**Source Code for Text-file Based System for Storing and Updating Teacher Records**

**Teacher.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Text\_file\_Based\_System\_Lib

{

[Serializable]

public class Teacher

{

public int ID { get; set; }

public string Name { get; set; }

public string Class { get; set; }

public string Section { get; set; }

}

}

**FileOperation.cs:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Text\_file\_Based\_System\_Lib;

namespace ConApp\_Text\_File\_project2

{

public class FileOperation

{

private const string FileName = @"D:\SimpliLearn\Phase End Project\C# Final project\Text-file Based System project 2\Record.txt";

public void AddTeacher(Teacher teacher)

{

using (StreamWriter writer = File.AppendText(FileName))

{

writer.WriteLine($"{teacher.ID},{teacher.Name},{teacher.Class},{teacher.Section}");

}

}

public List<Teacher> GetAllTeachers()

{

List<Teacher> teachers = new List<Teacher>();

if (File.Exists(FileName))

{

foreach (string line in File.ReadAllLines(FileName))

{

string[] parts = line.Split(',');

if (parts.Length == 4)

{

teachers.Add(new Teacher

{

ID = int.Parse(parts[0]),

Name = parts[1],

Class = parts[2],

Section = parts[3]

});

}

}

}

return teachers;

}

public void UpdateTeacher(int id, Teacher updatedTeacher)

{

List<string> lines = new List<string>();

if (File.Exists(FileName))

{

foreach (string line in File.ReadAllLines(FileName))

{

string[] parts = line.Split(',');

if (parts.Length == 4 && int.Parse(parts[0]) == id)

{

lines.Add($"{updatedTeacher.ID},{updatedTeacher.Name},{updatedTeacher.Class},{updatedTeacher.Section}");

}

else

{

lines.Add(line);

}

}

File.WriteAllLines(FileName, lines);

}

}

}

}

**Program.cs:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices.ComTypes;

using System.Runtime.Serialization.Formatters.Binary;

using System.Text;

using System.Threading.Tasks;

using Text\_file\_Based\_System\_Lib;

using System.Runtime.Serialization;

namespace ConApp\_Text\_File\_project2

{

internal class Program

{

static void Main(string[] args)

{

FileOperation manager = new FileOperation();

bool exit = false;

while (!exit)

{

Console.WriteLine("---------------------------------");

Console.WriteLine("Teacher Record Management");

Console.WriteLine("---------------------------------");

Console.WriteLine("1. Add Teacher");

Console.WriteLine("2. Display All Teachers");

Console.WriteLine("3. Update Teacher");

Console.WriteLine("4. Exit");

Console.WriteLine("---------------------------------");

Console.Write("Enter your choice: ");

int choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Console.Write("Enter Teacher ID: ");

int id = int.Parse(Console.ReadLine());

Console.Write("Enter Teacher Name: ");

string name = Console.ReadLine();

Console.Write("Enter Teacher Class: ");

string classValue = Console.ReadLine();

Console.Write("Enter Teacher Section: ");

string section = Console.ReadLine();

Teacher newTeacher = new Teacher { ID = id, Name = name, Class = classValue, Section = section };

manager.AddTeacher(newTeacher);

Console.WriteLine("Teacher added successfully.\n\n");

break;

case 2:

var teachers = manager.GetAllTeachers();

foreach (var teacher in teachers)

{

Console.WriteLine($"ID: {teacher.ID}, Name: {teacher.Name}, Class: {teacher.Class}, Section: {teacher.Section}");

}

break;

case 3:

Console.Write("Enter Teacher ID to Update: ");

int teacherIdToUpdate = int.Parse(Console.ReadLine());

Console.Write("Enter Updated Teacher Name: ");

string uName = Console.ReadLine();

Console.Write("Enter Updated Teacher Class: ");

string uClass = Console.ReadLine();

Console.Write("Enter Updated Teacher Section: ");

string uSection = Console.ReadLine();

Teacher updatedTeacher = new Teacher { ID = teacherIdToUpdate, Name = uName, Class = uClass, Section = uSection };

manager.UpdateTeacher(teacherIdToUpdate, updatedTeacher);

Console.WriteLine("Teacher updated successfully.\n\n");

break;

case 4:

exit = true;

break;

default:

Console.WriteLine("Invalid choice!!!!!!");

break;

}

Console.ReadKey();

}

}

}

}