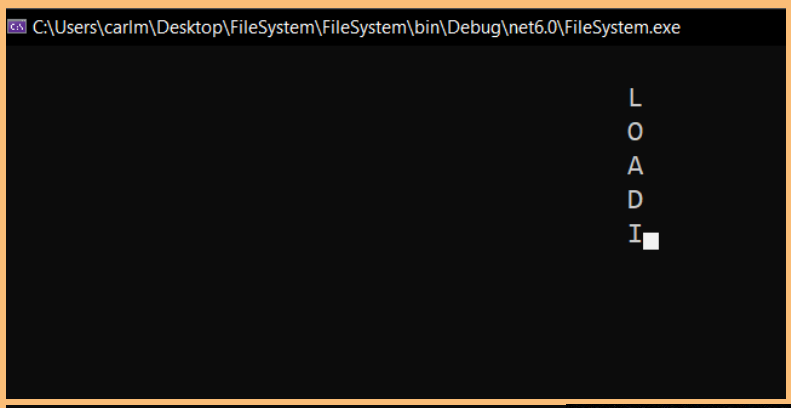
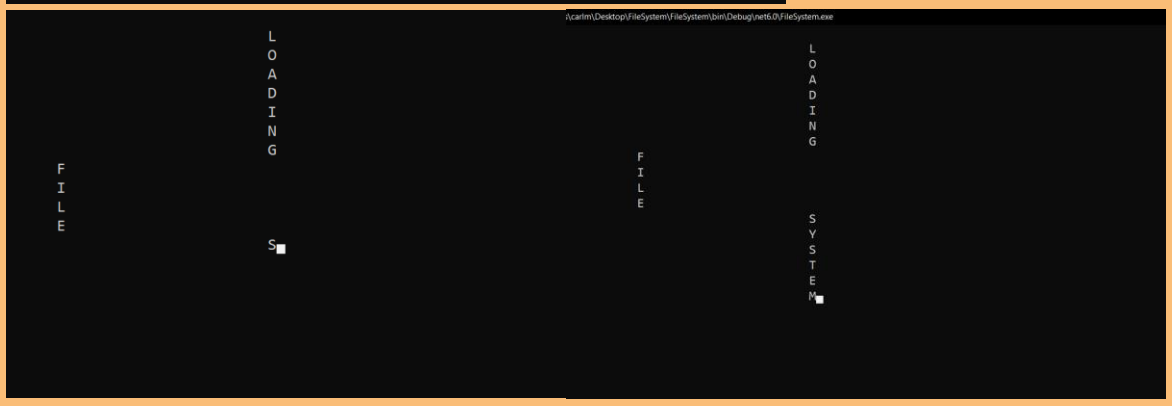
CODE DOCUMENTATION

```
0 references
public class FileAPP
{
    0 references
    public static void Main(string[] args)
    {
        Dots();
        int n = 1;
        do
        {

            if (n < 1)
            {
```

***We Initialize the main method in part of a class as C# is class based language, but you can choose to not use class system otherwise.**

***We used the Dots() method as I have formulated this to perform a simple yet edgy animation**





```
19          do
20          {
21
22
23              Console.Clear();
24              Console.WriteLine("\t\t\t\t\t" +
25               "***EXISTING FILES WITHIN THE DIRECTORY***");
26              var txtFiles = new DirectoryInfo("C:\\File System").GetFiles("*.txt");
27              for (int i = 0; i < txtFiles.Length; i++)
28              {
29                  var firsttxtFilename = txtFiles[i].Name;
30                  Console.WriteLine(firsttxtFilename);
31              }
32
33              Console.WriteLine("[1] Create A New File");
34              Console.WriteLine("[2] Open Existing File");
35              Console.WriteLine("[3] Manage Data");
36              Console.WriteLine("[4] Delete a File");
37              Console.WriteLine("[5] Exit");
38              Console.Write("Your Choice: ");
39              var choice1st = Console.ReadLine();
40
```

***We then initialized a do statement in line 19 for looping a process, code lines 23-40 is all about setting up the main ui: clearing the console so that every loop the MAIN MENU UI continues to be clean, then retrieving the existing files within the directory by using a for loop for that loops till all txt files are printed in the console as a display.**

***We set up a do statement earlier right? So in order for the sequence to complete we need to set a while statement**
that has a condition of If n is equal to 1(which was our default value) to continually do the process all over again.

```
1006
1007                }
1008                n--;
1009                if (n < 1)
1010                {
1011                    Console.Write("\n" + "Type YES/yes If You Still Want To Use The System: ");
1012                    string? v = Console.ReadLine();
1013                    string decision = v.ToUpper();
1014
1015                    if (decision == "YES")
1016                    {
1017                        n++;
1018                    }
1019                    else
1020                    {
1021                        Console.WriteLine("Thank you for using this Program :>")
1022                    }
1023                }
1024            } while (n == 1);
1025        }
```

***in code line 1008 we have decremented n so that the if statement line 1009 will be triggered as**
the default(n=1) will be decremented unto 0 that satisfies the condition of n assumed a lower number than 1,
its supposed default value.

If the user chooses to input yes then the n value would be incremented by 1 satisfying the do while condition, therefore
Allowing another loop to the very first process within the program. If the user on the other hand chooses to not input yes

Then the else statement would be triggered thanking the user for its time using the program then the whole program will
be exterminated as the condition of the whole do while loop will not be met.

```
43
44          switch (choice1st)...
1008        n--;
1009        if (n < 1)
1010        {
1011            Console.Write("\n" + "Type YES/yes If You Still Want To Use The System: ");
1012            string? v = Console.ReadLine();
1013            string decision = v.ToUpper();
1014
1015            if (decision == "YES")
1016            {
1017                n++;
1018            }
1019            else
1020            {
1021                Console.WriteLine("Thank you for using this Program :>");
1022            }
1023        }
1024    } while (n == 1);
1025
1026
```

Within lines 44-1024, I choose to switch statement as we need to actually rely on the user's choice which was
represented by lines 33-39 by a variable named <u>choice1st</u>

```
33          Console.WriteLine("[1] Create A New File");
34          Console.WriteLine("[2] Open Existing File");
35          Console.WriteLine("[3] Manage Data");
36          Console.WriteLine("[4] Delete a File");
37          Console.WriteLine("[5] Exit");
38          Console.Write("Your Choice: ");
39          var choice1st = Console.ReadLine();
```

Switch(choice1st) breakdown

Case "1" → Creates a new File

```
case "1":
    {
        Dots();
        Console.Write("What is your desired file name: ");
        string? filename = Console.ReadLine();
        string filepath = fileLocation(filename);
        if (!File.Exists(filepath))
        {
            Console.WriteLine("Do you want to Create This File?" + "\n Type YES/yes or NO/no");
            string createDecision = Console.ReadLine().ToUpper();
            fileMaker(createDecision, filepath);
            string textChoice3 = ("-------------------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastnam
            File.AppendAllText(filepath, textChoice3);
            Dots();

        }
        else
        {
            Console.WriteLine("\n\n\t\t\t\t\t\t\t" +
                "*File already Exists :<*\n\n");
            break;
        }
        break;
    }
```

Case “2” → Opens a new File.

```
    }
case "2":
    {
        Dots();
        Console.Write("Name of the file that you want to Open: ");
        string? filename1 = Console.ReadLine();
        var filepath1 = fileLocation(filename1);


        if (!File.Exists(filepath1))
        {
            Console.WriteLine("Do you want to Create This File?" + "\n Type YES/yes or NO/no");
            string createDecision = Console.ReadLine().ToUpper();
            fileMaker(createDecision, filepath1);
            string textChoice3 = ("-------------------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastname
            File.AppendAllText(filepath1, textChoice3);
            Dots();

        }


        StreamReader fileOpen = new StreamReader(filepath1);
        string fO = fileOpen.ReadToEnd();
        Console.WriteLine(fO);
        fileOpen.Close();

        break;
    }
```

Case “3” → Manage Data within the File

```
case "3":
    {
        Console.Write("What is your desired file name: ");
        string? modifyfile = Console.ReadLine();
        string Location = fileLocation(modifyfile);
        string Location1 = fileLocation(modifyfile);

        if (!File.Exists(Location))
        {
            Dots();
            Console.WriteLine("\n\t\t\t\t\t\t" + "***The Desired Text File is not existing!!!***");
            break;
        }
        else
        {
            Console.WriteLine("\n");
        }
        StringBuilder newString = new StringBuilder();
        Console.WriteLine("\n" + "[A]dd  [E]dit [D]elete  [S]ort  [F]ilter  e[X]it");

        string? manageInput = Console.ReadLine();

        Dots();
        string alltext = File.ReadAllText(Location);
```

IF manageInput == “A”

```
if (manageInput.ToUpper() == "A")
{
    Dots();
    Console.Clear();
    //ID PROCESSING
    Console.Write("ID NUMBER: ");
    string idProcess = Console.ReadLine();

    //Conditions of ID input
    if (!Regex.IsMatch(idProcess, @"^[0-9]+$")) //Assumes a situation that the inputted value is not numeric
    {
        Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
        break;
    }
    else if (idProcess.Length != 8) //VSU ID format is only 8 length value
    {

        Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed wrong format of an ID number****");
        break;
    }
    else
    {
        Console.WriteLine("");
    }

    //Creates the format of the final output of ID VALUE
    string idVal = idProcess.Substring(0, 2) + "-" + idProcess.Substring(2, 1) + "-" + idProcess.Substring(3, 5);

    //LAST NAME PROCESSING
    Console.Write("LAST NAME: ");
    string lastVal = Console.ReadLine().ToUpper();

    //Conditions of Last Name Output
    if (!Regex.IsMatch(lastVal, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to the
    {
        Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
        break;
    }
    else
    {
        Console.WriteLine("");
    }

    //FIRST NAME PROCESSING
    Console.Write("FIRST NAME: ");
    string firstVal = Console.ReadLine().ToUpper();

    //Conditions of First Name Output
    if (!Regex.IsMatch(firstVal, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to th
    {
        Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
        break;
    }
    else
    {
        Console.WriteLine("");
    }
```

```
//BIRTHDATE PROCESSING
Console.WriteLine("###BIRTHDATE###");

//year processing
bool priorDecision = false;
Console.Write("Year:");
string yearString = Console.ReadLine();
if (!Int32.TryParse(yearString, out int yearInt)) //assumes a situation that there is no equivalent int to the string value
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
    break;
}
else if (yearInt >= 2023 || yearInt <= 0) //Limits the input to a realistic year choice.
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid YEAR input****");
    break;
}
else
{
    Console.WriteLine("");
}
string yearVal = (yearInt % 100).ToString("00"); //takes the remainder of the modulo 100 process to make the format /00/ - /99/
bool finalDecision = leapOrNot(yearInt, priorDecision); //creates a condition for a value to hold if it is leap year or not, true or false return


//month processing
Console.Write("Month:");
string monthString = Console.ReadLine();
if (!Int32.TryParse(monthString, out int monthInt)) //assumes a situation that there is no equivalent int to the string value
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
    break;
}
else if (monthInt > 12 || monthInt <= 0) //Limits the input to a realistic month choice.
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid MONTH input****");
    break;
}
else
{
    Console.WriteLine("");
}
string monthVal = String.Format("{0:00}", monthInt); //Formats the month to have preceeding 0 if it lesser than 10


//day processing
int dayLimit = daysOfMonth(monthInt, finalDecision); //creates a condition for the day input to be tested if it is a genuine day within that month
                                                     //based if it either meets the leap year day count change of February or the regular count day coun

Console.Write("Day:");
string dayString = Console.ReadLine();
if (!Int32.TryParse(dayString, out int dayInt))
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
    break;
}
else if (dayInt > dayLimit || dayInt <= 0)
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid Day input****");
    break;
}
else
{
    Console.WriteLine("");
}
string dayVal = String.Format("{0:00}", dayInt); //Formats the day to have preceeding 0 if it lesser than 10
```

```
//birthdate value processing
string birthVal = monthVal + "/" + dayVal + "/" + yearVal;

//GENDER PROCESSING
Console.Write("GENDER: ");
string genderVal = Console.ReadLine().ToUpper(); //sets the supposed to be assumed value by genderVal variable to be uppercase
if (genderVal != "M" && genderVal != "F")
{
    Console.WriteLine("\t\t\t\t" + "Input is Invalid, inputed value is not either of the choices");
    break;
}
else
{
    Console.WriteLine("");
}
//



string numberingFormat = numberingLines(Location).ToString();

string[] dataOutput = dataInputs(idVal, lastVal, firstVal, birthVal, genderVal);

string dataOutputString = "\t" + dataOutput[0] + "\t" + dataOutput[1] + "\t" + dataOutput[2] + "\t" + "\t" + "\t" + dataOutput[3] + "\t" + "\t" + dataOutput[4] + "\t";


Console.WriteLine("-----------------------------------------------------------------------------------------------\r\n\tStudent ID\tLastname\t\tFirstname\t\tBirthDat

Console.WriteLine(dataOutputString);

Console.WriteLine("Enter V if you want to save your input");
string? saveOrNot = Console.ReadLine();
string decisionV = saveOrNot.ToUpper();

if (decisionV == "V")
{
    File.AppendAllText(Location, "\n" + numberingFormat + dataOutputString);

}
else
{
    Console.WriteLine("\n\t\t\t\t\t\t\t" + "Data was not Saved!");
}
```

IF manageInput == “E”

```
if (manageInput.ToUpper() == "E")//Start of edit
{
    bool parser = true;
    do
    {
        Console.Clear();

        string readEdit = File.ReadAllText(Location);
        Console.WriteLine(readEdit);
        string[] strings = Regex.Split(readEdit, Environment.NewLine);
        string Numbering;

        for (int p = 0; p < strings.Length; p++)
        {
            if (p == strings.Length - 1)
            {
                Console.Write("Which Record Line do you want to edit:");
                string lineChoice = Console.ReadLine();

                List<string> quotelist = File.ReadAllLines(Location).ToList();

                if (int.TryParse(lineChoice, out int lineChoiceInt) & quotelist.Count >= lineChoiceInt + 3) //controls that the user should input integer else the system de
                {
                    lineChoiceInt = lineChoiceInt + 2;
                    string[] fields = quotelist[lineChoiceInt].Split('\t');

                    showField(fields);


                    Console.WriteLine("[Input this to choose]Choose data to change: [I]ID-NUMBER, [L]LASTNAME, [F]FIRSTNAME, [B]BIRTHDATE, [G]GENDER, [A]ALL");
                    string valueDecision = Console.ReadLine().ToUpper();
```

Data to change → ID

```
switch (valueDecision.ToUpper())
{

    case "I":
        {
            Console.Write("New ID Value: ");
            string newID = Console.ReadLine();
            if (!Regex.IsMatch(newID, @"^[0-9]+$")) //Assumes a situation that the inputted value is not numeric
            {
                Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
                break;
            }
            else if (newID.Length != 8) //VSU ID format is only 8 length value
            {
                Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed wrong format of an ID number****");
                break;
            }
            else
            {
                Console.WriteLine("");
            }


            Console.WriteLine("----------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastname\t\tFirstname\t\tBirthDate\t\

            //formatting id
            string idVal = newID.Substring(0, 2) + "-" + newID.Substring(2, 1) + "-" + newID.Substring(3, 5);
            fields[1] = idVal;

            //formatting output
            string dataOutputStringID = fields[0] + "\t" + fields[1] + "\t" + fields[2] + "\t" + fields[3] + "\t" + "\t" + "\t" + fields[6] + "\t" + "\t" + fields[8] + "\t";
            string dataOutputHolder = dataOutputStringID;
            Console.WriteLine(dataOutputHolder);

            //change processing
            Console.WriteLine("\n\nDo you want to save this Change/s?, [1] if yes or [2] if no");
            string idChange = Console.ReadLine();
            if (idChange == "2")
            {
                Console.WriteLine("Changes has been Aborted!");
            }
            else if (idChange == "1")
            {
                //Actual value changing processs
                valueChange(Location, quotelist, lineChoiceInt, dataOutputStringID);
            }
            else
            {
                Console.WriteLine("Invalid Input");
                break;
            }



            break;
        }
```

Data to change → LastName

```
    }
case "L":
    {
        Console.Write("New LastName Value: ");
        string newlName = Console.ReadLine().ToUpper();
        //Conditions of Last Name Output
        if (!Regex.IsMatch(newlName, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to the condition that limits its values to be only letters
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
            break;
        }
        else
        {
            Console.WriteLine("");

        }

        Console.WriteLine("----------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastname\t\tFirstname\t\tBirthDate\t\tGe

        //formatting lastname
        fields[2] = newlName;

        //formatting output
        string dataOutputStringlastName = fields[0] + "\t" + fields[1] + "\t" + fields[2] + "\t" + fields[3] + "\t" + "\t" + "\t" + fields[6] + "\t" + "\t" + fields[8] + "\t";

        Console.WriteLine(dataOutputStringlastName);

        //change processing
        Console.WriteLine("\n\nDo you want to save this Change/s?, [1] if yes or [2] if no");
        string lastNChange = Console.ReadLine();
        if (lastNChange == "2")
        {
            Console.WriteLine("Changes has been Aborted!");
            break;
        }
        else if (lastNChange == "1")
        {
            //Actual value changing processs
            valueChange(Location, quotelist, lineChoiceInt, dataOutputStringlastName);
            break;
        }
        else
        {
            Console.WriteLine("Invalid Input");
            break;
        }
        break;
    }
```

Data to change → FirstName

```
case "F":
    {
        Console.Write("New FirstName Value: ");
        string newFName = Console.ReadLine().ToUpper();
        //Conditions of First Name Output
        if (!Regex.IsMatch(newFName, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to the condition that limits its values to be only letters
        {
            Console.WriteLine("\t\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        Console.WriteLine("-------------------------------------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastname\t\tFirstname\t\tBirthDate\t\t

        //formatting firstname
        fields[3] = newFName;

        //Formatting output
        string dataOutputStringfirstName = fields[0] + "\t" + fields[1] + "\t" + fields[2] + "\t" + fields[3] + "\t" + "\t" + "\t" + fields[6] + "\t" + "\t" + fields[8] + "\t";

        Console.WriteLine(dataOutputStringfirstName);

        //change processing
        Console.WriteLine("\n\nDo you want to save this Change/s?, [1] if yes or [2] if no");
        string firstNChange = Console.ReadLine();
        if (firstNChange == "2")
        {
            Console.WriteLine("Changes has been Aborted!");
        }
        else if (firstNChange == "1")
        {
            //Actual value changing processs
            valueChange(Location, quotelist, lineChoiceInt, dataOutputStringfirstName);
        }
        else
        {
            Console.WriteLine("Invalid Input");
        }
        break;

    }
case "B":
```

Data to change → BirthDate

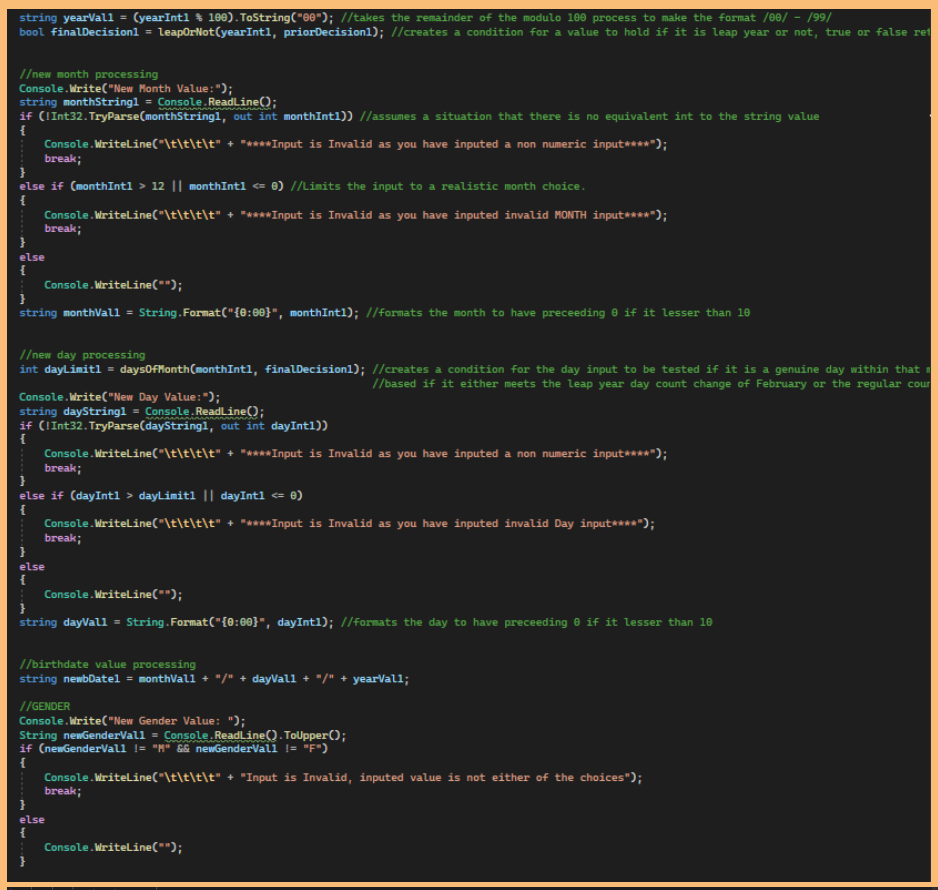
```
case "B":
    {
        //New BIRTHDATE PROCESSING
        Console.WriteLine("### NEW BIRTHDATE VALUE###");

        //new year processing
        bool priorDecision = false;
        Console.Write("New Year Value:");
        string yearString = Console.ReadLine();
        if (!Int32.TryParse(yearString, out int yearInt)) //assumes a situation that there is no equivalent int to the string value
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
            break;
        }
        else if (yearInt >= 2023 || yearInt <= 0) //Limits the input to a realistic year choice.
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid YEAR input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        string yearVal = (yearInt % 100).ToString("00"); //takes the remainder of the modulo 100 process to make the format /00/ - /99/
        bool finalDecision = leapOrNot(yearInt, priorDecision); //creates a condition for a value to hold if it is leap year or not, true or false


        //new month processing
        Console.Write("New Month Value:");
        string monthString = Console.ReadLine();
        if (!Int32.TryParse(monthString, out int monthInt)) //assumes a situation that there is no equivalent int to the string value
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
            break;
        }
        else if (monthInt > 12 || monthInt <= 0) //Limits the input to a realistic month choice.
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid MONTH input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        string monthVal = String.Format("{0:00}", monthInt); //formats the month to have preceeding 0 if it lesser than 10


        //new day processing
        int dayLimit = daysOfMonth(monthInt, finalDecision); //creates a condition for the day input to be tested if it is a genuine day within that
                                                             //based if it either meets the leap year day count change of February or the regular co
        Console.Write("New Day Value:");
        string dayString = Console.ReadLine();
        if (!Int32.TryParse(dayString, out int dayInt))
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
            break;
        }
        else if (dayInt > dayLimit || dayInt <= 0)
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid Day input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        string dayVal = String.Format("{0:00}", dayInt); //formats the day to have preceeding 0 if it lesser than 10
```

Data to change → Gender

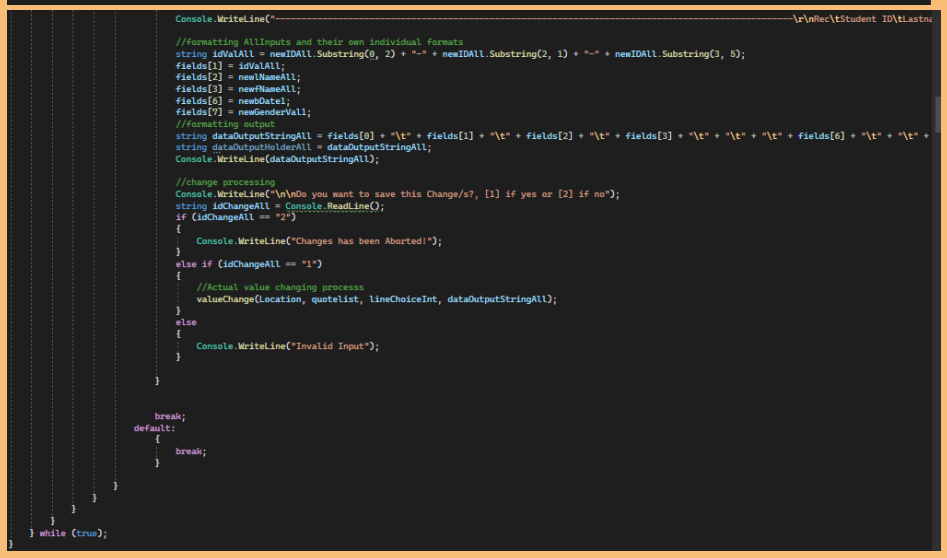
```
//New Gender Processing
case "G":
    {
        Console.Write("New Gender Value: ");
        String newGenderVal = Console.ReadLine().ToUpper();
        if (newGenderVal != "M" && newGenderVal != "F")
        {
            Console.WriteLine("\t\t\t\t" + "Input is Invalid, inputed value is not either of the choices");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        fields[8] = newGenderVal;

        //formatting output
        string dataOutputStringGender = fields[0] + "\t" + fields[1] + "\t" + fields[2] + "\t" + fields[3] + "\t" + "\t" + "\t" + fields[6] + "\t" + "\t" + fields[8] + "\t";

        Console.WriteLine(dataOutputStringGender);

        //change processing
        Console.WriteLine("\n\nDo you want to save this Change/s?, [1] if yes or [2] if no");
        string genderChange = Console.ReadLine();
        if (genderChange == "2")
        {
            Console.WriteLine("Changes has been Aborted!");
            break;
        }
        else if (genderChange == "1")
        {
            //Actual value changing processs
            valueChange(Location, quotelist, lineChoiceInt, dataOutputStringGender);
            break;
        }
        else
        {
            Console.WriteLine("Invalid Input");
            break;
        }
        break;
    }
case "A":
```

Data to change → All

```
case "A":
    {
        //ID
        Console.Write("New ID Value: ");
        string newIDAll = Console.ReadLine();

        //Conditions of Id Output
        if (!Regex.IsMatch(newIDAll, @"^[0-9]+$")) //Assumes a situation that the inputted value is not numeric
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
            break;
        }
        else if (newIDAll.Length != 8) //VSU ID format is only 8 length value
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed wrong format of an ID number****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }


        //LAST NAME
        Console.Write("New LastName Value: ");
        string newlNameAll = Console.ReadLine().ToUpper();

        //Conditions of Last Name Output
        if (!Regex.IsMatch(newlNameAll, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to the condition that limits its
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
            break;
        }
        else
        {
            Console.WriteLine("");

        }

        //FIRST NAME
        Console.Write("New FirstName Value: ");
        string newfNameAll = Console.ReadLine().ToUpper();

        //Conditions of First Name Output
        if (!Regex.IsMatch(newfNameAll, @"^[a-zA-Z]+$")) //assumes a situation that the input value is not able to meet to the condition that limits its
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non letter input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }

        //New BIRTHDATE PROCESSING
        Console.WriteLine("### NEW BIRTHDATE VALUE###");

        //new year processing
        bool priorDecision1 = false;
        Console.Write("New Year Value:");
        string yearString1 = Console.ReadLine();
        if (!Int32.TryParse(yearString1, out int yearInt1)) //assumes a situation that there is no equivalent int to the string value
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
            break;
        }
        else if (yearInt1 >= 2023 || yearInt1 <= 0) //Limits the input to a realistic year choice.
        {
            Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid YEAR input****");
            break;
        }
        else
        {
            Console.WriteLine("");
        }
        string yearAll = (yearInt1 % 100).ToString("00"); //takes the remainder of the module 100 process to give the format /00/ - /99/
```



```
string yearVal1 = (yearInt1 % 100).ToString("00"); //takes the remainder of the modulo 100 process to make the format /00/ - /99/
bool finalDecision1 = leapOrNot(yearInt1, priorDecision1); //creates a condition for a value to hold if it is leap year or not, true or false ret


//new month processing
Console.Write("New Month Value:");
string monthString1 = Console.ReadLine();
if (!Int32.TryParse(monthString1, out int monthInt1)) //assumes a situation that there is no equivalent int to the string value
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
    break;
}
else if (monthInt1 > 12 || monthInt1 <= 0) //Limits the input to a realistic month choice.
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid MONTH input****");
    break;
}
else
{
    Console.WriteLine("");
}
string monthVal1 = String.Format("{0:00}", monthInt1); //formats the month to have preceeding 0 if it lesser than 10


//new day processing
int dayLimit1 = daysOfMonth(monthInt1, finalDecision1); //creates a condition for the day input to be tested if it is a genuine day within that m
                                                        //based if it either meets the leap year day count change of February or the regular coun
Console.Write("New Day Value:");
string dayString1 = Console.ReadLine();
if (!Int32.TryParse(dayString1, out int dayInt1))
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed a non numeric input****");
    break;
}
else if (dayInt1 > dayLimit1 || dayInt1 <= 0)
{
    Console.WriteLine("\t\t\t\t" + "****Input is Invalid as you have inputed invalid Day input****");
    break;
}
else
{
    Console.WriteLine("");
}
string dayVal1 = String.Format("{0:00}", dayInt1); //formats the day to have preceeding 0 if it lesser than 10


//birthdate value processing
string newbDate1 = monthVal1 + "/" + dayVal1 + "/" + yearVal1;

//GENDER
Console.Write("New Gender Value: ");
String newGenderVal1 = Console.ReadLine().ToUpper();
if (newGenderVal1 != "M" && newGenderVal1 != "F")
{
    Console.WriteLine("\t\t\t\t" + "Input is Invalid, inputed value is not either of the choices");
    break;
}
else
{
    Console.WriteLine("");
}
```



```
                    Console.WriteLine("----------------------------------------------------------------------------------------------------\r\nRec\tStudent ID\tLastna

                    //formatting AllInputs and their own individual formats
                    string idValAll = newIDAll.Substring(0, 2) + "-" + newIDAll.Substring(2, 1) + "-" + newIDAll.Substring(3, 5);
                    fields[1] = idValAll;
                    fields[2] = newlNameAll;
                    fields[3] = newfNameAll;
                    fields[6] = newbDate1;
                    fields[7] = newGenderVal1;
                    //formatting output
                    string dataOutputStringAll = fields[0] + "\t" + fields[1] + "\t" + fields[2] + "\t" + fields[3] + "\t" + "\t" + "\t" + fields[6] + "\t" + "\t" +
                    string dataOutputHolderAll = dataOutputStringAll;
                    Console.WriteLine(dataOutputStringAll);

                    //change processing
                    Console.WriteLine("\n\nDo you want to save this Change/s?, [1] if yes or [2] if no");
                    string idChangeAll = Console.ReadLine();
                    if (idChangeAll == "2")
                    {
                        Console.WriteLine("Changes has been Aborted!");
                    }
                    else if (idChangeAll == "1")
                    {
                        //Actual value changing processs
                        valueChange(Location, quotelist, lineChoiceInt, dataOutputStringAll);
                    }
                    else
                    {
                        Console.WriteLine("Invalid Input");
                    }

                }


                break;
            default:
                {
                    break;
                }

        }
    }
}
} while (true);
}
```

IF manageInput == “D”

```
if (manageInput.ToUpper() == "D")
{
    var stringDel = File.ReadAllText(Location);
    Console.WriteLine(stringDel);
    string[] stringstoDel = Regex.Split(stringDel, Environment.NewLine);


    int decisionCounter = 1;

    for (int del = 0; del <= stringstoDel.Length; del++)
    {
        Console.WriteLine("What RECORD NUMBER do you want to delete?");
        string deleteDec = Console.ReadLine();
        Int32.TryParse(deleteDec, out int deleteDecInt);
        if (del + 2 < stringstoDel.Length+2)
        {
            File_DeleteLine(deleteDecInt + 3, Location);

        }
        else
        {
            Console.WriteLine("RECORD DOES NOT EXIST");
            break;
        }
        Console.WriteLine("Do you want to delete further?[Y]YES,[N]NO");
        String answer = Console.ReadLine().ToUpper();
        if (answer == "Y")
        {
            decisionCounter++;
        }
        else
        {
            break;
        }
    }
}
```

IF manageInput == “G”

IF manageInput == “S”

```
if (manageInput.ToUpper() == "S")
{
    bool parserR;
    do
    {
        var stringSort = File.ReadAllText(Location);
        Console.WriteLine(stringSort);


        List<string> listToDel = File.ReadAllLines(Location).ToList();


        for (int i = 1; i < listToDel.Count; i++)
        {
            if (i >= 3)
            {

                string[] fields = listToDel[i].Split('\t');

            }
        }
        Console.WriteLine("What value/values you want to be sorted?");

        Console.WriteLine("[R]RECORD NUMBER, [I]ID-NUMBER, [L]LASTNAME, [F]FIRSTNAME, [B]BIRTHDATE, [G]GENDER");
        string sortInput = Console.ReadLine().ToUpper();
        //rec
        if (sortInput == "R")
        {
            SortingRec(Location);
        }
        //id
        else if (sortInput == "I")
        {
            SortingID(Location);
        }
        //lastname
        else if (sortInput == "L")
        {
            SortinglName(Location);
        }
        //fname
        else if (sortInput == "F")
        {
            SortingfName(Location);
        }
        //bdate
        else if (sortInput == "B")
        {
            SortingbDate(Location);
        }
        //gender
        else if (sortInput == "G")
        {
            SortingGender(Location);
        }
```

```
            Console.WriteLine("Do you wish to do more Sorting Activity?, Input [Y] if yes; [N] if no");
            string SortRep1 = Console.ReadLine();
                if (SortRep1.ToUpper() == "Y")
                {
                    parserR = true;
                    Console.Clear();
                }
                else if (SortRep1.ToUpper() == "N")
                {
                    parserR = false;
                }
                else
                {
                    Console.WriteLine("Invalid Input");

                    break;
                }
        } while (parserR == true);
    }
```

IF manageinput == “F”

```
if (manageInput.ToUpper() == "F")
{
    Console.Clear();

    bool parserF = true;

    do
    {
        Dots();
        Console.WriteLine("\nWhat do you want to Filter: [I]ID ONLY, [L]LAST NAME ONLY, [F]FIRST NAME ONLY, [B] BIRTHDATE ONLY, [G] GENDER
        string filterDec = Console.ReadLine().ToUpper();

        if (filterDec == "I")
        {
            Filter(Location, 0, 1, "ID");
        }

        else if (filterDec == "L")
        {
            Filter(Location, 0, 2, "LastName");
        }

        else if (filterDec == "F")
        {
            Filter(Location, 0, 3, "FirstName");
        }
        else if (filterDec == "B")
        {
            Filter(Location, 0, 6, "BirthDate");
        }
        else if (filterDec == "G")
        {
            Filter(Location, 0, 8, "Gender");
        }

            Console.WriteLine("Do you wish to do more Filtering Activity?, Input [Y] if yes; [N] if no");
            string filRep1 = Console.ReadLine();
            if (filRep1.ToUpper() == "Y")
            {
                parserF = true;
            }
            else if (filRep1.ToUpper() == "N")
            {
                parserF = false;
            }
            else
            {
                break;
            }
    } while (parserF == true);
}
break;
```

IF manageinput == “X”

```
if(manageInput.ToUpper() == "X")
{
    break;
}
```

Case “4” → Delete a File

```
case "4":
    {
        Console.WriteLine("\n" + "What is the name of that you want to delete: ");
        string? fileDeleteChoice = Console.ReadLine();
        string fileDelete = fileLocation(fileDeleteChoice);

        if (File.Exists(fileDelete))
        {
            System.IO.File.Delete(fileDelete);

        }
        else
        {
            Console.WriteLine("\n" + "The File Does not Exist, therefore nothing is deleted!!!");
        }
        break;
    }
```

Case “5” → Exit

```
case "5":
    {
        Dots();

        Console.Write("\nThank you For Using Our System :D \n\n\n\n");
        System.Environment.Exit(1);
        break;
    }
```