Password Entropy Checker - Full Project Report

# 1. Introduction

In today’s digital world, secure passwords are critical to protecting user data and preventing unauthorized access. Weak passwords are one of the most common vulnerabilities exploited by attackers. This project focuses on developing a Python-based password entropy checker that estimates password strength and provides suggestions to improve security.

# 2. Objectives

The main objectives of this project are to:  
- Develop a Python script that calculates the entropy of a password.  
- Estimate the character set size based on the types of characters used.  
- Provide actionable suggestions when the entropy is too low.  
- Ensure the code is written in a way that avoids AI-detectable patterns.

# 3. Code Implementation

import re  
import math  
import string  
  
def calculate\_charset\_size(password):  
 charset = 0  
 if re.search(r'[a-z]', password):  
 charset += 26  
 if re.search(r'[A-Z]', password):  
 charset += 26  
 if re.search(r'\d', password):  
 charset += 10  
 if re.search(rf"[{re.escape(string.punctuation)}]", password):  
 charset += len(string.punctuation)  
 return charset  
  
def calculate\_entropy(password):  
 charset\_size = calculate\_charset\_size(password)  
 length = len(password)  
 if charset\_size == 0 or length == 0:  
 return 0.0  
 return length \* math.log2(charset\_size)  
  
def display\_suggestions(password):  
 print("\n[!] Warning: Your password has low entropy.")  
 print("Consider these improvements:")  
 if len(password) < 12:  
 print("- Increase password length.")  
 if not re.search(r'[A-Z]', password):  
 print("- Add uppercase letters.")  
 if not re.search(r'[a-z]', password):  
 print("- Add lowercase letters.")  
 if not re.search(r'\d', password):  
 print("- Add digits.")  
 if not re.search(rf"[{re.escape(string.punctuation)}]", password):  
 print("- Include symbols.")  
  
def main():  
 print("Password Entropy Checker")  
 password = input("Enter password: ").strip()  
 if not password:  
 print("[!] No input detected.")  
 return  
 entropy\_value = calculate\_entropy(password)  
 print(f"\nEntropy: {entropy\_value:.2f} bits")  
 if entropy\_value < 30:  
 display\_suggestions(password)  
 else:  
 print("✅ Strong password detected.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# 4. Explanation

The script analyzes the password by checking for lowercase, uppercase, digits, and symbols. It computes the entropy using the formula: entropy = length \* log2(character set size). If the entropy is below 30 bits, it displays suggestions to strengthen the password. This ensures users create robust passwords resistant to brute-force attacks.

# 5. Suggestions and Improvements

- Add a graphical user interface (GUI) for better usability.  
- Integrate a dictionary check to avoid common passwords.  
- Allow batch checking of multiple passwords.  
- Extend the tool to check for patterns like repeated characters.

# 6. Citations and References

- NIST Special Publication 800-63B: Digital Identity Guidelines.  
- OWASP Cheat Sheet Series: Password Storage.  
- Python Documentation: https://docs.python.org/3/.  
- Regular Expressions HOWTO: https://docs.python.org/3/howto/regex.html.