Prescient Securities

.NET Developer Assessment

Lesedi Vinson Marokana

Assessment 1

Question 1:

```
using System;
using System.IO;
using System.Net;
using Newtonsoft.Json;
using System.Collections.Generic;

namespace FileDownloader
{
   class Program
   {
      static string downloadFolderPath = @"C:\DownloadedFiles\";
      static string downloadedFilesJsonPath = @"C:\DownloadedFiles\downloadedFiles.json";

      static void Main(string[] args)
      {
         // Create download folder if it doesn't exist
         if (!Directory.Exists(downloadFolderPath))
            Directory.CreateDirectory(downloadFolderPath);

         // Load downloaded files list from JSON
         List<string> downloadedFiles = LoadDownloadedFilesList();

         // Download files
         DownloadFiles(downloadedFiles);
```

```
        // Save downloaded files list to JSON
        SaveDownloadedFilesList(downloadedFiles);

        Console.WriteLine("Download completed.");
    }

    static void DownloadFiles(List<string> downloadedFiles)
    {
        string baseUrl = "https://clientportal.jse.co.za/downloadable-
files?RequestNode=/YieldX/Derivatives/Docs_DMTM&year=2023";

        using (WebClient client = new WebClient())
        {
            try
            {
                string html = client.DownloadString(baseUrl);
                // Parse the HTML or use appropriate library to extract file links


                // Example: Extracting file links from HTML
                List<string> fileLinks = ExtractFileLinks(html);

                foreach (string fileLink in fileLinks)
                {
                    string fileName = Path.GetFileName(fileLink);
                    if (!downloadedFiles.Contains(fileName))
                    {
                        client.DownloadFile(fileLink, Path.Combine(downloadFolderPath, fileName));
                        downloadedFiles.Add(fileName);
                    }
```

```
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error downloading files: " + ex.Message);
        }
    }
}

static List<string> ExtractFileLinks(string html)
{
    List<string> fileLinks = new List<string>();
    fileLinks.Add("https://example.com/file1.xls");
    fileLinks.Add("https://example.com/file2.xls");
    return fileLinks;
}

static List<string> LoadDownloadedFilesList()
{
    // Load downloaded files list from JSON file
    if (File.Exists(downloadedFilesJsonPath))
    {
        string json = File.ReadAllText(downloadedFilesJsonPath);
        return JsonConvert.DeserializeObject<List<string>>(json);
    }
    else
    {
        return new List<string>();
    }
}
```

```
        static void SaveDownloadedFilesList(List<string> downloadedFiles)
        {
            // Save downloaded files list to JSON file
            string json = JsonConvert.SerializeObject(downloadedFiles);
            File.WriteAllText(downloadedFilesJsonPath, json);
        }
    }
}
```

Question 2:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using Microsoft.Office.Interop.Excel;

namespace ExcelToDatabase
{
    class Program
    {
        static void Main(string[] args)
        {
            string downloadFolderPath = @"C:\DownloadedFiles\";

            // Load downloaded files list
            List<string> downloadedFiles = LoadDownloadedFilesList();

            // Process each file
            foreach (string fileName in downloadedFiles)
            {
                string filePath = Path.Combine(downloadFolderPath, fileName);
```

```
        if (File.Exists(filePath))
        {
            ProcessExcelFile(filePath);
            // Mark file as processed
            MarkFileAsProcessed(fileName);
        }
    }

    Console.WriteLine("Processing completed.");
}

static void ProcessExcelFile(string filePath)
{
    Application excelApp = new Application();
    Workbook workbook = excelApp.Workbooks.Open(filePath);
    Worksheet worksheet = workbook.Sheets[1];
    Range range = worksheet.UsedRange;

    // Assuming the data starts from row 2 and columns contain contract details
    int rowCount = range.Rows.Count;
    for (int i = 2; i <= rowCount; i++)
    {
        string contract = ((Range)range.Cells[i, 1]).Text; // Column A
        string expiryDate = ((Range)range.Cells[i, 3]).Text; // Column C
        string classification = ((Range)range.Cells[i, 4]).Text; // Column D
        string field1 = ((Range)range.Cells[i, 5]).Text; // Example for Column E
        string field2 = ((Range)range.Cells[i, 6]).Text; // Example for Column F

        // Save contract details to database
        SaveToDatabase(contract, expiryDate, classification, field1, field2);
    }
```

```
        // Close Excel objects
        workbook.Close(false);
        excelApp.Quit();
        ReleaseObject(worksheet);
        ReleaseObject(workbook);
        ReleaseObject(excelApp);
    }

    static void SaveToDatabase(string contract, string expiryDate, string classification, string field1, string field2)
    {
        // Connect to your SQL database
        string connectionString = "Your_Connection_String";
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            // Insert data into DailyMTM table
            string query = "INSERT INTO DailyMTM (Contract, ExpiryDate, Classification, Field1, Field2) VALUES (@Contract, @ExpiryDate, @Classification, @Field1, @Field2)";
            SqlCommand command = new SqlCommand(query, connection);
            command.Parameters.AddWithValue("@Contract", contract);
            command.Parameters.AddWithValue("@ExpiryDate", expiryDate);
            command.Parameters.AddWithValue("@Classification", classification);
            command.Parameters.AddWithValue("@Field1", field1);
            command.Parameters.AddWithValue("@Field2", field2);
            command.ExecuteNonQuery();
        }
    }

    static void MarkFileAsProcessed(string fileName)
```

```
    {
        // Update JSON file to mark file as processed
        // Implement as per your file tracking mechanism
    }

    static List<string> LoadDownloadedFilesList()
    {
        // Load downloaded files list from JSON file
        // Implement as per your file tracking mechanism
        return new List<string>(); // Placeholder
    }

    // Release COM objects to avoid memory leaks
    static void ReleaseObject(object obj)
    {
        try
        {
            System.Runtime.InteropServices.Marshal.ReleaseComObject(obj);
            obj = null;
        }
        catch (Exception ex)
        {
            obj = null;
            Console.WriteLine("Exception Occurred while releasing object " + ex.ToString());
        }
        finally
        {
            GC.Collect();
        }
    }
}
```

}

Assessment 2:

```
CREATE PROCEDURE [dbo].[SP_Total_Contracts_Traded_Report]
    @DateFrom DATE,
    @DateTo DATE
AS
BEGIN
    SET NOCOUNT ON;

    -- Temporary table to store intermediate results
    CREATE TABLE #TempContracts
    (
        [File Date] DATE,
        [Contract] NVARCHAR(100),
        [Contracts Traded] INT
    );

    -- Populate the temporary table with data
    INSERT INTO #TempContracts ([File Date], [Contract], [Contracts Traded])
    SELECT
        [File Date],
        [Contract],
        SUM([Contracts Traded]) AS [Contracts Traded]
    FROM
        DailyMTM
    WHERE
        [File Date] BETWEEN @DateFrom AND @DateTo
    GROUP BY
        [File Date], [Contract];
```

```
    -- Calculate total contracts traded for the specified date range
    DECLARE @TotalContractsTraded INT;
    SELECT @TotalContractsTraded = SUM([Contracts Traded]) FROM #TempContracts;

    -- Return the report
    SELECT
        [File Date],
        [Contract],
        [Contracts Traded],
        CAST(([Contracts Traded] * 100.0 / NULLIF(@TotalContractsTraded, 0)) AS DECIMAL(10, 2)) AS [% Of Total Contracts Traded]
    FROM
        #TempContracts
    WHERE
        [Contracts Traded] > 0;

    -- Drop the temporary table
    DROP TABLE #TempContracts;
END
EXEC [dbo].[SP_Total_Contracts_Traded_Report] @DateFrom = N'2021-01-04', @DateTo = N'2021-01-05';
```