Project Report: Password Generator GUI
Project Title: Random Password Generator

Overview:

To design and develop a GUI-based password generator using Python's Tkinter library that allows users to create strong, random passwords based on customizable criteria. The application supports multiple character set options, applies security rules to prevent weak patterns, evaluates password strength, and stores the history of generated passwords in a file for later reference.

Technologies Used:

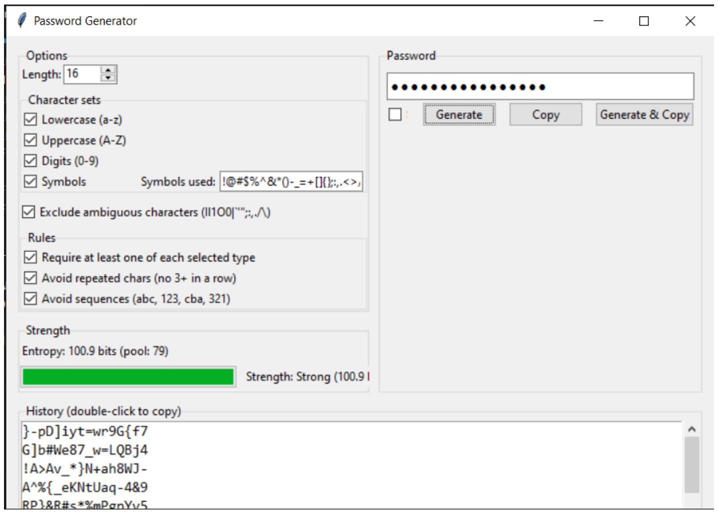
Programming Language: Python
GUI Toolkit: Tkinter (standard Python library)
Libraries:

- `secrets` for cryptographically secure random password generation
- `random.SystemRandom` for secure shuffling
- `string` for character sets
- `math` for entropy calculation
- `os` for file handling (password history)
- `tkinter.ttk` for themed widgets

Project Features:

- **User-friendly GUI** to set:
  - Password length (4–128 characters)
  - Inclusion/exclusion of lowercase, uppercase, digits, and symbols
  - Optional exclusion of ambiguous characters
- **Security Rules:**
  - Require at least one character from each selected type
  - Avoid 3+ repeated characters in a row
  - Avoid sequential patterns like `abc` or `123`
- **Password Strength Meter:**
  - Displays entropy in bits and categorizes strength as Weak, Okay, or Strong
- **Clipboard Integration:**
  - One-click **Copy** or **Generate & Copy** options
- **Password History:**
  - In-app history of the last 100 generated passwords
  - Persistent storage in `password_history.txt` for later use

Output Example:



Password Generator

Options

Length: 16

Character sets

☑ Lowercase (a-z)

☑ Uppercase (A-Z)

☑ Digits (0-9)

☑ Symbols   Symbols used: !@#$%^&*()-_=+[]{};:,.<>,

☑ Exclude ambiguous characters (Il1O0|`'";:,./\)

Rules

☑ Require at least one of each selected type

☑ Avoid repeated chars (no 3+ in a row)

☑ Avoid sequences (abc, 123, cba, 321)

Strength

Entropy: 100.9 bits (pool: 79)

Strength: Strong (100.9 I

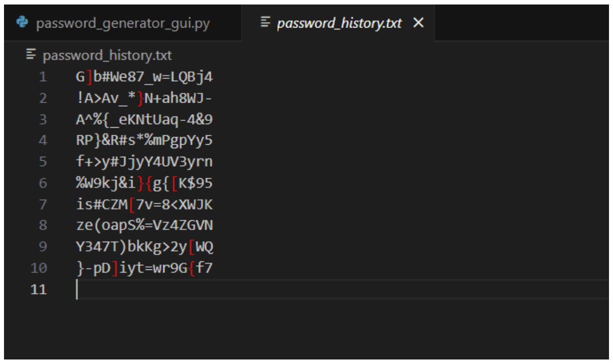
Password

●●●●●●●●●●●●●●●●

☐ Generate | Copy | Generate & Copy

History (double-click to copy)

```
}-pD]iyt=wr9G{f7
G]b#We87_w=LQBj4
!A>Av_*}N+ah8WJ-
A^%{_eKNtUaq-4&9
RP}&R#s*%mPgpYy5
```



password_generator_gui.py | password_history.txt

password_history.txt

```
G]b#We87_w=LQBj4
!A>Av_*}N+ah8WJ-
A^%{_eKNtUaq-4&9
RP}&R#s*%mPgpYy5
f+>y#JjyY4UV3yrn
%W9kj&i}{g{[K$95
is#CZM[7v=8<XWJK
ze(oapS%=Vz4ZGVN
Y347T)bkKg>2y[WQ
}-pD]iyt=wr9G{f7
```

Conclusion:

I used Python and Tkinter to build a secure, flexible, and user-friendly desktop application. This project helped me understand cryptographically secure randomness, user input handling, applying password strength rules, and managing persistent storage. The final application balances security and usability, making it practical for everyday password creation.