using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.RegularExpressions;

class Program

{

    // Method to validate the username

    static bool ValidateUsername(string username, out int uppercaseCount, out int lowercaseCount, out int digitCount, out int underscoreCount)

    {

        uppercaseCount = 0;

        lowercaseCount = 0;

        digitCount = 0;

        underscoreCount = 0;

        // Username should start with a letter and should only contain letters, digits, and underscores

        if (!Regex.IsMatch(username, @"^[a-zA-Z]"))

        {

            Console.WriteLine("Username must start with a letter.");

            return false;

        }

        // Username length should be between 5 and 15 characters

        if (username.Length < 5 || username.Length > 15)

        {

            Console.WriteLine("Username length must be between 5 and 15 characters.");

            return false;

        }

        // Validate that the username only contains allowed characters (letters, digits, underscores)

        if (!Regex.IsMatch(username, @"^[a-zA-Z0-9\_]+$"))

        {

            Console.WriteLine("Username can only contain letters, digits, and underscores.");

            return false;

        }

        // Count uppercase, lowercase, digits, and underscores

        foreach (char c in username)

        {

            if (char.IsUpper(c)) uppercaseCount++;

            if (char.IsLower(c)) lowercaseCount++;

            if (char.IsDigit(c)) digitCount++;

            if (c == '\_') underscoreCount++;

        }

        return true;

    }

    // Method to generate a secure random password

    static string GeneratePassword()

    {

        Random rand = new Random();

        string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

        string lower = "abcdefghijklmnopqrstuvwxyz";

        string digits = "0123456789";

        string specialChars = "!@#$%^&\*";

        // Generate at least 2 uppercase, 2 lowercase, 2 digits, 2 special characters

        string password = new string(Enumerable.Range(0, 2).Select(\_ => upper[rand.Next(upper.Length)]).ToArray()) +

                          new string(Enumerable.Range(0, 2).Select(\_ => lower[rand.Next(lower.Length)]).ToArray()) +

                          new string(Enumerable.Range(0, 2).Select(\_ => digits[rand.Next(digits.Length)]).ToArray()) +                          new string(Enumerable.Range(0, 2).Select(\_ => specialChars[rand.Next(specialChars.Length)]).ToArray());

        // Fill up the rest of the password with random characters

        string allChars = upper + lower + digits + specialChars;

        password += new string(Enumerable.Range(0, 12 - password.Length).Select(\_ => allChars[rand.Next(allChars.Length)]).ToArray());

        return new string(password.OrderBy(c => rand.Next()).ToArray()); // Shuffle password

    }

    // Method to check password strength

    static string CheckPasswordStrength(string password)

    {

        int score = 0;

        if (password.Length >= 12) score++;

        if (password.Any(char.IsUpper)) score++;

        if (password.Any(char.IsLower)) score++;

        if (password.Any(char.IsDigit)) score++;

        if (password.Any(c => "!@#$%^&\*".Contains(c))) score++;

        if (score <= 2) return "Weak";

        if (score == 3) return "Medium";

        return "Strong";

    }

    static void Main(string[] args)

    {

        Console.WriteLine("Enter usernames (separated by commas):");

        string input = Console.ReadLine();

        var usernames = input.Split(',').Select(u => u.Trim()).ToList();

      int totalUsernames = usernames.Count;

        int validUsernames = 0;

        int invalidUsernames = 0;

        List<string> invalidList = new List<string>();

        // List to hold the username and password details

        List<string> outputLines = new List<string>();

        foreach (var username in usernames)

        {

            int uppercaseCount, lowercaseCount, digitCount, underscoreCount;

            if (ValidateUsername(username, out uppercaseCount, out lowercaseCount, out digitCount, out underscoreCount))

            {

                validUsernames++;

                string password = GeneratePassword();

                string passwordStrength = CheckPasswordStrength(password);

                outputLines.Add($"{username} - Valid");

                outputLines.Add($"   Letters: {uppercaseCount + lowercaseCount} (Uppercase: {uppercaseCount}, Lowercase: {lowercaseCount}), Digits: {digitCount}, Underscores: {underscoreCount}");

                outputLines.Add($"   Generated Password: {password} (Strength: {passwordStrength})");

            }

            else

            {

                invalidUsernames++;

                invalidList.Add(username);

            }

        }

        // Show Summary in Console

        Console.WriteLine("\nValidation Results:");

        foreach (var line in outputLines)

        {

            Console.WriteLine(line);

        }

        Console.WriteLine("\nSummary:");

        Console.WriteLine($"- Total Usernames: {totalUsernames}");

        Console.WriteLine($"- Valid Usernames: {validUsernames}");

        Console.WriteLine($"- Invalid Usernames: {invalidUsernames}");

        if (invalidUsernames > 0)

        {

            Console.WriteLine("\nInvalid Usernames: " + string.Join(", ", invalidList));

            Console.WriteLine("\nDo you want to retry invalid usernames? (y/n):");

            string retryChoice = Console.ReadLine().ToLower();

            if (retryChoice == "y")

            {

                Console.WriteLine("Enter invalid usernames:");

                string retryInput = Console.ReadLine();

                var retryUsernames = retryInput.Split(',').Select(u => u.Trim()).ToList();

                foreach (var retryUsername in retryUsernames)

                {

                    int u, l, d, uCount;

                    if (ValidateUsername(retryUsername, out u, out l, out d, out uCount))

                    {

                        validUsernames++;

                        string retryPassword = GeneratePassword();

                        string retryPasswordStrength = CheckPasswordStrength(retryPassword);

                        outputLines.Add($"{retryUsername} - Valid");

                        outputLines.Add($"   Letters: {u + l} (Uppercase: {u}, Lowercase: {l}), Digits: {d}, Underscores: {uCount}");

                        outputLines.Add($"   Generated Password: {retryPassword} (Strength: {retryPasswordStrength})");

                    }

                }

            }

        }

      // Write results to file

        string filePath = "UserDetails.txt";

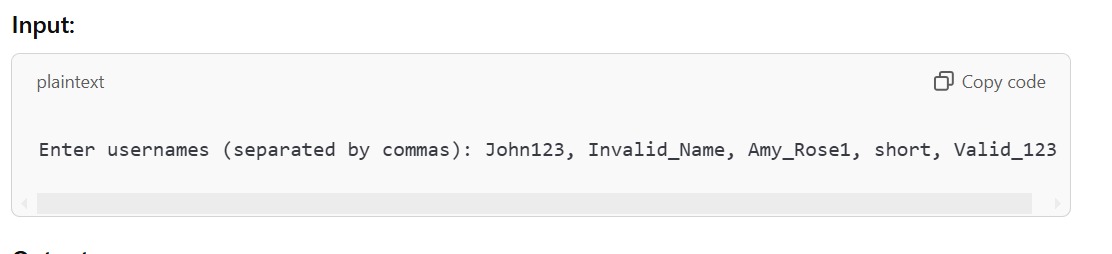
        File.WriteAllLines(filePath, outputLines);

        Console.WriteLine("\nResults saved to UserDetails.txt");

    }

}

**INPUT**

****

**OUTPUT**

