Project Title: Password Strength Analyzer

Introduction:

This project is a command-line tool, a type of program you run directly from your computer's terminal. Its main goal is to help you figure out how strong a password is and why it might be weak. Think of it as a helpful security assistant that can give you advice on how to make your passwords harder for hackers to guess.

Purpose of the Tool:

Most people know that a good password needs to be long and use a mix of letters, numbers, and symbols. But a common password like "password123!" follows all these rules and is still very weak. Why? Because hackers don't just guess randomly; they use smart programs that try millions of common and leaked passwords. This project was built to show that a truly secure password isn't just about length, it's about being unpredictable.

Functionality:

This tool is made of a few key parts that work together:

1. The Brain: The zxcvbn Engine
    - What it does: Instead of just checking if a password is long enough, this part of the program uses a powerful open-source library called zxcvbn. This library acts like a smart hacker, checking for common patterns. It looks for things like:
        - Common words: Is the password a word found in a dictionary?
        - Personal information: Does it contain names, birthdays, or other personal details?
        - Common patterns: Does it have simple sequences like "abc" or "123"?
        - Keyboard patterns: Does it follow a pattern on a keyboard, like "qwerty"?
    - The Score: The zxcvbn engine then gives a score from 0 (very weak) to 4 (very strong). We take that score and change it to a more understandable scale of 1-10, where a score of 10 means the password is very secure.
2. The Security Tester: Custom Wordlist Generation
    - What it does: This is a unique and important part of the tool. It creates a list of passwords that are easily guessed.
    - How it works: You give the program some personal information, such as a name, a date, and a pet's name. The tool then automatically creates hundreds of password combinations using that information, with variations like Leetspeak (a becomes 4, o becomes 0), capitalization, and adding common years.
    - Why it's important: This feature is not about creating good passwords. It's about demonstrating how easily a hacker could generate a list of weak passwords that are

based on information found on social media or elsewhere. It proves that using personal details in your password is a big security risk.

3. The Organizer: Command-Line Interface (CLI) 💻
    - What it does: The program uses a part of Python called argparse to create a simple menu you can use in your terminal. This makes it easy to tell the program what you want it to do.
    - How it works: Instead of clicking buttons, you type simple commands like python password_analyzer.py --password "example". This allows you to choose between analysing a single password, checking a whole list of passwords from a file (batch mode), or generating a wordlist for testing.

Technical Advantages:

This project demonstrates several key skills that are important for anyone working with software and technology:

- Python Programming: Writing clean and efficient code in Python.
- Command-Line Tool Development: Building a program that can be used directly from a terminal, a fundamental skill in software development.
- Using External Libraries: Learning how to use a pre-built tool (zxcvbn) to add powerful features to your own program.
- File Handling: The ability to read and write information to files, which is essential for tasks like reading a list of passwords or saving a report.

Conclusion:

This project is more than just a simple program; it's a practical example of a cybersecurity tool. It shows that I can not only write code but also apply security principles to build something useful and educational. It’s a great piece to show to future employers because it proves I can build a functional application and understand the important concepts behind it.

Results:

1. Executing single password.
2. Executing the suggested password to check whether it is succussed in generating the strong password.
3. Executing the all the passwords in the passwords.txt as a batch.

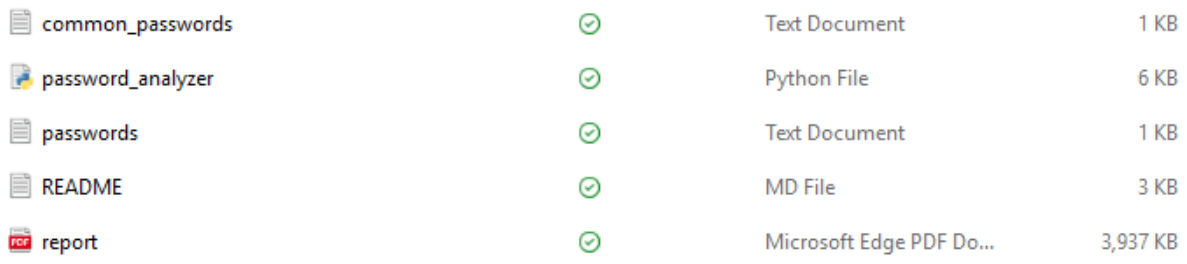
```
PS D:\password_analyzer_project> ls


    Directory: D:\password_analyzer_project


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
------        17-08-2025     16:07            200 common_passwords.txt
-a----        18-08-2025     00:22             94 passwords.txt
-a----        18-08-2025     00:32           5685 password_analyzer.py
-a----        18-08-2025     00:00           2116 README.md
-a----        17-08-2025     23:29        4031200 report.pdf


PS D:\password_analyzer_project> python3 password_analyzer.py -p "pass123"
Password: pass123
Analysis: Password is too weak. Score: 4/10. Suggestions: Add another word or two. Uncommon words are better.
Suggestions:
  - pass12d3
  - p!ass123
  - passi123
PS D:\password_analyzer_project> python3 password_analyzer.py -p "pass12d3"
Password: pass12d3
Analysis: Password is strong and secure. Score: 6/10.
PS D:\password_analyzer_project> python3 password_analyzer.py --batch passwords.txt
123456 -> This password is in the top leaked list. Very weak.
Password@2025 -> Password is strong and secure. Score: 6/10.
hello123 -> Password is too weak. Score: 2/10. Suggestions: Add another word or two. Uncommon words are better.
qwerty -> This password is in the top leaked list. Very weak.
StrongPass!2025 -> Password is strong and secure. Score: 10/10.
letmein -> This password is in the top leaked list. Very weak.
Test@123 -> Password is strong and secure. Score: 6/10.
password -> This password is in the top leaked list. Very weak.
MySecurePwd99! -> Password is strong and secure. Score: 10/10.
PS D:\password_analyzer_project>
```

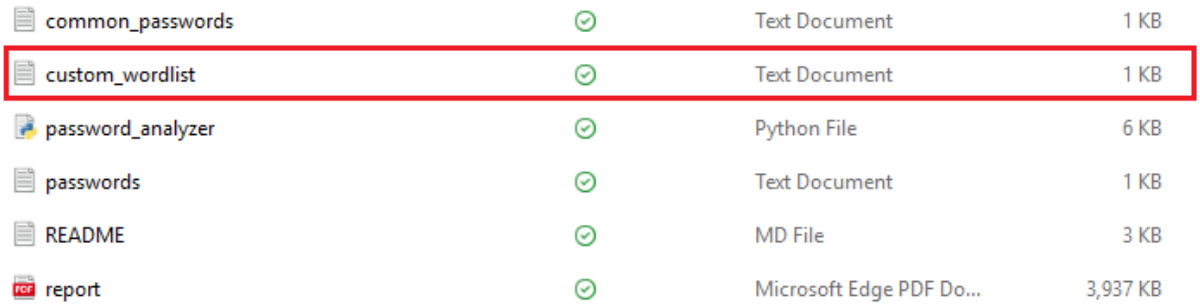
4. Existing files before generation of custom wordlist file.



5. Executing the custom wordlist generation command.

```
PS D:\password_analyzer_project> python3 password_analyzer.py --generate_wordlist --name kali --date 3-1-2000 --pet jimmy
Custom wordlist generated in 'custom_wordlist.txt'
PS D:\password_analyzer_project>
```

6. Generation of custom wordlist is success.



7.Generated Custom wordlist.

```
3-1-2000
3-1-20001999
3-1-20002000
3-1-20002024
3-1-20002025
Jimmy
Kali
j1mmy
j1mmy1999
j1mmy2000
j1mmy2024
j1mmy2025
jimmy
jimmy1999
jimmy2000
jimmy2024
jimmy2025
k411
k4111999
k4112000
k4112024
k4112025
kali
kali1999
kali2000
kali2024
kali2025
```