RANDOM PASSWORD GENERATOR

P SAIKARTHIK

VU21CSEN0100398

1. Abstract

This report provides a detailed analysis of a Python-based password generator program. The program is designed to create random, secure passwords of user-specified lengths. It combines uppercase letters, lowercase letters, digits, and special characters to ensure strong password generation. The user is given the option to accept or regenerate the password until satisfied. This report explains the code structure, functionality, and the importance of each component in ensuring the effectiveness of the password generator.

2. Introduction

Password security is critical in today's digital world, where data breaches are common. Creating strong, random passwords is a fundamental step in protecting sensitive information. This password generator program assists users in generating secure passwords by leveraging Python's random module to ensure randomness and unpredictability.

3. Program Structure

The program consists of a simple, interactive Python script that includes the following components:

Character Sets: The program defines four different character sets: uppercase letters, lowercase letters, digits, and special characters.

Password Generation Logic: The core of the program involves selecting random characters from these sets to create a password of the desired length.

```
import random
print("\t\t\t\t\t\t\t\tPASSWORD GENERATOR\t\t\t")
def fun():
    j=int(input("Enter the desired length to generate password :"))
    b= [str(i) for i in range(10)]
    a= [chr(i) for i in range(65, 91)]
    f= [chr(i) for i in range (97,123)]
    g=['#','&','@','*','(',')','!','$','%']
    c=a+b+f+g
    d=""
    d=d+random.choice(a)
    for i in range(j-1):
        k=random.choice(c)
        d=d+k
    print(f"The generated password of length {j} is : {d}")



fun()
while(1):
    l=input("\nAre you satisfied with the password (Y/N) :")
    print()
    if(l=='Y' or l=='y'):
        print("Thank You !! have a nice day :)")
        break
    elif(l=='N' or l=='n'):
        fun()
    else:
        print("Enter valid option !!!")
```

4. Code Explanation

4.1. Character Arrays

a: Uppercase letters (A-Z)

b: Digits (0-9)

f: Lowercase letters (a-z)

g: Special characters (#, &, @, *, (,), !, $, %)

These arrays are combined into a single list c, which is used to generate the password.

4.2. Password Generation (fun() function)

The function begins by asking the user to input the desired password length.

It ensures that the generated password includes at least one uppercase letter by starting the password string d with a random choice from the a array.

The remaining characters are selected randomly from the combined list c to complete the password.

The password is printed to the user.

4.3. User Interaction Loop

After generating the password, the program asks the user if they are satisfied with the password.

If the user is satisfied (Y/y), the program ends with a thank you message.

If not (N/n), the program calls the fun() function again to generate a new password.

The loop continues until the user is satisfied.

5. Implementation Details

Randomness and Security: The use of random.choice ensures that each character is chosen independently and uniformly, which is crucial for creating unpredictable and secure passwords.

Password Composition: By guaranteeing that at least one character from the uppercase set is included, the program avoids generating all-lowercase passwords, which are generally weaker.

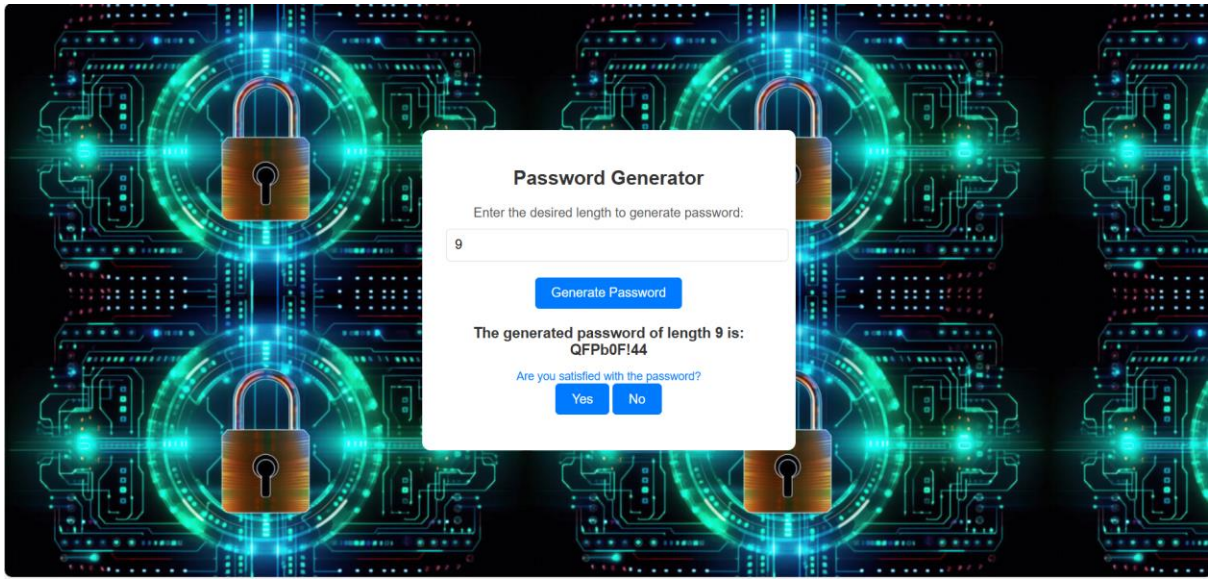
User Experience: The program is interactive, allowing users to continuously generate new passwords until they are satisfied, ensuring that the final password meets the user's expectations.

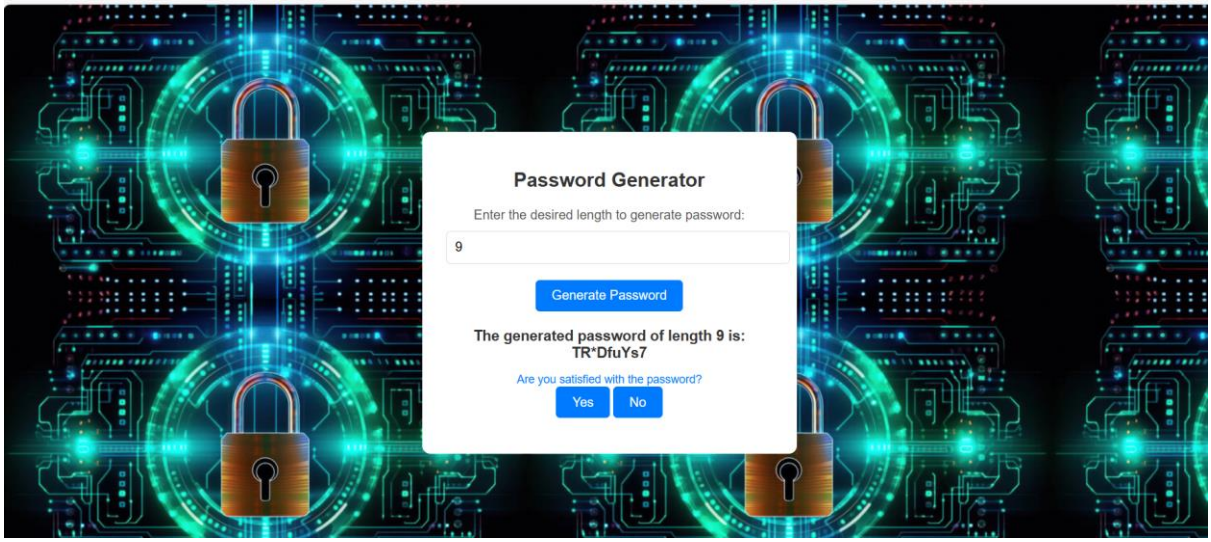
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Password Generator</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            background-image: url('https://t3.ftcdn.net/jpg/06/04/74/88/360_F_604748806_gQSyPrazhAocHefqrUtieGBKK22PS5QZ.jpg');
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .container {
            background-color: #fff;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
            text-align: center;
            max-width: 400px;
            width: 100%;
        }
        h1 {
            margin-bottom: 20px;
            font-size: 24px;
            color: #333;
        }
        label {
```

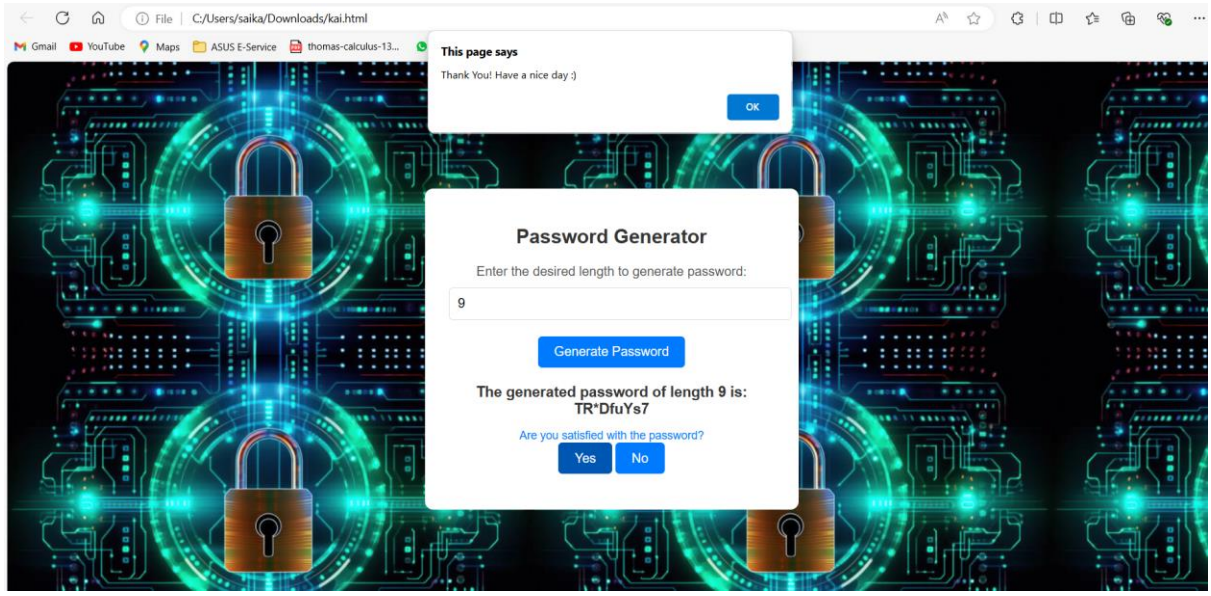
```
        function generatePassword() {
            const length = document.getElementById('length').value;
            const numbers = '0123456789';
            const uppercase = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
            const lowercase = 'abcdefghijklmnopqrstuvwxyz';
            const symbols = '#&@*()!$%';
            const allChars = uppercase + numbers + lowercase + symbols;

            let password = uppercase.charAt(Math.floor(Math.random() * uppercase.length));

            for (let i = 1; i < length; i++) {
                const char = allChars.charAt(Math.floor(Math.random() * allChars.length));
                password += char;
            }

            document.getElementById('result').innerText = `The generated password of length ${length} is: ${password}`;
            promptFeedback();
        }

        function promptFeedback() {
            const feedback = document.getElementById('feedback');
            feedback.innerHTML = `
                <div>
                    <p>Are you satisfied with the password? <br><button onclick="satisfied()">Yes</button> <button onclick="generatePassword()">No</button><
                </div>
```

6. Conclusion

This password generator program is an effective tool for creating strong, random passwords. It combines user interaction with secure password generation practices, ensuring that users can easily obtain passwords that meet their security needs. The program can be further enhanced by adding more character sets or incorporating additional security features such as length recommendations or password strength indicators.







7. Recommendations

Enhance Security: Consider using a more secure random number generator, such as secrets module, which is designed for cryptographic applications.

Password Strength Indication: Adding a feature to evaluate and display the strength of the generated password could help users understand how secure their password is.

Customization Options: Allow users to customize the inclusion of different character sets (e.g., excluding special characters) based on their specific needs.

🌐 Sources

This report was based on general programming knowledge and principles of password security best practices. The following are references for further reading:

Python Official Documentation - random module

NIST Guidelines on Password Security

Python Official Documentation - secrets module