COMP1551 APPLICATION DEVELOPMENT

Tejesh Ramesh Bawa
Id No: 001173097-8

Table of Contents

Task 1

The Education Centre has now decided to transition from a Traditional Paper-Based Book-Keeping to a Modern Desktop Information System, which is expected to process information about 3 User Groups: Teaching Staff, Administration, and Students.

The system processes the same general information for all the user groups, which is the Name, Telephone Number, Email, and their Respective Role. The system will also process specified information on their Role, where the Teaching Staff is Salary and Names of 2 Subjects, Administration is Salary, Full Time/Part Time and Working Hours and the Students is Names of 2 Current Subjects and Names of 2 Previously Studied Subjects.

The system will have the functionality to Add New Data, View All Existing Data, View Existing Data by User Group, Edit, and Delete Existing Data. This range of functionalities will allow the Education Centre to stay informed about the following User Groups. This will allow the Education Centre to analyse the data to make informed decisions.

The system is expected to be used on the Microsoft Windows Operating System, as it is the most used platform around the world, therefore, making the adaptability to use the system will be quicker by the users. The system will be accessible on different hardware devices like desktop computers, laptops, and tablets, whereby it will be convenient for the users to access the system on their choice of devices.

In conclusion, the system will handle the adding, viewing, editing, and deleting of data from 3 different user groups. This will allow the educational system to generate reports, which will provide insights into the operations and services which are being exercised by the education centre, thereby allowing the stakeholders to make informed decisions on the improvements of the services provided by the Education Centre.

Introduction

Purpose

This Software Requirements Specification defines the requirements for a Desktop Information System for an education centre. This system will allow the education centre to migrate from a traditional paper-based book-keeping system to a desktop information system, which is more efficient and secure ways of managing information about the three user groups: Teaching Staff, Administration, and Students.

Project Scope

The system will allow the management to input data of the 3 user groups into the new system. This system will also allow the management to view all data, view data by specific user group, edit and delete data.

Overall Description

Product

The Desktop Information System is a simple software application which will allow the Education Centre to insert and manage data of the 3 User Group.

Users

The system will used by the management to input and manage data from the Teaching Staff, Administration, and Students. They have the general data like the name, telephone number, email, and role, as well as specific data corresponding to their user type.

Operational Environment

The system has to be installed into the education centre devices, which should have Windows 10 or later operating system, which is a widely used operating system, whereby easy making it easy to use.

System Features

Description

The system will insert and manage data from each of the user group.

Functional Requirements

The system will allow user to Add data of different user groups. All user groups have the same general information like the Name, Telephone Number, Email and Role. Furthermore, each user group have specified information like Teacher's Salary and 2 Subjects, Administration's Salary, Full-time or Part-time and Working Hours and Student's 2 Previous and Current Subjects .

Users will also be able to View all the data or specific user data.

Users will also be able to Edit and Delete the data.

User Interface Requirements

The system should be completely intuitive and user friendly for the user, whereby allowing them to navigate the system easier.

The system should allow complete navigation by only using the keyboard, in addition to the mouse and keyboard.

Platform Requirements

The system will be installed on the education centre devices, which should be having Windows 10 or later operating systems.

Quality Attributes

Performance

The system needs to be able to hold and process large amounts of data, without giving a runtime error.

Security

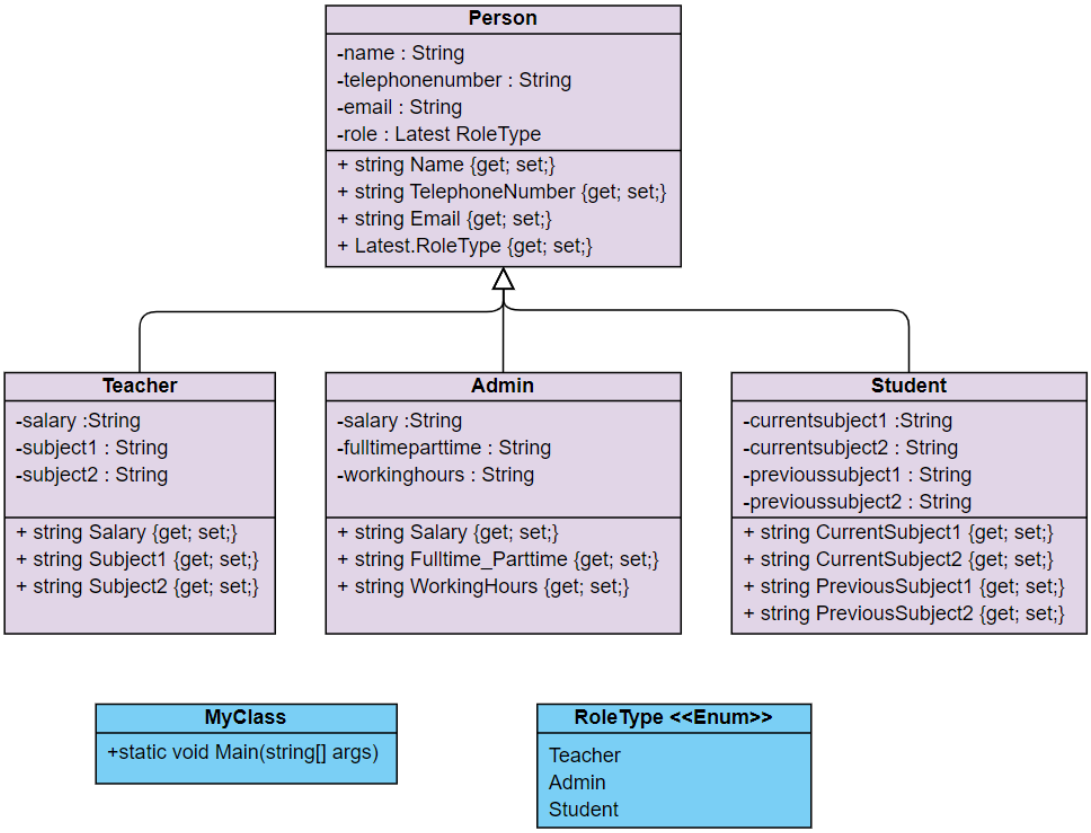
It is important that the system have an authentication, which will not allow unauthorized access to the system, whereby compromising the data withheld by the system.

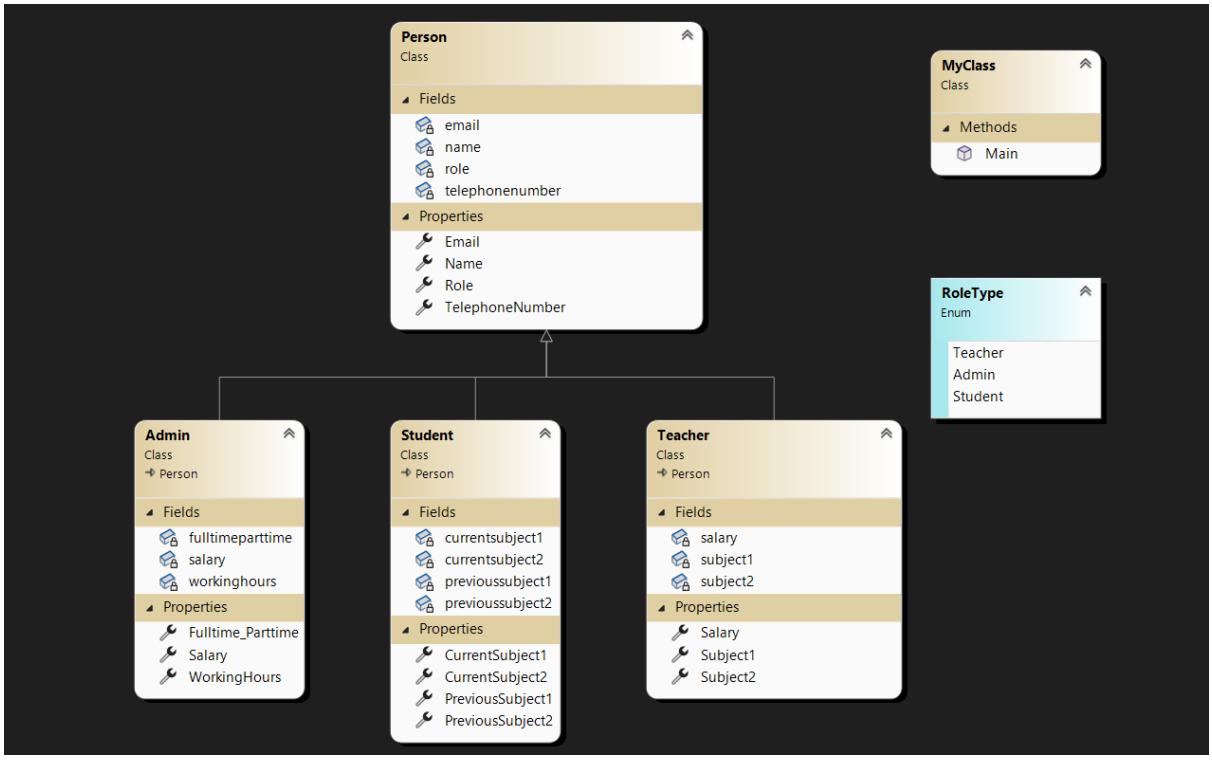
Safety

The system should not allow accidentally deletion of user data. It is important that the system should have a backup mechanism to retrieve the deleted data or system malfunction.

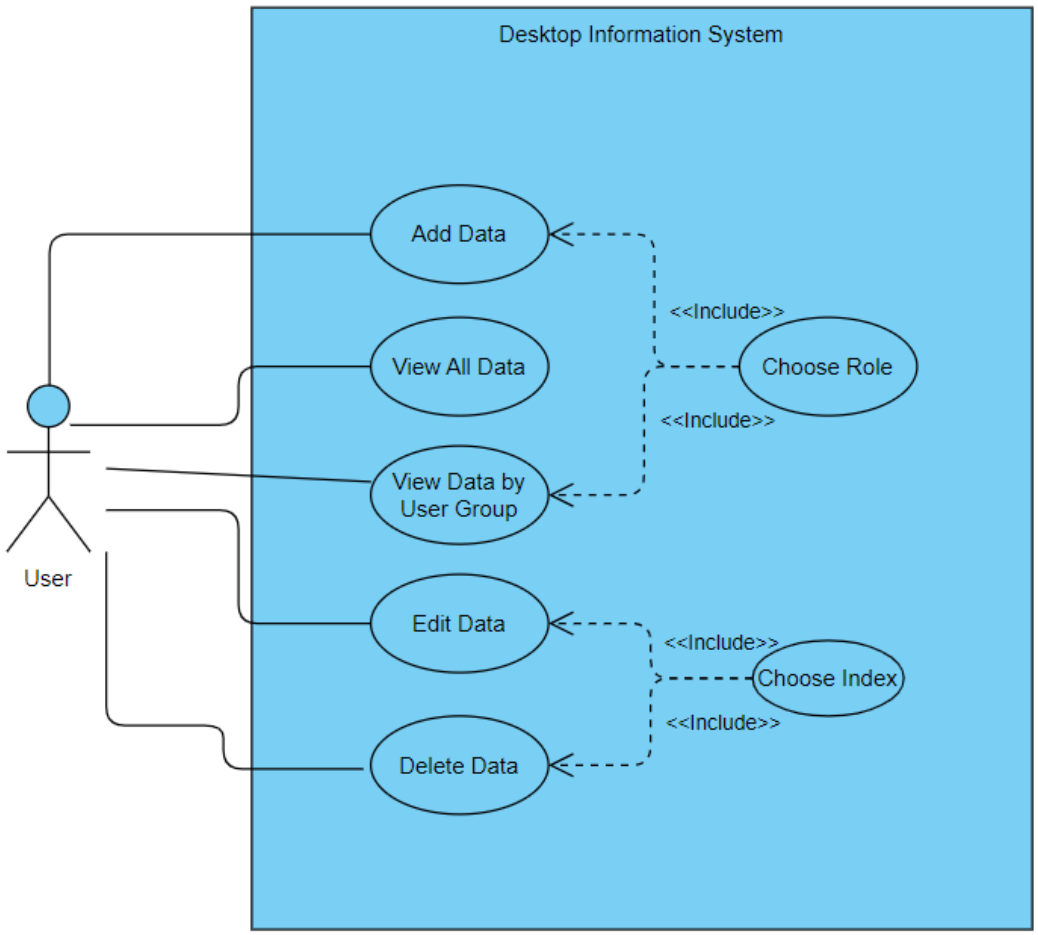
Task 2

Class Diagram





Use Case Diagram



Task 3

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

// namespace named Coursework
namespace Coursework
{
    // enumeration named "RoleType" that defines three values
    public enum RoleType
    {
        Teacher,
        Admin,
        Student
    }
    // class named Person
    class Person
    {
        // private variables that stored in the Person object
        private string name = "";
        private string telephonenumber = "";
        private string email = "";
        private RoleType role = RoleType.Student;

        // public property Name to give access to the Person object
        public string Name
        {
            get { return (name); }
            set { name = value; }
```

```
        }
        // public property TelephoneNumber to give access to the Person object
        public string TelephoneNumber
        {
            get { return (telephonenumber); }
            set { telephonenumber = value; }
        }
        // public property Email to give access to the Person object
        public string Email
        {
            get { return (email); }
            set { email = value; }
        }
        // public property RoleType to give access to the Person object
        public RoleType Role
        {
            // property returns the role of the Person object
            get { return (role); }
            // property sets the role of the Person object to possible value in RoleType enumeration
            set { if (((int)value >= (int)RoleType.Teacher) && ((int)value <= (int)RoleType.Student)) { role =
value; } }


        }
    }


    // class named Teacher that inherits from the Person class
    class Teacher : Person
    {
        // private variables that stored in the Teacher object
        private string salary = "";
        private string subject1 = "";
```

```
    private string subject2 = "";

    // public property Salary to give access to the Teacher object
    public string Salary
    {
        get { return salary; }
        set { salary = value; }
    }
    // public property Subject1 to give access to the Teacher object
    public string Subject1
    {
        get { return subject1; }
        set { subject1 = value; }
    }
    // public property Subject2 to give access to the Teacher object
    public string Subject2
    {
        get { return subject2; }
        set { subject2 = value; }
    }
}
// class named Admin that inherits from the Person class
class Admin : Person
{
    // private variables that stored in the Admin object
    private string salary = "";
    private string fulltimeparttime = "";
    private string workinghours = "";

    // public property Salary to give access to the Admin object
    public string Salary
```

```
    {
        get { return salary; }
        set { salary = value; }
    }
    // public property Fulltime_Parttime to give access to the Admin object
    public string Fulltime_Parttime
    {
        get { return fulltimeparttime; }
        set { fulltimeparttime = value; }
    }
    // public property WorkingHours to give access to the Admin object
    public string WorkingHours
    {
        get { return workinghours; }
        set { workinghours = value; }
    }
}
// class named Student that inherits from the Person class
class Student : Person
{
    // private variables that stored in the Student object
    private string currentsubject1 = "";
    private string currentsubject2 = "";
    private string previoussubject1 = "";
    private string previoussubject2 = "";

    // public property CurrentSubject1 to give access to the Student object
    public string CurrentSubject1
    {
        get { return currentsubject1; }
        set { currentsubject1 = value; }
```

```
    }
    // public property CurrentSubject2 to give access to the Student object
    public string CurrentSubject2
    {
        get { return currentsubject2; }
        set { currentsubject2 = value; }
    }
    // public property PreviousSubject1 to give access to the Student object
    public string PreviousSubject1
    {
        get { return previoussubject1; }
        set { previoussubject1 = value; }
    }
    // public property PreviousSubject2 to give access to the Student object
    public string PreviousSubject2
    {
        get { return previoussubject2; }
        set { previoussubject2 = value; }
    }
}

// main program MyClass
public class MyClass
{
    public static void Main(string[] args)
    {
        int i = 0;
        int c = 1;
        int r = 0;
        // List of objects
        List<Person> Person = new List<Person>();
```

```
while (c < 6)
{
   // writes the table for the users to choose their choice
   Console.WriteLine("\n Main Menu \n");
   Console.WriteLine("1. Add new data");
   Console.WriteLine("2. View all existing data");
   Console.WriteLine("3. View existing data by user group");
   Console.WriteLine("4. Edit existing user");
   Console.WriteLine("5. Delete existing user");
   Console.WriteLine("6. Exit");
   c = Convert.ToInt32(Console.ReadLine());
   switch (c)
   {
      // Adding new data
      case 1:
         // writes line
         Console.WriteLine("\n Adding new record \n");
         Person newPerson;
         Console.Write("Enter name: ");
         // reads line to store data in name
         string name = Console.ReadLine();
         Console.Write("Enter telephone number: ");
         // reads line to store data in telephonenumber
         string telephoneNumber = Console.ReadLine();
         Console.Write("Enter email: ");
         // reads line to store data in email
         string email = Console.ReadLine();
         // prompts user to choose the corresponding index
         Console.Write("Enter a role (0: Teacher, 1: Admin, 2: Student): ");
         int role = Convert.ToInt32(Console.ReadLine());
```

```
// after the role is chosen it will switch to the chosen role
switch ((RoleType)role)
{
    case RoleType.Teacher:
        newPerson = new Teacher();
        Console.Write("Enter salary: ");
        // reads line to store data in Salary
        ((Teacher)newPerson).Salary = Console.ReadLine();
        Console.Write("Enter subject 1: ");
        // reads line to store data in Subject1
        ((Teacher)newPerson).Subject1 = Console.ReadLine();
        Console.Write("Enter subject 2: ");
        // reads line to store data in Subject2
        ((Teacher)newPerson).Subject2 = Console.ReadLine();
        break;
    case RoleType.Admin:
        newPerson = new Admin();
        Console.Write("Enter salary: ");
        // reads line to store data in Salary
        ((Admin)newPerson).Salary = Console.ReadLine();
        Console.Write("Enter full-time/part-time: ");
        // reads line to store data in Fulltime_Parttime
        ((Admin)newPerson).Fulltime_Parttime = Console.ReadLine();
        Console.Write("Enter working hours: ");
        // reads line to store data in WorkingHours
        ((Admin)newPerson).WorkingHours = Console.ReadLine();
        break;
    case RoleType.Student:
        newPerson = new Student();
        Console.Write("Enter current subject 1: ");
        // reads line to store data in CurrentSubject1
```

```
            ((Student)newPerson).CurrentSubject1 = Console.ReadLine();
            Console.Write("Enter current subject 2: ");
            // reads line to store data in CurrentSubject2
            ((Student)newPerson).CurrentSubject2 = Console.ReadLine();
            Console.Write("Enter previous subject 1: ");
            // reads line to store data in PreviousSubject1
            ((Student)newPerson).PreviousSubject1 = Console.ReadLine();
            Console.Write("Enter previous subject 2: ");
            // reads line to store data in PreviousSubject2
            ((Student)newPerson).PreviousSubject2 = Console.ReadLine();
            break;
        default:
            Console.WriteLine("Invalid role");
            continue;
    }
    // all data is stored in newPerson
    newPerson.Name = name;
    newPerson.TelephoneNumber = telephoneNumber;
    newPerson.Email = email;
    newPerson.Role = (RoleType)role;
    // data added from newPerson to Person
    Person.Add(newPerson);
    break;
// listing all the records
case 2:
    Console.WriteLine("\n Listing all records \n");
    i = 1;
    // depending on the roletype it displays the data
    foreach (Person b in Person)
    {
        // displays data for teacher roletype
```

```
                if (b.Role == RoleType.Teacher)
                    Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Salary:{5} Subject1:{6} Subject2:{7}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Teacher)b).Salary, ((Teacher)b).Subject1, ((Teacher)b).Subject2);

                // displays data for admin roletype
                if (b.Role == RoleType.Admin)
                    Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Salary:{5} Fulltime/Parttime:{6} Working Hours:{7}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Admin)b).Salary, ((Admin)b).Fulltime_Parttime, ((Admin)b).WorkingHours);

                // displays data for student roletype
                if (b.Role == RoleType.Student)
                    Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Current Subject1:{5} Current Subject2:{6} Previous Subject1:{7} Previous Subject2:{8}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Student)b).CurrentSubject1, ((Student)b).CurrentSubject2, ((Student)b).PreviousSubject1, ((Student)b).PreviousSubject2);

                i = i + 1;
            }
            break;
        // listing records by role
        case 3:
            // prompts users to choose the index
            Console.Write("Enter a role  (0: Teacher, 1: Admin, 2: Student): ");
            // reads the chosen index
            r = Convert.ToInt32(Console.ReadLine());
            // prints the roletype chosen
            Console.WriteLine("You selected {0}: ", (RoleType)r);
            i = 1;
            foreach (Person b in Person)
            {
                if (r == (int)b.Role)
```

```
                {
                    // prints data for teacher roletype
                    if (b.Role == RoleType.Teacher)
                        Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Salary:{5} Subject1:{6} Subject2:{7}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Teacher)b).Salary, ((Teacher)b).Subject1, ((Teacher)b).Subject2);

                    // prints data for admin roletype
                    if (b.Role == RoleType.Admin)
                        Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Salary:{5} Fulltime/Parttime:{6} Working Hours:{7}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Admin)b).Salary, ((Admin)b).Fulltime_Parttime, ((Admin)b).WorkingHours);

                    // prints data for student roletype
                    if (b.Role == RoleType.Student)
                        Console.WriteLine("{0}. Name:{1} Telephone Number:{2} Email:{3} Role:{4} Current Subject1:{5} Current Subject2:{6} Previous Subject1:{7} Previous Subject2:{8}", i, b.Name, b.TelephoneNumber, b.Email, b.Role, ((Student)b).CurrentSubject1, ((Student)b).CurrentSubject2, ((Student)b).PreviousSubject1, ((Student)b).PreviousSubject2);

                    i = i + 1;
                }
            }
            break;
        // editing data by choosing the index
        case 4:
            // choosing index
            Console.Write("Enter the index of the record you want to edit: ");
            int index = Convert.ToInt32(Console.ReadLine());
            if (index >= 1 && index <= Person.Count)
            {
                // saves the updated data as existingPerson
                Person existingPerson = Person[index - 1];
```

```
                // Enter updated data
                //update for name
                Console.Write("Enter updated name (or press enter to keep existing value): ");
                string updatedName = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedName))
                {
                   existingPerson.Name = updatedName;
                }
                //update for telephonenumber
                Console.Write("Enter updated telephone number (or press enter to keep existing
value): ");
                string updatedTelephoneNumber = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedTelephoneNumber))
                {
                   existingPerson.TelephoneNumber = updatedTelephoneNumber;
                }
                //update for email
                Console.Write("Enter updated email (or press enter to keep existing value): ");
                string updatedEmail = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedEmail))
                {
                   existingPerson.Email = updatedEmail;
                }

                // update data depending on teacher roletype
                if (existingPerson.Role == RoleType.Teacher)
                {
                   Teacher existingTeacher = (Teacher)Person[index - 1];
```

```
    //update for salary for teacher
    Console.Write("Enter updated salary (or press enter to keep existing value): ");
    string updatedSalary = Console.ReadLine();
    // updates details
    if (!string.IsNullOrEmpty(updatedSalary))
    {
        existingTeacher.Salary = updatedSalary;
    }
    //update for subject1 for teacher
    Console.Write("Enter updated subject1 (or press enter to keep existing value): ");
    string updatedSubject1 = Console.ReadLine();
    // updates details
    if (!string.IsNullOrEmpty(updatedSubject1))
    {
        existingTeacher.Subject1 = updatedSubject1;
    }
    //update for subject2 for teacher
    Console.Write("Enter updated subject2 (or press enter to keep existing value): ");
    string updatedSubject2 = Console.ReadLine();
    // updates details
    if (!string.IsNullOrEmpty(updatedSubject2))
    {
        existingTeacher.Subject2 = updatedSubject2;
    }
}
// update data depending on admin roletype
if (existingPerson.Role == RoleType.Admin)
{
    Admin existingAdmin = (Admin)Person[index - 1];
    //update for salary for admin
    Console.Write("Enter updated salary (or press enter to keep existing value): ");
```

```
                    string updatedSalaryAdmin = Console.ReadLine();
                    // updates details
                    if (!string.IsNullOrEmpty(updatedSalaryAdmin))
                    {
                        existingAdmin.Salary = updatedSalaryAdmin;
                    }
                    //update for fulltime/parttime for admin
                    Console.Write("Enter updated Fulltime/Part (or press enter to keep existing value):
");
                    string updatedFullPart = Console.ReadLine();
                    // updates details
                    if (!string.IsNullOrEmpty(updatedFullPart))
                    {
                        existingAdmin.Fulltime_Parttime = updatedFullPart;
                    }
                    //update for workinghours for admin
                    Console.Write("Enter updated Work Hour (or press enter to keep existing value):
");
                    string updatedWorkHour = Console.ReadLine();
                    // updates details
                    if (!string.IsNullOrEmpty(updatedWorkHour))
                    {
                        existingAdmin.WorkingHours = updatedWorkHour;
                    }
                }
                // update data depending on student roletype
                if (existingPerson.Role == RoleType.Student)
                {
                    Student existingStudent = (Student)Person[index - 1];
                    //update for currentsubject1 for student
                    Console.Write("Enter updated Current Subject1 (or press enter to keep existing
value): ");
```

```
                string updatedCurrentSubject1 = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedCurrentSubject1))
                {
                   existingStudent.CurrentSubject1 = updatedCurrentSubject1;
                }
                //update for currentsubject2 for student
                Console.Write("Enter updated Current Subject2 (or press enter to keep existing
value): ");
                string updatedCurrentSubject2 = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedCurrentSubject2))
                {
                   existingStudent.CurrentSubject2 = updatedCurrentSubject2;
                }
                //update for pervioussubject1 for student
                Console.Write("Enter updated Previous Subject1 (or press enter to keep existing
value): ");
                string updatedPreviousSubject1 = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedPreviousSubject1))
                {
                   existingStudent.PreviousSubject1 = updatedPreviousSubject1;
                }
                //update for pervioussubject2 for student
                Console.Write("Enter updated Previous Subject2 (or press enter to keep existing
value): ");
                string updatedPreviousSubject2 = Console.ReadLine();
                // updates details
                if (!string.IsNullOrEmpty(updatedPreviousSubject2))
                {
                   existingStudent.PreviousSubject2 = updatedPreviousSubject2;
```

```
                    }
                }
            }
            break;
        // deleting data by choosing index
        case 5:
            // prints line
            Console.WriteLine("\n Deleting a record \n");
            // choosing the index of the data to delete
            Console.Write("Enter the index of the record to be deleted: ");
            // reads the choose index
            index = Convert.ToInt32(Console.ReadLine());
            if (index >= 1 && index <= Person.Count)
            {
                // deletes the data
                Person.RemoveAt(index - 1);
                Console.WriteLine("Record deleted successfully!");
            }
            else
            {
                Console.WriteLine("Invalid index!");
            }
            break;

        }
    }
  }
}
```

Results

Main Menu

```
 Main Menu

1. Add new data
2. View all existing data
3. View existing data by user group
4. Edit existing user
5. Delete existing user
6. Exit
```

Adding New Data

New Teacher Data

```
 Adding new record

Enter name: John Doe
Enter telephone number: 987654321
Enter email: johndoe@gmail.com
Enter a role (0: Teacher, 1: Admin, 2: Student): 0
Enter salary: 10000
Enter subject 1: Web Programming
Enter subject 2: Computer Science
```

New Admin Data

```
 Adding new record

Enter name: Susan Smith
Enter telephone number: 543216789
Enter email: susan@gmail.com
Enter a role (0: Teacher, 1: Admin, 2: Student): 1
Enter salary: 10000
Enter full-time/part-time: Full Time
Enter working hours: 40
```

New Student Data

```
 Adding new record

Enter name: Tejesh Ramesh
Enter telephone number: 123456789
Enter email: tb8667n@gre.ac.uk
Enter a role (0: Teacher, 1: Admin, 2: Student): 2
Enter current subject 1: Application Development
Enter current subject 2: Agile Scrum
Enter previous subject 1: Web Programming
Enter previous subject 2: Software Application
```

View All Existing Data

```
Listing all records

1. Name:Tejesh Ramesh Telephone Number:123456789 Email:tb8667n@gre.ac.uk Role:Student Current Subject1:Application Development
Current Subject2:Agile Scrum Previous Subject1:Web Programming Previous Subject2:Software Application
2. Name:John Doe Telephone Number:987654321 Email:johndoe@gmail.com Role:Teacher Salary:10000 Subject1:Web Programming Subject2
:Computer Science
3. Name:Susan Smith Telephone Number:543216789 Email:susan@gmail.com Role:Admin Salary:10000 Fulltime/Parttime:Full Time Workin
g Hours:40
```

View Existing Data by User Group

```
Enter a role  (0: Teacher, 1: Admin, 2: Student): 0
You selected Teacher:
1. Name:John Doe Telephone Number:987654321 Email:johndoe@gmail.com Role:Teacher Salary:10000 Subject1:Web Programming Subject2
:Computer Science
2. Name:Dana White Telephone Number:543219876 Email:dana@gmail.com Role:Teacher Salary:10000 Subject1:Chemistry Subject2:Physic
s
```

```
Enter a role  (0: Teacher, 1: Admin, 2: Student): 1
You selected Admin:
1. Name:Susan Smith Telephone Number:543216789 Email:susan@gmail.com Role:Admin Salary:10000 Fulltime/Parttime:Full Time Working
 Hours:40
```

```
Enter a role  (0: Teacher, 1: Admin, 2: Student): 2
You selected Student:
1. Name:Tejesh Ramesh Telephone Number:123456789 Email:tb8667n@gre.ac.uk Role:Student Current Subject1:Application Development
 Current Subject2:Agile Scrum Previous Subject1:Web Programming Previous Subject2:Software Application
```

Editing Existing User

```
Enter the index of the record you want to edit: 4
Enter updated name (or press enter to keep existing value): White Dana
Enter updated telephone number (or press enter to keep existing value): 111111111
Enter updated email (or press enter to keep existing value): white@gmail.com
Enter updated salary (or press enter to keep existing value): 20000
Enter updated subject1 (or press enter to keep existing value): English
Enter updated subject2 (or press enter to keep existing value): Math
```

```
Listing all records

1. Name:Tejesh Ramesh Telephone Number:123456789 Email:tb8667n@gre.ac.uk Role:Student Current Subject1:Application Development
 Current Subject2:Agile Scrum Previous Subject1:Web Programming Previous Subject2:Software Application
2. Name:John Doe Telephone Number:987654321 Email:johndoe@gmail.com Role:Teacher Salary:10000 Subject1:Web Programming Subject
2:Computer Science
3. Name:Susan Smith Telephone Number:543216789 Email:susan@gmail.com Role:Admin Salary:10000 Fulltime/Parttime:Full Time Worki
ng Hours:40
4. Name:White Dana Telephone Number:111111111 Email:white@gmail.com Role:Teacher Salary:20000 Subject1:English Subject2:Math
```

Deleting Existing User

```
Deleting a record

Enter the index of the record to be deleted: 4
Record deleted successfully!
```

```
Listing all records

1. Name:Tejesh Ramesh Telephone Number:123456789 Email:tb8667n@gre.ac.uk Role:Student Current Subject1:Application Development
 Current Subject2:Agile Scrum Previous Subject1:Web Programming Previous Subject2:Software Application
2. Name:John Doe Telephone Number:987654321 Email:johndoe@gmail.com Role:Teacher Salary:10000 Subject1:Web Programming Subject
2:Computer Science
3. Name:Susan Smith Telephone Number:543216789 Email:susan@gmail.com Role:Admin Salary:10000 Fulltime/Parttime:Full Time Worki
ng Hours:40
```