COM741 Web Application Development

Practical 2 Solution

1. Updated IStudentService Interface

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

using SMS.Data.Models;

namespace SMS.Data.Services
{
    // This interface describes the operations that a
    // StudentService class should implement
    public interface IStudentService
    {
        // Initialise the list
        void Initialise();

        // new interface method
        Student GetStudentByEmail(string email);

        // ---------------- Student Management --------------
        IList<Student> GetStudents();
        Student GetStudent(int id);
        Student AddStudent(string name, string course, string email,
                           int age, double grade);
        Student UpdateStudent(Student updated);
        bool DeleteStudent(int id);

    }

}
```

2. StudentServiceList implementation

```
using System;
using System.Linq;
using System.Collections.Generic;
using SMS.Data.Models;
```

```
using SMS.Data.Repository;

namespace SMS.Data.Services
{
    public class StudentServiceDb : IStudentService
    {
        private readonly DataContext db;

        public StudentServiceDb()
        {
            db = new DataContext();
        }

        public void Initialise()
        {
            db.Initialise(); // recreate database
        }

        // -------- Student Related Operations ------------

        // retrieve list of Students
        public List<Student> GetStudents()
        {
            return db.Students.ToList();
        }

        // Retrive student by Id
        public Student GetStudent(int id)
        {
            return db.Students.FirstOrDefault(s => s.Id == id);
        }

        // Add a new student checking email is unique
        public Student AddStudent(string name, string course, string
                                  int age, double grade)
        {
            // check if student with email exists
            var exists = GetStudentByEmail(email);
            if (exists != null)
            {
                return null;
            }

            // create new student
            var s = new Student
            {
                Name = name,
                Course = course,
```

```
                Email = email,
                Age = age,
                Grade = grade
            };
            db.Students.Add(s); // add student to the list
            db.SaveChanges();
            return s; // return newly added student
        }

        // Delete the student identified by Id returning true if
        // deleted and false if not found
        public bool DeleteStudent(int id)
        {
            var s = GetStudent(id);
            if (s == null)
            {
                return false;
            }
            db.Students.Remove(s);
            db.SaveChanges();
            return true;
        }

        // Update the student with the details in updated
        public Student UpdateStudent(Student updated)
        {
            // verify the student exists
            var student = GetStudent(updated.Id);
            if (student == null)
            {
                return null;
            }
            // update the details of the student retrieved and save
            student.Name = updated.Name;
            student.Email = updated.Email;
            student.Course = updated.Course;
            student.Age = updated.Age;
            student.Grade = updated.Grade;

            db.SaveChanges();
            return student;
        }

        public Student GetStudentByEmail(string email)
        {
            return db.Students.FirstOrDefault(s => s.Email == email);
        }
    }
```

```
}
```

3. Add Duplicate Student Test

```
[Fact] // --- AddStudent Duplicate Test
public void AddStudent_WhenDuplicateEmail_ShouldReturnNull()
{
   // act
   var s1 = svc.AddStudent("XXX", "xxx@email.com", "Computing", 20, (
   // this is a duplicate as the email address is same as previous st
   var s2 = svc.AddStudent("XXX", "xxx@email.com", "Computing", 20, (

   // assert
   Assert.NotNull(s1); // this student should have been added correct
   Assert.Null(s2); // this student should NOT have been added
}
```

4. Optional Questions

```
using System;
using System.Linq;                  // Linq Collection extension me
thods
using System.Collections.Generic;   // generic list classes

using SMS.Data.Models;      // student class
using SMS.Data.Extensions;  // List extension methods (ToPrettyStri
ng)
using SMS.Data.Services;    // Student Service

namespace SMS.Data  {

    public class Program
    {
        public static void Main (string[] args)
        {
            // call relevant methods here to test
            Question7_1();
            Question7_2();
            Question7_3();
        }

        public static void Question7_1()
        {
            Console.WriteLine("\nQuestion 7.1 - List all Student Name

            IStudentService svc = new StudentServiceDb();
            Seed(svc); // initialise the database
```

```
        // retrieve all students (List<Student>) and print studer
        var students = svc.GetStudents();
        students.ForEach(s => Console.WriteLine(s.Name));

        // alternatively select just the name out of the result l
        var names = svc.GetStudents().Select(s => s.Name).ToList(
        Console.WriteLine(names.ToPrettyString());
    }

    public static void Question7_2()
    {
        Console.WriteLine("\nQuestion 7.2 - List students in Grad

        IStudentService svc = new StudentServiceDb();
        Seed(svc); // initialise the database

        // retrieve all students - order by grade ascending
        // and print students name course and grade
        var students = svc.GetStudents().OrderBy(s => s.Grade).To

        // using a traditional loop to print each student
        // foreach(var s in students)
        // {
        //     Console.WriteLine($"{s.Name} {s.Course} {s.Grade}"
        // }

        // use List ForEach method to print the student details
        students.ForEach(s => Console.WriteLine($"{s.Name} {s.Cou

    }

    public static void Question7_3()
    {
        Console.WriteLine("\nQuestion 7.3 - List Students with Gr

        IStudentService svc = new StudentServiceDb();
        Seed(svc); // add seed data

        // print name and grade of students with Grade >= 60
        var students = svc.GetStudents().Where(s => s.Grade >= 60
        students.ForEach(s => Console.WriteLine($"{s.Name} {s.Gra
    }

    // ===== Utility add dummy student data via service ======
    private static void Seed(IStudentService svc)
    {
        svc.Initialise();
```

```
            svc.AddStudent("Homer", "Physics", "joe@mail.com", 42, 5(
            svc.AddStudent("Marge", "English", "marge@mail.com", 38,
            svc.AddStudent("Lisa", "Maths", "lisa@mail.com", 14, 80);
            svc.AddStudent("Bart", "Computing", "bart@mail.com", 12,
        }

    }
}
```