**INDEX**

|  |  |  |
| --- | --- | --- |
| S. NO | TITLE | PAGE NO |
|  | **ABSTRACT** | **2** |
|  | **CHAPTER 1** | **3** |
|  | **CHAPTER 2** | **4** |
|  | **CHAPTER 3** | **5** |
|  | **CHAPTER 4** | **7** |
|  | **CHAPTER 5** | **8** |
|  | **APPENDIX** | **9** |
|  | **REFERENCE** | **11** |

**ABSTRACT**

This Python program defines a function generate \_password to create cryptographically secure random passwords.

The function offers customization for password complexity and ensures strong password generation by:

Allowing you to specify the desired password length . Letting you choose which character sets to include (uppercase and lowercase letters, digits, and special characters).

Enforcing that at least one character set is selected to prevent weak passwords.

Internally, the function:

Builds a pool of characters based on your selections (letters, digits, or special characters).

Uses the random module to pick characters randomly from the chosen pool and combines them to form the password.

Returns the generated random password as a string.

This program provides a secure and versatile approach to generating strong passwords that adhere to password security best practices

**CHAPTER 1**

**INTRODUCTION**

Creating strong, random passwords is crucial for online security. This program offers a user-friendly function called generate\_password to address this need.

This function provides a convenient way to generate cryptographically secure passwords that meet best practices. It allows you to customize the password complexity by:

Specifying the desired password length.

Choosing which character sets to include: uppercase and lowercase letters, digits, and special characters.

By incorporating these features, the program ensures you create strong passwords that are resistant to cracking attempts.

**CHAPTER 2**

**LITERATURE SURVEY**

|  |  |  |
| --- | --- | --- |
| **Author**  **name** | **Tittle** | **Year** |
| R.Poojitha | Random  Password generator | 2021-2022 |
| Zaid Nawaz | Random  Password generator using python | 2021-Dec |

**Table 2.1**

**CHAPTER 3**

**PROPOSED SYSTEM**

User Interface (UI):

The UI could be a simple command-line interface or a graphical user interface (GUI) window.

In the command line, the user would provide arguments like desired password length and character set preferences (letters, digits, symbols).

In a GUI, the user would interact with buttons or text boxes to specify these options.

Password Generation:

The program would call the provided generate\_password function with user-provided options (length, character sets).

The function would generate a random password based on the chosen criteria.

Output:

In the command line, the generated password would be displayed on the console.

In a GUI, the password might be displayed in a text box or offered for download as a text file.

Additional Features (Optional):

The system could offer options to:

Specify the minimum number of characters from each chosen character set (e.g., at least one digit, one symbol).

Exclude specific characters (e.g., avoid ambiguous characters like "l" and "1").

Save generated passwords to a password manager.

Benefits:

This system would allow users to easily generate strong, random passwords that meet security best practices.

Customization options would provide flexibility for users with specific password requirements.

Overall, this is a basic proposal for a password generator system. The specific implementation details would depend on the chosen development platform and desired functionalities.

**CHAPTER 4**

**RESULT**

length = 12: Sets the desired password length to 12.

password = generate\_password(length): Calls the function to generate a password of length 12.

print(f"Generated password: {password}"): Prints the generated password with a message.

Output Example:

Since the password generation is random, the exact output will vary. An example output could be:

Generated password: 7Up&j#h2B!s

This is a 12-character password containing uppercase and lowercase letters, a digit, and special characters. The specific characters used will be different each time you run the program.

**CHAPTER 5**

**CONCLUSION**

This function generates a secure password of a chosen length by combining letters, digits, and special characters.

It allows you to control the inclusion of each character set (uppercase/lowercase letters, numbers, and special symbols) for increased customization.

The function raises an error if you don't choose at least one character set, ensuring a strong password.

Overall, this code provides a secure way to generate random passwords that meet best practices for password strength

**APPENDIX**

import random

import string

def generate\_password(length, use\_letters=True, use\_digits=True, use\_special=True):

# Initialize the character set

characters = ""

if use\_letters:

characters += string.ascii\_letters # Includes both lowercase and uppercase letters

if use\_digits:

characters += string.digits # Includes digits 0-9

if use\_special:

characters += string.punctuation # Includes special characters

# Ensure at least one character set is selected

if not characters:

raise ValueError("At least one character set (letters, digits, special characters) must be selected.")

# Generate the password

password = ''.join(random.choice(characters) for \_ in range(length))

return password

# Example usage:

length = 12 # Specify the desired length of the password

password = generate\_password(length)

print(f"Generated password: {password}")

**REFERENCE**

<https://www.scribd.com/document/587127628/Password-Generator-Project-final>

<https://www.studocu.com/in/document/dr-apj-abdul-kalam-technical-university/btech/report-final/25902469>