Random Password Generator

Joseph Binson
25BAI10774

INTRODUCTION

This report outlines the development and functionality of a **Random Password Generator** application**.**

- ## Project Goal

The primary objective of this project is to create a reliable and secure tool that generates strong, randomized passwords based on user-defined criteria. In today's digital landscape, the use of unique and complex passwords is vital for cybersecurity, and this tool aims to address the common difficulty of manually creating truly random and strong passwords**.**

- ## Key Features

Customizable Length: Allow the user to specify the desired length of the password.
Character Set Selection: Enable selection of which character types to include:

- Uppercase letter
- Lowercase Letter
- Numbers
- Symbols or Special charaters

DESIGN AND IMPLEMENTATION

- ## Technology Stack

The password generator can be implemented using various programming languages. A common choice for rapid development and accessibility is **Python**, utilizing its built-in `random` module for true randomness and string manipulation capabilities.

Component	Technology	Rationale
Programing language	Python	Simplicity, readability and extensive libraries.
Source	Modules of Python Essentials	Ensures to learn python program and other tools in python.
User Interface	Command-Line Interface (CLI) or basic GUI	Provides simple input for length and character sets.

- ## Code Algorithm

*Define Character Pools***:** Create separate strings or lists for each character type.

Combine Pools: Based on the user's selections e.g., including uppercase, lowercase, and numbers, concatenate the required character pools into a single master pool. **Ensure**

Inclusion (Strength): To guarantee the password meets the minimum criteria e.g., must contain at least one of each selected type, select one character randomly from *each* selected pool and place them into the resulting password.

Fill Remaining Length: Calculate the **remaining length** of the password. Fill this remaining length by repeatedly and randomly selecting characters from the **master pool**.

Shuffle: Randomly shuffle the final set of characters to ensure the character types are not grouped together (e.g., all symbols at the beginning), thus maximizing randomness and entropy.

- **Password Strength Calculation (Optional)**

The strength of a password is often measured by its entropy (measured in bits), which is the number of possibilities available. The formula for maximum password entropy (H) is:

$$H = L.\log_2(C)$$

Where L is the length and C is the size of the character set.

- **Use of CSPRNG**

CSPRNG - *Cryptographically Secure Pseudo-Random Number Generator*.
The most critical security feature is the use of a CSPRNG. Unlike simple pseudo-random generators, a CSPRNG is seeded using sources of true entropy from the operating system e.g., hardware interruptions, timing variations, making the generated sequence practically impossible for an attacker to predict or reproduce.

USAGE

Scenario	Used for	Benefit
New Account Setup	Generate a unique, high-entropy password for every new website, application, or service registration.	Prevents Credential Stuffing: If one site is breached, the password cannot be used to compromise other accounts.
Periodic Password Rotation	When an organization mandates a password change, the RPG ensures the new password is not a simple variation of the old one.	Mitigates Guessing: Eliminates predictable patterns or minor changes in sequential passwords.

System Administration	Generating complex credentials for back-end databases, API keys, SSH keys, service accounts, and Wi-Fi networks.	Secures Infrastructure: Provides high-entropy keys for non-human accounts that are often targets of automated attacks.
Local File	Creating strong, one-time passwords for encrypting sensitive containers	Ensures Data Confidentiality: The encryption key is resistant to brute-force recovery.

FUTURE ENHANCEMENTS

- Passphrase Generation Mode: Implement a mode that generates strong, but memorable, multi-word passphrases using a dictionary source.
- Entropy Meter: Integrate a real-time visual meter that calculates and displays the entropy (in bits) as the user adjusts the length and character sets.
- GUI Development: Transition the user interface to a desktop application (GUI) for broader accessibility and ease of use.

CONCLUSION

The Random Password Generator project successfully delivers a secure, customizable, and high-entropy password generation tool. By focusing on CSPRNG implementation and a robust mixing algorithm, it provides a crucial layer of defense against modern brute-force and dictionary attacks.