Name-Kundan Kumar

Registration Number : 12218104

DATE-12-01-2026

ASSESSMENT QUESTION 1

Case Study: Student Attendance Logger (C# File Handling)

---

1. Project Overview

Objective:
Develop a C# console application that logs student attendance details into a text file and allows viewing of stored records using file handling concepts.

---

2. Functional Requirements

A. File Operations

- Create a file named `attendance.txt` if it does not exist
- Append attendance entries in the format:
  `Date | StudentId | StudentName | Status`
- Read and display all attendance records

B. User Operations

- Add attendance record
- View attendance log
- Exit application

---

3. Interface Requirements

- Menu-driven console application

- File handling using `System.IO`
- Graceful handling of invalid inputs

---

4. Constraints

[ ] Use `StreamWriter` and `StreamReader`
[ ] Handle `IOException`
[ ] File path must be constant
[ ] No abrupt program termination

---

5. Sample Input

```
1
101
Ravi
Present
2
3
```

---

6. Sample Output

```
Attendance recorded.

--- Attendance Log ---
01/02/2026 | 101 | Ravi | Present
```

---

7. Test Cases

Test Case	Input	Expected Output
TC1	Add valid attendance	Record added to file
TC2	View attendance	Display file contents
TC3	View without file	"No attendance records found."
TC4	Invalid menu choice	"Invalid choice"

```
using System;
```

```
using System.IO;
using Microsoft.Win32.SafeHandles;
namespace StudentLoggerNamespace
{
  public class StudentLogger
  {
    private const string filePath = "attendance.txt";
    public static void WriteToFile(string data)
    {
      // string filePath = @"attendance.txt";
      try
      {
        using (FileStream fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write))
        using (StreamWriter streamWriter = new StreamWriter(fileStream))
        {
          streamWriter.WriteLine(data);
          streamWriter.Flush();
          streamWriter.Close();
          fileStream.Close();
        }
      }
      catch (UnauthorizedAccessException ioEx)
      {
        System.Console.WriteLine("Error: You do have permission to access the file.");
      }
```

```
  catch (IOException ex)
  {
    System.Console.WriteLine("I/O Error :", ex.Message);
  }
  catch (Exception ex)
  {
    System.Console.WriteLine("Unexpected Error : ", ex.Message);
  }
}
public static void AddStudentAttendance()
{
  System.Console.WriteLine("Enter the StudentId : ");
  int.TryParse(Console.ReadLine(), out int studentId);
  System.Console.WriteLine("Enter the Student Name: ");
  string studentName = Console.ReadLine();
  System.Console.WriteLine("Enter status (present/absent)");
  string status = Console.ReadLine();
  string LogEntry = $"{DateTime.Now.ToShortDateString()} | {studentId} | {studentName} | {status}";
  WriteToFile(LogEntry);

}

// public static void ViewStudentAttendance()
// {
//   if (!File.Exists(filePath))
```

```
//   {
//     Console.WriteLine("No records found.");
//     return;
//   }
//   Console.WriteLine("\n--- Attendance Records ---");
//   System.Console.WriteLine("Date | StudentId | Name | Status");
//   Console.WriteLine(File.ReadAllText(filePath));
// }
public static void ViewStudentAttendance()
{
  if (!File.Exists(filePath))
  {
    Console.ForegroundColor = ConsoleColor.Red;
    Console.WriteLine("\n[!] No records found.");
    Console.ResetColor();
    return;
  }

  Console.WriteLine("\n" + new string('=', 60));
  Console.ForegroundColor = ConsoleColor.Cyan;

  // Header with fixed widths: Date(12), ID(10), Name(20), Status(10)
  Console.WriteLine("{0,-12} | {1,-10} | {2,-20} | {3,-10}", "DATE", "ID", "NAME", "STATUS");
  Console.WriteLine(new string('-', 60));
  Console.ResetColor();
```

```
  string[] records = File.ReadAllLines(filePath);
  foreach (string record in records)
  {
    string[] parts = record.Split('|');
    if (parts.Length == 4)
    {
      // Trim to remove any accidental extra spaces
      Console.WriteLine("{0,-12} | {1,-10} | {2,-20} | {3,-10}",
          parts[0].Trim(), parts[1].Trim(), parts[2].Trim(), parts[3].Trim());
    }
  }
  Console.WriteLine(new string('=', 60) + "\n");
}

public static void Main()
{
  while (true)
  {
    System.Console.WriteLine("Enter the Choice: ");
    System.Console.WriteLine("1. Add Attentance Records.");
    System.Console.WriteLine("2. View Attentance Records.");
    System.Console.WriteLine("3. Exit.");
    int.TryParse(Console.ReadLine(), out int choice);
    switch (choice)
```

```
      {
        case 1:
          AddStudentAttendance();
          break;
        case 2:
          ViewStudentAttendance();
          break;
        case 3:
          System.Console.WriteLine("Ending...");
          return;
        default:
          System.Console.WriteLine("Invalid Choice .please Enter correct options");
          break;
      }
    }
  }
 }
}
```

ASSESSMENT QUESTION 2

Case Study: Application Error Logger (C# File Handling)

---

1. Project Overview

Objective:
Create a console utility that logs application errors into a file for debugging purposes.

---

2. Functional Requirements

- Create `error_log.txt`
- Append error messages with timestamp
- Display all logged errors
- Clear the error log

---

3. Interface Requirements

- Menu-driven application
- Exception handling using try-catch

---

4. Constraints

[ ] Append mode only
[ ] Safe file deletion
[ ] Use `System.IO.File`

---

5. Sample Input

```
1
NullReferenceException occurred
2
3
4
```

---

6. Sample Output

```
Error logged successfully.

01/02/2026 : NullReferenceException occurred

Log cleared successfully.
```

---

7. Test Cases

Test Case	Input	Expected Output
TC1	Log error	Error saved to file
TC2	View errors	Display all entries
TC3	Clear log	File deleted
TC4	View after clear	"No errors logged."

```
using System.IO;

using System;

using System.Runtime.InteropServices;

using System.Security.AccessControl;

namespace ErrorLoggerApp

{


 public class Program

 {

  private static string filePath = "error_log.txt";

  public static void Main()

  {

   int choice;

   do
```

```
{
  System.Console.WriteLine("Application Error Logger");
  System.Console.WriteLine("1. Log Error");
  System.Console.WriteLine("2. View All Error");
  System.Console.WriteLine("3. Clear All Error");
  System.Console.WriteLine("4. Exit");
  if (int.TryParse(Console.ReadLine(), out choice))
  {
    switch (choice)
    {
      case 1:
        LogError();
        break;
      case 2:
        ViewLog();
        break;
      case 3:
        DeleteLog();
        break;
      case 4:
        System.Console.WriteLine("Ending...");
        break;
      default:
        System.Console.WriteLine("Invalid Choice.");
        break;
```

```
      }
    }
  } while (choice != 4);
}
static void LogError()
{
  try
  {
    System.Console.Write("Enter the Log: ");
    string message = Console.ReadLine();
    string logEntry = $"{DateTime.Now:MM/dd/yyyy} : {message}{Environment.NewLine}";
    File.AppendAllText(filePath, logEntry);
    System.Console.WriteLine("Error Logged Successfully.");
  }
  catch (UnauthorizedAccessException ex)
  {
    System.Console.WriteLine($"Ex: {ex.Message}");
  }
  catch (Exception ex)
  {
    System.Console.WriteLine($"Ex: {ex.Message}");
  }
}
static void ViewLog()
{
```

```
try
{

  if (File.Exists(filePath))
  {
    string LogEntry = File.ReadAllText(filePath);
    if (string.IsNullOrWhiteSpace(LogEntry))
    {
      System.Console.WriteLine("No Error Logged");
    }
    else
    {
      System.Console.WriteLine($"\nLogged Entries\n{LogEntry}");
    }
  }
  else
  {
    System.Console.WriteLine("No LogError Exits");
  }


}
catch (UnauthorizedAccessException ex)
{
  System.Console.WriteLine($"Ex: {ex.Message}");
}
```

```
    catch (Exception ex)
    {
      System.Console.WriteLine($"Ex: {ex.Message}");
    }
  }
  static void DeleteLog()
  {
    try
    {
      if (File.Exists(filePath))
      {
        File.Delete(filePath);
        System.Console.WriteLine("Log Cleared Successfully.");
      }
      else
      {
        System.Console.WriteLine("Log Does not Exists");
      }
    }
    catch (Exception ex)
    {
      System.Console.WriteLine("Ex: " + ex.Message);
    }
  }
}
```

}

ASSESSMENT QUESTION 3

Case Study: Employee Record Manager (File IO + Serialization)

---

1. Scenario Overview

Develop a console application to store employee records permanently using JSON serialization.

---

2. Technical Requirements

- File name: `employees.json`
- Serialize and deserialize employee objects
- Persist data across executions

---

3. Data Model

```
public class Employee
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Department { get; set; }
}
```

---

4. Sample Input

```
101
Ravi
IT
102
Anita
HR
```

---

5. Expected Output

```
101 - Ravi - IT
102 - Anita - HR
```

---

6. Test Cases

Test Case	Scenario	Expected Result
TC1	Add employees	JSON file created
TC2	Restart app	Data loaded from file
TC3	Empty file	Start with empty list
TC4	Invalid JSON	Error handled gracefully

---

7. C# Implementation

```
using System;

using System.Text.Json.Serialization;

using System.IO;

using System.Text.Json;

namespace EmployeeRecordManagerApp

{

  public class Program

  {

    private static string filePath = "employees.json";

    public static void Main()

    {
```

```
List<Employee> employees = LoadEmployee();
int choice;
do
{
  Console.WriteLine("\n1. Add Employee\n2. View Employees\n3. Save & Exit");
  int.TryParse(Console.ReadLine(), out choice);
  switch (choice)
  {
    case 1:
      Console.Write("Enter ID: ");
      int id = int.Parse(Console.ReadLine());
      Console.Write("Enter Name: ");
      string name = Console.ReadLine();
      Console.Write("Enter Department: ");
      string dept = Console.ReadLine();
      employees.Add(new Employee { Id = id, Name = name, Department = dept });
      break;
    case 2:
      if (employees.Count == 0) { System.Console.WriteLine("No Record Found"); }
      employees.ForEach(e => System.Console.WriteLine($"{e.Id} - {e.Name} - {e.Department}"));
      break;
    case 3:
      SaveEmployees(employees);
      break;
    default:
```

```
        System.Console.WriteLine("Invalid Choice.");
        break;
    }
  } while (choice != 3);
}
public static List<Employee> LoadEmployee()
{
  try
  {
    if (!File.Exists(filePath))
    {
      System.Console.WriteLine("JSON file created");
      return new List<Employee>();
    }
    System.Console.WriteLine("Data loaded from file");
    string jsonString = File.ReadAllText(filePath);
    return JsonSerializer.Deserialize<List<Employee>>(jsonString) ?? new List<Employee>();
  }
  catch (JsonException)
  {
    System.Console.WriteLine("Warning Employee Data File is Corrupted . Start with empty list.");
    return new List<Employee>();
  }
}
```

```
    public static void SaveEmployees(List<Employee> empList)
    {
      string jsonString = JsonSerializer.Serialize(empList, new JsonSerializerOptions { WriteIndented = true
});
      File.WriteAllText(filePath, jsonString);
    }
  }
  public class Employee
  {
    public int Id { get; set; }
    public string Name { get; set; }
    public string Department { get; set; }
  }

}
```