SECURE PASSWORD GENERATOR

21CSS101J - PROGRAMMING FOR PROBLEM SOLVING

Mini Project Report

Submitted by

Anu Shree VS [Reg No.: RA2211003010739]

B.Tech. CSE-CORE

Cavin Shree Ramesh Kumar [Reg No.: RA2211003010740]

B.Tech. CSE-CORE

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act,1956)

S.R.M NAGAR, KATTANKULATHUR – 603203

KANCHEEPURAM DISTRICT

December 2022

TABLE OF CONTENTS

Problem Statement

On a typical day in one's life, one visits numerous websites from e-commerce websites to learning websites and coding platforms. Most of these platforms require he/she to log in with their credentials and have a strong password.

Coming up with a strong password that meet security constraints can be challenging. Trivial passwords are not that strong and are susceptible to security threats.

Having a weak password is not good for a system that demands high confidentiality and security of user credentials. It turns out that people find it hard to make up a strong password that is secure enough to prevent unauthorised users from memorizing it.

This is what we aim to achieve with our programme, to be able to create a secure password that is tailored to the users demands; so that their data is protected.

Methodology / Procedure

This program uses the modules- string and secrets.

- secrets module to generate sequences of random characters that are secure for cryptographic applications
- string module to create string of possible characters to be used

1. First, a string is created using strings module that contains uppercase and lowercase letter, digits, and special characters.
2. Required password length is taken as input from user.
3. If user chooses 'no constraints':
   3.1. Using secrets module, random characters are generated from alphabet and stored in a string pwd till pwd is of required length.
   3.2. Password pwd is printed.
4. If user chooses 'yes constraints':
   4.1. Required constraints are taken as input from the user.
   4.2. Using secrets module, random characters are generated from alphabet and stored in a string pwd till pwd is of required length.
   4.3. Password pwd is checked if it matches given constraints.
      4.3.1. If it matches constraints, the password is printed.
      4.3.2. If it doesn't match the constraints, a new password pwd is generated till it matches the given constraints

Programme

```
# necessary imports
import secrets
import string

# define the alphabet
lowercase_letters = string.ascii_lowercase
uppercase_letters = string.ascii_uppercase
digits = string.digits
special_chars = string.punctuation

alphabet = lowercase_letters + uppercase_letters + digits + special_chars

# fix password length
pwd_length=int(input("Enter length of password required : "))

choice=input("\nWant to set any constraints? (YES/NO) : ")
if choice=='NO':
    # generating password
    pwd = ''
    for i in range(pwd_length):
        pwd += ''.join(secrets.choice(alphabet))
    print(pwd)
elif choice=="YES":
    # setting constraints
    print("\nEnter number of -")
```

```
ucl_n=int(input("uppercase letters : "))
lcl_n=int(input("lowercase letters : "))
dig_n=int(input("digits : "))
splchr_n=int(input("special chars : "))
# generating password
if ucl_n+lcl_n+dig_n+splchr_n==pwd_length:
    while True:
        pwd=''
        for i in range(pwd_length):
            pwd+=''.join(secrets.choice(alphabet))

        # checking if generated password matches the constraints
        if (sum(char in special_chars for char in pwd)==splchr_n)and(sum(char in digits for char in pwd)==dig_n)and(sum(char in lowercase_letters for char in pwd)==lcl_n)and(sum(char in uppercase_letters for char in pwd)==ucl_n):
            break
    print("\n"+pwd)
else:
    print("\nrequired password length and constraints don't match up")
```

Results

Sample output 1 :

```
Shell
Enter length of password required : 16
Want to set any constraints? (YES/NO) : NO
JOCK<\8/[xuD3E%s
>
```

Sample output 2 :

```
Shell
Enter length of password required : 16
Want to set any constraints? (YES/NO) : YES
Enter number of -
uppercase letters : 4
lowercase letters : 4
digits : 4
special chars : 4
fVu^o5a9Y-Z8_N"1
>
```

Sample output 3 :

```
Shell
Enter length of password required : 16
Want to set any constraints? (YES/NO) : YES
Enter number of -
uppercase letters : 3
lowercase letters : 3
digits : 6
special chars : 10
required password length and constraints don't match up
>
```

Sample output 4 :

```
Shell
Enter length of password required : 16
Want to set any constraints? (YES/NO) : YES
Enter number of -
uppercase letters : 1
lowercase letters : 2
digits : 3
special chars : 4
required password length and constraints don't match up
>
```

Result :

The programme has been successfully executed and meets all requirements.

Conclusion

After rigorously testing the performance of our coding system we are well satisfied with its credibility and speed. For this project we have used the Python programming language. Our project is very convenient and efficient, it gives a very strong password within seconds.

We have immense pleasure in expressing our deepest gratitude to our beloved and highest esteemed institution SRM University for giving this golden opportunity. We would also like to thank and express our sincere gratitude to Dr Ramesh S for his valuable guidance and support without which would not have been able to complete this project.

This project was made by Anu Shree VS and Cavin Shree Ramesh Kumar from K1 section.