Logo, company name

Description automatically generated

**COMSATS University Islamabad (CUI)**

**Lab Terminal**

**Submitted to: Sir Bilal Bukhari**

**Submitted By: Sohaib Safeer**

**Reg No: FA21-BCS-054**

**Course: Topics in Computer Science**

**Date: 3 jan, 2025**

**Question 6 : Write a C# program using Regular Expressions to validate usernames, generate secure passwords, and provide additional features such as batch processing, saving results to a file, and evaluating password strength.**

**using System;**

**using System.Collections.Generic;**

**using System.IO;**

**using System.Linq;**

**using System.Text.RegularExpressions;**

**class Program**

**{**

**static void Main()**

**{**

**// Get usernames from the user**

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

**string input = Console.ReadLine();**

**string[] usernames = input.Split(',');**

**// List to store the validation results**

**List<string> validationResults = new List<string>();**

**int validCount = 0, invalidCount = 0;**

**foreach (var username in usernames)**

**{**

**string trimmedUsername = username.Trim();**

**// Part 1: Username Validation**

**if (ValidateUsername(trimmedUsername, out string validationMessage))**

**{**

**validCount++;**

**var (upper, lower, digits, underscores) = CountUsernameComponents(trimmedUsername);**

**string password = GeneratePassword();**

**string passwordStrength = CheckPasswordStrength(password);**

**validationResults.Add($"{trimmedUsername} - Valid\nLetters: {upper + lower} (Uppercase: {upper}, Lowercase: {lower}), Digits: {digits}, Underscores: {underscores}\nGenerated Password: {password} (Strength: {passwordStrength})\n");**

**}**

**else**

**{**

**invalidCount++;**

**validationResults.Add($"{trimmedUsername} - Invalid ({validationMessage})\n");**

**}**

**}**

**// Part 2: Display Validation Results**

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

**foreach (var result in validationResults)**

**{**

**Console.WriteLine(result);**

**}**

**// Part 3: Display Summary of Valid and Invalid Usernames**

**Console.WriteLine("\nSummary:");**

**Console.WriteLine($"Total Usernames: {validCount + invalidCount}");**

**Console.WriteLine($"Valid Usernames: {validCount}");**

**Console.WriteLine($"Invalid Usernames: {invalidCount}");**

**// Part 3: Save results to a file**

**SaveResultsToFile(validationResults, validCount, invalidCount);**

**// Part 4: Retry Invalid Usernames**

**if (invalidCount > 0)**

**{**

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

**string retryResponse = Console.ReadLine();**

**if (retryResponse.ToLower() == "y")**

**{**

**Console.Write("Enter invalid usernames: ");**

**string retryUsernames = Console.ReadLine();**

**string[] retryList = retryUsernames.Split(',');**

**foreach (var retry in retryList)**

**{**

**string trimmedRetry = retry.Trim();**

**if (ValidateUsername(trimmedRetry, out string retryMessage))**

**{**

**validCount++;**

**var (upper, lower, digits, underscores) = CountUsernameComponents(trimmedRetry);**

**string password = GeneratePassword();**

**string passwordStrength = CheckPasswordStrength(password);**

**validationResults.Add($"{trimmedRetry} - Valid\nLetters: {upper + lower} (Uppercase: {upper}, Lowercase: {lower}), Digits: {digits}, Underscores: {underscores}\nGenerated Password: {password} (Strength: {passwordStrength})\n");**

**}**

**else**

**{**

**invalidCount++;**

**validationResults.Add($"{trimmedRetry} - Invalid ({retryMessage})\n");**

**}**

**}**

**SaveResultsToFile(validationResults, validCount, invalidCount);**

**}**

**}**

**Console.WriteLine("\nProcessing complete.");**

**}**

**static bool ValidateUsername(string username, out string message)**

**{**

**// Username rules**

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

**{**

**message = "Username length must be between 5 and 15 characters";**

**return false;**

**}**

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

**{**

**message = "Username must start with a letter";**

**return false;**

**}**

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

**{**

**message = "Username can only contain letters, numbers, and underscores";**

**return false;**

**}**

**message = "";**

**return true;**

**}**

**static (int upper, int lower, int digits, int underscores) CountUsernameComponents(string username)**

**{**

**int upper = username.Count(c => char.IsUpper(c));**

**int lower = username.Count(c => char.IsLower(c));**

**int digits = username.Count(c => char.IsDigit(c));**

**int underscores = username.Count(c => c == '\_');**

**return (upper, lower, digits, underscores);**

**}**

**static string GeneratePassword()**

**{**

**Random rand = new Random();**

**string upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";**

**string lower = "abcdefghijklmnopqrstuvwxyz";**

**string digits = "0123456789";**

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

**string allChars = upper + lower + digits + special;**

**string password = "";**

**password += upper[rand.Next(upper.Length)];**

**password += upper[rand.Next(upper.Length)];**

**password += lower[rand.Next(lower.Length)];**

**password += lower[rand.Next(lower.Length)];**

**password += digits[rand.Next(digits.Length)];**

**password += digits[rand.Next(digits.Length)];**

**password += special[rand.Next(special.Length)];**

**password += special[rand.Next(special.Length)];**

**for (int i = 8; i < 12; i++)**

**{**

**password += allChars[rand.Next(allChars.Length)];**

**}**

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

**}**

**static string CheckPasswordStrength(string password)**

**{**

**if (password.Length >= 12 &&**

**password.Any(char.IsUpper) &&**

**password.Any(char.IsLower) &&**

**password.Any(char.IsDigit) &&**

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

**{**

**return "Strong";**

**}**

**else if (password.Length >= 8)**

**{**

**return "Medium";**

**}**

**else**

**{**

**return "Weak";**

**}**

**}**

**static void SaveResultsToFile(List<string> results, int validCount, int invalidCount)**

**{**

**string filePath = "UserDetails.txt";**

**using (StreamWriter sw = new StreamWriter(filePath))**

**{**

**sw.WriteLine("Validation Results:");**

**foreach (var result in results)**

**{**

**sw.WriteLine(result);**

**}**

**sw.WriteLine("\nSummary:");**

**sw.WriteLine($"Total Usernames: {validCount + invalidCount}");**

**sw.WriteLine($"Valid Usernames: {validCount}");**

**sw.WriteLine($"Invalid Usernames: {invalidCount}");**

**}**

**}**

**}**