MINI PROJECT REPORT

on

PASSWORD GENERATOR using Python

Submitted by

M N ADITYA **(RA2311053010090)**
M GIRIJESH (RA2311053010073)

Semester – II

Academic Year: 2023-24 Even

Under the guidance of

Dr. Arumbu V N
Assistant Professor, Department of ECE

In partial fulfilment for the Course

of

21CSS101J -PROGRAMMING FOR PROBLEM SOLVING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING College of Engineering and Technology,

SRM Institute of Science and Technology

SRM Nagar, Kattankulathur – 603203, Kancheepuram District, Tamil Nadu.

April 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this activity report for the course 21CSS101J -PROGRAMMING FOR PROBLEM SOLVING is the bonafide work of **M N ADITYA (RA2311053010090), M GIRIJESH (RA2311053010090)** who carried out the work under my supervision.

SIGNATURE

Dr. Arumbu V N

Department of ECE

Academic Coordinator

SRMIST Kattankulathur

SIGNATURE

Assistant Professor

TABLE OF CONTENTS

ABSTRACT

In the digital age, ensuring the security of online accounts has become paramount. This abstract introduces a Python-based solution for generating secure passwords and checking their strength. Leveraging Python's versatility and robust libraries, this system provides users with a reliable tool for creating strong passwords and evaluating their resilience against potential attacks.

Utilising Python's extensive ecosystem, the system employs cryptographic libraries and randomisation techniques to generate passwords that are resistant to brute-force and dictionary attacks. Users can specify the desired length and complexity of the password, ensuring it meets their security requirements.

Furthermore, the system incorporates a strength checker component that analyses the generated passwords and provides feedback on their strength. By assessing factors such as length, character diversity, and entropy, users can gauge the effectiveness of their passwords in thwarting unauthorised access attempts.

Additionally, the system offers features for real-time feedback and guidance, assisting users in crafting stronger passwords. Through intuitive user interfaces and informative prompts, the system aims to educate users on best practices for password security.

Moreover, leveraging Python's data processing capabilities, the system can analyze patterns and trends in password usage. Administrators can gain insights into common vulnerabilities and trends, enabling them to implement proactive measures for enhancing overall system security.

In conclusion, the development of a secure password generator and strength checker system using Python empowers users to safeguard their digital identities effectively. By combining robust algorithms, user-friendly interfaces, and data-driven insights, this system contributes to the advancement of password security practices in the digital realm.

Introduction

In the digital age, ensuring the security of online accounts has become increasingly crucial. With the rise of cyber threats and data breaches, the need for strong and unique passwords has never been greater. Password generators and strength checkers have emerged as indispensable tools for individuals seeking to fortify their online security.

Python, renowned for its versatility and extensive libraries, is at the forefront of password security solutions. Its simplicity and readability make it an ideal choice for developing robust password generation and strength evaluation systems. By harnessing Python's capabilities, developers can craft tailored solutions to meet the specific security needs of users.

This project delves into the realm of password security using Python, exploring the essential concepts of cryptographic algorithms, user interaction, and data processing. Through practical implementation, we will create a command-line application that functions as a password generator and strength checker, empowering users to generate secure passwords and assess their strength with ease.

By embarking on this project, we not only gain insights into Python's application in cybersecurity but also appreciate the importance of proactive measures in safeguarding digital identities. This project offers a valuable opportunity to explore the intersection of technology and security, equipping us with the knowledge and tools to enhance online safety. So let's dive into the world of password security with Python as our guide.

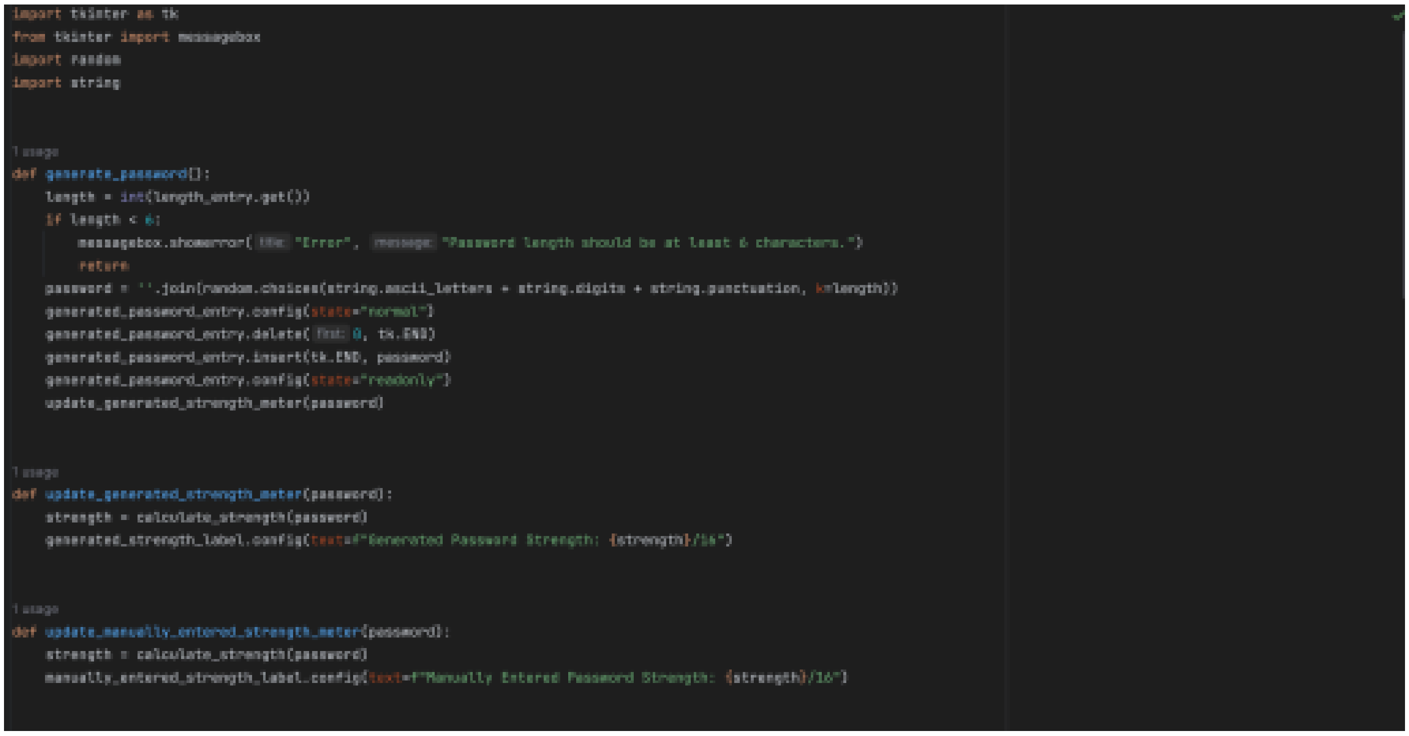
Objective

The objective of this project is to create a comprehensive and user-friendly password generator and strength checker system using the Python programming language. The goal is to streamline the process of generating secure passwords and evaluating their strength, thereby enhancing online security across various domains.

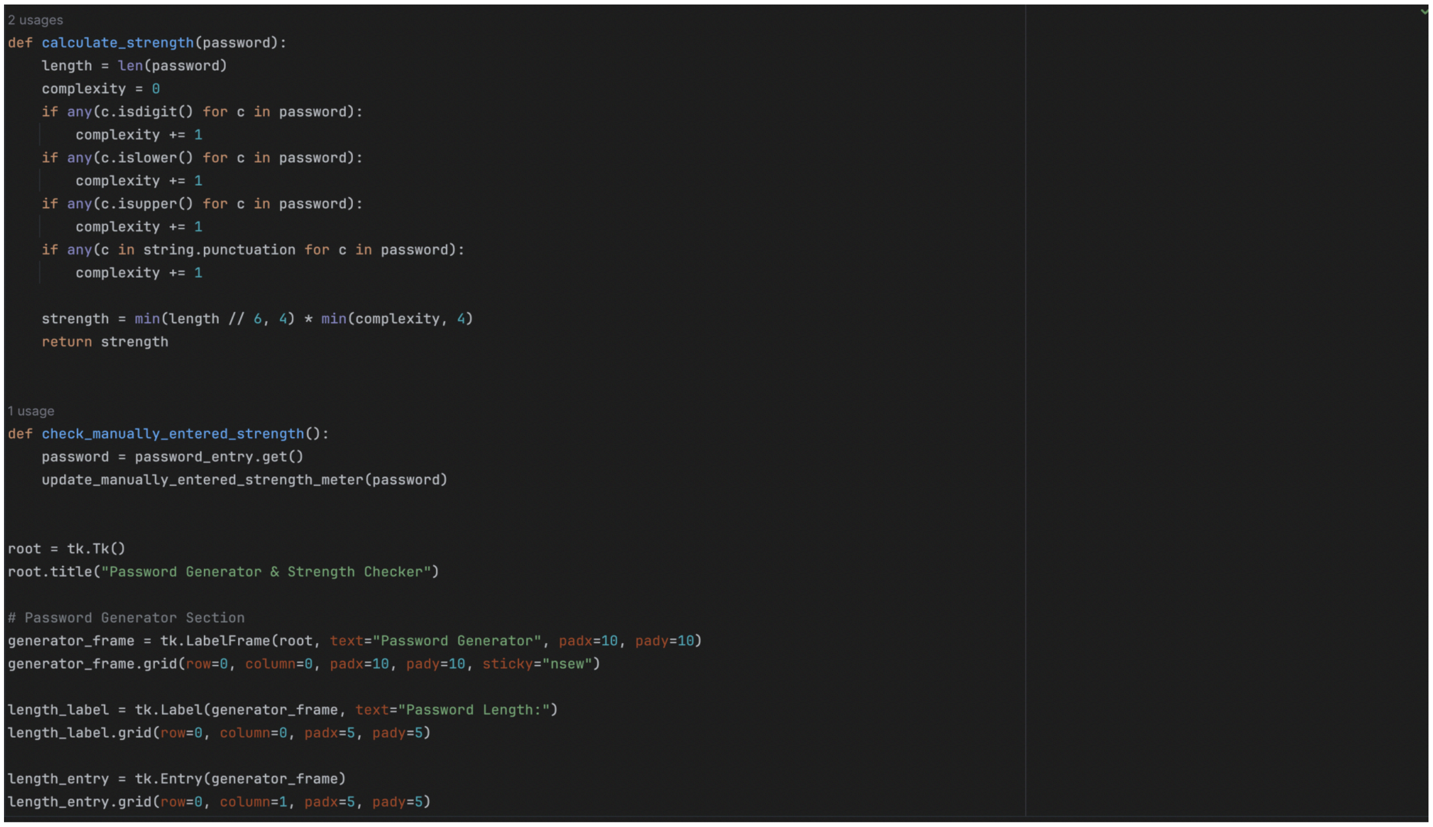
This project aims to leverage Python's robust libraries, frameworks, and data processing capabilities to develop a scalable, secure, and efficient platform for password management. By harnessing Python's versatility, the system will provide users with a seamless experience for generating strong passwords tailored to their specific requirements.

Furthermore, the system will include a strength checker component to assess the resilience of generated passwords against potential attacks. Through real-time feedback and insights, users will be empowered to make informed decisions about their password security.

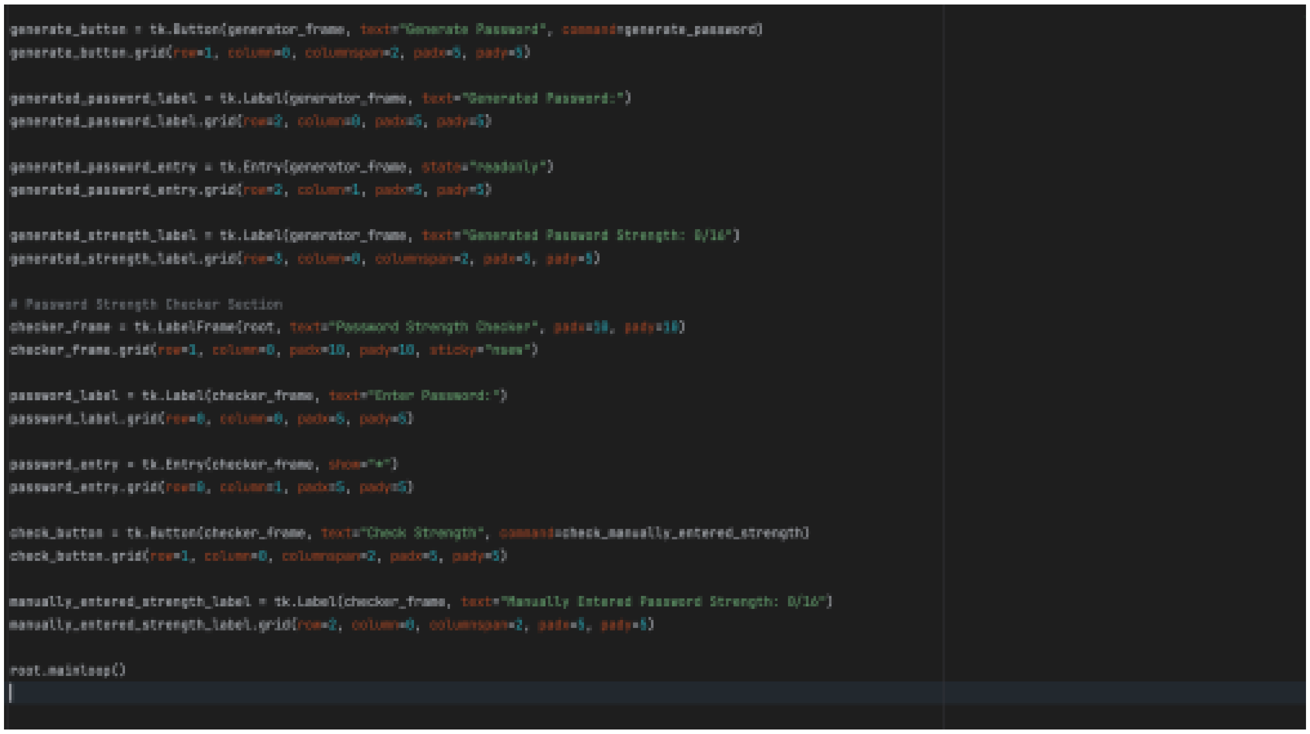
Ultimately, this project seeks to optimise password security practices in the digital era by offering a sophisticated yet accessible solution for generating and evaluating passwords. By combining Python's capabilities with intuitive design and efficient algorithms, the system aims to enhance overall online security and user confidence in password management.



```
import tkinter as tk
from tkinter import messagebox
import random
import string


1 usage
def generate_password():
    length = int(length_entry.get())
    if length < 6:
        messagebox.showerror( title: "Error",  message: "Password length should be at least 6 characters.")
        return
    password = ''.join(random.choices(string.ascii_letters + string.digits + string.punctuation, k=length))
    generated_password_entry.config(state="normal")
    generated_password_entry.delete( first: 0, tk.END)
    generated_password_entry.insert(tk.END, password)
    generated_password_entry.config(state="readonly")
    update_generated_strength_meter(password)


1 usage
def update_generated_strength_meter(password):
    strength = calculate_strength(password)
    generated_strength_label.config(text=f"Generated Password Strength: {strength}/16")


1 usage
def update_manually_entered_strength_meter(password):
    strength = calculate_strength(password)
    manually_entered_strength_label.config(text=f"Manually Entered Password Strength: {strength}/16")
```

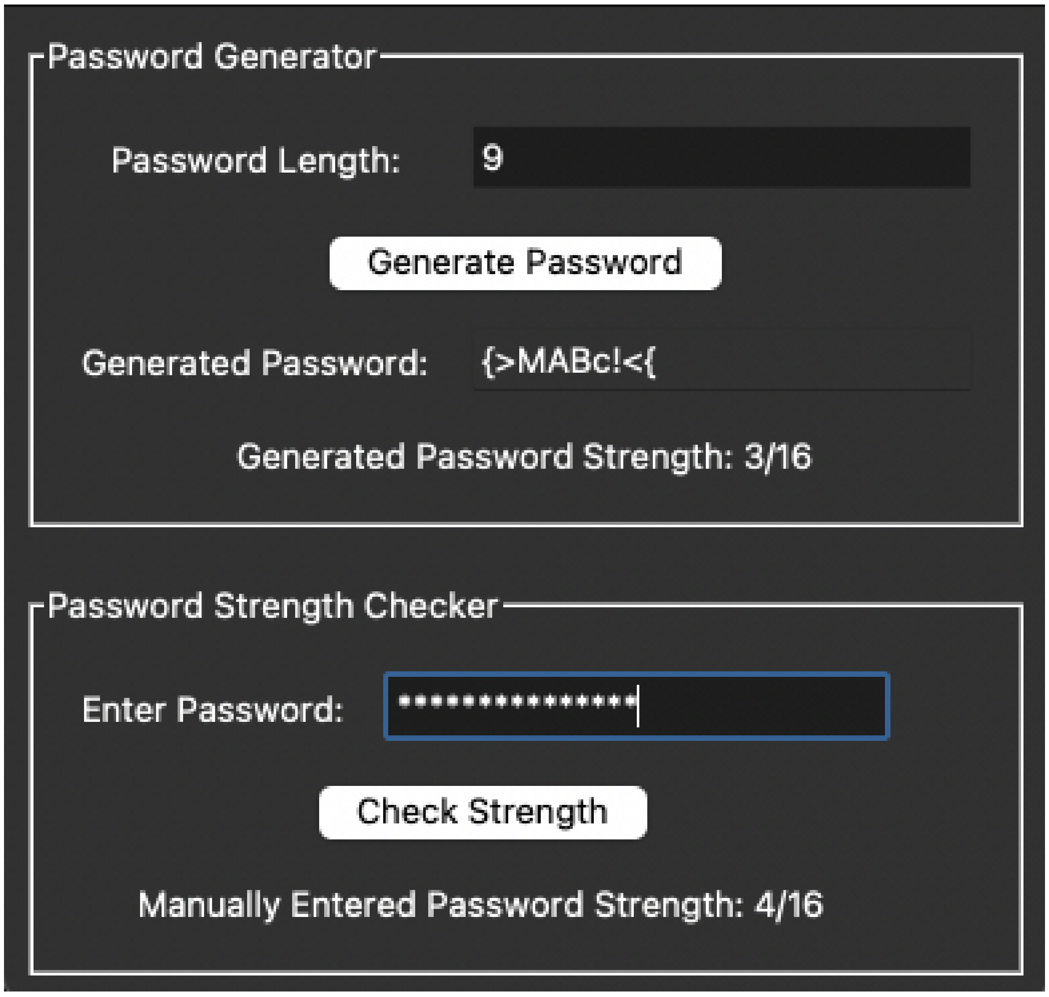


```
2 usages
def calculate_strength(password):
    length = len(password)
    complexity = 0
    if any(c.isdigit() for c in password):
        complexity += 1
    if any(c.islower() for c in password):
        complexity += 1
    if any(c.isupper() for c in password):
        complexity += 1
    if any(c in string.punctuation for c in password):
        complexity += 1

    strength = min(length // 6, 4) * min(complexity, 4)
    return strength


1 usage
def check_manually_entered_strength():
    password = password_entry.get()
    update_manually_entered_strength_meter(password)


root = tk.Tk()
root.title("Password Generator & Strength Checker")

# Password Generator Section
generator_frame = tk.LabelFrame(root, text="Password Generator", padx=10, pady=10)
generator_frame.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")

length_label = tk.Label(generator_frame, text="Password Length:")
length_label.grid(row=0, column=0, padx=5, pady=5)

length_entry = tk.Entry(generator_frame)
length_entry.grid(row=0, column=1, padx=5, pady=5)
```



```python
generate_button = tk.Button(generator_frame, text="Generate Password", command=generate_password)
generate_button.grid(row=1, column=0, columnspan=2, padx=5, pady=5)

generated_password_label = tk.Label(generator_frame, text="Generated Password:")
generated_password_label.grid(row=2, column=0, padx=5, pady=5)

generated_password_entry = tk.Entry(generator_frame, state="readonly")
generated_password_entry.grid(row=2, column=1, padx=5, pady=5)

generated_strength_label = tk.Label(generator_frame, text="Generated Password Strength: 0/16")
generated_strength_label.grid(row=3, column=0, columnspan=2, padx=5, pady=5)

# Password Strength Checker Section
checker_frame = tk.LabelFrame(root, text="Password Strength Checker", padx=10, pady=10)
checker_frame.grid(row=1, column=0, padx=10, pady=10, sticky="nsew")

password_label = tk.Label(checker_frame, text="Enter Password:")
password_label.grid(row=0, column=0, padx=5, pady=5)

password_entry = tk.Entry(checker_frame, show="*")
password_entry.grid(row=0, column=1, padx=5, pady=5)

check_button = tk.Button(checker_frame, text="Check Strength", command=check_manually_entered_strength)
check_button.grid(row=1, column=0, columnspan=2, padx=5, pady=5)

manually_entered_strength_label = tk.Label(checker_frame, text="Manually Entered Password Strength: 0/16")
manually_entered_strength_label.grid(row=2, column=0, columnspan=2, padx=5, pady=5)

root.mainloop()
```

RESULTS


Password Generator

Password Length: 9

Generate Password

Generated Password: {>MABc!<{

Generated Password Strength: 3/16

Password Strength Checker

Enter Password: ****************

Check Strength

Manually Entered Password Strength: 4/16

REFERENCES :

GOOGLE