# MIS2402 – Assignment 07: Checking Password Strength

## Scenario

In this assignment you will use at least one of JavaScript’s string functions to check the strength of a password.

In our imaginary environment, passwords must be at least 10 characters long, contain ***exactly one*** exclamation point (!), and contain a combination of upper-case letters, lower-case letters, and digits. No other characters are allowed.

## Step‑by‑Step Instructions

1. **Log in to the class server.** Use the username and password your instructor gave you. The server’s address is misdemo.temple.edu. After you log in, click the **New terminal console** icon in Bitvise.
2. **Download and prepare the assignment.** At the prompt, type each of these commands exactly as shown:

cd wwwroot

wget https://misdemo.temple.edu/assignments/assignment07.zip

unzip assignment07.zip

rm assignment07.zip

cd assignment07

1. **Open the start file in your browser.** Replace <yourusername> with your own account name:

* https://misdemo.temple.edu/<yourusername>/assignment07/validator2.html
* You should see a form with a single text box and a “Check Password Strength” button. If so, you are ready to begin. (Note, typically we would not allow a user to see their password in clear text on the screen, but this activity is more about using JavaScript strings than any sort of attempt to make this realistic.)

1. **Open the HTML file for editing.** Return to your terminal and open the page with nano:  
     
   nano validator2.html

* Scroll down to the <script> tag near the bottom. You will see several empty functions: checkStrength() , countUpperCaseLetters() , countLowerCaseLetters() , and countDigits().
* Write your code inside these functions. **Do not change the names or parameters.**

1. **Complete countUpperCaseLetters().** This function has one parameter input. You may assume that input is a string.
   * Declare a variable called countU with an initial value of zero. This variable will hold the number of upper-case characters.
   * Write a loop that starts at i=0 and steps by one for as long as i is less than the length of the string.
   * In each step of the loop, check to see if input[i] is between “A” and “Z”, inclusive. If that is true, then add one to countU
   * When the loop is over, return countU
2. **Test countUpperCaseLetters()** Press **Ctrl‑O** to save in nano and refresh your browser page. Then, using the Web Developer tools in Chrome test the function. For example:
   * **countUpperCaseLetters("FROG");** // returns 4
   * **countUpperCaseLetters("A Z");** // returns 2
   * **countUpperCaseLetters("Frog");** // returns 1
   * **countUpperCaseLetters("1234");** // returns 0
3. **Repeat steps 5 and 6 for countLowercaseLetters() and countDigits().** Adapt the code accordingly.
4. **Complete checkStrength()**. This function has one parameter input. You may assume that input is a string. This function returns one of two possible values "Weak" or "Strong"
   * Check the length of input storing it in a variable called inputlength
   * If inputlength is less than 10, return "Weak" and stop.
   * Use the JavaScript indexOf() method to test for the presence of "!". Remember if the method returns -1 it means that the character was not found in the string. If there is no "!" then return "Weak" and stop.
   * Call **countUpperCaseLetters()** and store the result in a variable for later use. If this count is zero, return "Weak" and stop.
   * Call **countLowerCaseLetters()** and store the result in a variable for later use. If this count is zero, return "Weak" and stop.
   * Call **countDigits()** and store the result in a variable for later use. If this count is zero, return "Weak" and stop.
   * The sum of all those values, plus one, should be equal to the value stored in the inputlength variable. If it is, then return "Strong" and stop. Otherwise, return "Weak" and stop.
5. **Save and test your code.** Press **Ctrl‑O** to save in nano and refresh your browser page. Click the button to verify that the correct results appear. ( Some sample screenshots are on the next page.)
6. Determine the **URL** to your improved treadmill page. It will look like:

* https://misdemo.temple.edu/<yourusername>/assignment07/validate2.html
* Replace <yourusername> with your own account and test the link to make sure it works.

1. Go to the assignment in Canvas and paste your URL into the submission field. Be sure your page loads without errors.

## Hints & Reminders

* Always begin a function with parameters by validating the inputs. Return the error message immediately if the data is bad.
* Use return to send a result back to the caller. Do not rely on console.log() for the final answer.

## Grading rubric:

* **100 points:** You provided a valid URL, all three functions work correctly, and your code follows the rules.
* **80 points:** Minor problems such as a small miscalculation or formatting issue.
* **50 points:** Multiple problems but the page does not crash and you attempted the assignment.
* **0 points:** The file is missing, the URL is wrong, or the code uses features not covered in class.

## SAMPLE OUTPUT

A screenshot of a computer password

AI-generated content may be incorrect.

A screenshot of a computer password

AI-generated content may be incorrect.

A screenshot of a computer password

AI-generated content may be incorrect.

A screenshot of a computer password

AI-generated content may be incorrect.