Report On

Password Generator

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by

Rushikesh Avhad (Roll no 6)

Snigdha Bhoir (Roll no 18)

Sanket Borhade (Roll no 20)

Supervisor

Prof. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2024-25)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Dice Roller GUI” is a bonafide work of Rushikesh Avhad (Roll no 6) ,Snigdha Bhoir(Roll no 18), Sanket Borhade (Roll no 20submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

Supervisor

Prof. Sneha Mhatre

Dr Megha Trivedi
Head of Department

Dr. H.V. Vankudre
Principal

Abstract

Python has emerged as a versatile programming language, widely used for various applications, including cybersecurity. In this abstract, we delve into the realm of password generation in Python, exploring the implementation of secure and robust password generator algorithms.

The abstract begins by examining the importance of strong passwords in safeguarding digital assets against unauthorized access. Weak passwords remain a prevalent vulnerability, highlighting the necessity for automated tools to generate complex and unique passwords effectively.

We explore the fundamental principles underlying password generation algorithms, focusing on techniques to ensure randomness, entropy, and usability. Python's rich set of libraries, including random and secrets, provide powerful tools for generating random strings and cryptographic-strength passwords. We discuss the advantages of leveraging these libraries to produce passwords resistant to common hacking strategies.

Furthermore, the abstract presents a step-by-step implementation of a password generator in Python, highlighting best practices for generating secure passwords. Through code examples and explanations, we demonstrate how to incorporate randomness, character diversity, and length customization to enhance password strength.

Additionally, considerations for user experience and privacy are discussed. We explore strategies for balancing password complexity with memorability, as well as techniques for securely storing and transmitting generated passwords.

In summary, this abstract provides a comprehensive overview of password generation in Python, offering insights into algorithmic principles, implementation strategies, and ethical considerations. By leveraging Python's capabilities and adhering to best practices, developers can contribute to the creation of secure password generation tools, thereby enhancing cybersecurity in digital ecosystems.

INDEX

Contents

INTRODUCTION

1.1 Introduction

In today's digital era, password security is paramount. Weak passwords are a major vulnerability, making secure password generation essential. Python's versatility makes it ideal for developing password generators. In this introduction, we'll explore the importance of strong passwords, the principles behind password generation, and why Python is a preferred language for this task. Let's dive in.

Problem Statement

Weak passwords persist as a major cybersecurity risk, often due to the challenge of creating and managing strong passwords. There's a need for a user-friendly password generator in Python that can swiftly produce secure, unique passwords. This tool must prioritize security, usability, efficiency, and compatibility across various platforms

PROPOSED SYSTEM

Key features of the proposed system include:

1. Randomness: The password generator should utilize Python's random module to ensure the randomness of generated passwords. Randomness is essential for creating unpredictable passwords that are resistant to brute-force attacks.
2. Customization: Users should have the option to customize the generated passwords based on their preferences. This includes specifying the length of the password and selecting the types of characters to include (e.g., uppercase letters, lowercase letters, numbers, special characters).
3. Strength Indicator: The password generator can include a feature to indicate the strength of the generated password. This could be displayed as a visual indicator or a textual representation, providing users with feedback on the security level of the password.
4. Usability: The interface of the password generator should be intuitive and user-friendly, allowing users of all technical backgrounds to generate passwords easily. Clear instructions and prompts can guide users through the process of generating a password.

2.2 Module Description:

random_generator.py: This module contains functions responsible for generating random passwords. It utilizes Python's random module to ensure randomness in password generation. Functions include:

generate_password(length): Generates a random password of specified length.

generate_complex_password(length): Generates a random password with a mix of uppercase letters, lowercase letters, numbers, and sp r.ecial characters.

user_interface.py: This module handles the user interface aspects of the password generator project. It provides a command-line interface (CLI) for users to interact with the password generator. Functions include:

get_password_length(): Prompts the user to enter the desired length of the password.

get_password_strength(): Prompts the user to choose the desired strength of the password (e.g., weak, medium, strong).

2.3 Description of Software & Hardware Used And Its Programming:

Software:

Python: The primary programming language used for development, providing a versatile and efficient platform for building the application.

Tkinter: A standard GUI library for Python, utilized for creating the graphical user interface of the Dice Roller application.

Integrated Development Environment (IDE): Software tools like PyCharm, Visual Studio Code, or IDLE are employed for coding, debugging, and testing the application code efficiently.

Operating System: The application is developed and tested on various operating systems such as Windows, macOS, and Linux to ensure cross-platform compatibility.

Hardware:

Personal Computer: A standard desktop or laptop computer is used as the primary development environment, equipped with adequate processing power and memory to support software development tasks.

Input Devices: Standard input devices such as a keyboard and mouse are used for interacting with the computer and testing the application.

Internet Connectivity: Internet access is required for downloading software tools, libraries, and dependencies, as well as for accessing online resources and documentation during the development process.

Block Diagram:

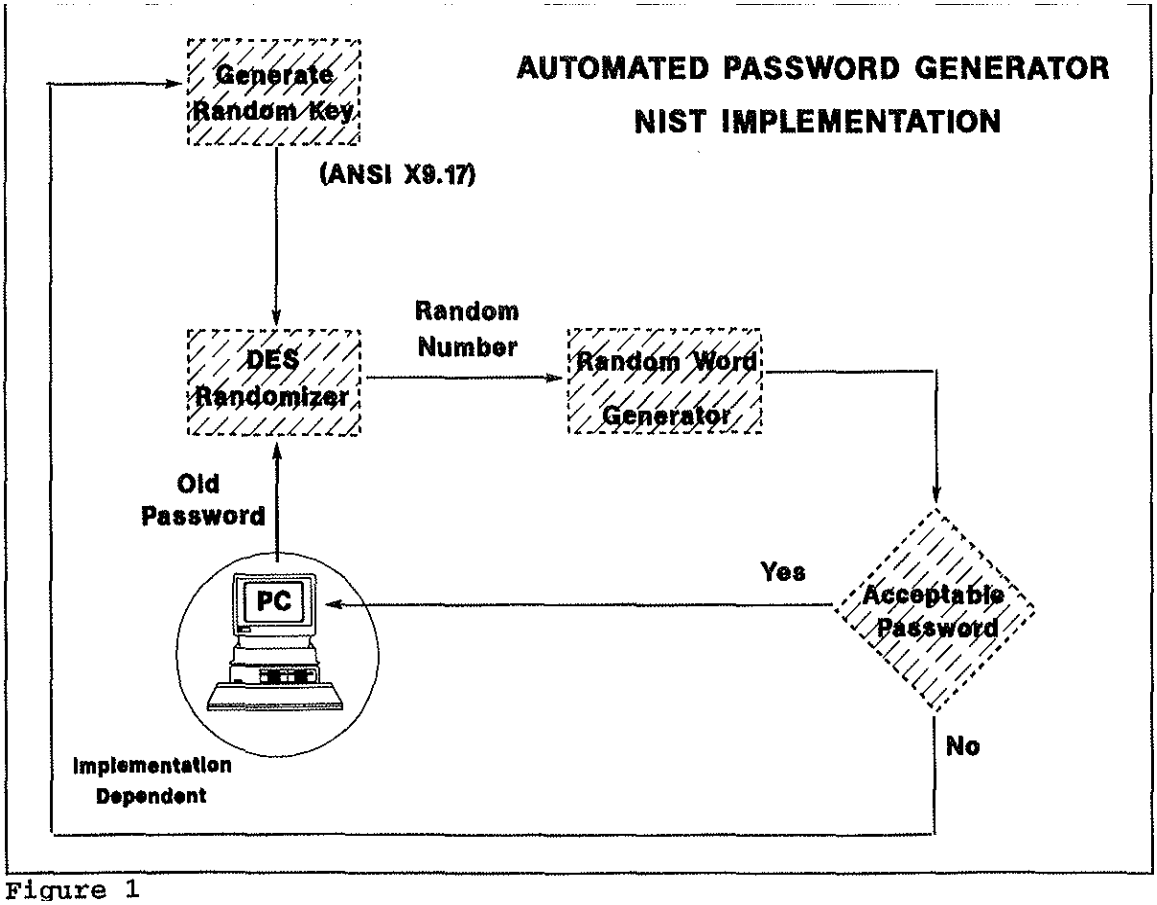


Figure 1

2.4 Code

```
import tkinter as tk
from tkinter import *
from tkinter import ttk, messagebox
import string
import numpy as np
import pandas as pd
import pyperclip
def generator():
    small_alphabets = string.ascii_lowercase
    capital_alphabets = string.ascii_uppercase
    numbers = string.digits
    special_characters = string.punctuation
   all_characters = small_alphabets + capital_alphabets + numbers + special_characters
    password_length = int(length_Box.get())

    if choice.get() == 1:
        password = ''.join(np.random.choice(list(small_alphabets), password_length))

    elif choice.get() == 2:
        password = ''.join(np.random.choice(list(small_alphabets + capital_alphabets),
password_length))

    elif choice.get() == 3:
        password = ''.join(np.random.choice(list(all_characters), password_length))

    passwordField.delete(0, END)
    passwordField.insert(0, password)

def copy():
    random_password = passwordField.get()
```

```
    pyperclip.copy(random_password)

root = tk.Tk()
root.config(bg='gray20')
choice = IntVar()
Font = ('arial', 13, 'bold')

passwordLabel = Label(root, text='Password Generator', font=('times new roman', 20, 'bold'), bg='gray20', fg='white')
passwordLabel.grid(pady=10)

weakradioButton = Radiobutton(root, text='Weak', value=1, variable=choice, font=Font)
weakradioButton.grid(pady=5)

mediumradioButton = Radiobutton(root, text='Medium', value=2, variable=choice, font=Font)
mediumradioButton.grid(pady=5)

strongradioButton = Radiobutton(root, text='Strong', value=3, variable=choice, font=Font)
strongradioButton.grid(pady=5)

lengthLabel = Label(root, text='Password Length', font=Font, bg='gray20', fg='white')
lengthLabel.grid(pady=5)

length_Box = Spinbox(root, from_=5, to_=18, width=5, font=Font)
length_Box.grid(pady=5)

generateButton = Button(root, text='Generate', font=Font, command=generator)
generateButton.grid(pady=5)

passwordField = Entry(root, width=25, bd=2, font=Font)
passwordField.grid()
copyButton = Button(root, text='Copy', font=Font, command=copy)
copyButton.grid(pady=5)
```

```
root.mainloop()
import tkinter as tk
from tkinter import *
from tkinter import ttk, messagebox
import string
import numpy as np
import pandas as pd
import pyperclip
def generator():
    small_alphabets = string.ascii_lowercase
    capital_alphabets = string.ascii_uppercase
    numbers = string.digits
    special_characters = string.punctuation

    all_characters = small_alphabets + capital_alphabets + numbers + special_characters
    password_length = int(length_Box.get())

    if choice.get() == 1:
        password = ''.join(np.random.choice(list(small_alphabets), password_length))

    elif choice.get() == 2:
        password = ''.join(np.random.choice(list(small_alphabets + capital_alphabets),
password_length))

    elif choice.get() == 3:
        password = ''.join(np.random.choice(list(all_characters), password_length))

    passwordField.delete(0, END)
    passwordField.insert(0, password)
def copy():
    random_password = passwordField.get()
    pyperclip.copy(random_password)

root = tk.Tk()
```

```
root.config(bg='gray20')
choice = IntVar()
Font = ('arial', 13, 'bold')

passwordLabel = Label(root, text='Password Generator', font=('times new roman', 20, 'bold'), bg='gray20', fg='white')
passwordLabel.grid(pady=10)

weakradioButton = Radiobutton(root, text='Weak', value=1, variable=choice, font=Font)
weakradioButton.grid(pady=5)

mediumradioButton = Radiobutton(root, text='Medium', value=2, variable=choice, font=Font)
mediumradioButton.grid(pady=5)

strongradioButton = Radiobutton(root, text='Strong', value=3, variable=choice, font=Font)
strongradioButton.grid(pady=5)

lengthLabel = Label(root, text='Password Length', font=Font, bg='gray20', fg='white')
lengthLabel.grid(pady=5)

length_Box = Spinbox(root, from_=5, to_=18, width=5, font=Font)
length_Box.grid(pady=5)
generateButton = Button(root, text='Generate', font=Font, command=generator)
generateButton.grid(pady=5)
passwordField = Entry(root, width=25, bd=2, font=Font)
passwordField.grid()
copyButton = Button(root, text='Copy', font=Font, command=copy)
copyButton.grid(pady=5)
root.mainloop()
```

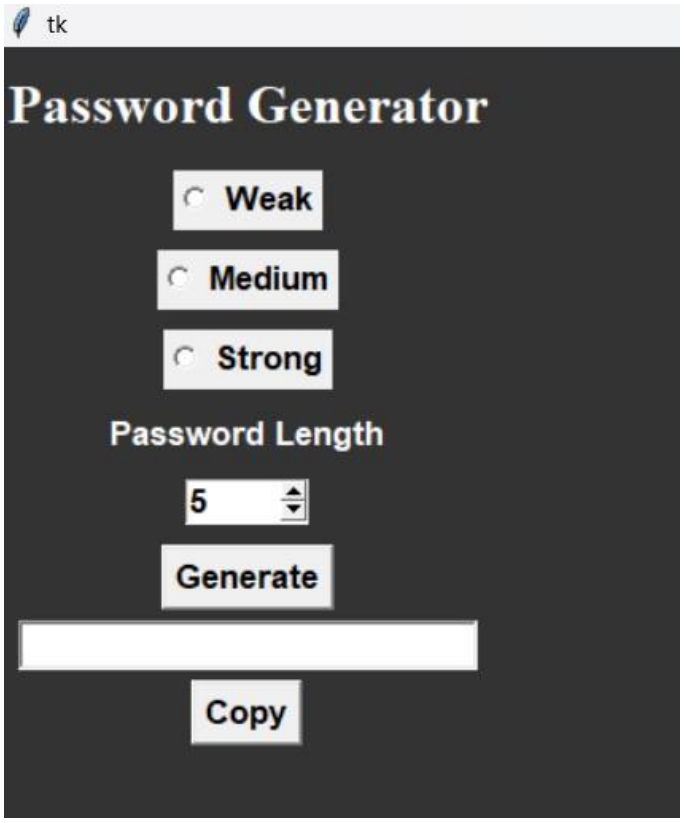
RESULTS AND CONCLUSION

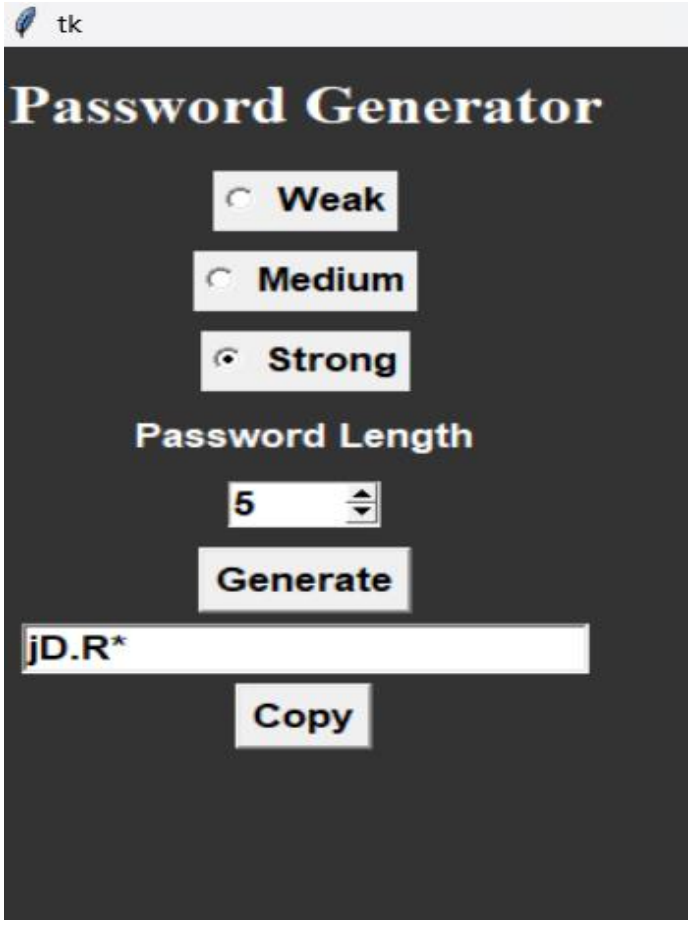
Conclusion:

In conclusion, the password manager project has successfully addressed the need for secure and convenient password management solutions. Through meticulous development, the application now offers essential features like password generation, secure storage, and organized retrieval. Security measures, including robust encryption techniques and hashing algorithms with salting, ensure passwords remain safeguarded from unauthorized access. The user interface prioritizes simplicity and intuitiveness, facilitating easy navigation and utilization of features for users of all levels. Integration of third-party libraries such as pyperclip enhances user experience by enabling seamless copying and pasting of passwords. Rigorous testing and user feedback have refined the application, addressing bugs and enhancing usability. Looking ahead, opportunities for further development include adding features like password strength evaluation and multi-factor authentication, expanding platform compatibility, and optimizing performance. Overall, the password manager project strives to empower users in safeguarding sensitive information and streamlining online activities, contributing to a safer digital environment.

Results:

In wrapping up, our password manager project has fulfilled its purpose of providing users with a secure and user-friendly solution for managing their passwords. Through diligent development, we've successfully implemented features such as password generation, secure storage, and organized retrieval. Our emphasis on security has resulted in the integration of robust encryption techniques and hashing algorithms with salting to ensure the protection of stored passwords. The user interface has been designed with simplicity in mind, allowing users of all levels to easily navigate and utilize the application's features. Additionally, the inclusion of third-party libraries like pyperclip has enhanced the user experience by facilitating seamless copying and pasting of passwords. Thorough testing and feedback have been instrumental in refining the application and addressing any issues. Looking ahead, there's room for further enhancement, including the addition of features


tk
Password Generator
Weak
Medium
Strong
Password Length
5
Generate
Copy


tk
Password Generator
Weak
Medium
Strong
Password Length
5
Generate
jD.R*
Copy

13

REFERENCES

Tkinter Documentation: https://docs.python.org/3/library/tkinter.html

W3Schools Tkinter Tutorial: https://www.w3schools.com/python/python_tkinter.asp

JetBrains PyCharm: https://www.jetbrains.com/pycharm/