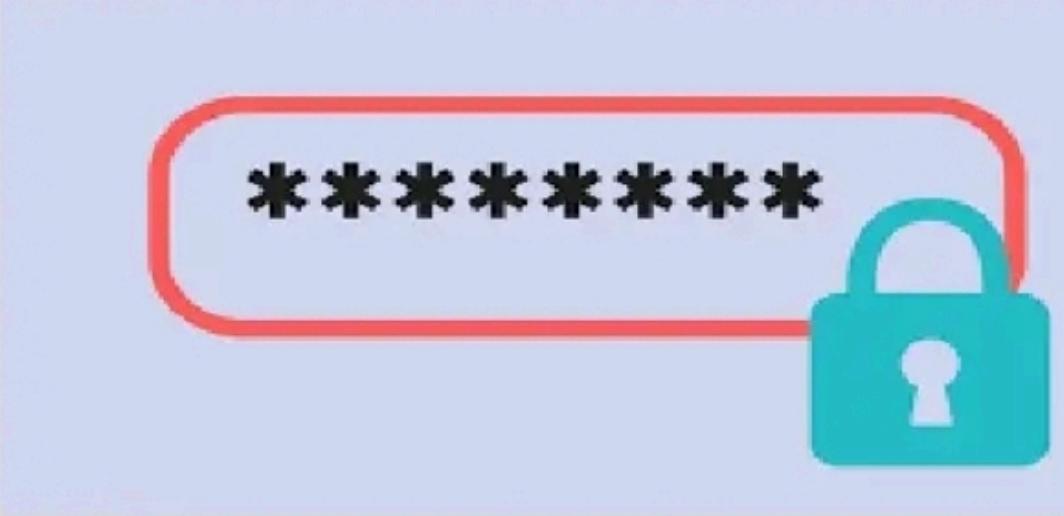


Introduction

- Passwords are the first line of defense in digital security.
- Weak passwords are vulnerable to attacks.
- This project evaluates password strength and guides users.





Methodology

- - Input: User-entered password.
- - Evaluation Criteria:
- - Length
- - Uppercase/lowercase
- - Numbers & special characters
- - Avoid common patterns
- - Output: Score and suggestions.



Tools and Technologies Used

- - Programming Language: Python
- - Libraries: re (Regex), tkinter (GUI)
- - Optional: Web version using HTML, CSS, JS



System Architecture

- Input Layer → Validation Logic → Result Display

```
total 12.53 .
4. Sep 15:53 ..
0. Sep 2015 bin -> usr/bin
19. Sep 09:31 boot
21. Sep 15:50 dev
19. Sep 09:32 etc
21. Sep 15:52 home
7 30. Sep 2015 lib -> usr/lib
34 23. Jul 10:01 lib64 -> usr/lib
96 1. Aug 22:45 lost+found
396 30. Sep 2015 mnt
16 21. Sep 15:52 opt
6 21. Sep 08:15 private -> /home/encrypted
4096 12. Aug 15:37 proc
560 21. Sep 15:58 root
7 30. Sep 2015 run
4096 30. Sep 2015 sbin -> usr/bin
6 21. Sep 15:51 sys
300 21. Sep 15:45 tmp
4096 12. Aug 15:39 usr
4096 23. Jul 10:25 var
4096 21. Sep 15:54
```

Password Strength Criteria

- Criteria:
- - Length ≥ 8 (+2)
- - Uppercase letter (+1)
- - Lowercase letter (+1)
- - Digit (+1)
- - Special character (+1)
- - No dictionary words (+1)

The screenshot shows a 'Password Verification' window. At the top, there are fields for 'Username' (containing 'user') and 'Password' (containing '*****'). Below these fields is a message: 'Password must meet the following requirements:' followed by a list of criteria with green checkmarks and red X's:

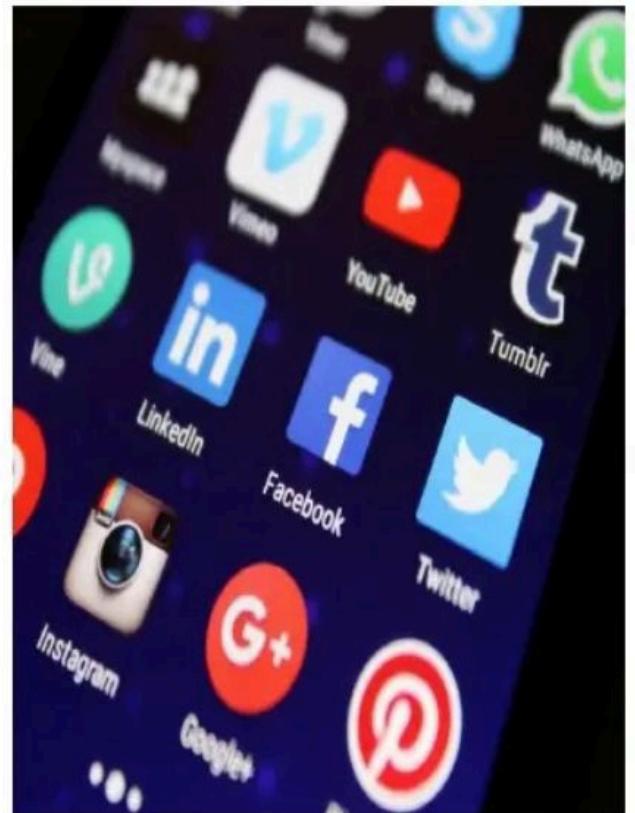
- ✓ At least one letter
- ✗ At least one capital letter
- ✓ At least one number
- ✗ Be at least 8 characters

Strength Levels

- - Weak: Score ≤ 3
- - Moderate: Score 4–5
- - Strong: Score 6+

```
.. Sep 15:53 .
. Sep 15:53 ..
0. Sep 2015 bin -> usr/bin
19. Sep 09:31 boot
21. Sep 15:50 dev
21. Sep 09:32 etc
1. 30. Sep 15:52 home
7. 30. Sep 2015 lib -> usr/lib
34. 23. Jul 10:01 lib64 -> usr/lib
96. 1. Aug 22:45 lost+found
896. 30. Sep 2015 mnt
16. 21. Sep 15:52 opt
9. 21. Sep 08:15 private -> /home/encrypted
4096. 12. Aug 15:37 proc
568. 21. Sep 15:50 root
7. 30. Sep 15:50 run
4096. 30. Sep 2015 sbin -> usr/bin
9. 21. Sep 15:51 sys
308. 21. Sep 15:45 usr
1. 4096. 12. Aug 15:39 var
14. 4096. 23. Jul 10:25 var

100% 4096 31. Sep 15:52
```



Sample Output

- Input: P@ssw0rd123
- Score: 6
- Strength: Strong
- Suggestion: Avoid common words like 'password'.



Results and Testing

- - Tested with various passwords.
- - Accurately identifies password strength.
- - Improved user awareness of password security.



Challenges Faced

- - Handling common patterns and dictionary words.
- - Balancing complexity and user-friendliness.
- - Educating users effectively



Future Enhancements

- - Integrate with password manager
- - AI-based analysis.
- - Multilingual feedback.
- - Browser extension/mobile app.



Conclusion

- - Helps users improve password hygiene.
- - Encourages better cybersecurity practices.
- - Can be extended to secure web applications.



Acknowledgement

- I would like to express my sincere gratitude to everyone who supported me throughout this project.
- Special thanks to my guide/mentor, Suraj Yadav Sir whose valuable insights and guidance helped me successfully complete this Password Strength Checker project.
- I also appreciate the support from my friends for their encouragement and motivation.
- Finally, I thank all the resources and tools that made this work possible.

References

- - OWASP Guidelines
- - NIST Digital Identity Guidelines
- - Python Documentation

