

CONTENTS

Introduction	1
Rich visualization	1
Peak Interactivity and exceptional loading time	2
Agility	4
Security	4
Web Technology Integration	5
References	7

Introduction

The approach chosen to implement the designed prototype was using HTML, CSS and dash of Bootstrap for the frontend purposes and JavaScript for both backend and frontend. Below are reasons for using the chosen approach:

Rich visualization: One of the main reasons why JavaScript was used implement the captcha and password feature was because JavaScript fully radiates the visualization aspect of rendering data on the fly when paired with HTML and CSS pages, which were used in the implementation of the prototype. As of the captcha method used which is the text-based type, the technologies used offer a perfect solution to render and display the random captcha text. A study done by (Andreeva, J. et al, 2012) illustrates how JavaScript neatly portrays information on an unprecedented level due to its customizable UI elements and responsive capability to load data dynamically when ever a user interacts with the HTML and CSS UI components

The snippet below is the source code of the prototype where UI implementation got into use. CDNs of CSS, Bootstrap, jQuery and JavaScript are imported into the project. This is done so the frameworks could be used during the development.

```

7  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
8
9  <!-- jQuery library -->
10 <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
11
12 <!-- Popper JS -->
13 <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
14
15 <!-- Latest compiled JavaScript -->
16 <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script>
17

```

The snippet below shows a function that displays output below the input password form that helps in guiding the user about the strength of his password as input is being passed in real-time.

```

25     function StrengthChecker(PasswordParameter){
26         // We then change the badge's color and text based on the password strength
27
28         if(strongPassword.test(PasswordParameter) && x.style.display === "none" ) {
29             strengthBadge.style.backgroundColor = "green"
30             strengthBadge.textContent = 'Strong'
31             x.style.display = "block"
32         }
33         else if(mediumPassword.test(PasswordParameter)){
34             strengthBadge.style.backgroundColor = 'blue'
35             strengthBadge.textContent = 'Medium'
36             x.style.display = "block"
37         } else{
38             x.style.display = "none"
39             strengthBadge.style.backgroundColor = 'red'
40             strengthBadge.textContent = 'Weak'
41             strengthBadge2.textContent = "Use at least 8 characters"
42             strengthBadge3.textContent = "Have at least one uppercase letter [A-Z]"
43             strengthBadge4.textContent = "Have at least one lowercase letter [a-z]"
44             strengthBadge5.textContent = "Have at least one digit [0-9]"
45             strengthBadge6.textContent = "Have at least one special character [*-~]"
46         }
47     }

```

Peak Interactivity and exceptional loading time: JavaScript when compared to other programming languages, performs better in terms of client to server communications as idle server request do not exist therefore, connections will only exist when prompted by defined user input. This results in fast executions and reduced lag time. (Andreeva, J. et al, 2012) states that programs built with JavaScript load the requested content and data only when it is demanded depending on the certain parameters passed as input from the client side during runtime.

From line 13 to line 16 on the snippet below shows declaration and initialization of a variable. Line 17 to 24 will generate text-based captcha using the predefined library and the declared variable only at runtime.

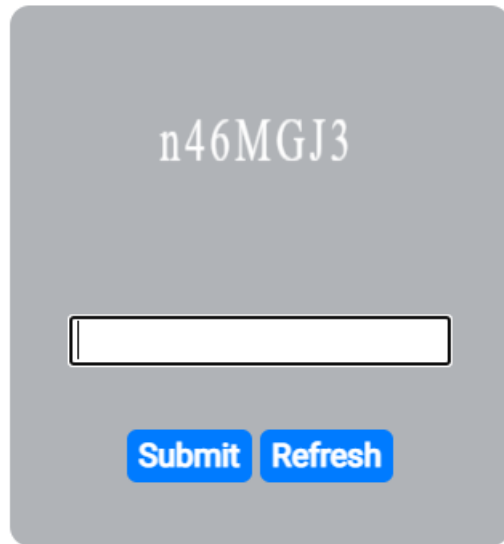
```

12 // alphaNums contains the characters with which you want to create the CAPTCHA
13 let alphaNums = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N',
14 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
15 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
16 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'];
17 let emptyArr = [];
18 // This loop generates a random string of 7 characters using alphaNums
19 // Further this string is displayed as a CAPTCHA
20 for (let i = 1; i <= 7; i++) {
21     emptyArr.push(alphaNums[Math.floor(Math.random() * alphaNums.length)]);
22 }
23 var c = emptyArr.join('');
24 ctx.fillText(emptyArr.join(''), captchaText.width/4, captchaText.height/2);

```

The screenshot below shows a captcha generated at runtime after a user's password has been approved.

Please enter the code below



n46MGJ3

Submit Refresh

Agility: Given the nature of the prototype specifications which, users must be redirected to a separate page once the password and captcha are accepted. JavaScript proved to be the ideal scripting language for the task as it can validate and redirect users to the appropriate HTML documents to be displayed prior to execution. (Yu, D., Chander et al, 2007) explains the nature of self-modification within the language which contributes to its impeccable ability to generate process, generate, and redirect sessions.

The line of code is used to redirect a user to the index page after a successful validation.

```
55 | window.location.href = "index.html"
```

Security: As to the security aspects of the prototype, a clear avoidance of probable threats like an SQL injection has been adequately avoided through utilizing data saving libraries found in the JavaScript API stack. Unlike using SQL to save data from an application which is susceptible to SQL injections, JavaScript uses predefined methods that utilize runtime techniques to store data and for this case the data is stored on a simple text file which is immune to script-based attacks. A study done by (Chugh, R., Meister. et al, 2009) on “Staged information flow for JavaScript” critically assesses the safety of JavaScript on web applications and one of the discoveries found was how efficient the language is in merging various API libraries together in conjunction of performing a more much broader and peculiar task.

Line 1 defines a method that will be used to save user credentials in a text file. Line 4-5 is a declaration of the elements that will be used to collect the data then in line 7-8 the data is collected and stored having predefined text at the beginning of every set of data.

```

1  let saveFile = () => {
2    // Get the data from each element on the form.
3
4    const username = document.getElementById("userEntry");
5    const password = document.getElementById("PassEntry");
6
7    // This variable stores all the data.
8    let data = "\r User Name: " + username.value +
9    " \r\n " + "Password: " + password.value;
10
11
12    console.log(data); //printing form data into the console
13    // Convert the text to BLOB.
14    const textToBLOB = new Blob([data], { type: "text/plain" });

```

Web Technology Integration: (Sun, K. and Ryu, S., 2017) states that JavaScript has high level integration with web technology APIs. One of these is the ability of JavaScript code to manipulate HTML documents due to the provision and presence of technologies such as the *Document Object Model* (DOM) which is found in almost any available web browser in the market. Therefore, this aspect influenced my choice of using HTML in the development of the prototype and the underlying fact that HTML documents are exceptionally portable, hence allowing the prototype to have a wide user base.

This property is demonstrated on the snippets below. First on the snippet below we see a UI div element being declared on line 46 and given “myDIV” as an identifier on a HTML document. This will be later used to access it on JavaScript code

```

44    <div class="row">
45        <div class="col"></div>
46        <div id="myDIV">

```

On line 15 the identifier “myDIV” is initialized with value of “x” on JavaScript. Then on line 16 “x” is set to not be visible when the HTML document is loaded on a browser.

```

15    let x = document.getElementById("myDIV")
16    x.style.display = "none"
17    strengthBadge.style.display = "none"

```

On line 28 there is an if-statement that checks the strength of the inputted password and also if the div element is displayed and if it’s not displayed line 31 displays the element.

```
28 if(strongPassword.test(PasswordParameter) && x.style.display === "none")
29     strengthBadge.style.backgroundColor = "green"
30     strengthBadge.textContent = 'Strong'
31     x.style.display = "block"
```

The screen shot below shows the div element that contains the proceed button being visible only after a user provides an acceptable password combination.

Create a new password



Use at least 8 characters
Have at least one uppercase letter [A-Z]
Have at least one lowercase letter [a-z]
Have at least one digit [0-9]
Have at least one special character [^~]

References

Andreeva, J., Dzhunov, I., Karavakis, E., Kokoszkiewicz, L., Nowotka, M., Saiz, P. and Tuckett, D., 2012, December. Designing and developing portable large-scale JavaScript web applications within the Experiment Dashboard framework. In *Journal of Physics: Conference Series* (Vol. 396, No. 5, p. 052069). IOP Publishing.

Yu, D., Chander, A., Islam, N. and Serikov, I., 2007. JavaScript instrumentation for browser security. *ACM SIGPLAN Notices*, 42(1), pp.237-249.

Chugh, R., Meister, J.A., Jhala, R. and Lerner, S., 2009, June. Staged information flow for JavaScript. In *Proceedings of the 30th ACM SIGPLAN conference on programming language design and implementation* (pp. 50-62).

Sun, K. and Ryu, S., 2017. Analysis of JavaScript programs: Challenges and research trends. *ACM Computing Surveys (CSUR)*, 50(4), pp.1-4.