

PHASE-1 : PROJECT DOCUMENT

.NET Console application:

Storing and Updating Teacher Data Using Text Files

Submitted by:

Chandramouli Kondamudi

ckondamudi@dovercorp.com

Contents:

1. Project Structure
2. Implementation Steps
 - a. Initializing file with data
 - b. Looping the Switch case for taking inputs from console.
 - c. Functionalities
 - i. DisplayElements
 - ii. Add Elements
 - iii. Update Data (id)
 - iv. Delete Record (id)
 - v. Exit
3. Output Screenshots
4. Github Link

1. Project Structure:

This project contains three files:

TeacherModel.cs: It is used to handle the teacher data (Id, name, class, section);

```
17 references
class TeacherModel
{
    9 references
    public int id { get; set; }
    6 references
    public string Name { get; set; }

    6 references
    public string Class { get; set; }

    6 references
    public string Section { get; set; }
}
```

Figure 1: TeacherModel class

TeacherBO.cs: This file contains the class TeacherBO, in which all the business logic is implemented. Business logic includes 1. Adding teacher data to the file 2. Updating teacher data in the file 3. Deleting teacher data in the file 4. Displaying teacher data from the file 5. Append Data to the file 6. Add data to the file (overwrite existing file with new data)

Program.cs: This file contains the main method from where we will access the business logic based on the input given by the user.

2. Implementation Steps:

a. Initializing file with data

Initially we store 5 teacher records in a text file (Phase1_Project.txt).

ID	Name	Class	Section
1	Chandra	X	A
2	Mouli	IX	B
3	Chiru	X	B
4	Balayya	IX	B
5	Venky	VIII	C

When the program starts, we store all the file data into a data structure (List)

- i) `TeacherBO context = new TeacherBO();` *// object created to TeacherBo where all the business logic is implemented*

```
namespace Assignments
{
    3 references
    class TeacherBO
    {
        13 references
        public List<TeacherModel> teachers { get; set; }

        1 reference
        public TeacherBO()
        {
            teachers = new List<TeacherModel>();
        }
    }
    4 references
}
```

Figure 2: Constructor of TeacherBO class

- ii) Context.CreateModel (); *// This method will create a teacherModel taking the string*

```
1 reference
public TeacherModel CreateModel(string str)
{
    string[] st = str.Split(" ");

    int id = Convert.ToInt32(st[0]);
    string Name = st[1];
    string Class = st[2];
    string Section = st[3];

    return new TeacherModel { id = id, Name = Name, Class = Class, Section = Section };
}
```

Figure 3: CreateModel implementation in TeacherBO class

- iii) Context.WriteToFile() : *// This method will write the data to the file*

```
2 references
public void WriteToFile(List<TeacherModel> teachers)
{
    int i = 0;
    foreach (TeacherModel t in teachers)
    {
        if (i == 0)
            Add(t);
        else
            Append(t);
        i++;
    }
}
```

Figure 4: WriteToFile implementation in TeacherBO class

- iv) Add(TeacherModel t): *// This will overwrite existing data*

```

1 reference
public void Add(TeacherModel t)
{
    string fpath = @"e:\Phase1_Project.txt";
    StreamWriter fname = new StreamWriter(fpath,false);

    fname.WriteLine($"{t.id} {t.Name} {t.Class} {t.Section}");
    fname.Close();
}

```

Figure 5: Add Method implementation in TeacherBO class

- v) Append(TeacherModel t) : // This will append to the existing file

```

2 references
public void Append(TeacherModel t)
{
    string fpath = @"e:\Phase1_Project.txt";
    StreamWriter fname = File.AppendText(fpath);

    fname.WriteLine($"{t.id} {t.Name} {t.Class} {t.Section}");
    fname.Close();
}

```

Figure 6 : Append Method implementation in TeacherBO class

- vi) CheckId(id): // This function will check if the given id exists in the database

```

3 references
public bool CheckId(int id)
{
    teachers=GetAllTeachers();
    int index = teachers.FindIndex(x => x.id == id);

    if (index == -1)
        return false;
    return true;
}

```

Figure 7 : CheckId implementation in TeacherBO class

c. Looping the switch case for taking inputs from the console

We will take input from the user, to perform his/her preferred operations on the file.

Below is the pseudo code, we will see the actual code of each case from the next step

```
int input_from_user;
while(true)
{
    ...
    ...
    //here we read input from user
    switch(x)
    {
        Case 1: context.DisplayAllData();

        Case 2: context.Append(teacher);

        Case 3: context.Update(teacher, update_id);

        Case 4: context.Delete(delete_id);

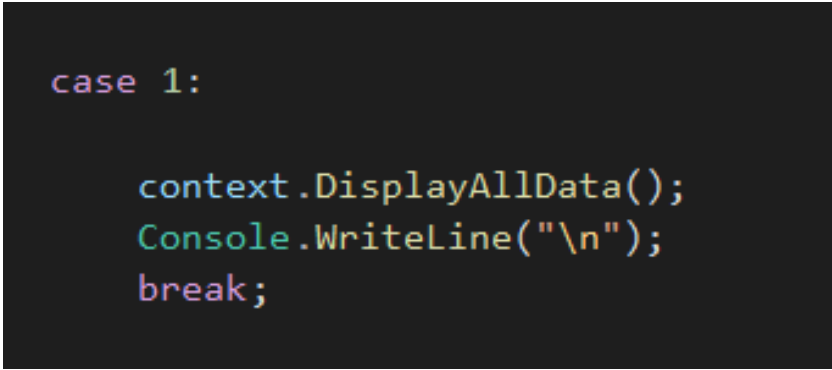
        Case 5: Exit;

        Default:
    }
}
```

d. Functionalities

- i) **DisplayAllData():** Display complete data

We will read the file completely and displays the data on the console



```
case 1:
    context.DisplayAllData();
    Console.WriteLine("\n");
    break;
```

Figure 8: DisplayAllData implementation to take input and call DisplayAllData() method

```

4 references
public List<TeacherModel> GetAllTeachers()
{
    string fpath = @"e:\Phase1_Project.txt";
    List<TeacherModel> teachers=new List<TeacherModel>();
    foreach(string line in File.ReadLines(fpath))
    {
        TeacherModel t = CreateModel(line);
        teachers.Add(t);
    }
    return teachers;
}

```

Figure 9: GetAllTeachers() method implementation in TeacherBO class

- ii) **Add Elements:** Append new teacher data to the end of file *// switch case implementation for add element*
 Case 2: **Context.Append(teacher)** ;
 break;


```

case 2:

    Console.WriteLine("Existing Data : ");
    context.DisplayAllData();
    Console.WriteLine("\n");
    Console.WriteLine("Adding to the existing Data ");
    Console.Write("Enter id number : ");
    try
    {
        int id = Convert.ToInt32(Console.ReadLine());
        if (context.CheckId(id))
        {
            Console.WriteLine($"this id :{id} already exists in the database");
        }
        else
        {
            Console.Write("Enter Teacher name : ");
            string Name = Console.ReadLine();
            Console.Write("Enter Class the teacher teaches : ");
            string Class = Console.ReadLine();
            Console.Write("Enter Section the teacher takes : ");
            string Section = Console.ReadLine();

            teacher = new TeacherModel { id = id, Name = Name, Class = Class, Section = Section };
            context.Append(teacher);

            Console.WriteLine(" data is added to the DataBase");
        }

        Console.WriteLine("\n");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error :"+ex.Message);
    }
    break;

```

Figure 10: Add Method implementation, calling of Append method in TeacherBO class

iii) **Update(Id):** Update the data with the given id

Here we will display the complete data to user and takes id as input and will update the files based on given input.

```

case 3:

    Console.WriteLine("Existing Data : ");
    context.DisplayAllData();
    Console.WriteLine("\n");
    Console.Write("Enter id number to be updated : ");

    try
    {
        int update_id = Convert.ToInt32(Console.ReadLine());

        if (context.CheckId(update_id))
        {
            Console.Write("Enter Teacher name : ");
            string Name = Console.ReadLine();
            Console.Write("Enter Class the teacher teaches : ");
            string Class = Console.ReadLine();
            Console.Write("Enter Section the teacher takes : ");
            string Section = Console.ReadLine();

            teacher = new TeacherModel { id = update_id, Name = Name, Class = Class, Section = Section };
            context.Update(teacher, update_id);

            Console.WriteLine($"Database is updated successfully");
            Console.WriteLine("\n Updated Database is : \n");
            context.DisplayAllData();
        }
        else
        {
            Console.WriteLine($"There is no entry available with the given id :{update_id}");
        }
        Console.WriteLine("\n");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error : " + ex.Message);
    }
}

```

Figure 11: Switch case implementation to take input and call the Update(teacher,id) method

```

1 reference
public void Update(TeacherModel t,int update_id)
{

    teachers = GetAllTeachers();
    int index = teachers.FindIndex(x => x.id == update_id);

    teachers[index] = t;

    WriteToFile(teachers);
}

```

Figure 12: update(id) method implementation from TeacherBO class

- iv) **Delete(id)** : Delete the teacher record with the given id

Here , we display the existing data and will ask user for the id to be deleted , then will delete that record from the database

```
case 4:

    Console.WriteLine("Existing Data ");
    context.DisplayAllData();
    Console.WriteLine("\n");
    Console.Write("Enter id number to be deleted : ");

    try
    {
        int delete_id = Convert.ToInt32(Console.ReadLine());

        if (context.CheckId(delete_id))
        {
            context.Delete(delete_id);
            Console.WriteLine($"Database with the id : {delete_id} is successfully deleted");
        }
        else
        {
            Console.WriteLine($"There is no record preseent with the id as {delete_id}");
        }

        Console.WriteLine("\n");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Error is : " + ex.Message) ;
    }

    break;
```

Figure13:switch case implementation to take input and call delete(id) method

```

1 reference
public void Delete(int id)
{
    teachers = GetAllTeachers();
    int index = teachers.FindIndex(x => x.id == id);

    teachers.RemoveAt(index);

    WriteToFile(teachers);
}

```

Figure 14: implementation of Delete(id) in the TeacherBO class

- v) **Exit**: Exists from the while loop

```

if (input_from_user == 5)
{
    Console.WriteLine($"User given input is {input_from_user} , exited from the application");
    break;
}

```

Figure 15: Switch case implementation to take input and exit from loop

3.Output Screenshot's:

```
C:\> E:\Assignments\Assignments\bin\Debug\netcoreapp3.1\Assignments.exe
```

```
Options for the user
```

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

```
Enter your input : █
```

```
C:\> E:\Assignments\Assignments\bin\Debug\netcoreapp3.1\Assignments.exe
```

```
Options for the user
```

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

```
Enter your input : 1
```

```
1 Chandra X A
```

```
2 Mouli IX B
```

```
3 Chiru X B
```

```
4 Balayya IX B
```

```
5 Venky VIII C
```

```
5 Nag X C
```

```
Options for the user
```

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

```
Enter your input :
```

options for the user

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

Enter your input : 2

Existing Data :

- 1 Chandra X A
- 2 Mouli IX B
- 3 Chiru X B
- 4 Balayya IX B
- 5 Venky VIII C
- 6 Nag X C

Adding to the existing Data

Enter id number : 7

Enter Teacher name : Pawan

Enter Class the teacher teaches : X

Enter Section the teacher takes : C

data is added to the DataBase

options for the user

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

Enter your input : 1

- 1 Chandra X A
- 2 Mouli IX B
- 3 Chiru X B
- 4 Balayya IX B
- 5 Venky VIII C
- 6 Nag X C
- 7 Pawan X C

```

E:\Assignments\Assignments\bin\Debug\netcoreapp3.1\Assignments.exe

```

```

3 Chiru X B
4 Balayya IX B
5 Venky VIII C
6 Nag X C
7 Pawan X C

```

```

options for the user

```

```

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

```

```

enter your input : 4

```

```

Existing Data

```

```

1 Chandra X A
2 Mouli IX B
3 Chiru X B
4 Balayya IX B
5 Venky VIII C
6 Nag X C
7 Pawan X C

```

```

enter id number to be deleted : 6

```

```

Database with the id : 6 is successfully deleted

```

```

options for the user

```

```

1. Display Elements
2. Add Elements
3. Update data
4. Delete record
5. Exit

```

```

enter your input : 1

```

```

1 Chandra X A
2 Mouli IX B
3 Chiru X B
4 Balayya IX B
5 Venky VIII C
7 Pawan X C

```

4.Github Link:

https://github.com/cmouli96/Simplelearn_Phase1_project