer)	It is more difficult to build a 3-tier application
4.	This software can be developed by different team, each team developing different layer and at last combining them together.	Physical separation can drop the performance

Table 6 the good and Drawbacks in Coding Architecture used in system.

Feedback:

Using three tiers architecture for the development of the system is good thing about this system. Generally 3 tier architecture is really advanced and used in big projects. This system can be done in single tire or double tier coding architecture but that would be like going backward direction. 4-tier architecture allows a limitless number of programs to run all together, direct data to one another, use altered protocols to communicate, and act together alongside. This permits for a greatly further powerful software. Generally 4 layer of 4 tiers architecture are:

1. Business Object [BO]
2. Business Access Layer [BAL]
3. Data Access Layer [DAL]
4. UI (4-Tier) folder [UI]

But this application is for a college hence developing powerful application using 4-tier architecture is not necessary.

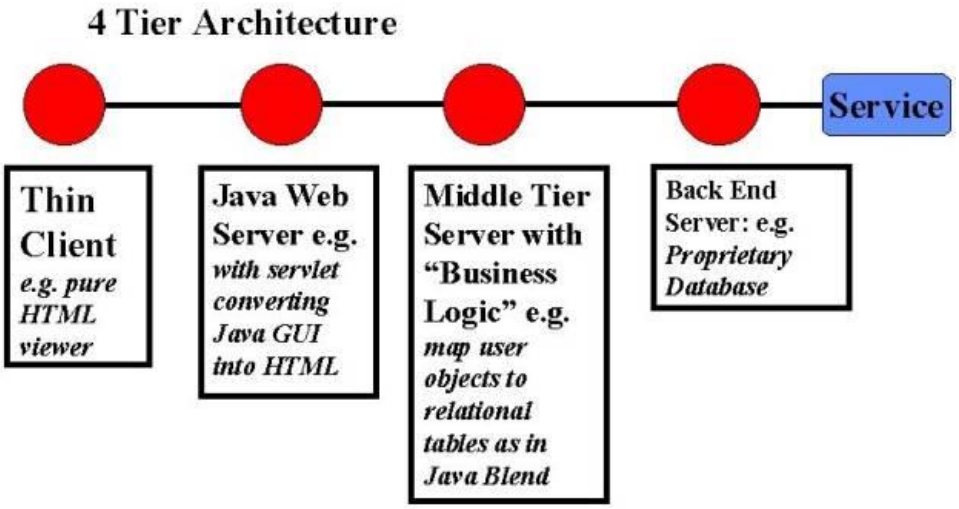


Figure 11 4-Tiers Architecture in Web Application

Security

Since the program has developed in three tiers architecture, data is secure enough until a hacker does not forcefully tries to hack the system. In one tier, if application fails chances of data loss is greatly higher than in three tier application where database is not directly accessed. Since the client doesn't have direct access to the database system data is more secure. Moreover, this system has login feature hence general users don't have access to the administrative controls.

After evaluating the security aspect of the system, the good and drawback of the system can be demonstrated in table below.

S.N.	The goods	Drawbacks
1.	Login features hence other users can't access the admin panel.	No mechanism to prevent SQL injection.
2.	Three tiers architecture enables data protection from client end. Since no direct access to database.	No data backup and restore mechanism
3.	Asks for confirmation before deleting data.	No data recovery mechanism if data gets deleted accidentally.

Table 7 the good and Drawbacks of system (security aspect)

Feedback:

Besides using three tiers architecture, program basically does not have any mechanism to prevent data theft. SQL injection is a code injection technique, used to attack data driven applications. SQL statement can be passed to the program to hack into the system easily. Once attacker successfully enters into database he/she can do anything with database from drop to insert update and delete. This system would be far more better is developer implement SQL injection mechanism.

- Use strongly typed parameterized query
- input validation technique in which user input is authenticated against a set of defined rules for length, type, and syntax and also against business rules

Error Tracing Mechanism

Error tracing mechanism is used for tracing error while developing the program. Evaluation of the system shows good use of error tracing mechanism in the system. Try-catch Exceptional Handling is used in ISMT MANAGEMENT system to handle and trace the errors. In try catch Try tries to execute an event or methods or codes if it fails catch provides alternative flow and throws error massage and prevents the system from crashing.

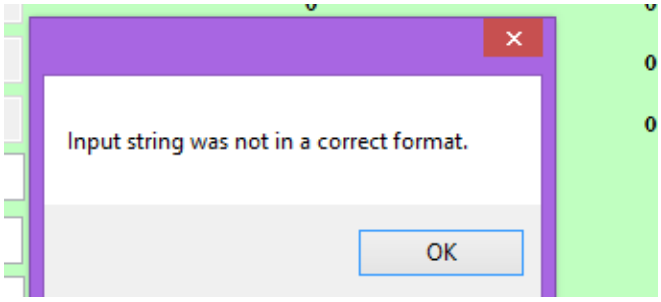


Figure 12 Error Message Appeared after use of try catches method.

```
try
            {
                int result = 0;
                string Batch = "";
                if (Mode == 1)

                    Batch = "insert into Batch values('" + BatchName + "','" + BatchDesc +
"')";
                else if (Mode == 2)
                    Batch = "Update Batch set BatchName='" + BatchName + "',BatchDesc='" +
BatchDesc + "' where BatchId='" + BatchId + "'";
                else if (Mode == 3)
                    Batch = "Delete from Batch where BatchId='" + BatchId + "'";
                SqlCommand cmd = new SqlCommand(Batch, conn);
                conn.Open();
                result = cmd.ExecuteNonQuery();
                conn.Close();
                return result;
            }
            catch (Exception ex)
            {

                throw ex;
            }
```

Table 8 Try Catch Patten in this application

Here are some good thing and some drawbacks in the system due to use of try catch .

SN.	The Good	The Drawbacks
1.	This helps to find the error in code.	Performance of the application may hurt because to compiler need to go through try catch even if there is no exceptional.
2.	Prevents system from crashing	

Feedback:

Use of Try Catch in ISMT MANAGEMENT system have done good job in handling the exceptional errors. This prevents system form crashing down in case of exceptional error occurrence.

Task 7 [P4.4]

Create onscreen help to assist the users of the computer program.

Help Menu

To provide the onscreen help to the user, I have created a help menu by help of which user can read on screen help just by clicking on the help icon.

Figure 13 Help Icon

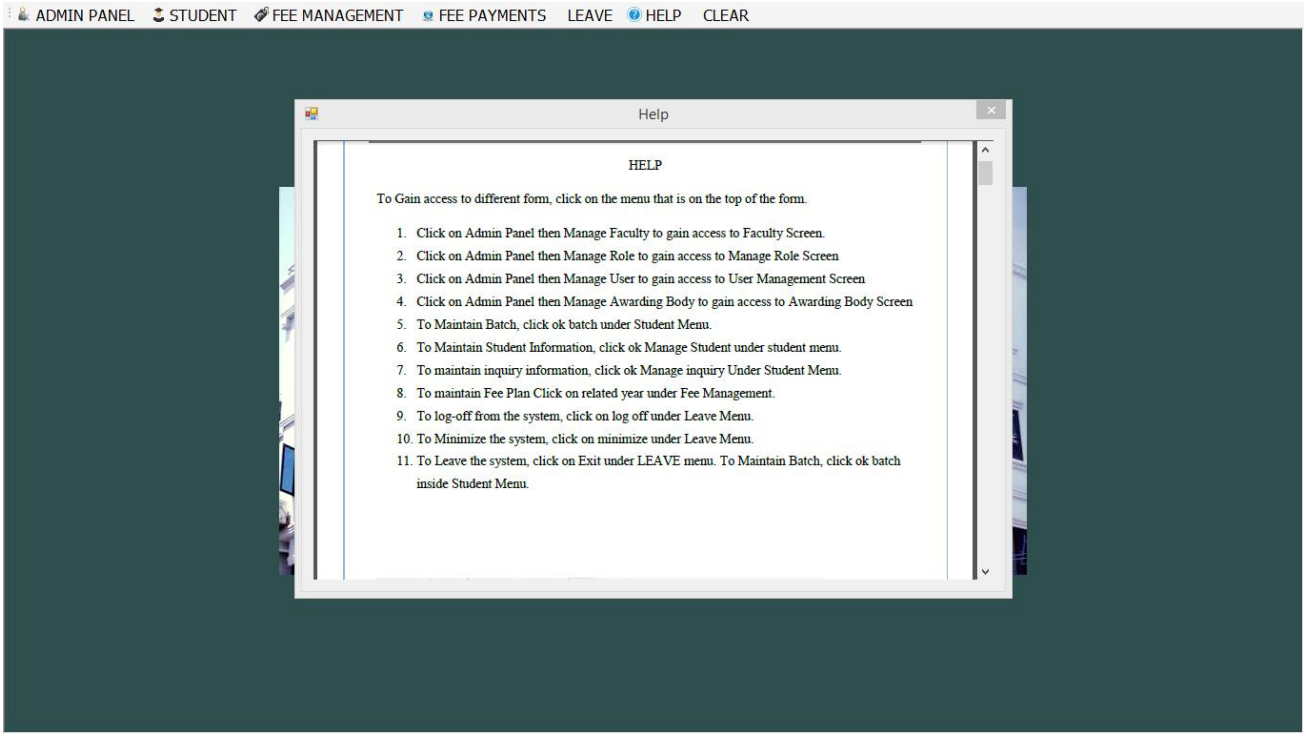


Figure 14 Onscreen Help

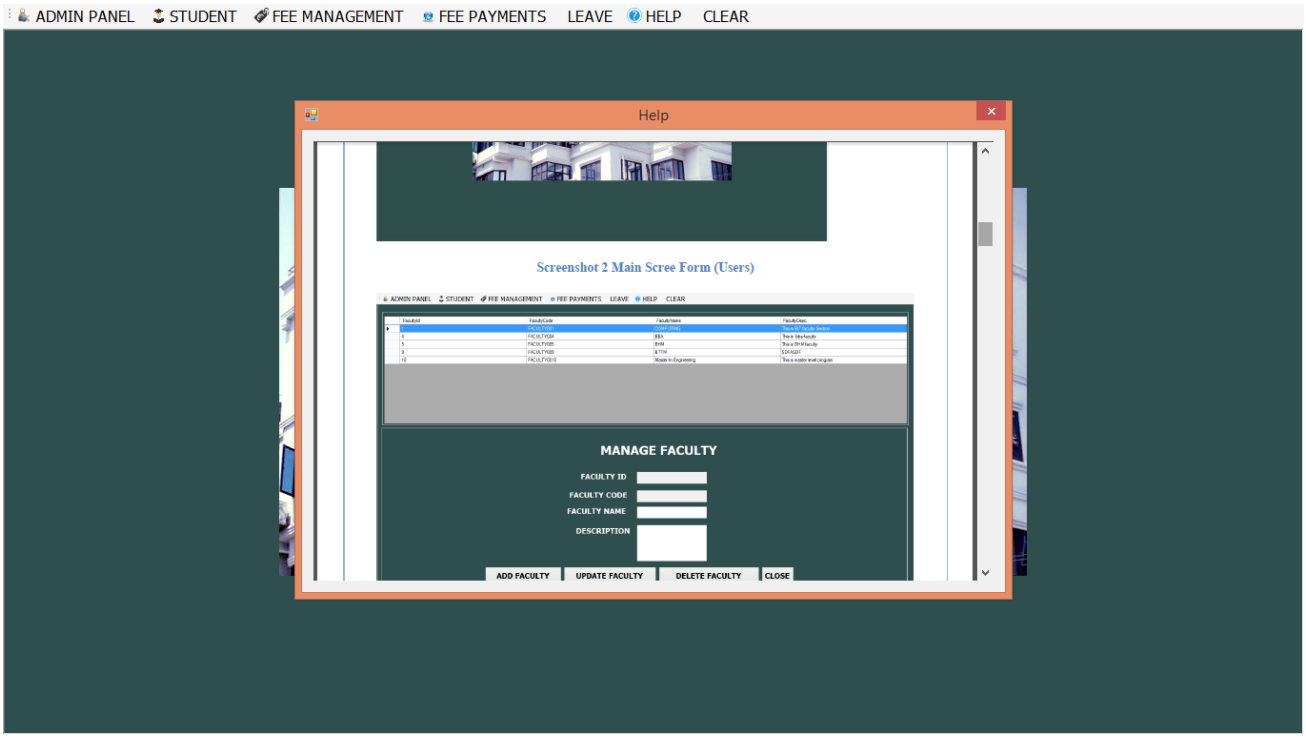


Figure 15 Help Menu(Screenshot 2)

Code for developing Help Menu:

```
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 ISMT_FEE_MANAGEMENT_BY_SECTION_D
{
    public partial class HelpForm : Form
    {
        public HelpForm()
        {
            InitializeComponent();
        }

        private void HelpForm_Load(object sender, EventArgs e)
        {
            axAcroPDF1.src = @"c:abc.pdf";
        }
    }
}
```

This help screen will assist user of the computer.

Task 8

Create Documentation for the support and maintenance of the computer. You need to create technical and non-technical documentation.

Technical Documentation:

The purpose of this document is to present the fundamental steps that will guide the Software Maintenance and the User Support tasks for the system. This software is developed form ISMT College to maintain their student and Finance information. They were facing problems like unmanaged database management due to traditional database management system. There were few important task needed to be addressed through this software. Those key tasks are:

- Student Information Management
- Fee Management
- Inquiry Management

This software; ISMT management system is able to address these areas. Separated Forms like Student Form management form. Fee management Form and Inquiry System is pretty much capable of performing these tasks which were the main reasons to develop this application.

Maintenance Guide:

To maintain this system technician may follow these key guides in order to perform software maintenances properly.

- Read the User manual to help users to use the Software.
- Since software does not have any inbuilt database back up system, perform database backup form SQL Server.
- In case of data loss due to poor handling of the software, restore the backed up database using SQL server.
- Create a desktop icon to help user opening the program directly.
- Copy abc.pdf from resources folder and save it in C: to make user able to utilize help feature of the program.

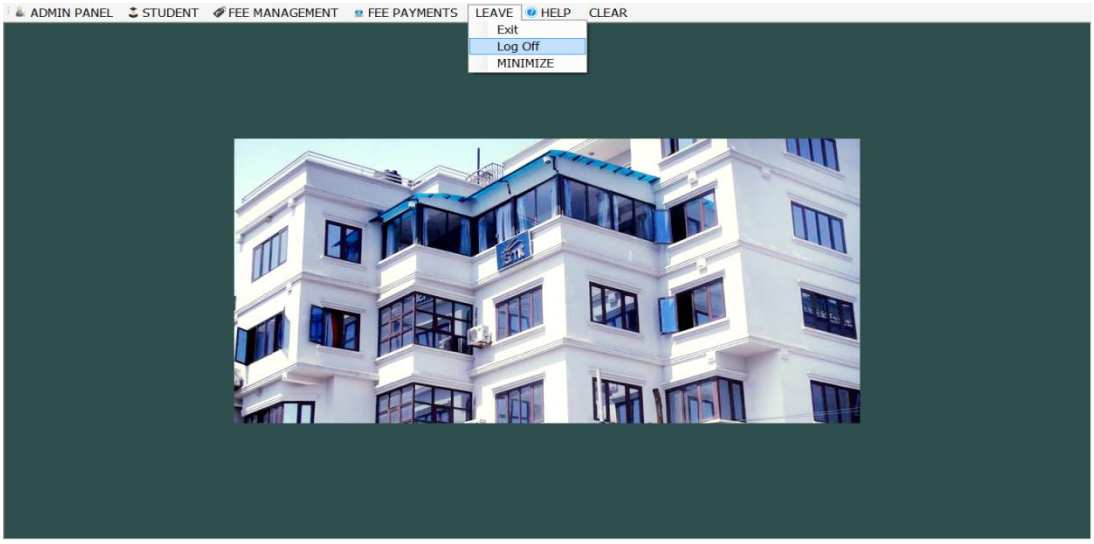
Gantt chart

The Gantt chart of the development of this application is shown below in separated page.

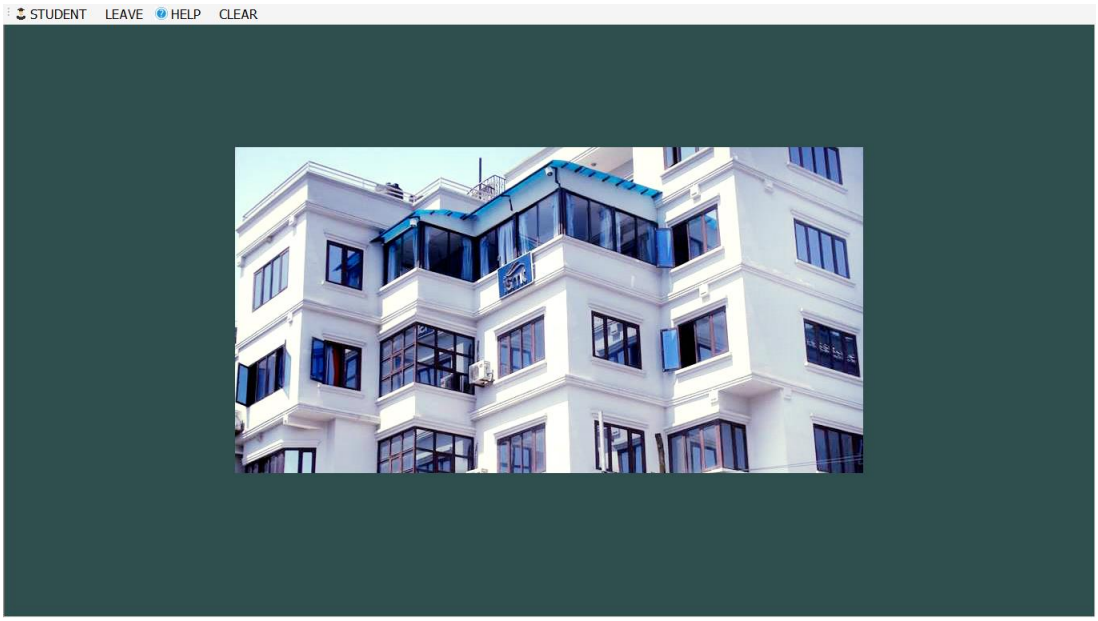
USER MANUAL

To Gain access to different form, click on the menu that is on the top of the form.

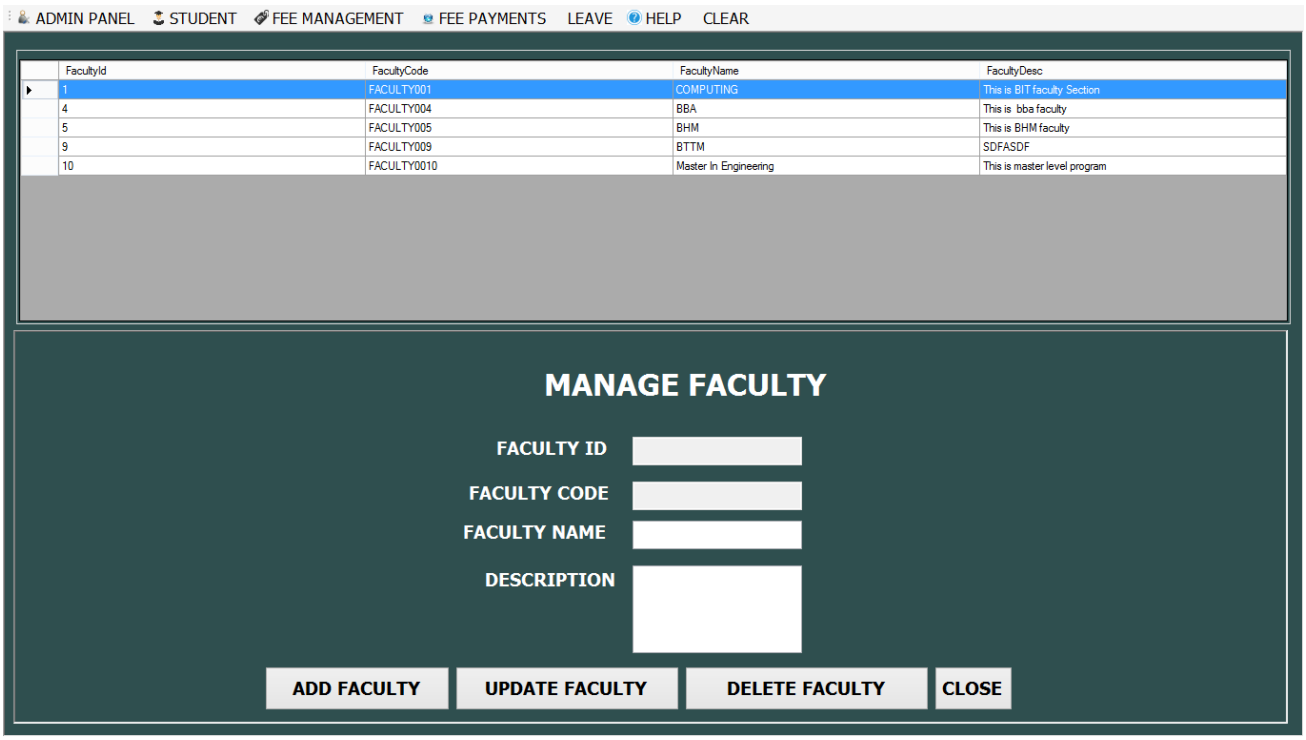
1. Click on Admin Panel Menu then Manage Faculty Menu to gain access to Faculty Screen.
2. Click on Admin Panel Menu then Manage Role Menu to gain access to Manage Role Screen
3. Click on Admin Panel Menu then Manage User Menu to gain access to User Management Screen
4. Click on Admin Panel Menu then Manage Awarding Body Menu to gain access to Awarding Body Screen
5. To Maintain Batch, click ok batch Menu under Student Menu.
6. To Maintain Student Information, click ok Manage Student Menu under student menu.
7. To maintain inquiry information, click ok Manage inquiry Menu Under Student Menu.
8. To maintain Fee Plan, Click on related year Menu under Fee Management.
9. To log-off from the system, click on log off under Leave Menu.
10. To Minimize the system, click on minimize under Leave Menu.
11. To LEAVE the system, click on Exit under LEAVE menu. To Maintain Batch, click ok batch inside Student Menu.



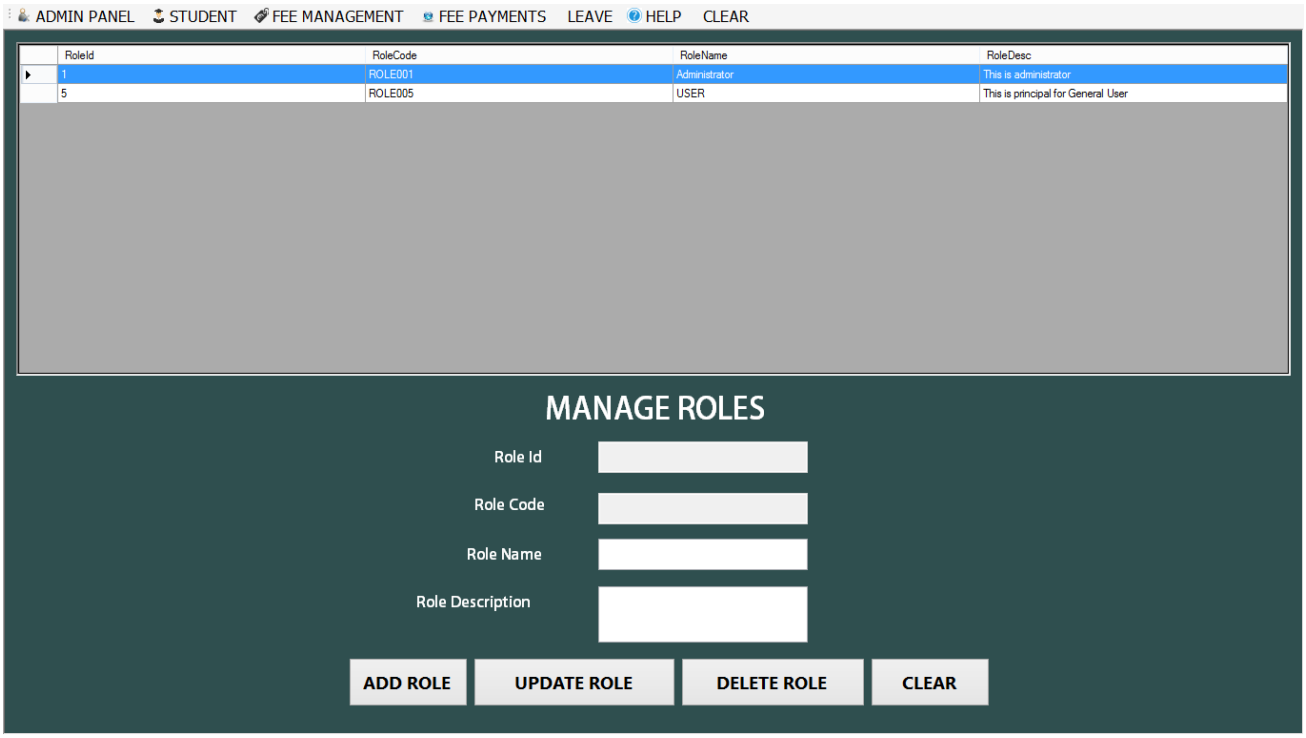
Screenshot 15 Main Screen Form (Admin)



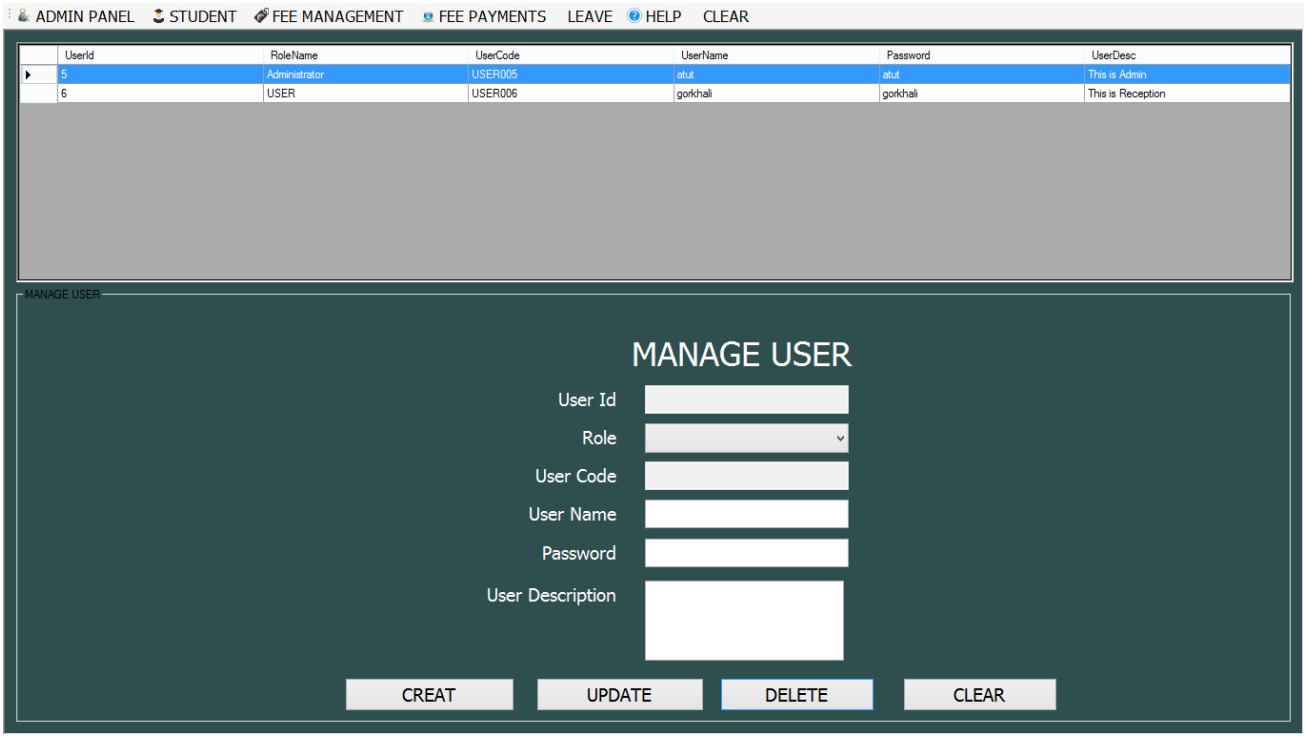
Screenshot 16 Main Scree Form (Users)



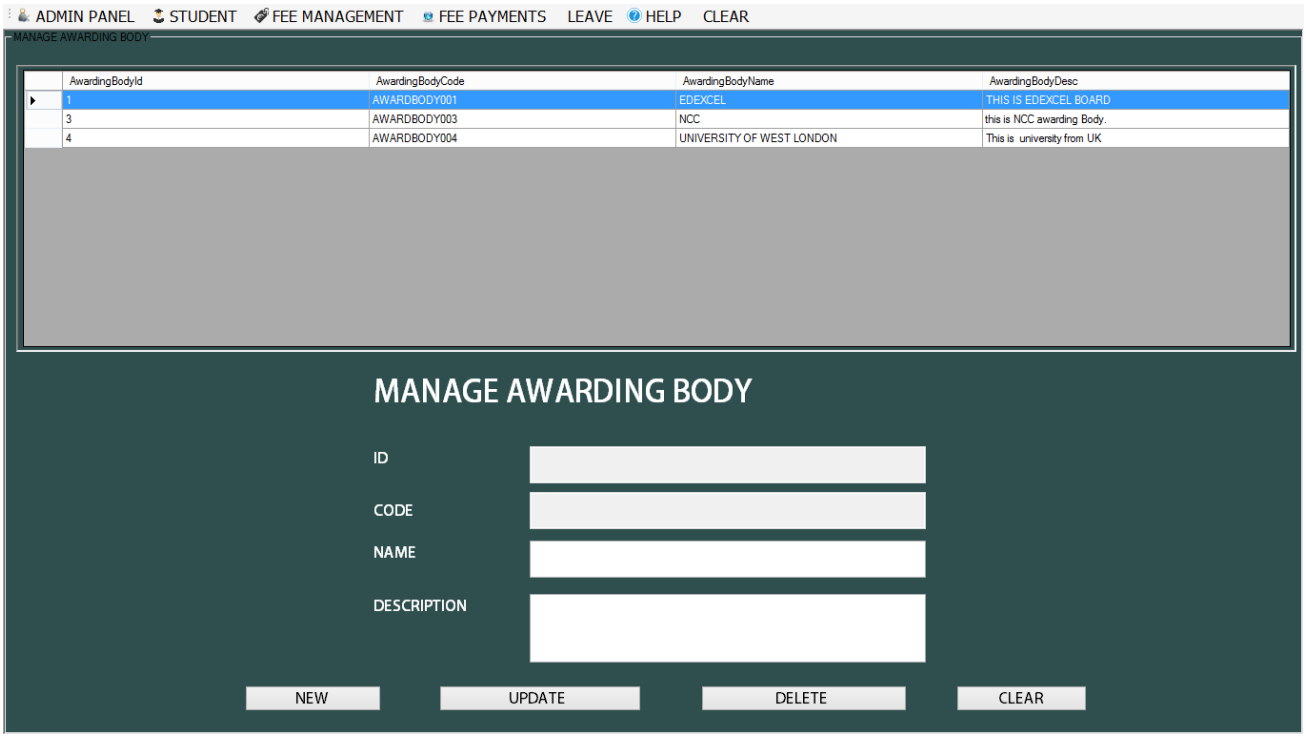
Screenshot 17 Faculty Form



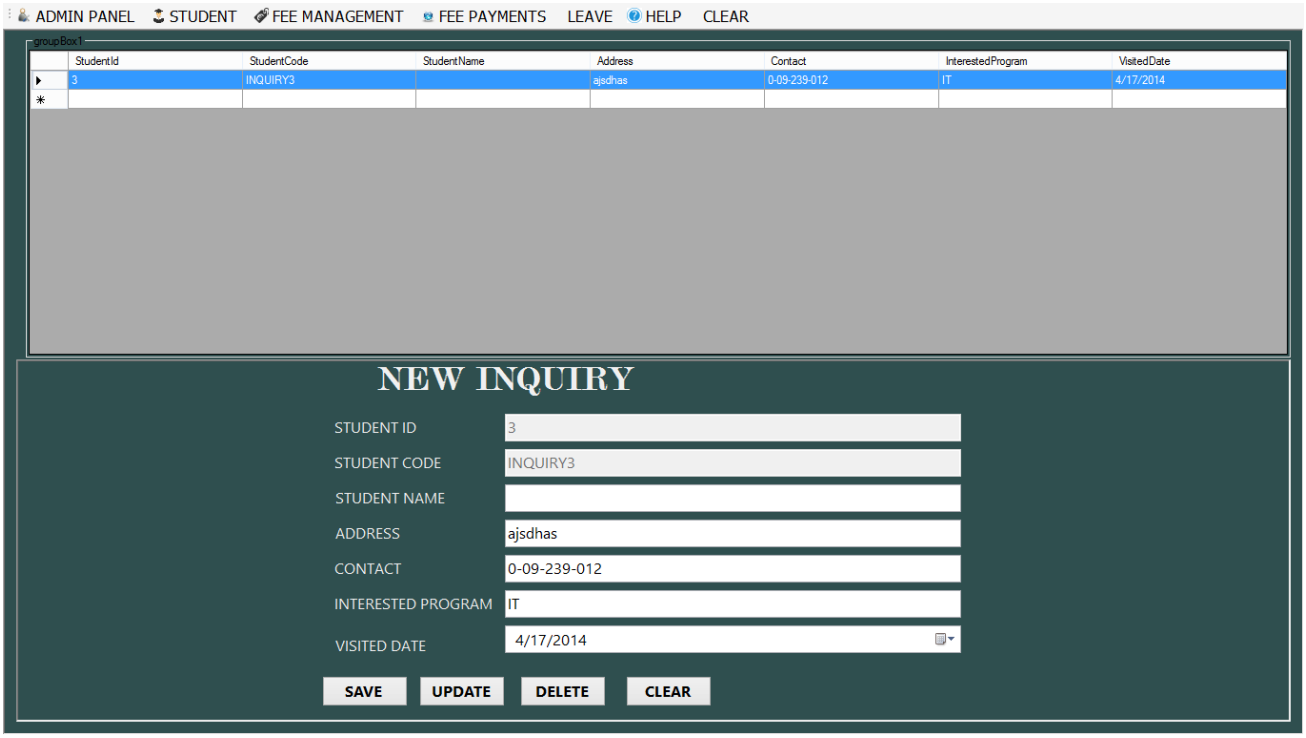
Screenshot 18 Manage Role Form



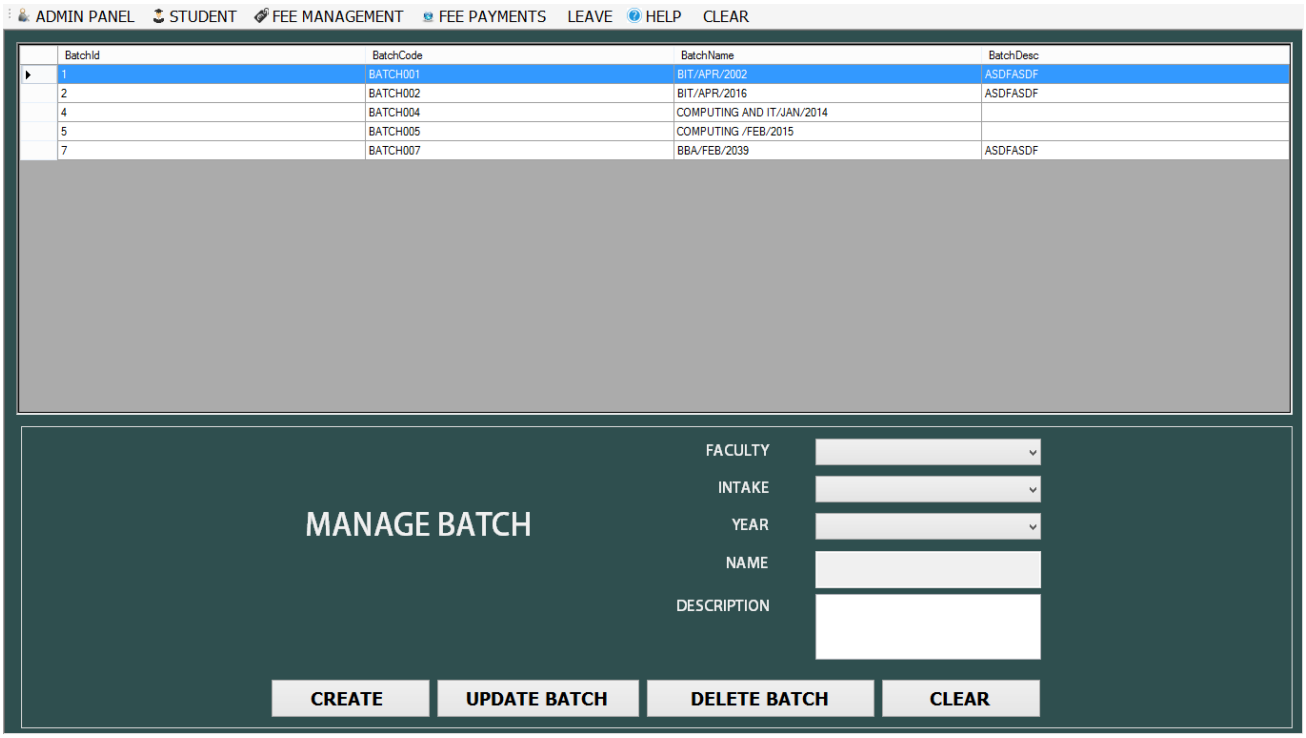
Screenshot 19 Manage User Form



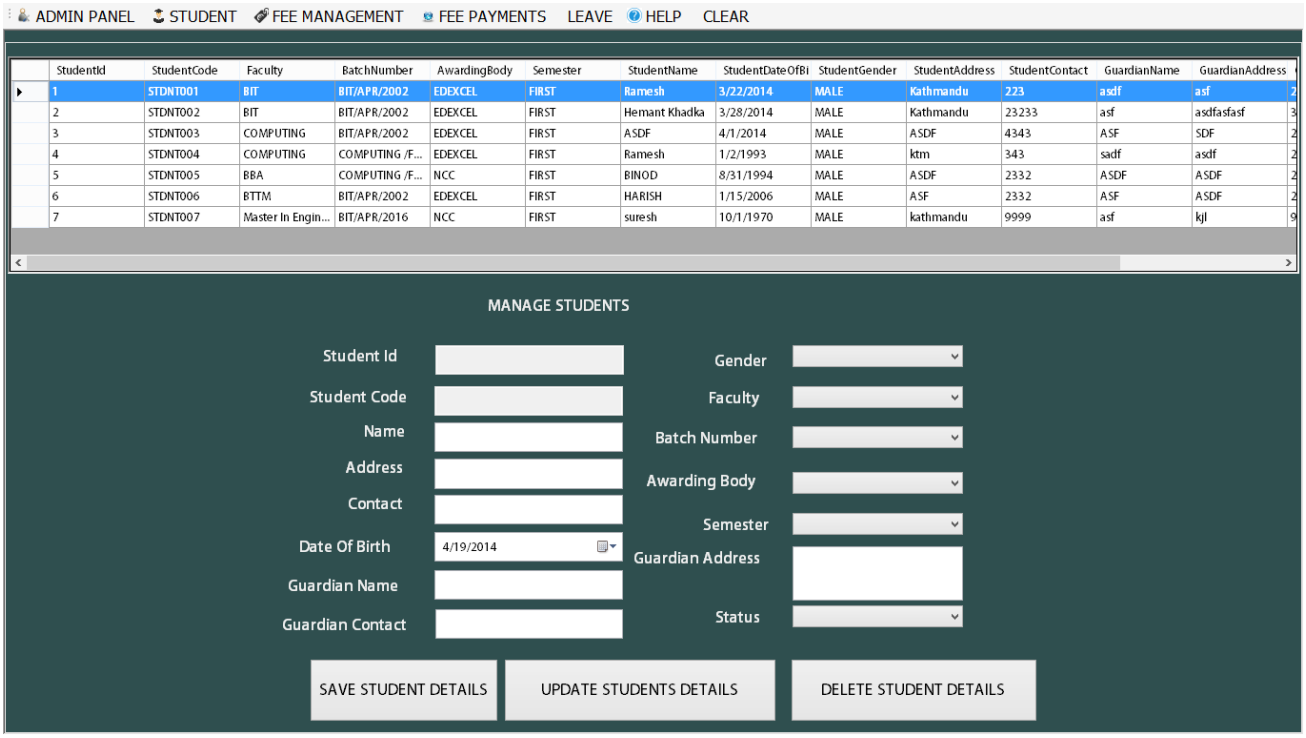
Screenshot 20 Manage Awarding Body Form



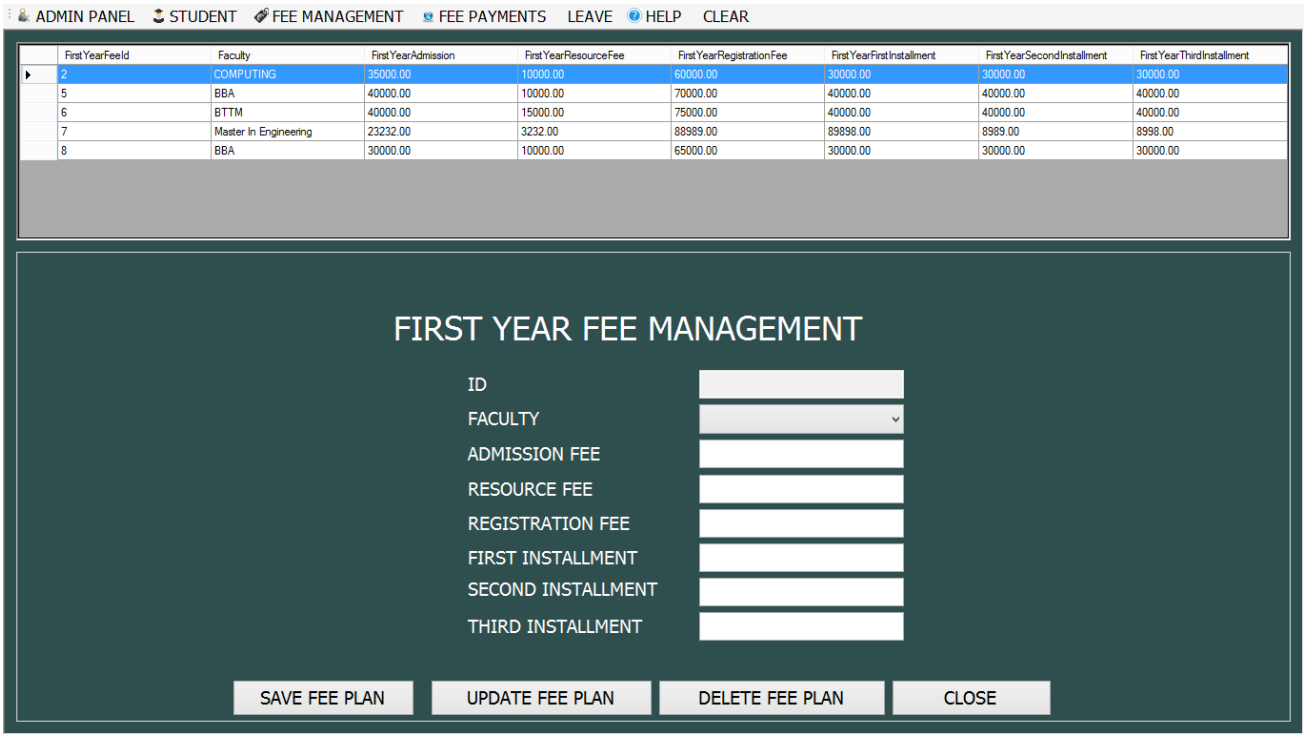
Screenshot 21 Inquiry Form



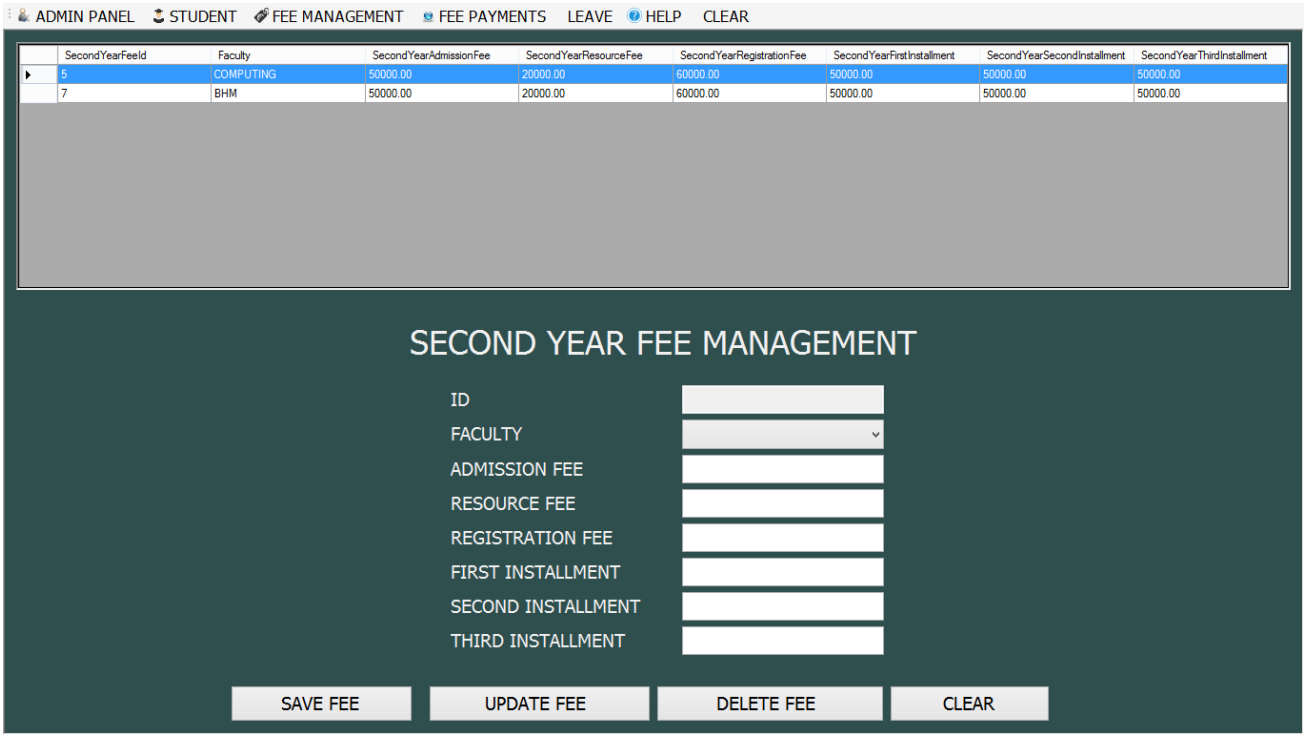
Screenshot 22 Manage Batch Form



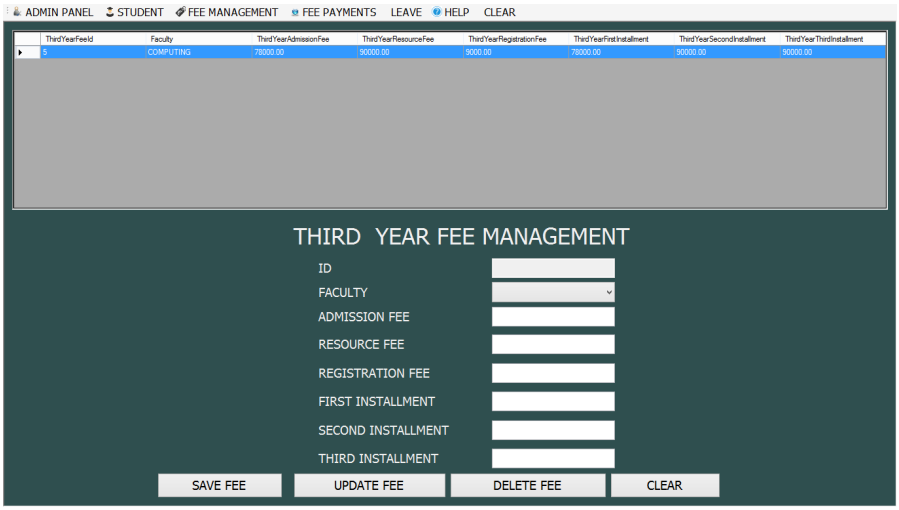
Screenshot 23 Manage Student



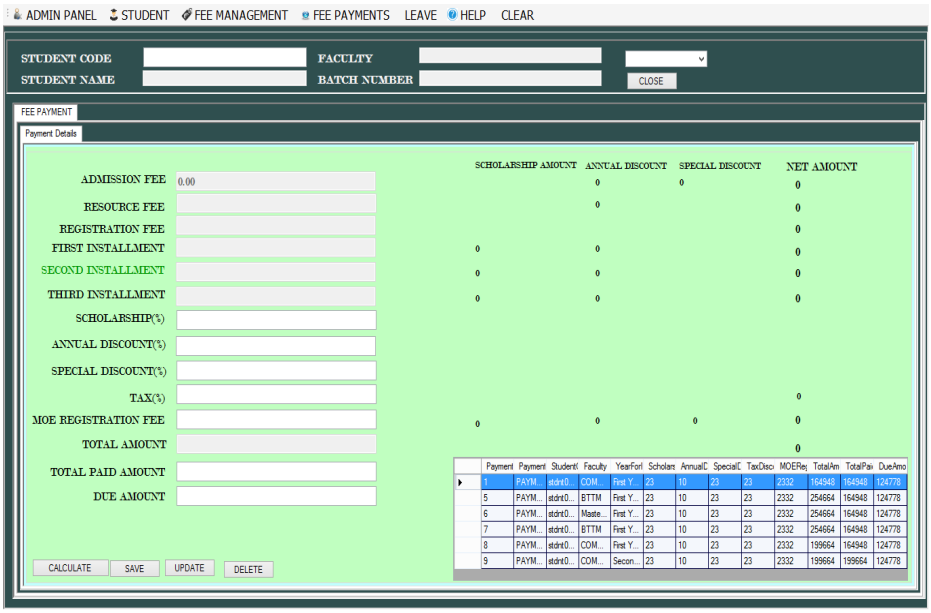
Screenshot 24 First Year Fee Management Form



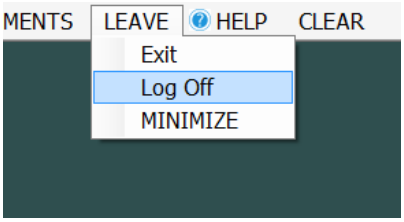
Screenshot 25 Second Year Fee Management Form



Screenshot 26 Third Year Fee Management Form



Screenshot 27 Fee Payment (Transaction) Form



Screenshot 28 Leaving the System

References

- 4inv (n.d.) *Four principles of object oriented programming:* Available: http://www.4inv.com/delphi/oo.htm Accessed [3/22/2014]
- Balagurusamy (2006) Object Oriented *Programming Using C++* New Delhi: Tata McGraw-Hill Publication
- Blackwasp (2007) *Creating a Simple Class in C#* Available: http://www.blackwasp.co.uk/CSharpSimpleClass.aspx Accessed [3/22/2014]
- ERPBASIC (n.d.) *Inheritance Advantages and Disadvantages* Available: http://erpbasic.blogspot.com/2012/01/inheritance-advantages-and.html Accessed [3/22/2014]
- Hagos, D. (2008) *Advantages and disadvantages of 3-Tier Architecture in Web Development* [online] Available at: **http://des-megan.blogspot.com/2008/11/advantages-and-disadvantages-of-3-tier.html** Accessed [03/21/2014]
- INTROPROGRAMMING (n.d.) *Object-Oriented Programming Principles (OOP)* Available: http://www.introprogramming.info/english-intro-csharp-book/read-online/chapter-20-object-oriented-programming-principles/#_Toc362296570 Accessed [3/22/2014]
- JANA, D. (2005) C++ AND OBJECT-*ORIENTED PROGRAMMING PARADIGM* New Delhi: Prentile-Hall
- Jones, R.S (1991) *C Programmer's Companion: ANSI C Library Functions.* Beverly Road: Silicon Press
- Rouse, M. (2008) *object-oriented programming (OOP)* [Online] Available: http://searchsoa.techtarget.com/definition/object-oriented-programming Accessed [3/22/2014]
- Singh, A., Mrs. Kaur, P.P (2008) *Object Oriented Programming Using C++* Pune: Technical Publication
- TUTORIALSPOINTS (n.d.) *Data Abstraction in C++* Available: http://www.tutorialspoint.com/cplusplus/cpp_data_abstraction.htm Accessed [3/22/2014]
- Virginia Tech (n.d.) *Object-Oriented Programming and Software Engineering* [online] Available at: http://people.cs.vt.edu/kafura/cs2704/oop.swe.html Accessed [03/21/2014]
- WOWWIKI (n.d.) Object-oriented programming Available at: **http://www.wowwiki.com/Object-oriented_programming** Accessed [03/21/2014]