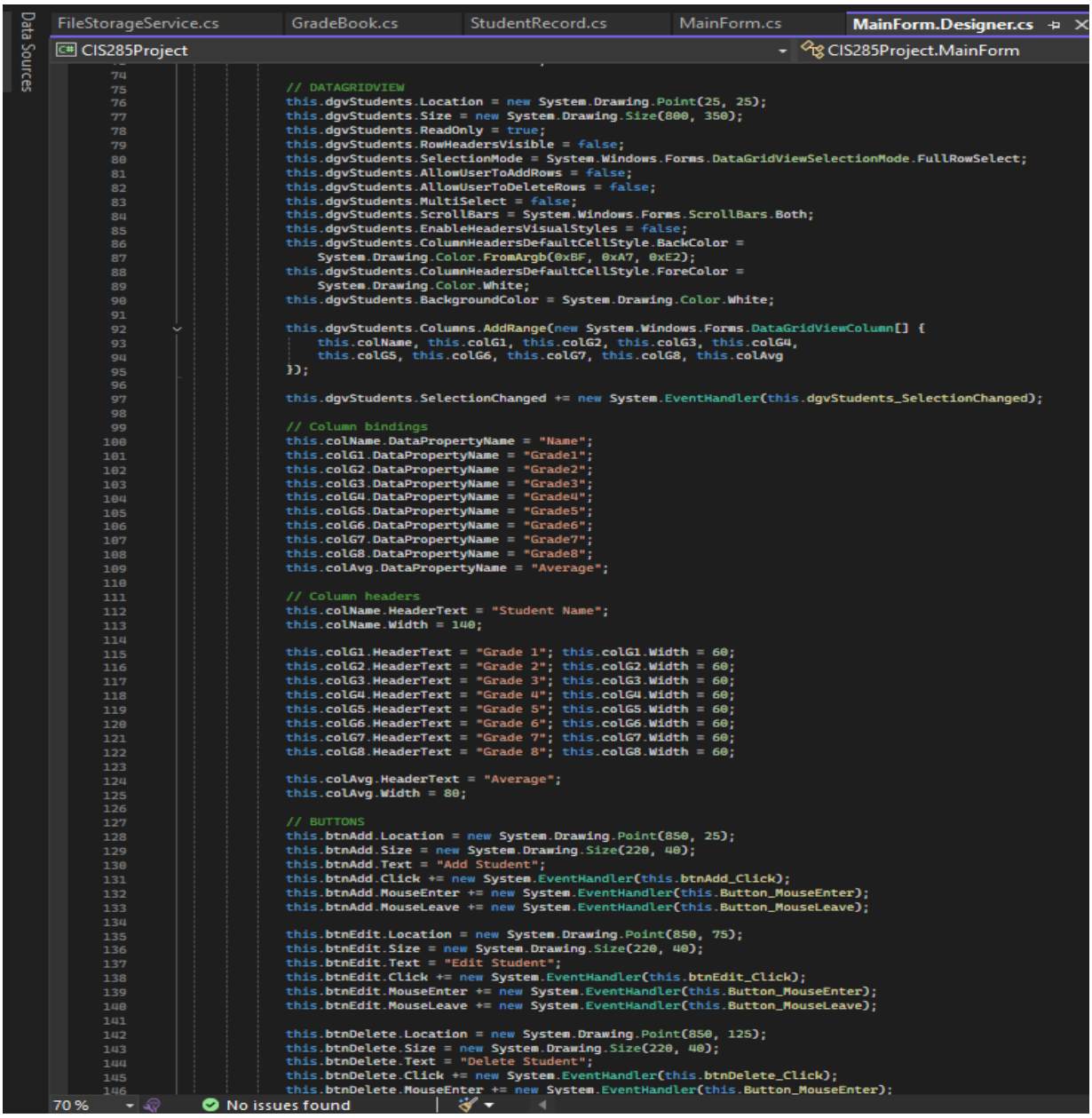
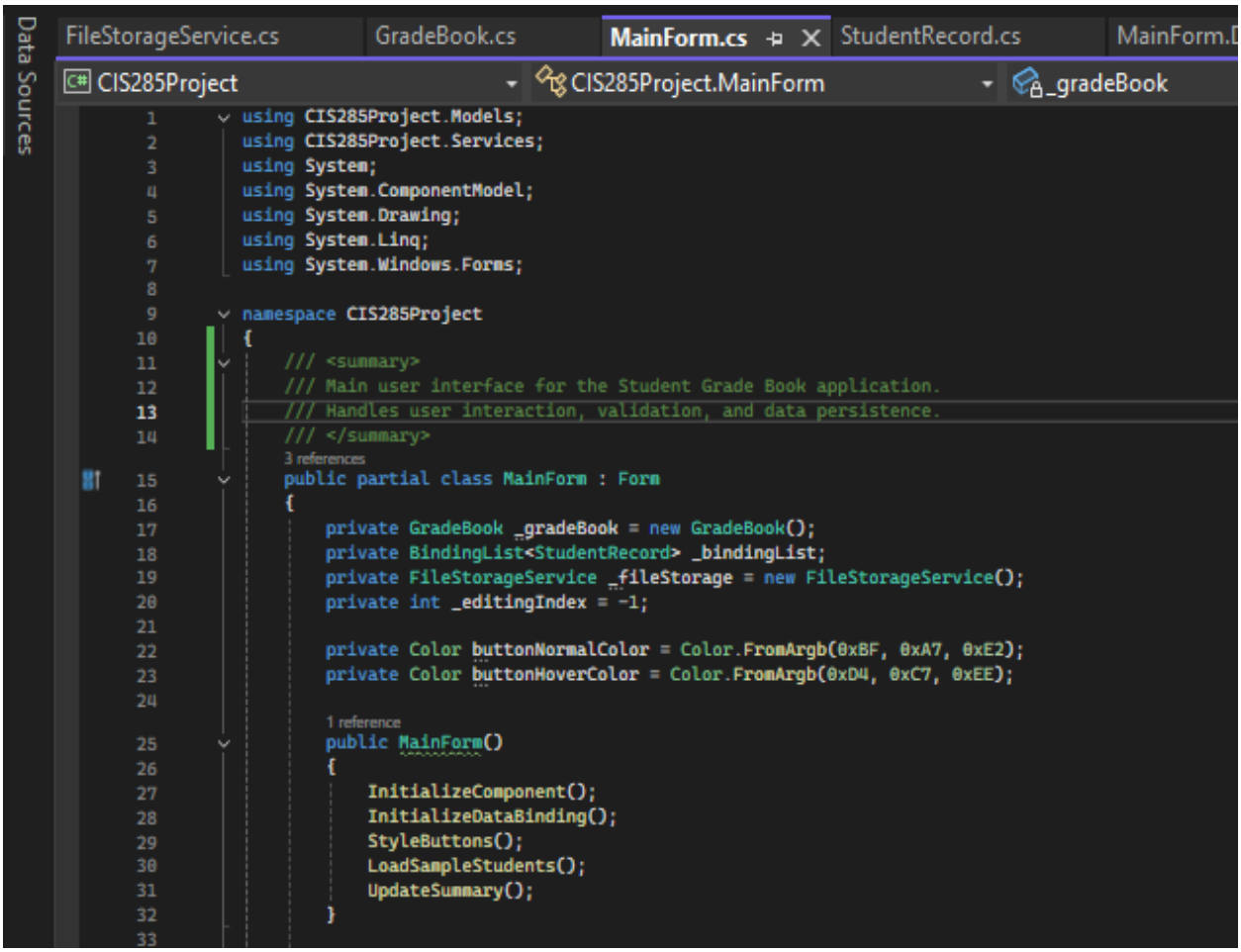
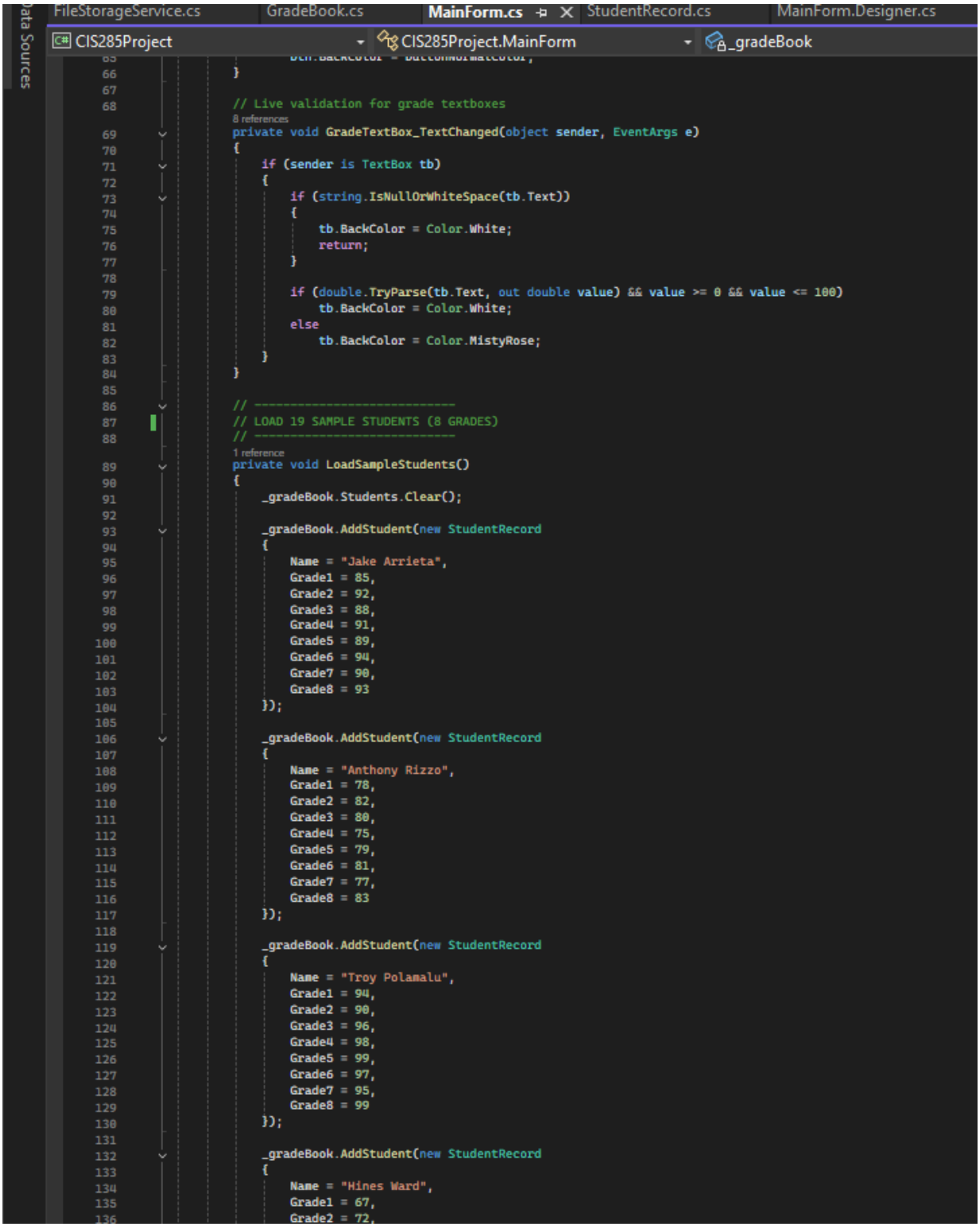
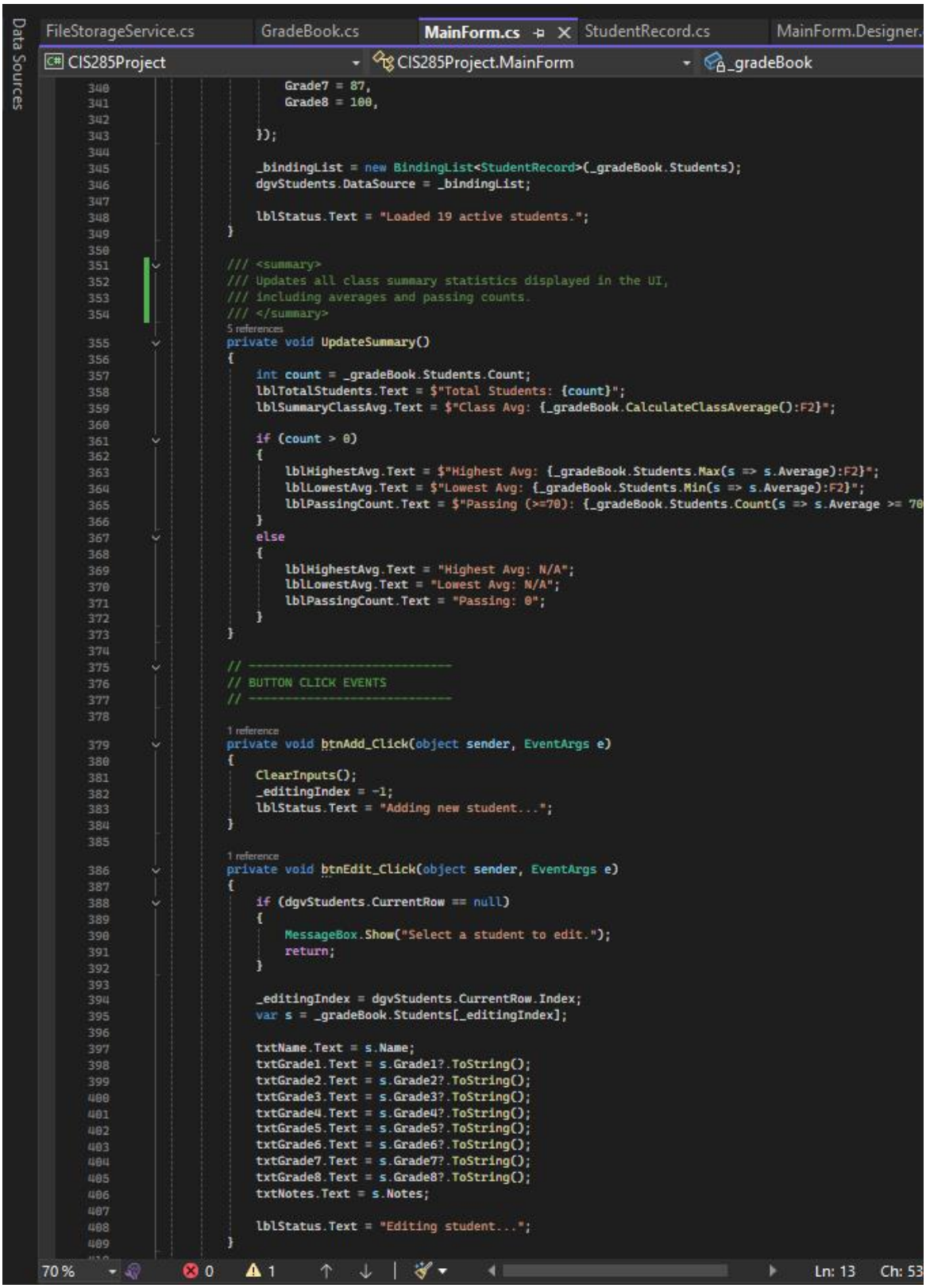
MAINFORMDESIGNER.cs

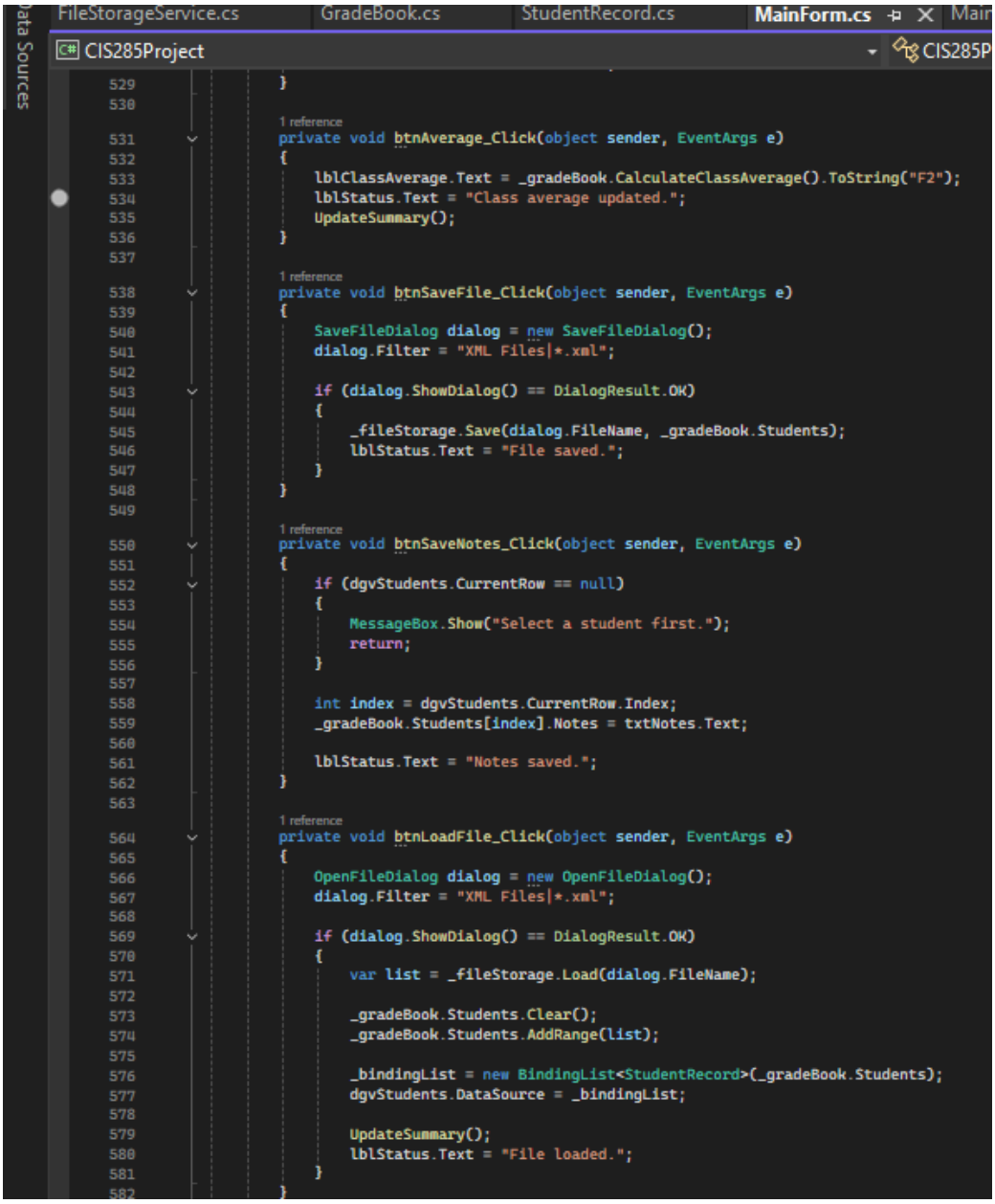


MAINFORM.cs (MULTIPLE SCREENSHOTS)



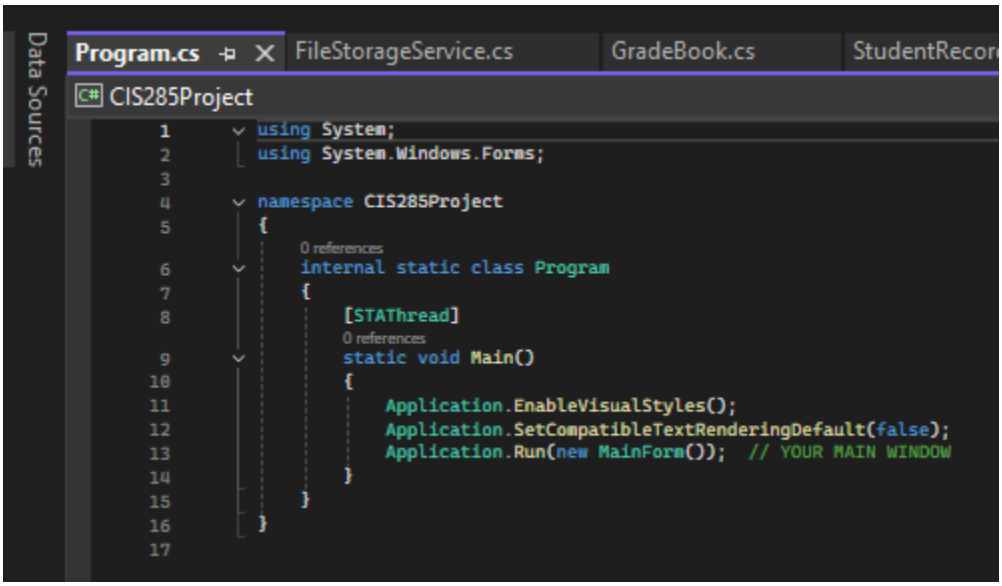

FileStorageService.cs
GradeBook.cs
MainForm.cs
StudentRecord.cs
MainForm.Designer.cs
CIS285Project
CIS285Project.MainForm
_gradeBook
Data Sources
}
// Live validation for grade textboxes
8 references
private void GradeTextBox_TextChanged(object sender, EventArgs e)
{
if (sender is TextBox tb)
{
if (string.IsNullOrWhiteSpace(tb.Text))
{
tb.BackColor = Color.White;
return;
}
if (double.TryParse(tb.Text, out double value) && value >= 0 && value <= 100)
tb.BackColor = Color.White;
else
tb.BackColor = Color.MistyRose;
}
}
// ----------------------------
// LOAD 19 SAMPLE STUDENTS (8 GRADES)
// ----------------------------
1 reference
private void LoadSampleStudents()
{
_gradeBook.Students.Clear();
_gradeBook.AddStudent(new StudentRecord
{
Name = "Jake Arrieta",
Grade1 = 85,
Grade2 = 92,
Grade3 = 88,
Grade4 = 91,
Grade5 = 89,
Grade6 = 94,
Grade7 = 90,
Grade8 = 93
});
_gradeBook.AddStudent(new StudentRecord
{
Name = "Anthony Rizzo",
Grade1 = 78,
Grade2 = 82,
Grade3 = 80,
Grade4 = 75,
Grade5 = 79,
Grade6 = 81,
Grade7 = 77,
Grade8 = 83
});
_gradeBook.AddStudent(new StudentRecord
{
Name = "Troy Polamalu",
Grade1 = 94,
Grade2 = 90,
Grade3 = 96,
Grade4 = 98,
Grade5 = 99,
Grade6 = 97,
Grade7 = 95,
Grade8 = 99
});
_gradeBook.AddStudent(new StudentRecord
{
Name = "Hines Ward",
Grade1 = 67,
Grade2 = 72,


Data Sources
FileStorageService.cs
GradeBook.cs
MainForm.cs
StudentRecord.cs
MainForm.Designer.
CIS285Project
CIS285Project.MainForm
_gradeBook
Grade7 = 87,
Grade8 = 100,
});
_bindingList = new BindingList<StudentRecord>(_gradeBook.Students);
dgvStudents.DataSource = _bindingList;
lblStatus.Text = "Loaded 19 active students.";
}
/// <summary>
/// Updates all class summary statistics displayed in the UI,
/// including averages and passing counts.
/// </summary>
5 references
private void UpdateSummary()
{
int count = _gradeBook.Students.Count;
lblTotalStudents.Text = $"Total Students: {count}";
lblSummaryClassAvg.Text = $"Class Avg: {_gradeBook.CalculateClassAverage():F2}";
if (count > 0)
{
lblHighestAvg.Text = $"Highest Avg: {_gradeBook.Students.Max(s => s.Average):F2}";
lblLowestAvg.Text = $"Lowest Avg: {_gradeBook.Students.Min(s => s.Average):F2}";
lblPassingCount.Text = $"Passing (>=70): {_gradeBook.Students.Count(s => s.Average >= 70
}
else
{
lblHighestAvg.Text = "Highest Avg: N/A";
lblLowestAvg.Text = "Lowest Avg: N/A";
lblPassingCount.Text = "Passing: 0";
}
}
// ----------------------------
// BUTTON CLICK EVENTS
// ----------------------------
1 reference
private void btnAdd_Click(object sender, EventArgs e)
{
ClearInputs();
_editingIndex = -1;
lblStatus.Text = "Adding new student...";
}
1 reference
private void btnEdit_Click(object sender, EventArgs e)
{
if (dgvStudents.CurrentRow == null)
{
MessageBox.Show("Select a student to edit.");
return;
}
_editingIndex = dgvStudents.CurrentRow.Index;
var s = _gradeBook.Students[_editingIndex];
txtName.Text = s.Name;
txtGrade1.Text = s.Grade1?.ToString();
txtGrade2.Text = s.Grade2?.ToString();
txtGrade3.Text = s.Grade3?.ToString();
txtGrade4.Text = s.Grade4?.ToString();
txtGrade5.Text = s.Grade5?.ToString();
txtGrade6.Text = s.Grade6?.ToString();
txtGrade7.Text = s.Grade7?.ToString();
txtGrade8.Text = s.Grade8?.ToString();
txtNotes.Text = s.Notes;
lblStatus.Text = "Editing student...";
}
70 %
0
1
Ln: 13
Ch: 53

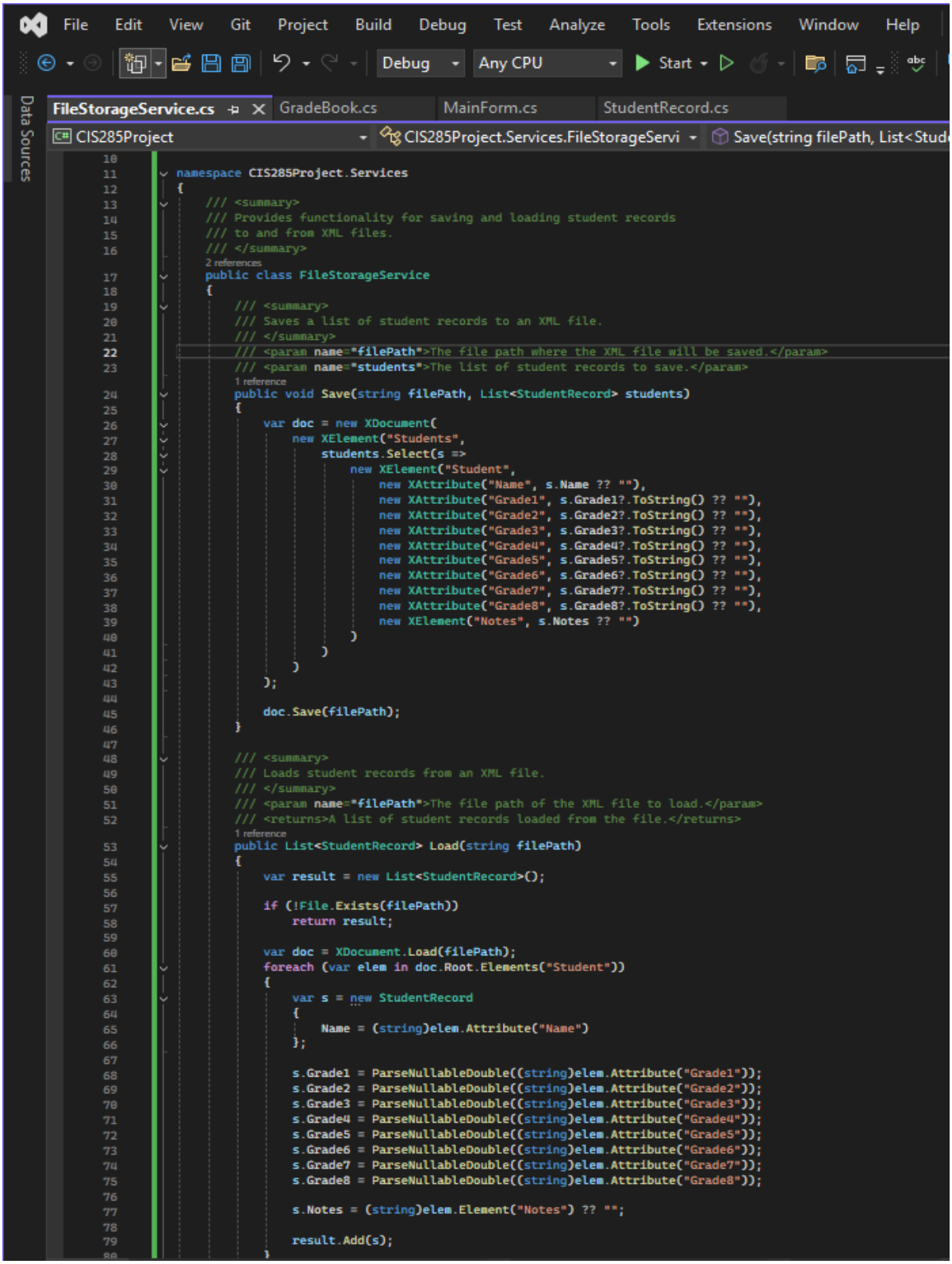



```
529         }
530
            1 reference
531         private void btnAverage_Click(object sender, EventArgs e)
532         {
533             lblClassAverage.Text = _gradeBook.CalculateClassAverage().ToString("F2");
534             lblStatus.Text = "Class average updated.";
535             UpdateSummary();
536         }
537
            1 reference
538         private void btnSaveFile_Click(object sender, EventArgs e)
539         {
540             SaveFileDialog dialog = new SaveFileDialog();
541             dialog.Filter = "XML Files|*.xml";
542
543             if (dialog.ShowDialog() == DialogResult.OK)
544             {
545                 _fileStorage.Save(dialog.FileName, _gradeBook.Students);
546                 lblStatus.Text = "File saved.";
547             }
548         }
549
            1 reference
550         private void btnSaveNotes_Click(object sender, EventArgs e)
551         {
552             if (dgvStudents.CurrentRow == null)
553             {
554                 MessageBox.Show("Select a student first.");
555                 return;
556             }
557
558             int index = dgvStudents.CurrentRow.Index;
559             _gradeBook.Students[index].Notes = txtNotes.Text;
560
561             lblStatus.Text = "Notes saved.";
562         }
563
            1 reference
564         private void btnLoadFile_Click(object sender, EventArgs e)
565         {
566             OpenFileDialog dialog = new OpenFileDialog();
567             dialog.Filter = "XML Files|*.xml";
568
569             if (dialog.ShowDialog() == DialogResult.OK)
570             {
571                 var list = _fileStorage.Load(dialog.FileName);
572
573                 _gradeBook.Students.Clear();
574                 _gradeBook.Students.AddRange(list);
575
576                 _bindingList = new BindingList<StudentRecord>(_gradeBook.Students);
577                 dgvStudents.DataSource = _bindingList;
578
579                 UpdateSummary();
580                 lblStatus.Text = "File loaded.";
581             }
582         }
```

PROGRAM.cs

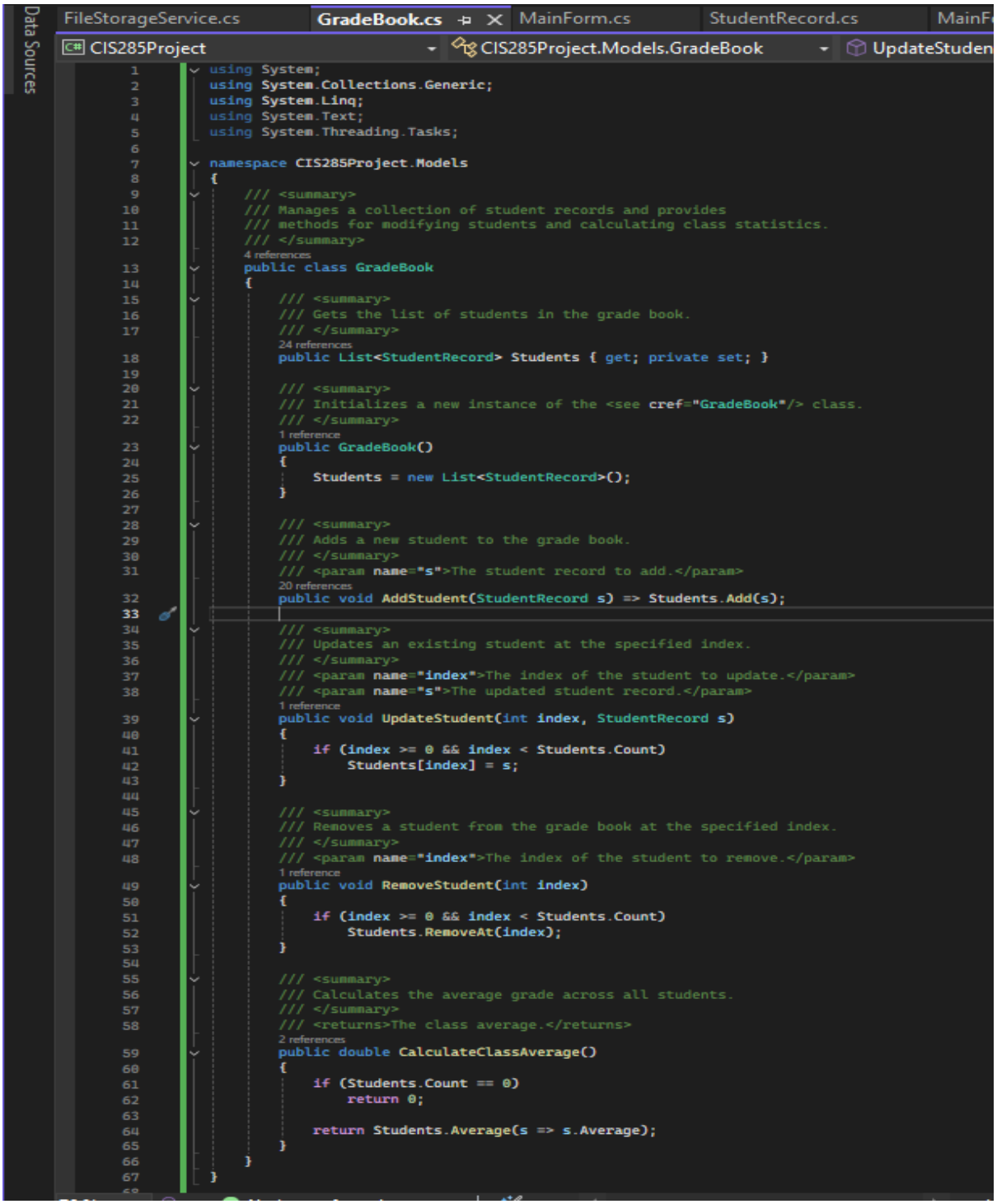

Data Sources
Program.cs
FileStorageService.cs
GradeBook.cs
StudentRecor
CIS285Project
using System;
using System.Windows.Forms;
namespace CIS285Project
{
0 references
internal static class Program
{
[STAThread]
0 references
static void Main()
{
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new MainForm()); // YOUR MAIN WINDOW
}
}
}

FILESTORAGESERVICE.cs



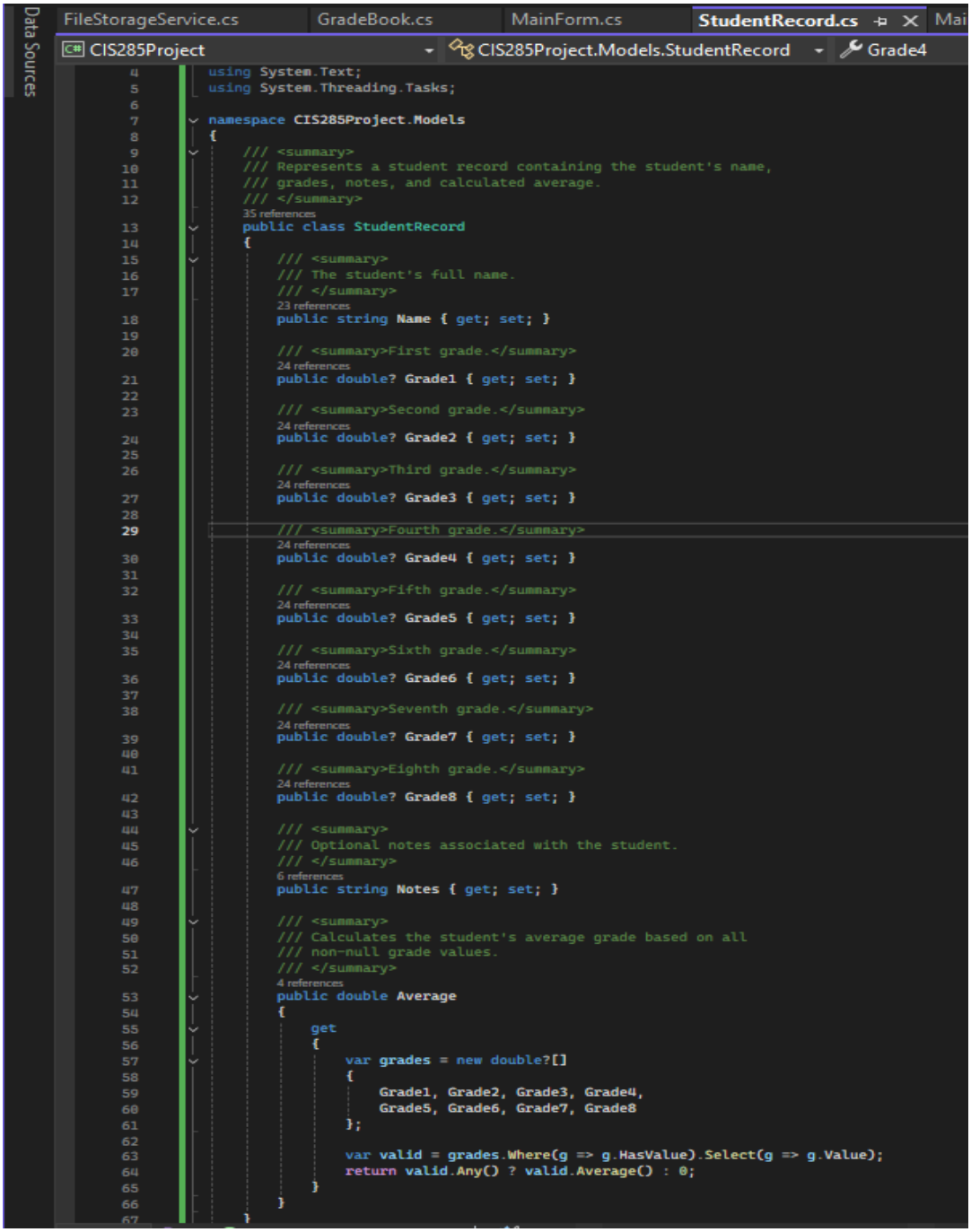

```
namespace CIS285Project.Services
{
    /// <summary>
    /// Provides functionality for saving and loading student records
    /// to and from XML files.
    /// </summary>
    2 references
    public class FileStorageService
    {
        /// <summary>
        /// Saves a list of student records to an XML file.
        /// </summary>
        /// <param name="filePath">The file path where the XML file will be saved.</param>
        /// <param name="students">The list of student records to save.</param>
        1 reference
        public void Save(string filePath, List<StudentRecord> students)
        {
            var doc = new XDocument(
                new XElement("Students",
                    students.Select(s =>
                        new XElement("Student",
                            new XAttribute("Name", s.Name ?? ""),
                            new XAttribute("Grade1", s.Grade1?.ToString() ?? ""),
                            new XAttribute("Grade2", s.Grade2?.ToString() ?? ""),
                            new XAttribute("Grade3", s.Grade3?.ToString() ?? ""),
                            new XAttribute("Grade4", s.Grade4?.ToString() ?? ""),
                            new XAttribute("Grade5", s.Grade5?.ToString() ?? ""),
                            new XAttribute("Grade6", s.Grade6?.ToString() ?? ""),
                            new XAttribute("Grade7", s.Grade7?.ToString() ?? ""),
                            new XAttribute("Grade8", s.Grade8?.ToString() ?? ""),
                            new XElement("Notes", s.Notes ?? "")
                        )
                    )
                )
            );

            doc.Save(filePath);
        }

        /// <summary>
        /// Loads student records from an XML file.
        /// </summary>
        /// <param name="filePath">The file path of the XML file to load.</param>
        /// <returns>A list of student records loaded from the file.</returns>
        1 reference
        public List<StudentRecord> Load(string filePath)
        {
            var result = new List<StudentRecord>();

            if (!File.Exists(filePath))
                return result;

            var doc = XDocument.Load(filePath);
            foreach (var elem in doc.Root.Elements("Student"))
            {
                var s = new StudentRecord
                {
                    Name = (string)elem.Attribute("Name")
                };

                s.Grade1 = ParseNullableDouble((string)elem.Attribute("Grade1"));
                s.Grade2 = ParseNullableDouble((string)elem.Attribute("Grade2"));
                s.Grade3 = ParseNullableDouble((string)elem.Attribute("Grade3"));
                s.Grade4 = ParseNullableDouble((string)elem.Attribute("Grade4"));
                s.Grade5 = ParseNullableDouble((string)elem.Attribute("Grade5"));
                s.Grade6 = ParseNullableDouble((string)elem.Attribute("Grade6"));
                s.Grade7 = ParseNullableDouble((string)elem.Attribute("Grade7"));
                s.Grade8 = ParseNullableDouble((string)elem.Attribute("Grade8"));

                s.Notes = (string)elem.Element("Notes") ?? "";

                result.Add(s);
            }
```

GRADEBOOK.cs




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

namespace CIS285Project.Models
{
    /// <summary>
    /// Manages a collection of student records and provides
    /// methods for modifying students and calculating class statistics.
    /// </summary>
    4 references
    public class GradeBook
    {
        /// <summary>
        /// Gets the list of students in the grade book.
        /// </summary>
        24 references
        public List<StudentRecord> Students { get; private set; }

        /// <summary>
        /// Initializes a new instance of the <see cref="GradeBook"/> class.
        /// </summary>
        1 reference
        public GradeBook()
        {
            Students = new List<StudentRecord>();
        }

        /// <summary>
        /// Adds a new student to the grade book.
        /// </summary>
        /// <param name="s">The student record to add.</param>
        20 references
        public void AddStudent(StudentRecord s) => Students.Add(s);

        /// <summary>
        /// Updates an existing student at the specified index.
        /// </summary>
        /// <param name="index">The index of the student to update.</param>
        /// <param name="s">The updated student record.</param>
        1 reference
        public void UpdateStudent(int index, StudentRecord s)
        {
            if (index >= 0 && index < Students.Count)
                Students[index] = s;
        }

        /// <summary>
        /// Removes a student from the grade book at the specified index.
        /// </summary>
        /// <param name="index">The index of the student to remove.</param>
        1 reference
        public void RemoveStudent(int index)
        {
            if (index >= 0 && index < Students.Count)
                Students.RemoveAt(index);
        }

        /// <summary>
        /// Calculates the average grade across all students.
        /// </summary>
        /// <returns>The class average.</returns>
        2 references
        public double CalculateClassAverage()
        {
            if (Students.Count == 0)
                return 0;

            return Students.Average(s => s.Average);
        }
    }
}
```

STUDENTRECORD.cs



Source Code Reference:

This project was developed by our group and was not adapted from any external repository or tutorial project.