Mini Project Report

KONSEQUENCES

CS633P – Design Patterns

SUBMITTED BY

ANDREW GABRIEL MANI 1960308

DISHTI KUNDRA 1960417

JASMINE BHUYAN 1960422

Department of Computer Science and Engineering

School of Engineering and Technology,

CHRIST(Deemed to be University),
Kumbalgodu, Bangalore - 74.

Mini Project Report

KONSEQUENCES

CS633P – Design Patterns

SUBMITTED BY

ANDREW GABRIEL MANI 1960308

DISHTI KUNDRA 1960417

JASMINE BHUYAN 1960422

Faculty-In-Charge Examiner Head of Department

Department of Computer Science and Engineering

School of Engineering and Technology,

CHRIST(Deemed to be University),
Kumbalgodu, Bangalore - 74.
April 2022.

ABSTRACT

The main objective of the project is to provide the examination result to the students and faculties in a simple way. This is done by the multi result option. This multi-result offers students the benefit to view their result, and offers faculties to view any student's individual result or class performance result, i.e, summary of results of all students in that class. The system is intended to provide the facilitation students and faculties (both come under different arena with different access privileges) result access. The whole result analyzer will be under the control of the administrator and the admin as the full privileges to read, write and execute the result, and type of result accessible options are approved through login's ID number (Reg No. or employee ID).

TABLE OF CONTENT

INTRODUCTION

KONSEQUENCES is a result management system which is more than just viewing student result. It can be accessed by both students and faculties. Main feature of KONSQUENCES is its multi result access privilege to students and faculties based on who is accessing the application. Authorization of faculty or students is done through ID numbers. Generic ID numbers are used to authorize, which differentiates between Student Register number or Employee ID number.

An individual report card of each student has to be displayed and printed at a keystroke according to any selected format. An important aid for teachers and students to judge their performance is provided through the class result report option. Merit list printing by totals for a class by individual subject marks for a class. Student performance in a particular subject or all the subjects must be expressed. Performance of subjects of various classes can be easily compared.

The application will manage the information about various students , the marks obtained by the various students in various subjects in different semesters. The application will greatly simplify and speed up the result preparation and management process as it is made accessible to two-tier userbase: Faculties and Students.

This application is prepared with C# and software design practices of using SOLID principles to define the classes, attributes relation, basically the whole design of the application.

2. DESIGN & IMPLEMENTATION

2.1 UML Diagram

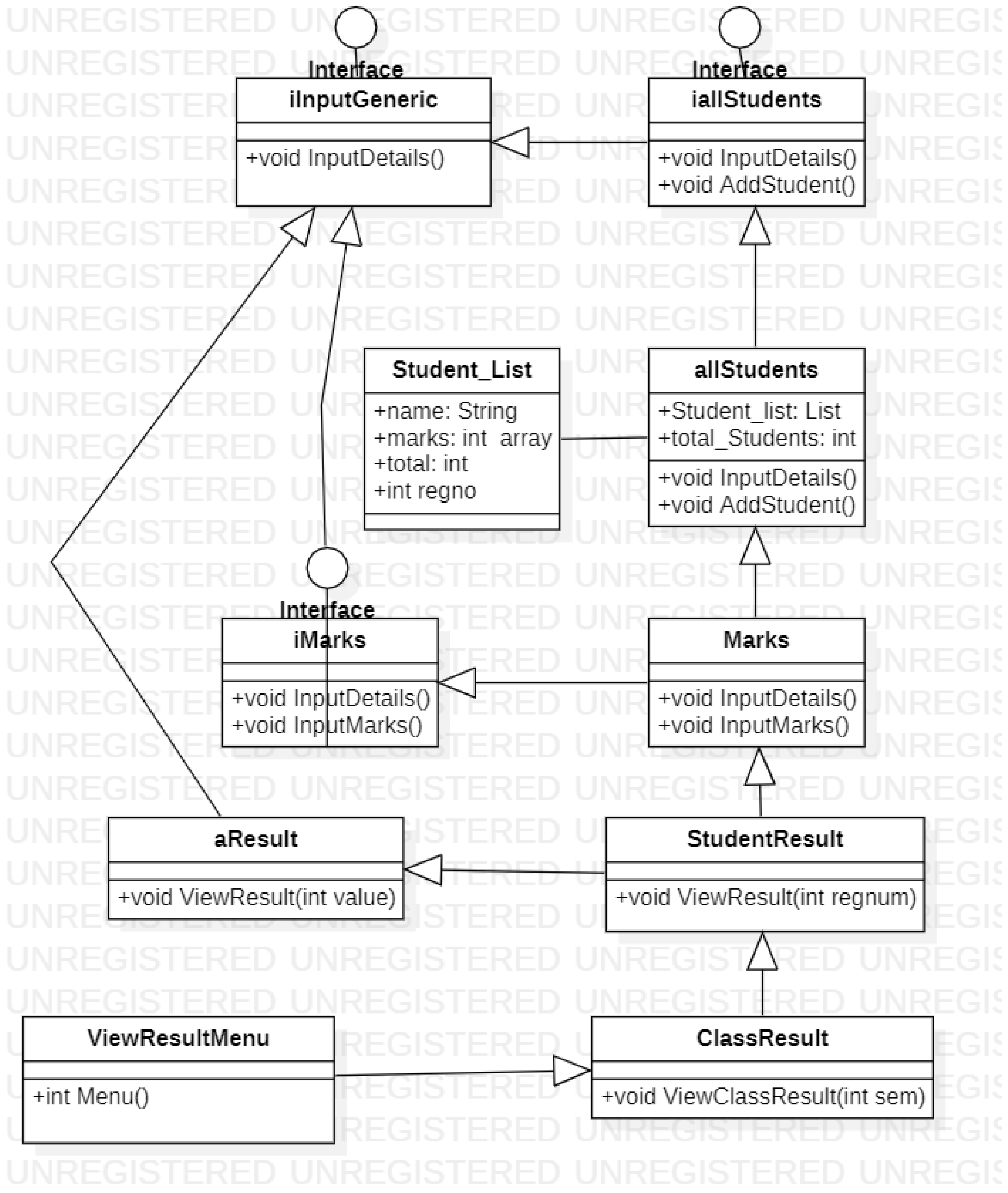


Fig 1

The above figure, Fig1, implements SOLID design principles to build KONSEQUENCES application.

Firstly, all classes are divided into single responsibilities

//R1 : allStudent : maintaining student database with respect to adding/registering students.

//R2: Marks : Responsible for maintaining student marks.

//R3: StudentResult: Responsible for accessing student report card

//R4: ClassResult: Responsible for accessing class performance report

//R5: ViewResultMenu: Responsible for presenting result menu

Secondly, through Open Close Principle, ViewResultClass is sealed, making the implementation open for extension but close for modification.

Third, through the Liskov Substitution Principle, contract and variant rules are applied, mainly to present options of different result accesses based on student or faculty through ID type. ID acts as our data invariant. If ID is less than 500, it is a student accessing application hence they can view only individual student result card. Above 500 to 800 there are faculties. Faculties can access both student report cards and overall class performance or class result cards. But what if a new student joins mid session. Then we'll have to raise an exception where the ID number above 500 is not faculty but a student. This exception is presented through case 3 where ID number 883 is a student.

Next, we apply Interface Segregation Principle, to segregate multi functionality interfaces such that each application or responsibility has its own interface, ruled by a binding generic interface, iInputGeneric.Furthermore, CRUD is applied as follows:

C - Create : allStudents

R - Read: allStudent (Read student details), Marks (Reads student marks)

Update - ViewResultMenu (type of result reports accessible is updates based on who is accessing application; accessing privileges are differentiated using their ID numbers)

Display - StudentResult, ClassResult (both are used to display results - individual student result, class/overall performance report, respectively)

2.2 Code Implementation

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;

namespace Konsequences
{
        public interface iInputGeneric
        {
        void InputDetails();
        }

        public abstract class aResult : Marks, iInputGeneric
        {
        public abstract void ViewResult(int value);
        }
        public interface iallStudents : iInputGeneric
   {
         void AddStudents();
         void InputDetails();
   }
        public interface iMarks : iInputGeneric
   {
        void InputDetails();
         void InputMarks();
   }
        public class Student
        {
        public string name;
        public int[] marks = new int[2];
        public int total;
        public int regno;
        }

        public class allStudents : Student, iallStudents
        {
        public List<Student> Student_list = new List<Student>();
        public int total_students;

        public void AddStudents()
        {
        Console.Write("Enter the number of students: ");
        int numStudents = Convert.ToInt32(Console.ReadLine());
        for (int i = 1; i <= numStudents; i++)
```

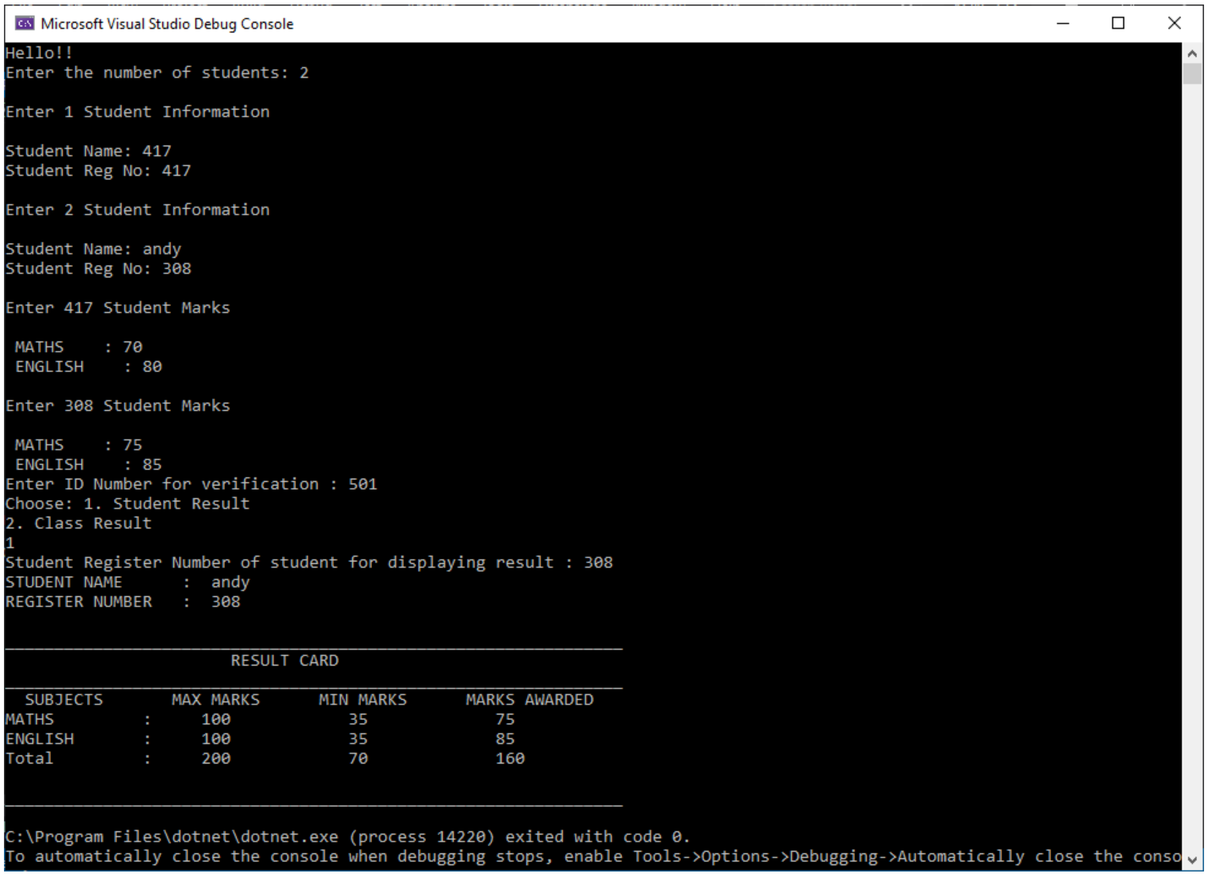
```
{
        Console.WriteLine("\nEnter " + i.ToString() + " Student Information\n");
        InputDetails();
}
}

public void InputDetails()
{

Student stu = new Student();

Console.Write("Student Name: ");
stu.name = Console.ReadLine();
Console.Write("Student Reg No: ");
stu.regno = Convert.ToInt32(Console.ReadLine());
Student_list.Add(stu);
total_students = Student_list.Count;
}
}

//R2
public class Marks : allStudents, iMarks
{
public void InputDetails()
{
string[] subjects = { "MATHS", "ENGLISH" };
int[] marks = new int[2];
for (int i = 0; i < total_students; i++)
{
        Console.WriteLine("\nEnter " + Student_list[i].regno + " Student Marks\n");

        for (int j = 0; j < 2; j++)
        {
        Console.Write(" " + subjects[j] + "     : ");
        marks[j] = Convert.ToInt32(Console.ReadLine());
        }
        Student_list[i].marks = marks;
        for (int j = 0; j < 2; j++)
        {
        Student_list[i].total += marks[j];
        }
}
}
public void InputMarks()
{
        InputDetails();

        }
        }
        class StudentResult : aResult
        {
        public override void ViewResult(int regnum)
        {
        int c = 0;
        for (int i = 0; i < total_students; i++)
        {
                if ((Student_list[i].regno) == regnum)
                {
                Console.WriteLine("STUDENT NAME        :  {0, -19}", Student_list[i].name);
                Console.WriteLine("REGISTER NUMBER   :  {0, -7}", Student_list[i].regno);
                Console.WriteLine();

Console.WriteLine("__________________________________________________________________
__");
                Console.WriteLine("                    RESULT CARD                         ");

Console.WriteLine("__________________________________________________________________
__");
                Console.WriteLine("  SUBJECTS     MAX MARKS         MIN MARKS
MARKS AWARDED   ");
                Console.WriteLine("MATHS          :         100             35                {0,
-7}", Student_list[i].marks[0]);
                Console.WriteLine("ENGLISH        :         100             35                {0,
-7}", Student_list[i].marks[1]);
                Console.WriteLine("Total      :         200             70              {0, -7}",
Student_list[i].total);
                Console.WriteLine();

Console.WriteLine("__________________________________________________________________
__");
                c = c + 1;
                break;
                }
        }
        if (c == 0)
        {
                Console.WriteLine("WRONG INPUT. PLEASE TRY AGAIN!");
        }
        }
        }
        class ClassResult : StudentResult, iInputGeneric
        {
        public void ViewClassResult(int sem)
        {

Console.WriteLine("___________________________________________________");
        Console.WriteLine("SNo Student Name       Sub1   Sub2   Total");

Console.WriteLine("___________________________________________________");
        for (int i = 0; i < total_students; i++)
        {
                Console.Write("{0, -5}", i + 1);
                Console.Write("{0, -19}", Student_list[i].name);
                Console.Write("{0, -7}", Student_list[i].marks[0]);
                Console.Write("{0, -7}", Student_list[i].marks[1]);
                Console.Write("{0, -7}", Student_list[i].total);
                Console.WriteLine();
        }

Console.WriteLine("______________________________________________________
__");
        }
        }
        sealed class ViewResultMenu : ClassResult
        {
        public int Menu()
        {
        Console.Write("Enter ID Number for verification : ");
        int id = Convert.ToInt32(Console.ReadLine());
        if(id<500)
        {
                return 1;
        }
                 else if ( id == 883)
                 {
                        return 1;
                 }
        else
        {
                Console.WriteLine("Choose: 1. Student Result\n2. Class Result");
                int ch = Convert.ToInt32(Console.ReadLine());
                if(ch == 1)
                return 1;
                else if(ch == 2)
                return 2;
                else
                return -99;
        }


        }
        }



        public class Program
        {
        public static void Main()
        {
        ViewResultMenu records = new ViewResultMenu();
        Console.WriteLine("Hello!!");
        records.AddStudents();
        records.InputMarks();
        int ch = records.Menu();
        if(ch == 1)
        {
                Console.Write("Student Register Number of student for displaying result : ");
                int rnum = Convert.ToInt32(Console.ReadLine());
                records.ViewResult(rnum);

        }
        else if(ch == 2)
        {
                Console.Write("Enter Semester for displaying class result: ");
                int semes = Convert.ToInt32(Console.ReadLine());
                records.ViewClassResult(semes);
        }
        }
        }
}
```

3. RESULT

As we discussed, the program offers multi-result management where accessibility to types of results privilege is dependent on ID number to differentiate between faculty and students through Employee ID and Registration Number respectively. For this system, we took students to have ID number less than till 500, above which is faculty ID. Here, we also have an exception case as told before, with Student ID number 883.

Case 1: Faculty accessing individual student result

Since ID number is 501, which falls under faculty category, the console offers privilege to access individual student results as well as class result. Here, faculty chooses individual student result and student result is printed.

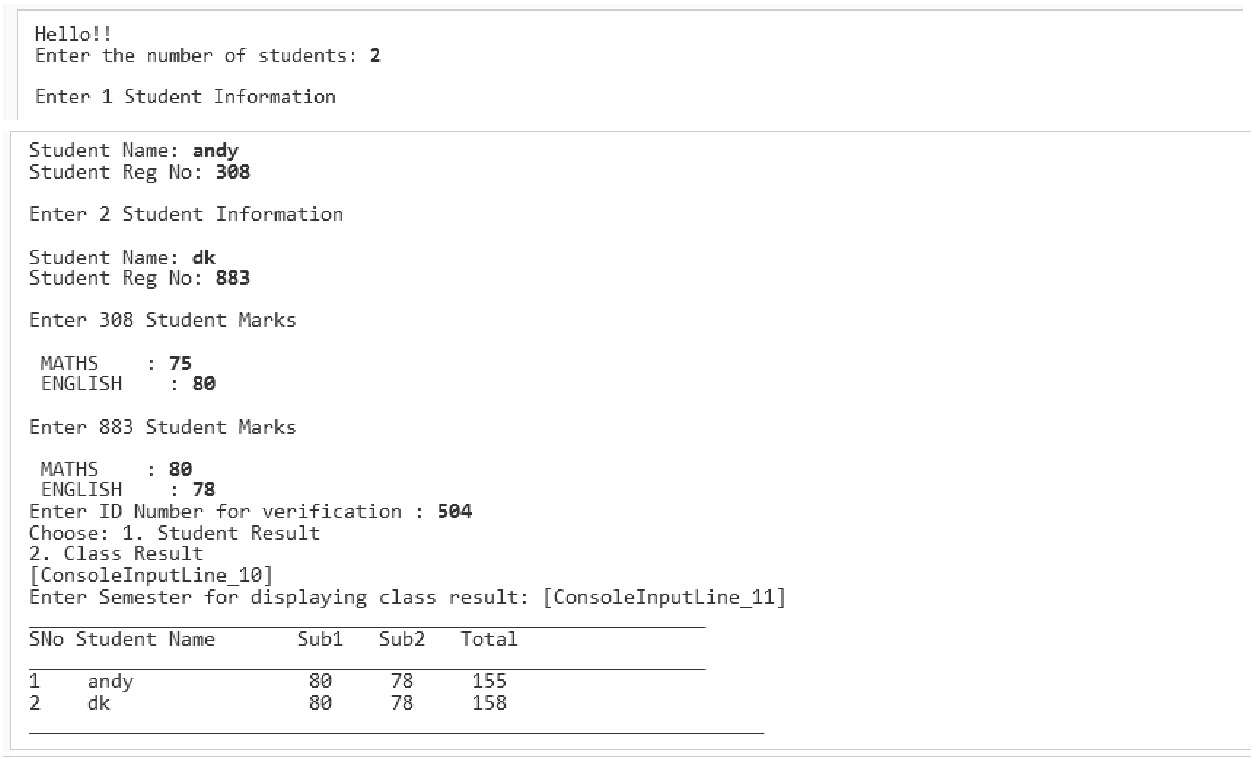


Microsoft Visual Studio Debug Console

```
Hello!!
Enter the number of students: 2

Enter 1 Student Information

Student Name: 417
Student Reg No: 417

Enter 2 Student Information

Student Name: andy
Student Reg No: 308

Enter 417 Student Marks

 MATHS    : 70
 ENGLISH    : 80

Enter 308 Student Marks

 MATHS    : 75
 ENGLISH    : 85
Enter ID Number for verification : 501
Choose: 1. Student Result
2. Class Result
1
Student Register Number of student for displaying result : 308
STUDENT NAME      :  andy
REGISTER NUMBER   :  308

_______________________________________________________________
                       RESULT CARD
_______________________________________________________________
  SUBJECTS        MAX MARKS      MIN MARKS      MARKS AWARDED
MATHS        :     100            35             75
ENGLISH      :     100            35             85
Total        :     200            70             160

_______________________________________________________________

C:\Program Files\dotnet\dotnet.exe (process 14220) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
```

Case 2: Faculty accessing class result

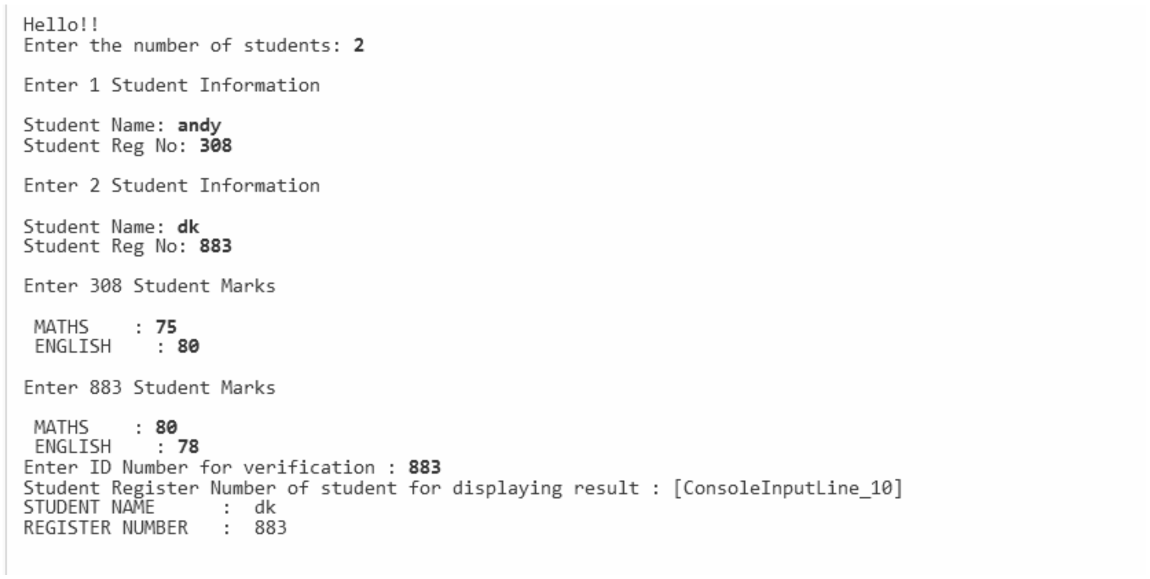
Since ID number is 504, which falls under faculty category, the console offers privilege to access individual student results as well as class result. Here, faculty chooses the class result report of class 6th semester.



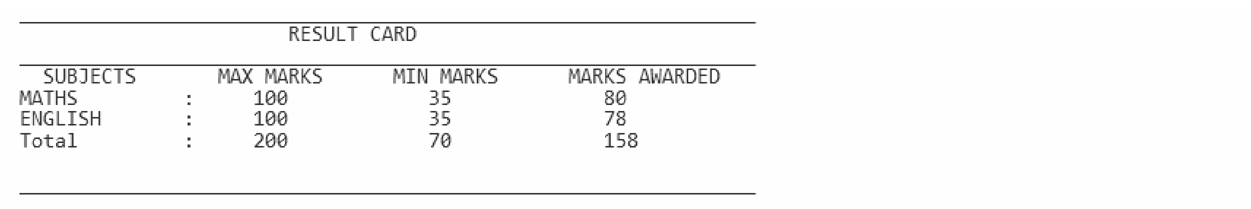
```
Hello!!
Enter the number of students: 2

Enter 1 Student Information

Student Name: andy
Student Reg No: 308

Enter 2 Student Information

Student Name: dk
Student Reg No: 883

Enter 308 Student Marks

 MATHS    : 75
 ENGLISH    : 80

Enter 883 Student Marks

 MATHS    : 80
 ENGLISH    : 78
Enter ID Number for verification : 504
Choose: 1. Student Result
2. Class Result
[ConsoleInputLine_10]
Enter Semester for displaying class result: [ConsoleInputLine_11]
__________________________________________________________
SNo Student Name       Sub1   Sub2   Total
__________________________________________________________
1    andy               80     78     155
2    dk                 80     78     158
_______________________________________________________________
```

Case 3: Exception student ID, accessing their own result.

Student ID 883 is an exception ID as by rule any ID above 500 is faculty, but this is Student ID. Hence, the console offers privilege to access individual student results only. Here, the user chooses to view their own result report.



```
Hello!!
Enter the number of students: 2

Enter 1 Student Information

Student Name: andy
Student Reg No: 308

Enter 2 Student Information

Student Name: dk
Student Reg No: 883

Enter 308 Student Marks

 MATHS    : 75
 ENGLISH    : 80

Enter 883 Student Marks

 MATHS    : 80
 ENGLISH    : 78
Enter ID Number for verification : 883
Student Register Number of student for displaying result : [ConsoleInputLine_10]
STUDENT NAME      :  dk
REGISTER NUMBER   :  883
```



```
________________________________________________________________
                       RESULT CARD
________________________________________________________________
  SUBJECTS        MAX MARKS       MIN MARKS       MARKS AWARDED
MATHS         :      100             35              80
ENGLISH       :      100             35              78
Total         :      200             70              158

________________________________________________________________
```

4. CONCLUSION & FUTURE SCOPE

We successfully Developed a Class result management system using SOLID principles and were able to implement all the features as required to fulfill the needs of the Teachers and Students.
This application serves multi-purpose, multi-user category that adds up to new benefits of this system

Now the developed System can be developed into a web based system, that can be made more easily accessible and operable for all user base. In the future, the results can be directly printed or to be shared by integrating it for sharing on multiple platforms, and this functionality can be made available to the user. This can also be enhanced by giving the user more services such as aggregate calculation etc.

5. REFERENCES

E. Chebanyuk and K. Markov, "An approach to class diagrams verification according to SOLID design principles," 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), 2016, pp. 435-441.

M. Gatrell, S. Counsell and T. Hall, "Design Patterns and Change Proneness: A Replication Using Proprietary C# Software," 2009 16th Working Conference on Reverse Engineering, 2009, pp. 160-164, doi: 10.1109/WCRE.2009.31.

M. Gatrell and S. Counsell, "Design patterns and fault-proneness a study of commercial C# software," 2011 FIFTH INTERNATIONAL CONFERENCE ON RESEARCH CHALLENGES IN INFORMATION SCIENCE, 2011, pp. 1-8, doi: 10.1109/RCIS.2011.6006827.

Agile Principles, Patterns, and Practices in C# (Robert C. Martin) | Guide books (acm.org)

Agerbo, E., Cornils, A.: How to Preserve the Benefits of Design Patterns. In: Proc. OOPLSA, pp. 134–143 (1998)

Baumgartner, G., Läufer, K., Russo, V.F.: On the interaction of object-oriented design patterns and programming languages. Technical Report CSR-TR-96-020, Purdue University (1996)