Password Strength Analyzer with Custom Wordlist Generator

Objective:

This project is designed to help users evaluate the strength of their passwords and generate custom wordlists for cybersecurity research, password audits, and penetration testing. It supports both GUI and CLI interfaces.

Tools Used:

- Python
- zxcvbn (for password strength estimation)
- NLTK (for dictionary filtering)
- Tkinter (for GUI)
- argparse (for CLI)

Functionalities:

1. Analyze password strength using zxcvbn.

2. Accept user details (name, pet name, year) to generate a personalized wordlist. 3. Apply common password mutation patterns (e.g., leetspeak, appending years). 4. Export generated wordlist in .txt format.

5. Provide GUI using Tkinter and a command-line interface via argparse.

Implementation Summary

The program consists of a main Python file (main.py) which detects CLI arguments. If provided, it runs in terminal mode; if not, it launches the GUI. Password strength is evaluated using zxcvbn and feedback is printed or displayed accordingly.

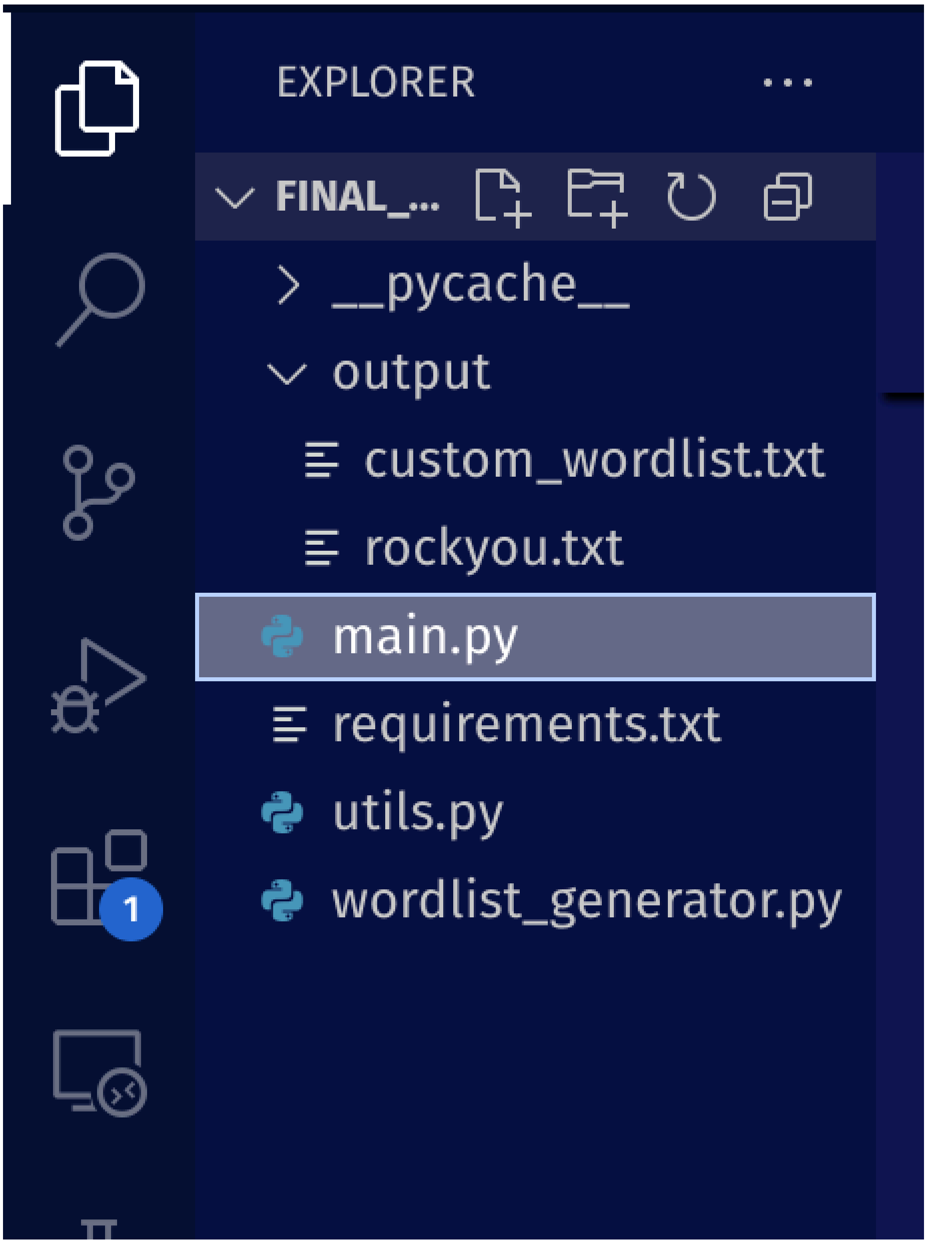
The wordlist is generated using the user's input and basic leetspeak transformations. The list is filtered against the NLTK dictionary to improve quality and remove meaningless entries.

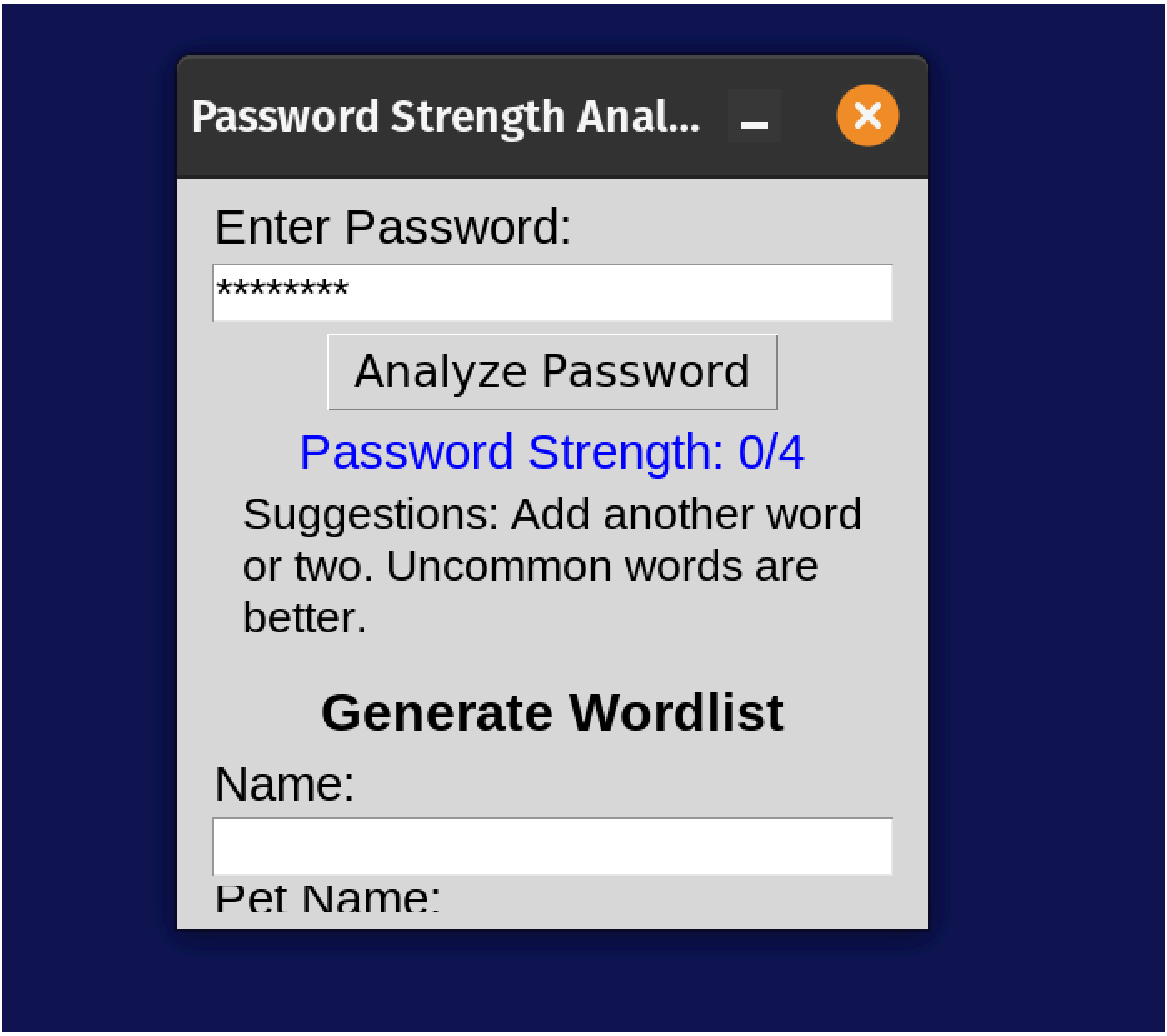
Output:

- Password score and feedback (printed or shown in GUI).
- Wordlist saved as output/custom_wordlist.txt.

Conclusion:

This tool enhances password awareness and helps simulate real-world attacks by showing how weak or guessable passwords may be constructed. It also provides cybersecurity students with a practical Python-based tool useful for learning and demonstrations.


EXPLORER
...
FINAL_...
__pycache__
output
custom_wordlist.txt
rockyou.txt
main.py
requirements.txt
utils.py
wordlist_generator.py
1



Enter Password:

****

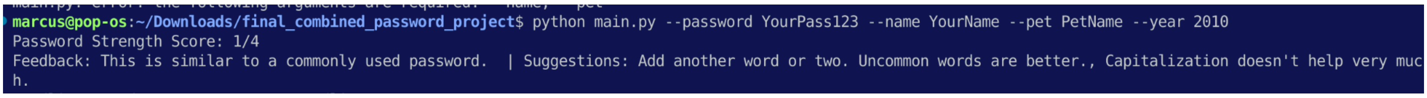
Password Strength: 0/4

Suggestions: Add another word or two. Uncommon words are better.

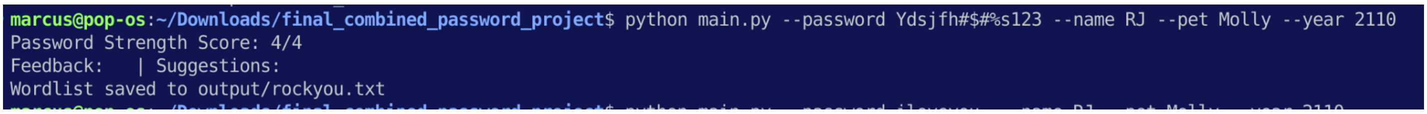
Generate Wordlist

Name:

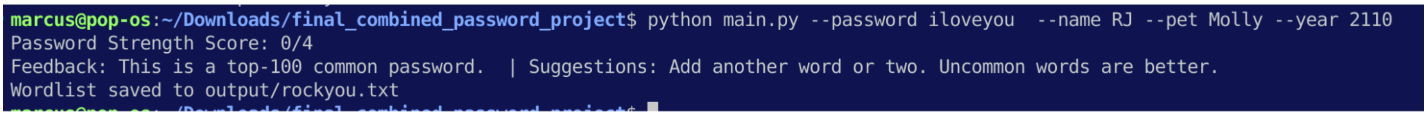
Pet Name:



```
main.py: error: the following arguments are required: --name, --pet
marcus@pop-os:~/Downloads/final_combined_password_project$ python main.py --password YourPass123 --name YourName --pet PetName --year 2010
Password Strength Score: 1/4
Feedback: This is similar to a commonly used password.  | Suggestions: Add another word or two. Uncommon words are better., Capitalization doesn't help very muc
h.
```



```
marcus@pop-os:~/Downloads/final_combined_password_project$ python main.py --password Ydsjfh#$#%s123 --name RJ --pet Molly --year 2110
Password Strength Score: 4/4
Feedback:   | Suggestions:
Wordlist saved to output/rockyou.txt
marcus@pop-os:~/Downloads/final_combined_password_project$ python main.py --password iloveyou  --name RJ --pet Molly --year 2110
```



```
marcus@pop-os:~/Downloads/final_combined_password_project$ python main.py --password iloveyou  --name RJ --pet Molly --year 2110
Password Strength Score: 0/4
Feedback: This is a top-100 common password.  | Suggestions: Add another word or two. Uncommon words are better.
Wordlist saved to output/rockyou.txt
marcus@pop-os:~/Downloads/final_combined_password_project$
```