

**Tech Saksham**

**Final Project Report**

**“Random password Generator Using HTML”**

**“Avanthi Scientific Technological and Research Academy”**

| **Roll No.** | **Student Name** |
| --- | --- |
| 19PT1A0520 | K. Anupriya Reddy |
| 19PT1A0507 | D. Nikitha |

**Trainer Name .**

**Uma Maheshwari**

Abstract

Password generators that generate site-specific passwords on demand are an alternative to password managers. Over the last 15 years a range of such systems have been described. We propose the first general model for such systems, and critically examine options for instantiating it. The model enables an objective assessment of the design of such systems; it has also been used to sketch a possible new scheme, AutoPass, intended to incorporate the best features of the prior art while addressing many of the shortcomings of existing systems.

A Password generator is a very useful application. This application can generate a random password, with the combination of letters, numerics, and special characters. One can mention the length of the password based on requirement and can also select the strength of the password.

**INDEX**

| **Sr. No.** | **Table of Contents** | **Page No.** |
| --- | --- | --- |
| 1 | Introduction | 5 |
| 2 | Project Definition | 6 |
| 3 | Objective | 7 |
| 4 | Architectur Project Architecture | 8 |
| 5 | Design Methodology | 10 |
| 6 | Flow chart Diagram | 12 |
| 7 | Program Screen | 13 |
| 8 | Program Code | 14 |
| 9 | Problem Statement & Solution | 33 |
| 10 | Conclusion | 34 |
| 11 | References | 35 |

**INTRODUCTION**

We are developing a webpage named Password generator. The Password generator tool creates a random and customized password for users that helps them to create a strong password which provides greater security. While there are many examples of "random" password generator programs available on the Internet, generating randomness can be tricky and many programs do not generate random characters in a way that ensures strong security.

A common recommendation is to use open source security tools where possible since they allow independent checks on the quality of the methods used. Note that simply generating a password at random does not ensure the password is a strong password, because it is possible, although highly unlikely, to generate an easily guessed or cracked password. In fact, there is no need at all for a password to have been produced by a perfectly random process: it just needs to be sufficiently difficult to guess. A password generator can be part of a password manager. When a password policy enforces complex rules, it can be easier to use a password generator based on that set of rules than to manually create passwords.

**PROJECT DEFINITION**

Speaking regarding the system, the user can create a random password according to various sizes.It additionally presents an aesthetic color-coded system which indicates the stamina of the password, beginning from Very Weak to Superb password strength. After creating a random password, the system presents it in the clipboard where the user can copy and paste easily.

Good password generators the following:

* Adjust guidelines to fit different sites' unique password requirements.
* Generate strong passwords using secure technology with built-in randomness.
* Are integrated into a password manager like Dashlane to create, manage, and easily use all of your strong passwords.

A password is your personal key to a computer system. Passwords help to ensure that only authorized individuals access computer systems. Passwords also help to determine accountability for all transactions and other changes made to system resources, including the data.Password generators are tools that allow the user to create random and customized strong passwords based on preferences.

**OBJECTIVE**

The main objective of the study is to develop a password generator.

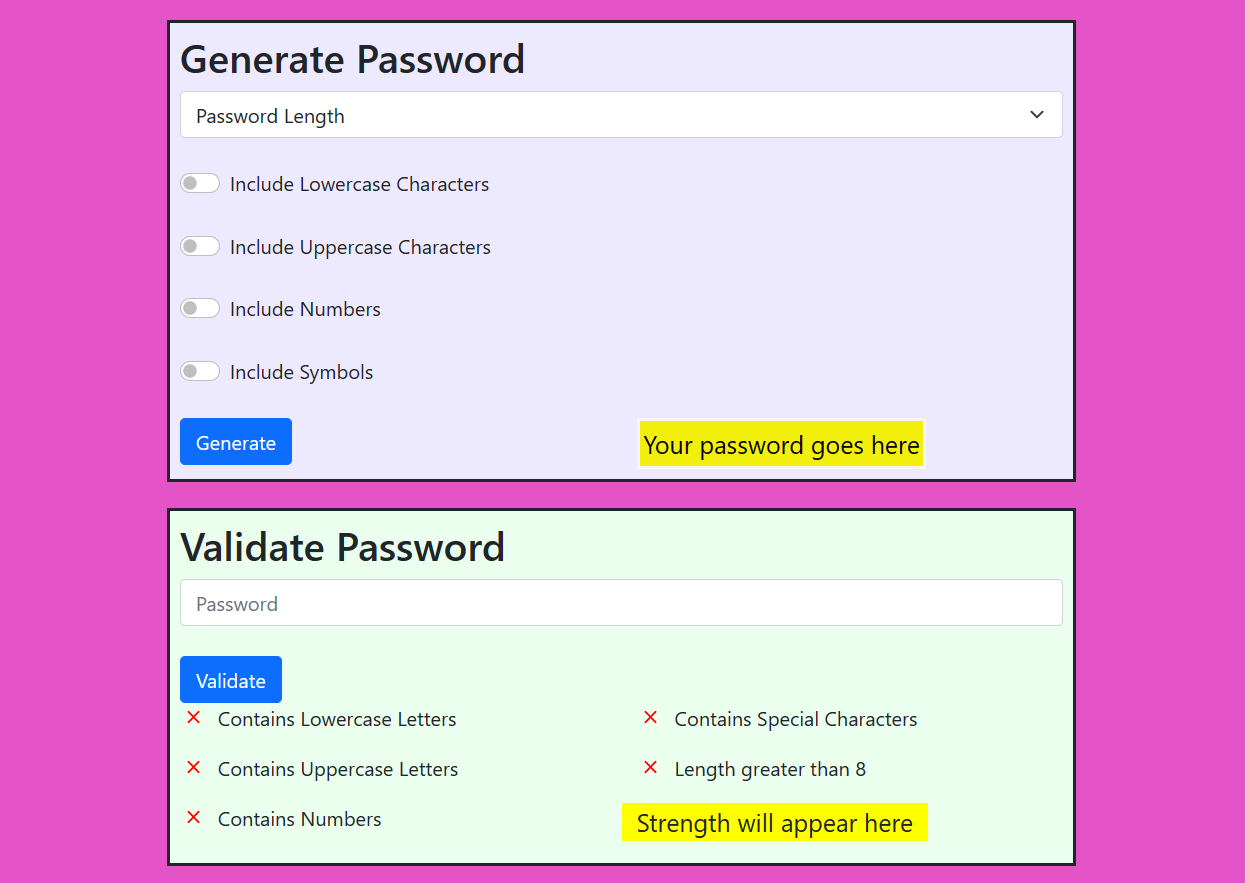
For a secure and strong password reminder the following:-

* Never use personal information.
* Include a combination of letters, numbers, and characters.
* Prioritize password length.
* Never repeat passwords.
* Avoid using real words.

Strong password:-

* **Long**: The longer a password, the more secure it is. A strong password should be at least 12 characters long.
* **Random**: Strong passwords use a combination of letters, numbers, cases, and symbols to form an unpredictable string of characters that doesn't resemble words or names.
* **Unique:** A strong password should be unique to each account to reduce vulnerability in the event of a hack**.**

**Project Architecture**

****

The architecture of a random password generator typically involves the following components:

* **User Interface:** This component provides the user interface for the password generator. It can be a graphical user interface (GUI) or a command-line interface (CLI).
* **Random Number Generator:** This component generates random numbers or characters that are used to create the password. The randomness of the generated characters is crucial for the security of the password.
* **Password Generator Logic:** This component defines the rules for generating a password. It specifies the length of the password, the types of characters to use (e.g., uppercase letters, lowercase letters, numbers, special characters), and any other requirements.
* **Password Strength Checker:** This component checks the strength of the generated password based on industry-standard metrics such as entropy, complexity, and length. It can also provide feedback to the user on how to improve the password's strength.
* **Password Storage:** This component stores the generated password securely. It can be stored locally or remotely, depending on the application's requirements.
* **Password Manager Integration:** This component integrates with popular password managers such as LastPass, 1Password, or Dashlane, enabling users to save the generated password and use it across different platforms.

Overall, the architecture of a random password generator should prioritize security, usability, and flexibility. It should generate passwords that are strong enough to resist brute-force attacks and easy to remember or manage by users.

**DESIGN METHODOLOGY:**

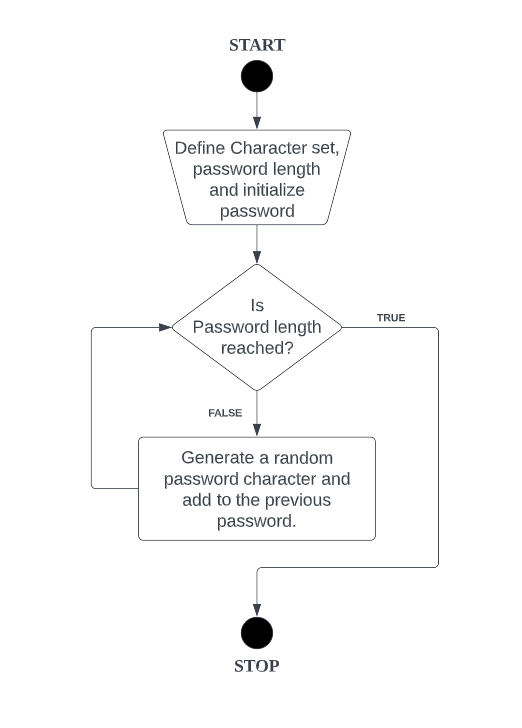
Steps required for building this project:

* All characters should be stored as a list. This can be done by choosing the instructions manually
* Ask the user for the length of the password.
* Use random shuffle to shuffle the characters.
* Create an empty list to store the password.
* Iterate password length to generate the password
* Choose a random character from all the characters using the random.choice method.
* Add the random character to the password.
* Randomize the resultant list of passwords.
* Use the generate method to create a string from the list of passwords.
* Print the password.

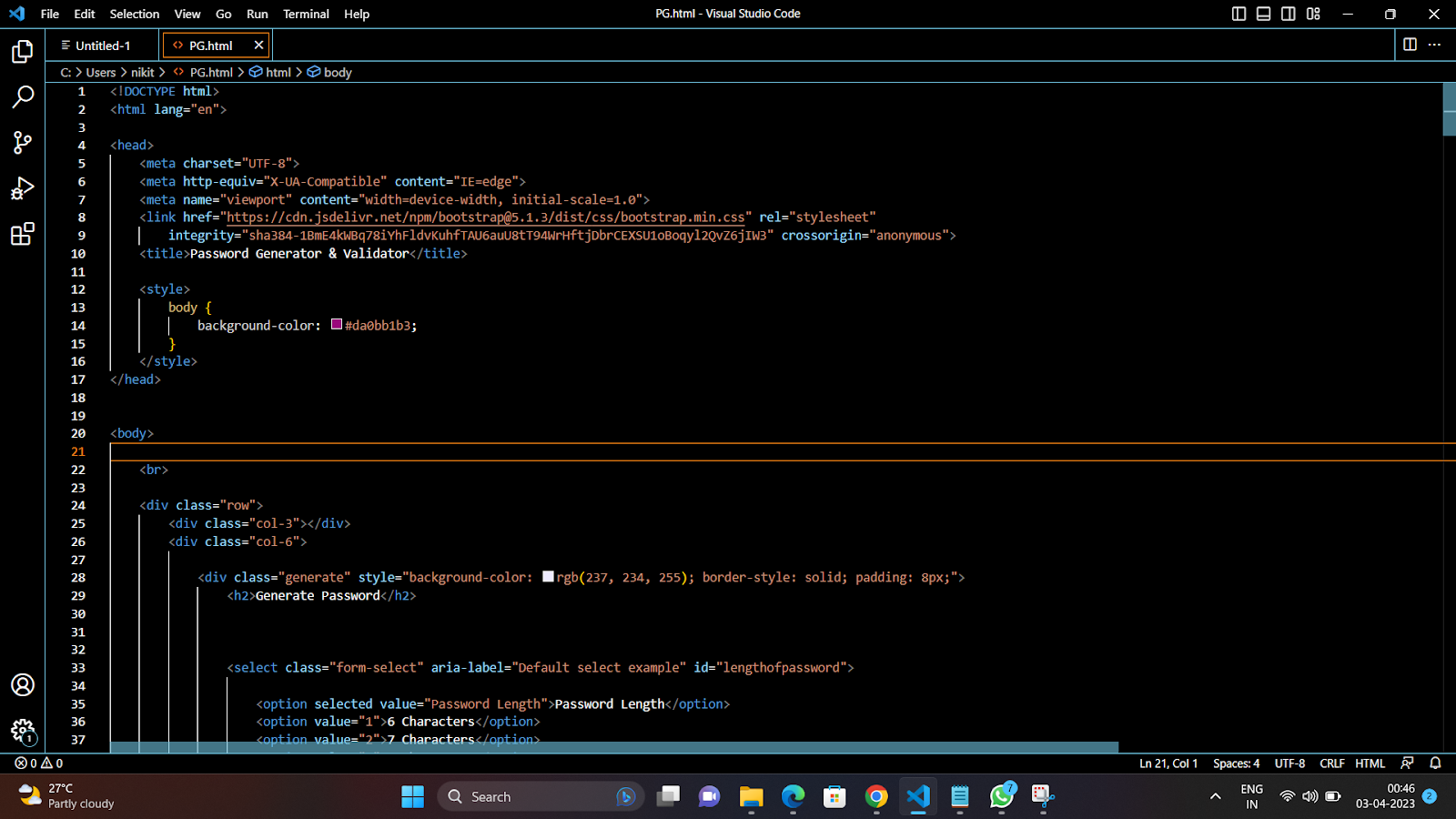
Here are the steps to design a methodology for a random password generator:

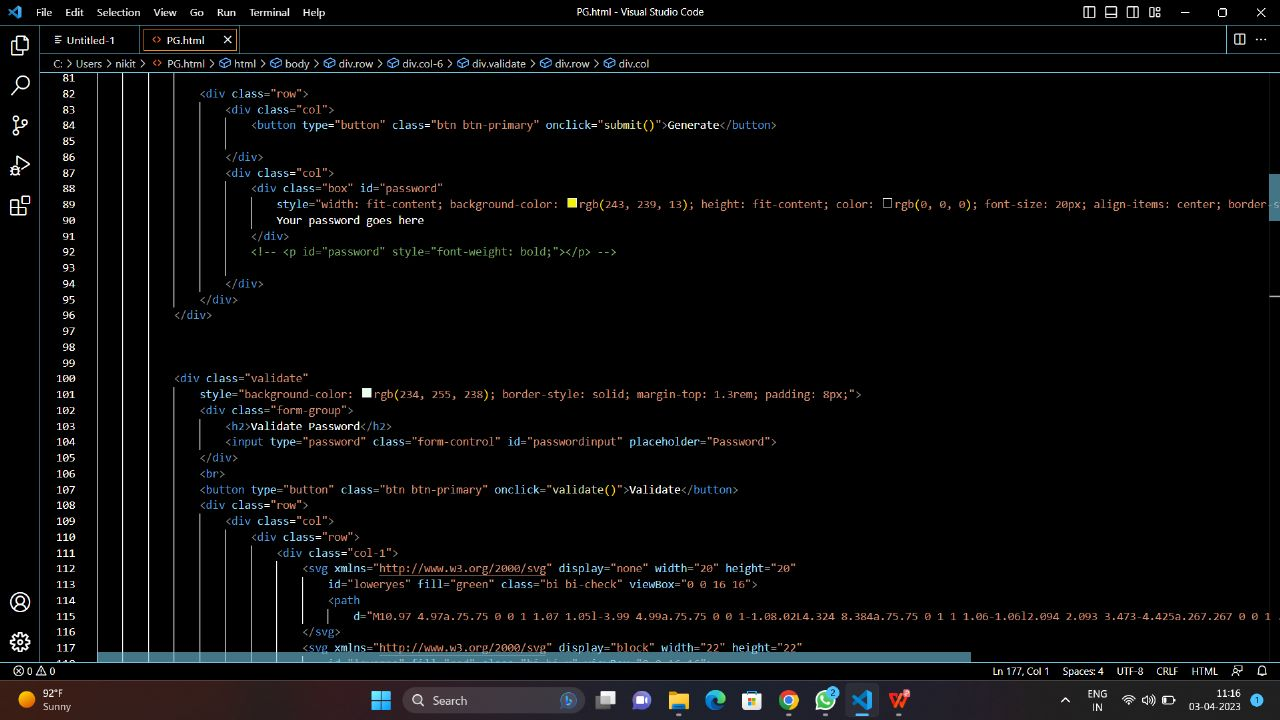
* **Determine the password requirements:** Start by defining the minimum and maximum length of the password, the type of characters to be included (uppercase letters, lowercase letters, numbers, special characters), and any other requirements (such as avoiding common words or patterns).
* **Choose a programming language:** Select a programming language that you are comfortable with and that is capable of generating random characters.
* **Define the algorithm:** Develop an algorithm that can generate random characters based on the password requirements. The algorithm should ensure that each character is selected randomly and meets the specified criteria.
* **Implement the algorithm:** Implement the algorithm in your chosen programming language. Test the code to ensure that it works as expected.
* **Create a user interface:** Create a simple user interface that allows the user to select the password requirements and generate a random password.
* **Test and debug:** Test the password generator thoroughly to ensure that it meets all requirements and is generating passwords randomly. Debug any issues that arise during testing.
* **Document the code:** Document the code to make it easier for others to understand and modify in the future.
* **Securely store passwords:** Finally, ensure that any generated passwords are stored securely and not stored in plain text. One way to do this is to hash the password and store the hash.

**FLOW CHART DIAGRAM**

****

**PROGRAM SCREEN**

****

****

**PROGRAM CODE**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

<title>Password Generator & Validator</title>

<style>

body

{

background-color: #da0bb1b3;

}

</style>

</head>

<body>

<br>

<div class="row">

<div class="col-3"></div>

<div class="col-6">

<div class="generate" style="background-color: rgb(237, 234, 255); border-style: solid; padding: 8px;">

<h2>Generate Password</h2>

<select class="form-select" aria-label="Default select example" id="lengthofpassword">

<option selected value="Password Length">Password Length</option>

<option value="1">6 Characters</option>

<option value="2">7 Characters</option>

<option value="3">8 Characters</option>

<option value="4">9 Characters</option>

<option value="5">10 Characters</option>

<option value="6">11 Characters</option>

<option value="7">12 Characters</option>

<option value="8">13 Characters</option>

<option value="9">14 Characters</option>

</select>

<br>

<div class="form-check form-switch">

<input class="form-check-input" type="checkbox" name="check" value="lowercase"

id="flexSwitchCheckDefault">

<label class="form-check-label" for="flexSwitchCheckDefault">Include Lowercase Characters</label>

</div>

<br>

<div class="form-check form-switch">

<input class="form-check-input" type="checkbox" name="check" value="uppercase"

id="flexSwitchCheckDefault">

<label class="form-check-label" for="flexSwitchCheckDefault">Include Uppercase Characters</label>

</div>

<br>

<div class="form-check form-switch">

<input class="form-check-input" type="checkbox" name="check" value="numbers"

id="flexSwitchCheckDefault">

<label class="form-check-label" for="flexSwitchCheckDefault">Include Numbers</label>

</div>

<br>

<div class="form-check form-switch">

<input class="form-check-input" type="checkbox" name="check" value="symbols"

id="flexSwitchCheckDefault">

<label class="form-check-label" for="flexSwitchCheckDefault">Include Symbols</label>

</div>

<br>

<div class="row">

<div class="col">

<button type="button" class="btn btn-primary" onclick="submit()">Generate</button>

</div>

<div class="col">

<div class="box" id="password"

style="width: fit-content; background-color: rgb(243, 239, 13); height: fit-content; color: rgb(0, 0, 0); font-size: 20px; align-items: center; border-style: solid; border-color: whitesmoke; padding: 3px; ">

Your password goes here

</div>

<!-- <p id="password" style="font-weight: bold;"></p> -->

</div>

</div>

</div>

<div class="validate"

style="background-color: rgb(234, 255, 238); border-style: solid; margin-top: 1.3rem; padding: 8px;">

<div class="form-group">

<h2>Validate Password</h2>

<input type="password" class="form-control" id="passwordinput" placeholder="Password">

</div>

<br>

<button type="button" class="btn btn-primary" onclick="validate()">Validate</button>

<div class="row">

<div class="col">

<div class="row">

<div class="col-1">

<svg xmlns="http://www.w3.org/2000/svg" display="none" width="20" height="20"

id="loweryes" fill="green" class="bi bi-check" viewBox="0 0 16 16">

<path

d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z" />

</svg>

<svg xmlns="http://www.w3.org/2000/svg" display="block" width="22" height="22"

id="lowerno" fill="red" class="bi bi-x" viewBox="0 0 16 16">

<path

d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z" />

</svg>

</div>

<div class="col">

<p>Contains Lowercase Letters</p>

</div>

</div>

<!-- ------- -->

<div class="row">

<div class="col-1">

<svg xmlns="http://www.w3.org/2000/svg" display="none" width="20" height="20"

id="upperyes" fill="green" class="bi bi-check" viewBox="0 0 16 16">

<path

d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z" />

</svg>

<svg xmlns="http://www.w3.org/2000/svg" display="block" width="22" height="22"

id="upperno" fill="red" class="bi bi-x" viewBox="0 0 16 16">

<path

d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z" />

</svg>

</div>

<div class="col">

<p>Contains Uppercase Letters</p>

</div>

</div>

<!-- ------- -->

<div class="row">

<div class="col-1">

<svg xmlns="http://www.w3.org/2000/svg" display="none" width="20" height="20"

id="numberyes" fill="green" class="bi bi-check" viewBox="0 0 16 16">

<path

d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z" />

</svg>

<svg xmlns="http://www.w3.org/2000/svg" display="block" width="22" height="22"

id="numberno" fill="red" class="bi bi-x" viewBox="0 0 16 16">

<path

d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z" />

</svg>

</div>

<div class="col">

<p>Contains Numbers</p>

</div>

</div>

<!-- ------- -->

</div>

<div class="col">

<div class="row">

<div class="col-1">

<svg xmlns="http://www.w3.org/2000/svg" display="none" width="20" height="20"

id="specialyes" fill="green" class="bi bi-check" viewBox="0 0 16 16">

<path

d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z" />

</svg>

<svg xmlns="http://www.w3.org/2000/svg" display="block" width="22" height="22"

id="specialno" fill="red" class="bi bi-x" viewBox="0 0 16 16">

<path

d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z" />

</svg>

</div>

<div class="col">

<p>Contains Special Characters</p>

</div>

</div>

<!-- ------- -->

<div class="row">

<div class="col-1">

<svg xmlns="http://www.w3.org/2000/svg" display="none" width="20" height="20"

id="lengthyes" fill="green" class="bi bi-check" viewBox="0 0 16 16">

<path

d="M10.97 4.97a.75.75 0 0 1 1.07 1.05l-3.99 4.99a.75.75 0 0 1-1.08.02L4.324 8.384a.75.75 0 1 1 1.06-1.06l2.094 2.093 3.473-4.425a.267.267 0 0 1 .02-.022z" />

</svg>

<svg xmlns="http://www.w3.org/2000/svg" display="block" width="22" height="22"

id="lengthno" fill="red" class="bi bi-x" viewBox="0 0 16 16">

<path

d="M4.646 4.646a.5.5 0 0 1 .708 0L8 7.293l2.646-2.647a.5.5 0 0 1 .708.708L8.707 8l2.647 2.646a.5.5 0 0 1-.708.708L8 8.707l-2.646 2.647a.5.5 0 0 1-.708-.708L7.293 8 4.646 5.354a.5.5 0 0 1 0-.708z" />

</svg>

</div>

<div class="col">

<p>Length greater than 8</p>

</div>

</div>

<div class="row" >

<!-- <div class="col-1"></div> -->

<div class="col-1" style="background-color: yellow; width:fit-content; font-size: 20px;" id="strength\_display" >Strength will appear here</div>

<!-- <p id="strength\_display">

Strength Will appear here

</p> -->

</div>

</div>

</div>

</div>

<div class="col-3"></div>

</div>

</div>

<script>

function submit() {

let str = ""

var pass = '';

var checkboxes = document.getElementsByName('check');

for (var checkbox of checkboxes) {

if (checkbox.checked) {

// document.body.append(checkbox.value + ' ');

console.log(checkbox.value)

if ((checkbox.value).localeCompare("lowercase") == 0) {

console.log("hello")

str = str + "abcdefghijklmnopqrstuvwxyz"

}

if ((checkbox.value).localeCompare("uppercase") == 0) {

console.log("hello")

str = str + "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

}

if ((checkbox.value).localeCompare("numbers") == 0) {

console.log("hello")

str = str + "1234567890"

}

if ((checkbox.value).localeCompare("symbols") == 0) {

console.log("hello")

str = str + "!@#$%^&\*()\_+={[}]':;?/><.,~"

}

}

}

var passwordlength = document.getElementById('lengthofpassword');

// console.log(passwordlength.value)

let textcomp = "Password Length"

if ((textcomp.localeCompare(passwordlength.value)) == 0) {

alert("Please select the password length");

return;

}

let lengthreq = parseInt(passwordlength.value) + 5

console.log(lengthreq)

for (i = 1; i <= lengthreq; i++) {

var char = Math.floor(Math.random()

\* str.length + 1);

pass += str.charAt(char)

}

console.log(pass)

pass.replace(/&/g, "&amp;");

pass.replace(/</g, "&lt;");

pass.replace(/>/g, "&gt;");

pass.replace(/"/g, "&quot;")

pass.replace(/'/g, "&#039;");

if (pass.length != lengthreq) {

submit();

}

else {

document.getElementById("password").innerHTML = pass;

}

let password = document.getElementById("passwordinput");

password.addEventListener("keyup", function () {

let pass = document.getElementById("password").value;

console.log(pass)

});

}

function reset\_checks() {

document.getElementById("loweryes").style.display = "none";

document.getElementById("lowerno").style.display = "inline";

document.getElementById("upperyes").style.display = "none";

document.getElementById("upperno").style.display = "inline";

document.getElementById("numberyes").style.display = "none";

document.getElementById("numberno").style.display = "inline";

document.getElementById("specialyes").style.display = "none";

document.getElementById("specialno").style.display = "inline";

document.getElementById("lengthyes").style.display = "none";

document.getElementById("lengthno").style.display = "inline";

}

function validate() {

reset\_checks();

let strength\_score = 0;

var strength = document.getElementById("passwordinput").value;

console.log(strength);

if (strength.match(/[a-z]/)) {

strength\_score += 1;

document.getElementById("loweryes").style.display = "inline";

document.getElementById("lowerno").style.display = "none";

}

if (strength.match(/[A-Z]/)) {

strength\_score += 1;

document.getElementById("upperyes").style.display = "inline";

document.getElementById("upperno").style.display = "none";

}

if (strength.match(/[1-9]/)) {

strength\_score += 1;

document.getElementById("numberyes").style.display = "inline";

document.getElementById("numberno").style.display = "none";

}

if (strength.match(/[^a-zA-Z\d]/g)) {

strength\_score += 1;

document.getElementById("specialyes").style.display = "inline";

document.getElementById("specialno").style.display = "none";

}

if (strength.length >= 8) {

strength\_score += 1;

document.getElementById("lengthyes").style.display = "inline";

document.getElementById("lengthno").style.display = "none";

}

if (strength\_score == 1) { document.getElementById("strength\_display").style.backgroundColor="red";

document.getElementById("strength\_display").innerHTML = "Weak Password";

}

else if (strength\_score == 2) { document.getElementById("strength\_display").style.backgroundColor="#F97D62";

document.getElementById("strength\_display").innerHTML = "Fair Password";

}

else if (strength\_score == 3) { document.getElementById("strength\_display").style.backgroundColor="#F9F962";

document.getElementById("strength\_display").innerHTML = "Better Password";

}

else if (strength\_score == 4) { document.getElementById("strength\_display").style.backgroundColor="#6286F9";

document.getElementById("strength\_display").innerHTML = "Strong Password";

}

else if (strength\_score == 5) {

document.getElementById("strength\_display").style.backgroundColor="#6CFF43";

document.getElementById("strength\_display").innerHTML = "Very Strong Password";

}

// if (strength\_score == 1 && strength.length >= 8) {

// final\_strength = "low and better"

// console.log(final\_strength);

// }

console.log(strength\_score)

}

</script>

</body>

</html>

**PROBLEM STATEMENT:**

In order to create a strong password, users can use this password generator to generate a random and customized password.

**RESULT AND DISCUSSION:**

**ADVANTAGES:**

1) Minimal Effort.

2) Secure & strong password.

3) Less time consuming.

4) Fast response.

5) complex for hackers.

**CONCLUSION**

The password generated using the alpha-numerical random password mechanism that was illustrated above is practical and can be used with great results. When the password is selected manually, most of the time, the user selects the password that is related to himself or herself and related to any of the events. This gives the space for the intruders to deploy various attacks in breaking the passwords. The random generated passwords avoid this particular situation. One of the drawbacks could be the difficulty in memorizing the randomly generated password. But when comparing the security achieved through the randomly generated password, it is much preferable than the manually chosen password. The encryption and decryption standard provided here also strengthens the security measures. Since, the encryption and decryption standards are simple, it is cost- effective. The above work also creates awareness and interest to start exploring this field more.

**REFERENCES**

1. https://www.geeksforgeeks.org/
2. https://www.wikipedia.org/
3. https://github.com/