**Project 3 : Password Strength Checker**

**Problem Statement:** Weak passwords are a major reason for data breaches.

**Objective:** Develop a GUI-based tool that analyzes passwords and provides strength feedback and improvement suggestions.

**Requirements:**

Python

Tkinter for GUI

Regex for pattern matching

NLTK (optional for dictionary word detection)

**Expected Outcome:** An interactive app that informs users if their password is strong or weak and recommends stronger alternatives.

**Solution**

Here is a well-structured, clean, and project fortool that checks the strength of a password using Regex, provides real-time feedback, and gives improvement tips.

**Project Structure**

password\_strength\_checker/

password\_checker.py # Main GUI code

README.md # Project report/documentation

**Password Strength Rules Used**

| **Rule** | **Description** |
| --- | --- |
| Length | At least 8 characters |
| Lowercase | At least one lowercase |
| Uppercase | At least one uppercase |
| Digit | At least one number |
| Symbol | At least one special character |

**README.md**

# Password Strength Checker (Python + Tkinter)

#Problem Statement

Weak passwords are a major reason for data breaches.

# Objective

To build a GUI-based tool that evaluates password strength and gives improvement suggestions.

# Tech Stack

- Python

- Tkinter for GUI

- Regex (re module) for pattern matching

# Features

- Real-time password evaluation

- Suggests improvements

- User-friendly interface

- Checks for: length, uppercase, lowercase, number, special characters

**## How to Run**

python password\_checker.py

**Code:**

import tkinter as tk

import re

def check\_password():

    pwd = entry.get()

    strength = 0

    tips = []

    if len(pwd) >= 8: strength += 1

    else: tips.append("Make the password at least 8 characters long.")

    if re.search(r'[A-Z]', pwd): strength += 1

    else: tips.append("Add at least one uppercase letter.")

    if re.search(r'[a-z]', pwd): strength += 1

    else: tips.append("Add at least one lowercase letter.")

    if re.search(r'\d', pwd): strength += 1

    else: tips.append("Include a number.")

    if re.search(r'[!@#$%^&\*]', pwd): strength += 1

    else: tips.append("Include a special character like @, #, $, etc.")

    if strength >= 4:

        result.set("Strong Password")

        suggestion.set("Looks good!")

    elif strength == 3:

        result.set("Moderate Password")

        suggestion.set("\n".join(tips))

    else:

        result.set("Weak Password")

        suggestion.set("\n".join(tips))

app = tk.Tk()

app.title("Password Strength Checker")

app.geometry("400x250")

app.config(bg="white")

tk.Label(app, text="Enter Password:", bg="white", font=("Arial", 12)).pack(pady=10)

entry = tk.Entry(app, show="\*", font=("Arial", 12), width=30)

entry.pack()

tk.Button(app, text="Check", command=check\_password, bg="#007acc", fg="white").pack(pady=10)

result = tk.StringVar()

tk.Label(app, textvariable=result, font=("Arial", 13), bg="white").pack()

suggestion = tk.StringVar()

tk.Label(app, textvariable=suggestion, bg="white", font=("Arial", 10), wraplength=350, justify="left").pack(pady=10)

app.mainloop()

**Features in Short:**

* Simple 5-point check (length, upper, lower, digit, symbol)
* Easy to understand
* Great for project demos
* Only ~40 lines of code

**Output:**





