Multimedia Systems

CA2 – Password Game

April 21, 2018

N00143888

Aaron O'Hare

Table of Contents

Introduction – Aims **3**

Interface Design Issues **3**

Implementation **3**

Checking Passwords **3**

Regular Expressions - Regex 4

Progressbar.js 5

Generating Passwords **6**

Simple Password genSimplePw() 6

Complex Password – genPw() 6

Hosting **8**

Materialize CSS **8**

Summary **9**

Conclusion **9**

Screenshots **10**

Resources **12**

Introduction – Aims

This project involved building a JavaScript based password game. The aim of this project was to provide a simple way for user's to check the security of their passwords, provide them with the ability to improve their passwords and to allow them to generate new passwords through the application. It needed to be easy for users' to understand and use. The application required a minimal amount of text content.

The application is a web-based application – due to the requirements of the continuous assessment. Throughout the development of this project git versioning control was used – this allowed for backup, access from multiple computers and the ability to roll back versions if needed. The repository is available here: https://github.com/aaronoh/MMSCA2-JSPasswordGame. The live application is also hosted online via Forge here: http://passwordcheck.getforge.io/.

Interface Design Issues

There are no outstanding design issues, the application is both desktop/laptop and mobile optimized through the use of the grid system provided by materialize CSS. The page will automatically resize based on the size of the devices' screen.

Implementation

Checking Passwords

The core function of this application is to verify the security of the users' password. The first step in this process was to build the homepage.

The home page is made up of a number of elements: The navigation, input field, buttons, modal window and the 'results' div which is populated by the getPw() function within checkPassword.js. This function is called when the value of the password input field changes. This allows for automatic feedback as the user modifies their password.

This is achieved by housing the getPW function within a key down even listener.

```
let input = document.getElementById('pwInput');
input.addEventListener('keyup', function getPw() {
                //functionality
});
```

Once the user presses a key on the keyboard, the getPw function is then executed. This function starts by setting the InnerHTML of the 'results' div to the required layout. This creates the card layout used to display the charts. There are three cards: Numbers, Symbols, Length and Total. Each one follows the same layout.

```
        <div class="col l3">\n' +
'                 <div class="card">\n' +
'                    <div class="card-content">\n' +
'                    <span id ="ctitle" class="card-title">Symbols</span>\n' +
'                       <div class="chart" id="symcircle"></div>\n' +
'                    </div>\n' +
'                 </div>\n' +
'                 </div>\n'
```

The users' input is then retrieved – this is the password which will be analysed via regular expressions.

```
let pw = document.getElementById('pwInput').value;
```

Regular Expressions - Regex

The password is then analysed using regex. A regex a search pattern, the expressions used in this application are designed specifically to extract the required information from the user input.

\d will extract all numbers from the string.

```
let numsregex = pw.match(/\d/g);
```

The results of this regex are then counted by getting the length of the array created. This provides us with the number of numbers in the user input.

```
if (numsregex) {
  nums = numsregex.length;
}
```

This same pattern was followed to extract symbols, with the regex for non language characters being \W:

```
let symaregex = pw.match(/\W/g);
```

Similarly, for upper and lower case letters:

```
let AZregex = pw.match(/[A-Z]/g);
let azregex = pw.match(/[a-z]/g);
```

Progressbar.js

Progress Bar is a JavaScript library used to create responsive progress bars for the web. They are used in this application to provide the user with visual feedback on the security level of their password.

You need to specify the shape, html div id, colour, duration of animation and the method of animation (easing).

The color is determined based on the result of the regex – red indicating a low score, yellow medium and green high.

```
var totalchart = new ProgressBar.Circle('#tcircle', {
    color: tcol,
    duration: 2000,
    easing: 'easeInOut'
});
```

You can then animate the chart to a defined value (between 0 and 1), at this point I also added the value text to the chart.

```
totalchart.animate(percentagescore / 100);

totalchart.setText(percentagescore + '%');
```

Generating Passwords

This application also allows the user to generate a new password. There are two options available, an easy to recall password or a password more suited to use with a password manager. The user simply clicks a button which calls the appropriate function.

```
<button id="gen" onclick="genPw()" class="btn waves-effect waves-light col l12" name="action">Generate
  <i class="material-icons right">lock</i>
</button>
```

Simple Password genSimplePw()

The simple password – while easy to recall, is still secure. It is made up of two random words of varying cases, two symbols and a random number. The array of strings which are used to create the two random words were extracted from the random-words node module. The random elements of the password are chosen using the Math.random function of JavaScript. Using this method, a random string is chosen from the array, followed by another random string converted to uppercase and finally a random number between 10 and 1000.

```
let w1 = wordList[Math.floor(Math.random()*wordList.length)];
let w2 = wordList[Math.floor(Math.random()*wordList.length)].toUpperCase();
let num = Math.floor(Math.random() * 1000);
```

Each of these elements is then used to construct a single string (with the addition of two symbols) before being returned to the user.

```
let simpPw =w1 +'!!' +w2 + num;
```

Complex Password – genPw()

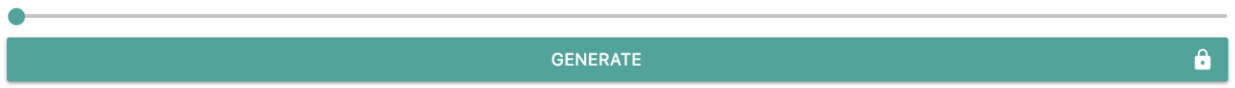
The complex password function generates passwords that are suited to password managers, this is an application that stores all of the users' password which is only accessible using the 'master password', which in theory is a very secure password set by the user. These passwords are made up of completely random characters making them impossible to guess.

This function firstly defines a set of characters that can be used in the generation of the password, omitting some of the commonly rejected characters such as ~.

```
let chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@£$%^&*()_+=-][{}';:/?.,><";
```

The user must specify the length of the password required via the use of a range slider.

```
<input type="range" id="range" min="10" max="20" />
```



Math.random is again used to choose a defined number of these characters at random.

```
for (var i = 0, n = chars.length; i < length; ++i) {
   pw += chars.charAt(Math.floor(Math.random() * n));
}
```

The password is then returned to the user via innerHTML

```
   document.getElementById('genPwOutput').innerHTML= `     <div class="row">\n
               <div class="col l12">\n
                 <div class="card">\n
                    <div class="card-content">\n
                      <span>${pw}</span>\n
                    </div>\n
                  </div>\n
               </div>\n
             </div>`
}
```

Hosting

The application is hosted using getforge. This service allows for simplified and fee hosting of HTML/CSS/JavaScript based projects. I simply uploaded the compressed project to their server via the web interface and my project is accessible at http://passwordcheck.getforge.io/.

The only manual configuration required was the process of enabling SSL. This is important as the website focuses on the creation and verification of secure passwords. A website without SSL will display a 'Not Secure' warning to the user, this would be very disconcerting to the user. Forge provide a simple method of enabling SSL via LetsEncrypt. However, unfortunately they have run out of SSL certificates for the week 23/04 – 29/04, this should be automatically rectified on 30/04.

Materialize CSS

Materialize CSS is a CSS library. It provides a number of pre-styled elements that can be incorporated into existing projects. This simplifies the layout/design process and provides the application with a clean, consistent style.

```
<div class="card">
                    <div class="card-content">
                   <span id ="ctitle" class="card-title">Symbols</span>
                       <div class="chart" id="symcircle"></div>
                    </div>
                </div>
```

Materialize CSS also provides a grid system which allows the developer to split the webpage into columns. This aids the design process for multiple screen sizes.

```
<div class="col l12">
```

Summary

The final product allows the user to check the quality of their password. It provides visual feedback on the makeup of the password, including the inclusion of numbers, symbols and the length of the password. Each of these categories are scored and graphed and the user receives a total score. All of this information is update as soon as the user types, allowing for instant feedback.

The user may also generate new passwords, choosing between a more secure password for use in a password manager or an easier to remember password.

The application is very visual, with the majority of text hidden unless the user specifically chooses to see it. These meets the requirements of the target user group.

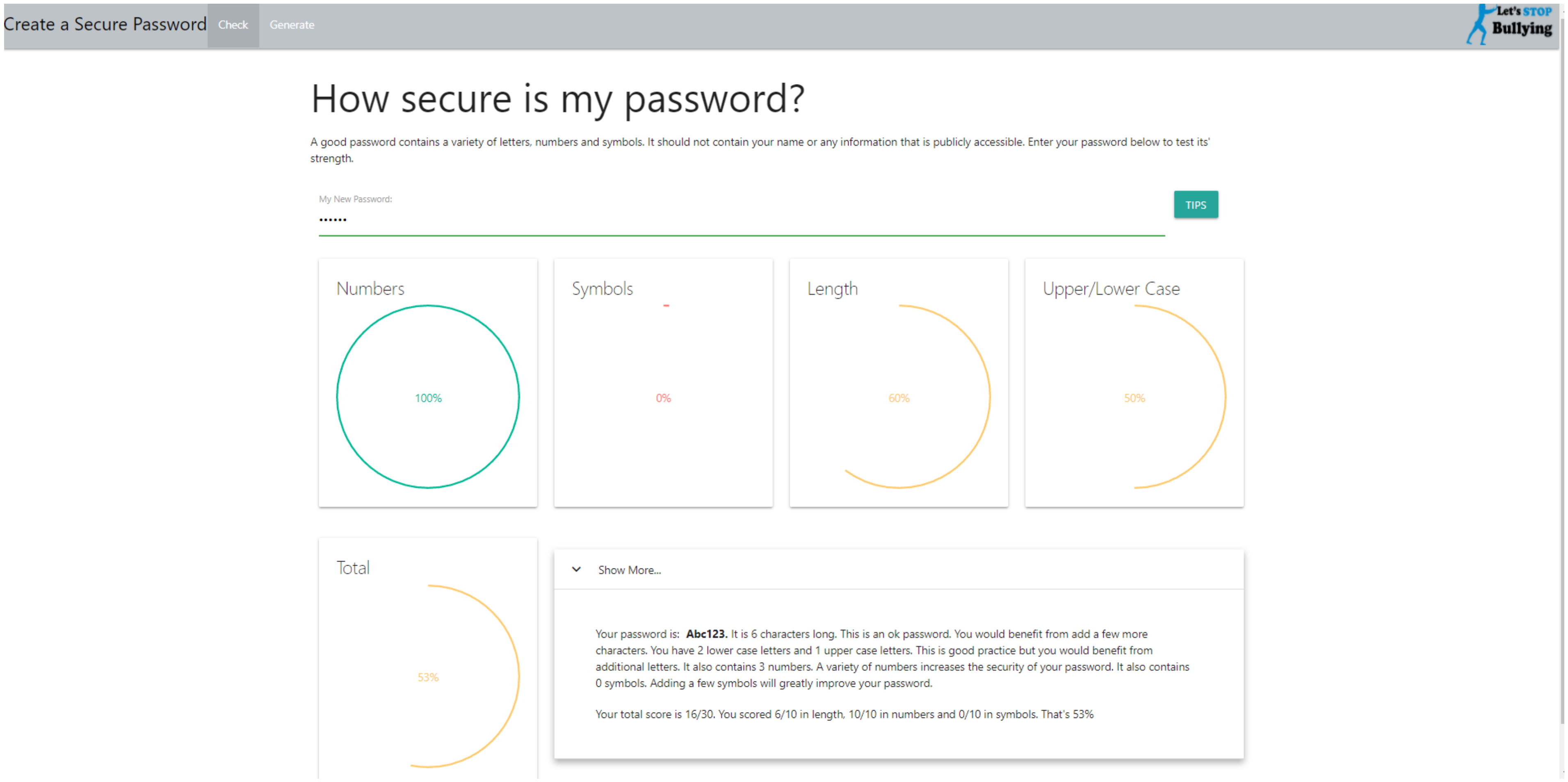
The project was built using HTML, CSS and JavaScript. Progressbar.js was used to construct the graphs, random-words was used to provide the array of words used in the simpler passwords, Materialize CSS was used to style and layout the project and Forge was used to host the project online.

Conclusion

Overall, I am happy with the progression of this project. I was reasonably comfortable with the JavaScript code behind the functionality of this project however I had never worked with regular expressions before and found them quite challenging. Materialize CSS was also new to me. It proved a worthwhile framework to learn as I have already began using it in other projects. This project also made me more comfortable with using git and introduced me to a new hosting provider, Forge. Forge specialize in static pages which is something I thought I would have difficulty finding as many of the hosting providers focus on more dynamic/backend heavy solutions.

If I was able to devote more time to this project I would focus on increasing the educational aspect of it. Using statistical information and multimedia to convey the importance of creating and maintaining secure passwords for your online presence.

Screenshots



How secure is my password?

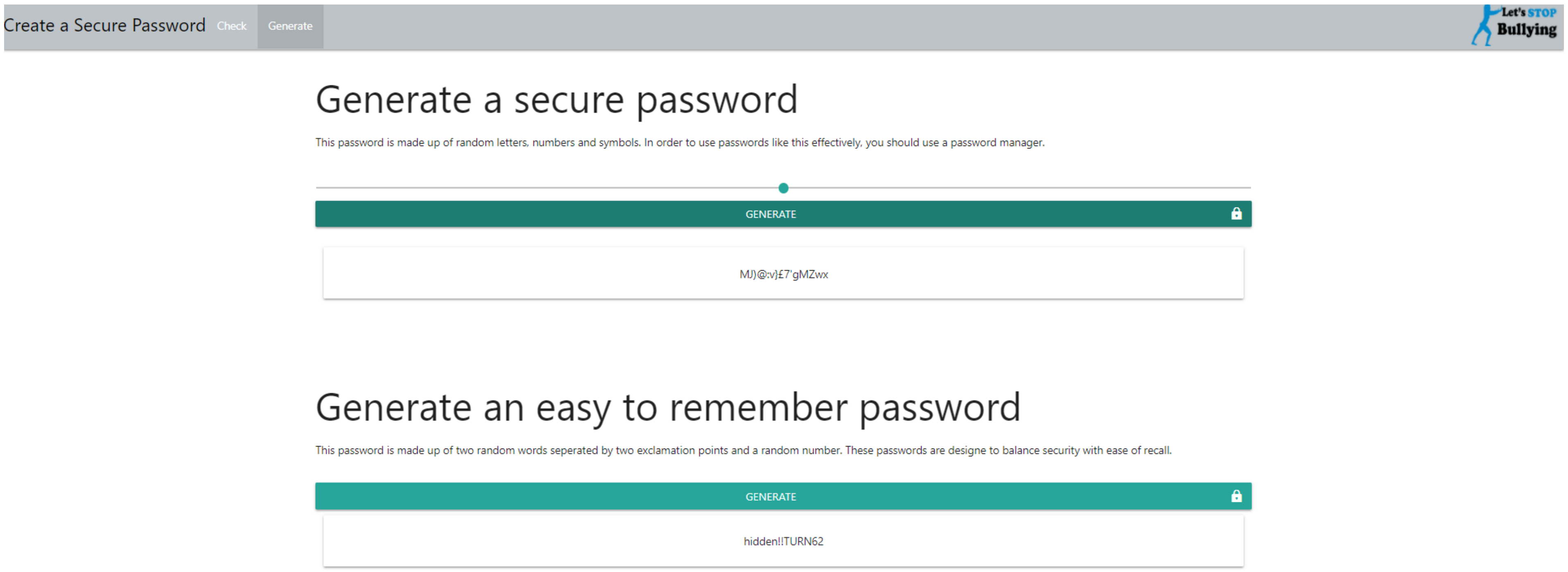
A good password contains a variety of letters, numbers and symbols. It should not contain your name or any information that is publicly accessible. Enter your password below to test its' strength.

My New Password:
••••••

Your password is: **Abc123.** It is 6 characters long. This is an ok password. You would benefit from add a few more characters. You have 2 lower case letters and 1 upper case letters. This is good practice but you would benefit from additional letters. It also contains 3 numbers. A variety of numbers increases the security of your password. It also contains 0 symbols. Adding a few symbols will greatly improve your password.

Your total score is 16/30. You scored 6/10 in length, 10/10 in numbers and 0/10 in symbols. That's 53%

Figure 1 – Check Password



Generate a secure password

This password is made up of random letters, numbers and symbols. In order to use passwords like this effectively, you should use a password manager.

MJ)@:v}£7'gMZwx

Generate an easy to remember password

This password is made up of two random words seperated by two exclamation points and a random number. These passwords are designe to balance security with ease of recall.

hidden!!TURN62

Figure 2 - Generate Password

Resources

https://regexr.com/

https://developer.mozilla.org/bm/docs/Web/JavaScript/Reference

https://github.com/punkave/random-words

http://materializecss.com/

https://getforge.com/