Project Report: Random Password Generator1. Executive SummaryIn an era of increasing digitalization and online security concerns, the need for strong and unique passwords is paramount. The Random Password Generator project aims to address this need by creating a user-friendly program that generates random passwords with customizable length and complexity. This report outlines the objectives, features, implementation details, and potential future enhancements of the Random Password Generator.2. Project ObjectivesThe primary objectives of the Random Password Generator project are as follows:Develop a program that generates random passwords based on user-defined criteria.Allow users to specify the length of the password.Enable users to choose the complexity of the password by including options for uppercase letters, lowercase letters, numbers, and symbols.Ensure that generated passwords are secure and meet industry standards for strength.3. Features3.1 User InterfaceThe program will have a user-friendly interface that allows users to input their preferences for password generation. This includes specifying the length of the password and selecting the desired complexity options.3.2 Password GenerationThe core functionality of the program will be the random generation of passwords based on user-defined criteria. The algorithm will ensure that the generated passwords are both random and secure.3.3 CustomizationUsers will have the flexibility to customize the complexity of the generated passwords. They can choose to include or exclude uppercase letters, lowercase letters, numbers, and symbols.3.4 Copy to ClipboardTo enhance user convenience, a 'Copy to Clipboard' feature will be implemented, allowing users to quickly copy the generated password for immediate use.4. Implementation Details4.1 Programming LanguageThe Random Password Generator will be implemented using a programming language such as Python, known for its simplicity and effectiveness in handling randomization.4.2 Randomization AlgorithmThe password generation algorithm will utilize a secure randomization process, ensuring that the generated passwords are unpredictable and resistant to attacks.4.3 User Interface DesignThe user interface will be designed to be intuitive and user-friendly, with clear options for password customization and a prominent display of the generated password.4.4 Security ConsiderationsThe program will follow best practices for password security, including the use of a secure randomization algorithm and adherence to industry standards for password strength.5. Future EnhancementsThe Random Password Generator project has the potential for future enhancements, including:Integration with password management tools.Implementing additional customization options, such as excluding specific characters.Providing password strength feedback to users.6. ConclusionThe Random Password Generator project addresses the growing need for secure and customizable passwords. By providing a user-friendly interface and implementing a robust randomization algorithm, the program aims to contribute to improved online security practices. With future enhancements and continued development, the Random Password Generator can evolve to meet the evolving demands of digital security.

Code :

# import all the modules for the gui

from tkinter import ttk

from tkinter import messagebox

import random

import string

import tkinter as tk

class Password:

    # constructor that takes the length of the password

    def \_\_init\_\_(self, length):

        self.length = length

        self.password = ""

    def \_\_init\_\_(self):

        self.length = random.randint(8, 16)

        self.password = ""

    # method to generate the password

    def generate(self):

        # get all the characters

        chars = string.ascii\_letters + string.digits + string.punctuation

        # generate the password

        for \_ in range(self.length):

            self.password += random.choice(chars)

        return self.password

    # method to get the password

    def get\_password(self):

        return self.password

    # method to get the length of the password

    def get\_length(self):

        return self.length

    # method to set the length of the password

    def set\_length(self, length):

        self.length = length

    # method for checking the strength of the password

    def check\_strength(self):

        # enum for the strength of the password

        class Strength:

            WEAK = 1

            MEDIUM = 2

            STRONG = 3

        # set the strength to weak of the enum

        strength = Strength.WEAK

        # check for the length of the password

        if self.length >= 8 and self.length <= 16:

            strength = Strength.MEDIUM

        elif self.length > 16:

            strength = Strength.STRONG

        # check for the digits in the password

        if any(char.isdigit() for char in self.password):

            strength = Strength.MEDIUM

        if any(char.isdigit() for char in self.password) and self.length > 16:

            strength = Strength.STRONG

        # check for the uppercase characters in the password

        if any(char.isupper() for char in self.password):

            strength = Strength.MEDIUM

        if any(char.isupper() for char in self.password) and self.length > 16:

            strength = Strength.STRONG

        # check for the lowercase characters in the password

        if any(char.islower() for char in self.password):

            strength = Strength.MEDIUM

        if any(char.islower() for char in self.password) and self.length > 16:

            strength = Strength.STRONG

        # check for the special characters in the password

        if any(char in string.punctuation for char in self.password):

            strength = Strength.MEDIUM

        if any(char in string.punctuation for char in self.password) and self.length > 16:

            strength = Strength.STRONG

        # return the strength of the password as a string

        if strength == Strength.WEAK:

            return "Weak"

        elif strength == Strength.MEDIUM:

            return "Medium"

        elif strength == Strength.STRONG:

            return "Strong"

    # method to clear the password

    def clear(self):

        self.password = ""

        self.length = 0

# class for the gui

class GUI:

    # constructor

    def \_\_init\_\_(self):

        # create the window

        self.window = tk.Tk()

        self.window.title("Password Generator")

        self.window.geometry("400x300")

        self.window.resizable(False, False)

        # create the frame

        self.frame = ttk.Frame(self.window)

        self.frame.pack()

        # create the label

        self.label = ttk.Label(self.frame, text="Password Generator")

        self.label.pack()

        # create the entry

        self.entry = ttk.Entry(self.frame, width=30)

        self.entry.pack()

        # create the button

        self.button = ttk.Button(self.frame, text="Generate", command=self.generate)

        self.button.pack()

        # create the strength label

        self.strength\_label = ttk.Label(self.frame, text="Strength: ")

        self.strength\_label.pack()

        # create the strength entry

        self.strength\_entry = ttk.Entry(self.frame, width=30)

        self.strength\_entry.pack()

        # create the copy button

        self.copy\_button = ttk.Button(self.frame, text="Copy", command=self.copy)

        self.copy\_button.pack()

        # create the clear button

        self.clear\_button = ttk.Button(self.frame, text="Clear", command=self.clear)

        self.clear\_button.pack()

        # create the exit button

        self.exit\_button = ttk.Button(self.frame, text="Exit", command=self.exit)

        self.exit\_button.pack()

        # create a length label with a menu

        self.length\_label = ttk.Label(self.frame, text="Length: ")

        self.length\_label.pack()

        # create a length dropdown menu

        self.length\_spinbox = tk.Spinbox(self.frame, from\_=8, to=32, width=5)

        self.length\_spinbox.pack()

        # create the password object

        self.password = Password()

        # start the main loop

        self.window.mainloop()

    # method to generate the password

    def generate(self):

        # clear the password

        self.password.clear()

        # get the length from the spinbox

        length = int(self.length\_spinbox.get())

        # set the length of the password

        self.password.set\_length(length)

        # generate the password

        self.password.generate()

        # set the password in the entry

        self.entry.delete(0, "end")

        self.entry.insert(0, self.password.get\_password())

        # set the strength of the password

        self.strength\_entry.delete(0, "end")

        self.strength\_entry.insert(0, self.password.check\_strength())

    # method to copy the password

    def copy(self):

        # copy the password to the clipboard

        self.window.clipboard\_clear()

        self.window.clipboard\_append(self.password.get\_password())

        # show a message box

        messagebox.showinfo("Password Generator", "Password copied to clipboard")

    # method to clear the password

    def clear(self):

        # clear the password

        self.password.clear()

        # clear the entry

        self.entry.delete(0, "end")

        self.strength\_entry.delete(0, "end")

    # method to exit the program

    def exit(self):

        # exit the program

        self.window.destroy()

# call the main function

def main():

    gui = GUI()

if \_\_name\_\_ == "\_\_main\_\_":

    main()