```
/--- Program Class/Main Function ---/

using System;

namespace TFMS
{
    internal class Program
    {
        static void Main(string[] args)
        {
            bool showMenu = true;
            while (showMenu)
            {
                showMenu = MainMenu();
            }
        }

        private static bool MainMenu()
        {
            Console.Clear();
            Console.WriteLine("----------------------------------------");
            Console.WriteLine("Rainbow School Teacher Management System");
            Console.WriteLine("----------------------------------------");
            StoreController.GetStoreStats();
            Console.WriteLine("----------------------------------------");
            Console.WriteLine("Choose an option:");
            Console.WriteLine("1) View All Teachers");
            Console.WriteLine("2) Add a new Teacher");
            Console.WriteLine("3) Filter Teachers");
            Console.WriteLine("4) Update Teacher");
            Console.WriteLine("5) Delete Teacher");
            Console.WriteLine("6) Exit");
            Console.Write("\r\nSelect an option: ");
            switch (Console.ReadLine())
            {
                case "1":
                    Teacher.GetAllTeachers();
                    return true;
                case "2":
                    Teacher.AddNewTeacher();
                    return true;
                case "3":
                    Teacher.FilterTeachers();
                    return true;
                case "4":
                    Teacher.UpdateTeacher();
                    return true;
                case "5":
                    Teacher.DeleteTeacher();
                    return true;
                case "6":
                    return false;
                default:
                    return true;
            }
        }
    }
}
```

```
/--- Teacher Class ---/


using System;
using System.Collections.Generic;

namespace TFMS
{
    public static class Teacher
    {
        public static void AddNewTeacher()
        {
            string name = GetInput("\nEnter Teacher name");
            string id = GetInput("Enter ID");
            string cls = GetInput("Enter class");
            string sec = GetInput("Enter Section").ToUpper();

            string data = $"'{id}','{name}','{cls}'/'{sec}'";

            StoreController.SaveData(data);
            GetFormattedResponse("Teacher added successfully.");
        }

        public static void FilterTeachers() {
            string filter = GetInput("\nEnter filter parameter (ID, Name or
Class/Section)");

            List<string> filteredList = StoreController.ReadData(filter);

            Console.WriteLine();
            Console.WriteLine("Result:");

            GetFormattedTableOutput(filteredList);
        }

        public static void GetAllTeachers()
        {
            List<string> teachers = StoreController.ReadAllData();

            GetFormattedTableOutput(teachers);
        }

        public static void UpdateTeacher()
        {
            Console.WriteLine();

            int index = GetIndexInput("Enter index of Teacher to update");



            string indexData = StoreController.GetDataAtIndex(index);

            Console.WriteLine("\nSelected Teacher: ");
            GetFormattedRow(indexData, true);
            Console.WriteLine();

            string[] splitData = indexData.Split(',');
            string newID = GetInput("Enter new ID", splitData[1]);
```

```
            string newName = GetInput("Enter new name", splitData[2]);
            string[] splitCS = splitData[3].Split('/');
            string newClass = GetInput("Enter new Class", splitCS[0]);
            string newSec = GetInput("Enter new Section", splitCS[1]).ToUpper();

            string newData = $"'{newID}','{newName}','{newClass}'/'{newSec}'";
            StoreController.UpdateData(newData, index);
            GetFormattedResponse($"Updated Teacher: {newData}");
        }

        public static void DeleteTeacher()
        {
            Console.WriteLine();

            int index = GetIndexInput("Enter index of Teacher to delete");

            /*do
            {
                index = Convert.ToInt32(GetInput("Enter index of Teacher to
delete"));
                if(index >= StoreController.GetNumberOfRows())
                {
                    Console.WriteLine($"Please enter index within range (0 -
{(StoreController.GetNumberOfRows() - 1)})");
                }
            } while (index >= StoreController.GetNumberOfRows());*/

            GetFormattedRow(StoreController.GetDataAtIndex(index), true);

            StoreController.DeleteData(index);
            GetFormattedResponse("Teacher deleted successfully.");
        }

        private static string GetInput(string Prompt)
        {
            string Result;
            do
            {
                Console.Write(Prompt + ": ");
                Result = Console.ReadLine();
                if (string.IsNullOrEmpty(Result))
                {
                    Console.WriteLine("Empty input, please try again");
                }
                else if (Result.Contains(","))
                {
                    Console.WriteLine("Contains an invalid character. Try again.");
                }
            } while (string.IsNullOrEmpty(Result) | Result.Contains(","));
            return Result;
        }

        private static string GetInput(string Prompt, string PrevValue)
        {
            string Result;
            Console.Write(Prompt + ": ");
            Result = Console.ReadLine();
            if (string.IsNullOrEmpty(Result))
```

```
            {
                Console.WriteLine("Empty input, retaining previous value.");
                return PrevValue.Trim('\'');
            }
            else if (Result.Contains(","))
            {
                Console.WriteLine("Contains an invalid character. Try again.");
            }
            return Result;
        }

        private static int GetIndexInput(string Prompt)
        {
            int index;
            do
            {
                index = Convert.ToInt32(GetInput(Prompt));
                if (index >= StoreController.GetNumberOfRows() || index < 0)
                {
                    Console.WriteLine($"Please enter index within range (0 -
{(StoreController.GetNumberOfRows() - 1)})");
                }
            } while (index >= StoreController.GetNumberOfRows() || index < 0);
            return index;
        }

        private static void GetFormattedTableOutput(List<string> data)
        {
            Console.WriteLine();
            Console.WriteLine("-----------------------------------------------------
--------");
            Console.WriteLine("Index| ID        | Name                    |
Class/Sec");
            Console.WriteLine("-----------------------------------------------------
--------");
            foreach (string item in data)
            {
                GetFormattedRow(item);
            }
            Console.WriteLine("-----------------------------------------------------
--------");
            Console.WriteLine();
            Console.Write("Press any key to return to main menu... ");
            Console.ReadKey();
        }

        private static void GetFormattedRow(string data)
        {
            // string[] splitData = data.Split(',');
            string[] splitData = GetFormatedItem(data.Split(','));
            Console.WriteLine(String.Format("{0,4} | {1,-9} | {2,-24} | {3,15}",
splitData[0], splitData[1], splitData[2], splitData[3]));
        }

        private static void GetFormattedRow(string data, bool showBorder)
        {
            if (showBorder)
            {
```

```
                Console.WriteLine("-----------------------------------------------
------------");
                Console.WriteLine("Index| ID        | Name                      |
Class/Sec");
                Console.WriteLine("-----------------------------------------------
------------");
                string[] splitData = GetFormatedItem(data.Split(','));
                Console.WriteLine(String.Format("{0,4} | {1,-9} | {2,-24} | {3,15}",
splitData[0], splitData[1], splitData[2], splitData[3]));
                Console.WriteLine("-----------------------------------------------
------------");
            }
        }

        private static void GetFormattedResponse(string response)
        {
            Console.WriteLine();
            Console.Write(response);
            GetAllTeachers();
        }

        private static string[] GetFormatedItem(string[] item)
        {
            string[] itemObj = item;
            for (int i = 0; i < item.Length; i++)
            {
                if (item[i].Contains("\'"))
                {
                    if(item[i].Contains("/"))
                    {
                        itemObj[i] =
$"{item[i].Split('/')[0].Trim('\'')}/{item[i].Split('/')[1].Trim('\'')}";
                    }
                    itemObj[i] = item[i].Trim('\'');
                }
            }
            return itemObj;
        }

    }
}
```

```
/--- Store Controller Class ---/


using System;
using System.IO;
using System.Collections.Generic;

namespace TFMS
{
    public static class StoreController
    {
        private static readonly string FilePath;

        static StoreController()
        {
            string appDirPath = Directory.GetCurrentDirectory(); // Get the current
directory path of the application
            string dataDirPath = Path.Combine(appDirPath, "data"); // Append "data"
folder to the current directory path

            // Create a new "data" subfolder within the app directory
            DirectoryInfo info = Directory.CreateDirectory(dataDirPath);
            // Console.WriteLine(info);

            // Create a file name for the new text file to store records in the data
directory
            string dataFile = "data.txt";
            FilePath = Path.Combine(dataDirPath, dataFile);

            if (!File.Exists(FilePath))
            {
                Console.WriteLine("Creating new file...");
                try
                {
                    FileStream f = File.Create(FilePath);
                    f.Close();
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                }
            }
        }

        // Method to perform writing to the file
        public static void SaveData(string data)
        {
            if (FilePath != null) {
                try
                {
                    using (StreamWriter sw = File.AppendText(FilePath))
                    {
                        sw.WriteLine(data);
                    }
                }
```

```
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                }
            }
        }

        // Method to retrieve all records from the file
        public static List<string> ReadAllData()
        {
            if (File.Exists(FilePath))
            {
                try
                {
                    int counter = 0;
                    string line;

                    // Read the file, add index to each line and assign it to a
returning List item
                    StreamReader file = new StreamReader(FilePath);
                    List<string> data = new List<string>();
                    while ((line = file.ReadLine()) != null)
                    {
                        data.Add($"{counter},{line}");
                        counter++;
                    }

                    file.Close();
                    return data;
                }

                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                    return null;
                }

            } else
            {
                return null;
            }
        }

        // Method to retrieve filtered data based on the paramaters
        public static List<string> ReadData(string fiter)
        {
            if (File.Exists(FilePath))
            {
                List<string> data = new List<string>();
                try
                {
                    int counter = 0;
                    foreach (string line in File.ReadLines(FilePath))
                    {
                        if (line.Split(',')[2].Contains("/"))
                        {
                            string[] d = line.Split(',');
```

```
                            string trimmedLine =
$"{d[0]},{d[1]},{d[2].Split('\'')[1].Trim('\'')}{d[2].Split('\'')[3].Trim('\'')}";
                            if (trimmedLine.ToLower().Contains(fiter.ToLower()))
                            {

data.Add($"{counter},{d[0]},{d[1]},{d[2].Split('/')[0]}/{d[2].Split('/')[1]}");
                                counter++;
                            }
                            counter++;
                        }
                        else
                        {
                            counter++;
                        }

                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                    return null;
                }

                return data;
            } else
            {
                return null;
            }
        }

        // Method to retrieve data at a specific index
        public static string GetDataAtIndex(int index)
        {
            if (File.Exists(FilePath))
            {
                try
                {
                    string[] fileData = File.ReadAllLines(FilePath);
                    return $"{index},{fileData[index]}";
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                    return null;
                }
            } else { return null; }
        }

        // Method to update data at a specific index
        public static void UpdateData(string data, int index)
        {
            if (File.Exists(FilePath))
            {
                try
                {
                    string[] fileData = File.ReadAllLines(FilePath);
                    fileData[index] = data;
                    File.WriteAllLines(FilePath, fileData);
```

```
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                }
            }
        }

        // Method to delete data at a specific index
        public static void DeleteData(int index)
        {
            if (File.Exists(FilePath))
            {
                try
                {
                    string[] fileData = File.ReadAllLines(FilePath);
                    List<string> fileDataList = new List<string>(fileData);
                    fileDataList.RemoveAt(index);
                    File.WriteAllLines(FilePath, fileDataList);
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Exception Occured: {ex.Message}");
                }
            }
        }

        public static void GetStoreStats()
        {
            DateTime dt = File.GetLastWriteTime(FilePath);

            int rowCount = File.ReadAllLines(FilePath).Length;

            Console.WriteLine("Last updated at: {0}.", dt);
            Console.WriteLine("No. of rows(s): {0}", rowCount);
        }

        public static int GetNumberOfRows()
        {
            return File.ReadAllLines(FilePath).Length;
        }
    }
}
```