CAPSTONE PROJECT

STRONG PASSWORD GENERATOR USING AI

PRESENTED BY

STUDENT NAME: S.SRIVALLI

COLLEGE NAME: MALLAREDDY ENGINEERING COLLEGE FOR WOMEN

DEPARTMENT: CSE(CYBER SECURITY)

EMAIL ID: S.SRIVALLI0511@GMAIL.COM

AICTE STUDENT ID: STU6662CFC511AD81717751749

OUTLINE

PROBLEM STATEMENT

In today's digital era, ensuring password security is critical for safeguarding personal and organizational data. Many users tend to create weak or repetitive passwords, making systems vulnerable to cyber threats such as brute-force attacks or credential stuffing. The problem becomes more acute when individuals do not follow the best practices of combining character types or setting adequate password lengths. This creates a significant security gap that can compromise sensitive information.

Moreover, a lack of awareness or discipline in following password creation guidelines—such as using a mix of characters, setting sufficient lengths, or avoiding common patterns—further increases the vulnerability of user accounts. This growing concern highlights a widespread issue in maintaining personal and organizational cybersecurity due to poor password practices.

PROPOSED SOLUTION

The issue of weak and reused passwords can be addressed through the development of an automated Strong Password Generator. This tool generates highly secure passwords using randomized combinations of characters, which can help users adopt safer practices without the cognitive load of manually creating or remembering complex strings. The solution is built as a lightweight Python application with a user-friendly interface.

- **DATA COLLECTION:**

This project does not use a dataset. Instead, it programmatically builds a "character pool" by collecting a predefined set of character types — uppercase letters, lowercase letters, digits, and special symbols — using Python's string module. These serve as the source for generating random combinations.

- **DATA PREPROCESSING**

No conventional data preprocessing is required. However, input validation is implemented to ensure that user inputs (e.g., desired password length) are acceptable and within a secure range. Internally, the application ensures a balanced representation of all character types in the generated password to maintain security strength.

- **MACHINE LEARNING ALGORITHM:**

This project does not involve machine learning algorithms. It relies on randomization logic using Python's random.sample() and random.shuffle() functions to generate unpredictable password strings. This manual logic mimics randomness, which is often used in cryptographic and security systems.

- **DEPLOYMENT:**

The application is deployed as a standalone Python program with a Tkinter GUI interface. Users can run the .py file locally, interact with a simple interface to choose the desired password length, and click a button to generate and copy the password.

- **EVALUATION:**

The generated passwords can be evaluated using entropy calculations or online password strength meters. Informally, evaluation can be done by verifying that each password includes at least one character from each category (uppercase, lowercase, digit, symbol) and is of a minimum secure length (typically ≥12 characters).

SYSTEM APPROACH

The Strong Password Generator follows a modular and user-driven design using Python. It integrates a Tkinter-based GUI with backend logic to generate secure passwords based on user-defined length.

User Interface (Tkinter):

Users enter the desired password length and click a button to generate the password. The GUI handles all inputs and displays outputs.

Password Generation:

Using the random module, characters are sampled and shuffled to ensure unpredictability. The logic ensures inclusion of varied character types for strength.

Output Display & Clipboard Copy:

The password is shown in the interface and optionally copied to the clipboard using pyperclip.

Input Validation:

Basic validation ensures users input a valid, secure length, preventing errors or weak outputs.

Extensibility:

The project is lightweight and can be expanded for web, mobile, or browser extensions with minimal changes.

SYSTEM REQUIREMENTS

- **HARDWARE REQUIREMENTS:**

  - ➢ Processor: Intel/AMD Dual Core or above
  - ➢ RAM: Minimum 2 GB (4 GB recommended)
  - ➢ Storage: Minimum 100 MB free space
  - ➢ Operating System: Windows, macOS, or any Linux distribution.

- **SOFTWARE REQUIREMENTS:**

  - ➢ Python (version 3.6 or above)
  - ➢ Tkinter library (comes bundled with standard Python installations)

LIBRARIES:

Library	Purpose
tkinter	For building the graphical user interface (GUI)
random	To randomly select and shuffle characters
string	To provide predefined sets of characters (letters, digits, punctuation)
pyperclip(optional, if used)	For copying the password to the clipboard

ALGORITHM & DEPLOYMENT

The project uses a randomization algorithm instead of a traditional machine learning algorithm. Functions like random.sample() and random.shuffle() help ensure that the output is both unpredictable and secure.

Algorithm Selection:

No ML algorithm is used. Instead, a randomized approach is chosen to generate secure, non-repetitive passwords dynamically.

Data Input:

- User provides the desired password length.
- Internally, a character set (uppercase, lowercase, digits, symbols) is used as the data source.

Training Process:

No model training occurs. However, logic is coded to ensure that:

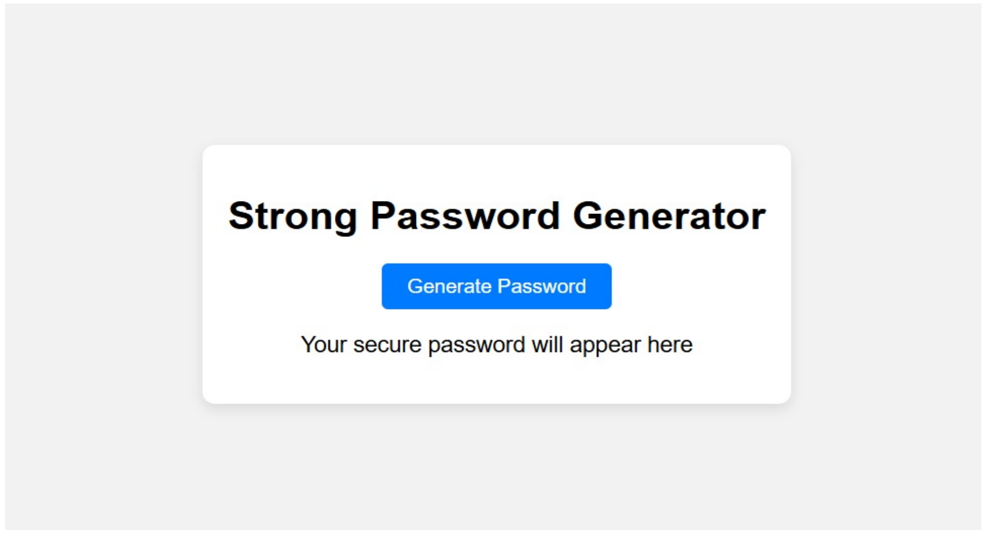
- Passwords meet minimum length.
- Different character types are included for strength.

Prediction Process:

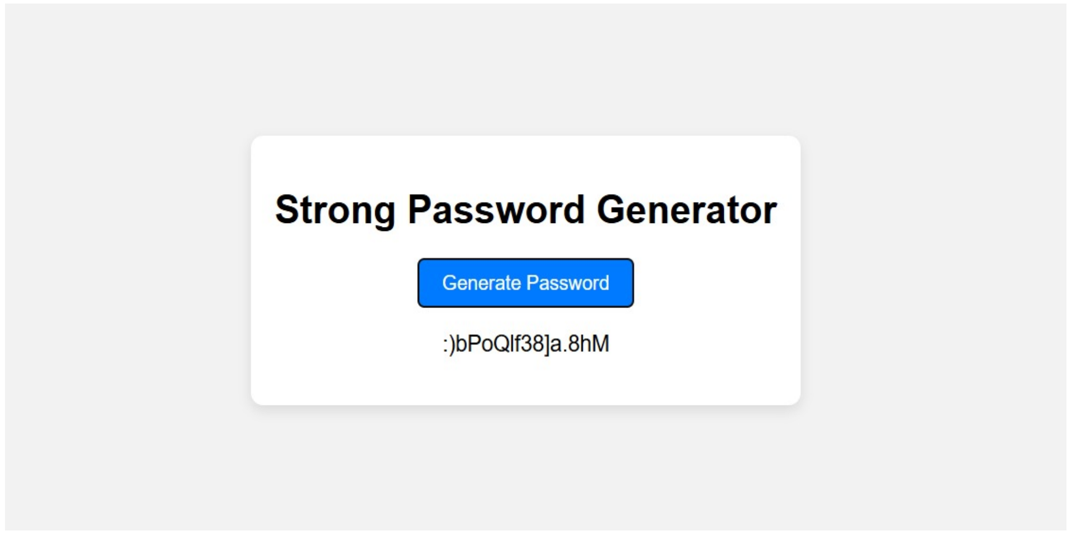
- Characters are randomly selected and shuffled.
- The final password is displayed on the interface and optionally copied to clipboard.

RESULT

The Strong Password Generator successfully generates secure, random passwords based on user-defined lengths. It ensures each password includes a combination of uppercase and lowercase letters, digits, and special characters, enhancing password strength and reducing vulnerability to cyber-attacks


Strong Password Generator

Your secure password will appear here


Strong Password Generator

```
:)bPoQlf38]a.8hM
```

CONCLUSION

The Strong Password Generator project successfully addresses the common issue of weak and easily guessable passwords by providing users with a reliable tool to create strong, complex, and random passwords. Through the use of Python’s built-in libraries and a simple GUI, the system ensures both usability and security, making it accessible even for non-technical users.

This project demonstrates how a lightweight, logic-based application can significantly enhance personal and organizational cybersecurity. While it doesn’t use advanced machine learning techniques, its effectiveness lies in simplicity, randomness, and adherence to strong password policies — making it a practical and valuable security tool.

FUTURE SCOPE

The Strong Password Generator has significant potential for future enhancements to increase its usability, functionality, and security. One major area of development is integrating a real-time password strength meter that visually indicates the quality of the generated password based on entropy and character diversity. The tool can also be connected to external APIs such as “Have I Been Pwned” to check if a generated or user-provided password has been compromised in any data breaches. Expanding beyond a desktop application, the project can be converted into a web-based platform, browser extension, or mobile app to allow cross-device usage and greater accessibility.

REFERENCES

- Python Software Foundation. (n.d.). Python 3 Documentation. Retrieved from: https://docs.python.org/3/
- Python random module documentation – for password character sampling.

  https://docs.python.org/3/library/random.html
- Python string module documentation – for character set constants.

  https://docs.python.org/3/library/string.html
- Tkinter GUI documentation – for building the user interface.

  https://docs.python.org/3/library/tkinter.html
- Pyperclip documentation – for copying password to clipboard.

  https://pypi.org/project/pyperclip/

GitHub Link: Link

Thank you