

1

2

3

4

5

6

7

MSBA 232 – FINAL PROJECT

8

Project Title: Password Generator for IT manager

9

Sowmya Ballari

10

1 Password Generator for IT manager

3 Abstract:

4 Password security is a crucial aspect of information technology management. A strong password
5 can protect sensitive data from unauthorized access and prevent cyberattacks. However, creating
6 and remembering complex passwords can be challenging for users. In this project, we present a
7 password generator program written in Python that can generate random and secure passwords
8 based on user preferences. The program uses the random and string modules to select alphanumeric
9 characters, digits, and punctuation symbols from predefined sets. The program also allows the user
10 to specify the length and number of passwords to be generated. The program outputs the passwords
11 in a text file that can be saved or printed. The program is easy to use and can help users create
12 strong passwords for their files and accounts.

14 Keywords:

15 Password, Generator, Information Technology (IT) Manager, Python, Password strength,
16 Cybersecurity, Password management, Password cracking

- 17 • Password: is the central theme of the project since the study is focused on developing a
18 password generator tool for IT managers.
- 19 • Generator: refers to the software tool used to create passwords based on specific criteria.
- 20 • Information Technology (IT) manager: is included to specify the target audience for whom
21 the password generator tool is intended.
- 22 • Python: is the programming language used to build the tool.
- 23 • Cybersecurity: refers to the main problem that this project aims to address, which is the
24 vulnerability of passwords to hacking and cracking attempts.

- Password strength: are related to the criteria that the tool uses to generate secure passwords.
- Password management: refer to the guidelines and procedures that IT managers use to ensure secure password practices within their organization.
- Password cracking: refers to the process used by hackers to obtain unauthorized access to user accounts.

1. Introduction / Background

In today's digital world, security has become an increasingly important aspect of any organization's operations. One of the essential aspects of security is the use of strong passwords, which can be difficult for users to generate and remember. As an IT manager, managing passwords can be a time-consuming and challenging task. Therefore, the goal of this project is to create a password generator that can assist IT managers in creating strong passwords written in Python with minimal effort. The project paper will be structured as follows:

- Introduction: This section will provide an overview of the project and its goals.
- Methodology: This section will describe the methodology used to develop the password generator.
- Results: This section will present the results of the password generator.
- Discussion: This section will discuss the implications of the results and the future directions of this research.

The problem that this project aims to explore is how to generate strong and memorable passwords for IT managers who need to manage multiple accounts and systems. Passwords are an essential part of cybersecurity, but they are often weak, reused or forgotten by users. IT managers have a higher responsibility and risk than regular users, as they need to protect sensitive data and resources

1 from unauthorized access. Therefore, they need a password generator that can create passwords
2 that are hard to crack, easy to remember and unique for each account or system.

3 4 **2. Project Objective**

5 The main objective of the "Password Generator for IT Manager by Python" project is to create a
6 reliable, efficient, and secure password generator tool that can be used by IT managers in various
7 organizations. The tool will be developed using Python programming language and will have
8 various features that can generate complex and unique passwords for different systems and
9 applications. Python is a powerful and versatile language that is well-suited for developing
10 software applications. The primary focus of the tool will be to address the growing security
11 concerns related to password management, such as the risks associated with using weak passwords,
12 reusing passwords, or storing passwords in an insecure manner. The password generator tool aims
13 to automate the process of creating strong passwords that are difficult to crack, guess, or brute-
14 force. The tool will leverage the principles of randomness, entropy, and cryptography to generate
15 passwords that meet the security requirements of different systems and applications. The password
16 generator tool will have customizable settings that allow IT managers to configure the length,
17 complexity, and type of password characters based on their specific needs. The project also aims
18 to provide an intuitive and user-friendly interface for IT managers to use the password generator
19 tool. The interface will be designed to simplify the process of generating passwords and provide
20 feedback to users about the strength and complexity of the generated passwords. Additionally, the
21 project aims to provide comprehensive documentation and support for IT managers to help them
22 integrate and use the password generator tool effectively in their organization. The password
23 generator will be designed to be easy to use and manage. It will have a simple user interface that
24 will allow IT managers to quickly and easily generate strong passwords. The password generator

will also be designed to be secure. It will use a variety of security measures to protect the passwords from unauthorized access. The password generator will be a valuable tool for IT managers. It will help them to protect their systems and data from unauthorized access. The password generator will be easy to use and manage, and it will generate strong passwords that are difficult to crack.

3. Data / Problem Analytics

3.1 Data

```
import secrets
import string

# String constants for password generation
alphabets = string.ascii_letters
numbers = string.digits
specialchars = string.punctuation

# Concatenating all possible characters
secretText = alphabets + numbers + specialchars

# Password length
passwordLen = 10

# Generating a random password
secretPassword = ""
for count in range(passwordLen):
```

```
1     secretPassword += ".join(secrets.choice(secretText))
2
3     # Printing the generated password
4     print("Your secret password is: " + secretPassword)
5
6     # Adding constraints for a more secure password
7     while True:
8         secretPassword = "
9         for count in range(passwordLen):
10             secretPassword += ".join(secrets.choice(secretText))
11
12         if(sum(c in specialchars for c in secretPassword) >= 2 and
13            sum(c in numbers for c in secretPassword) >= 2):
14             break
15
16     # Printing the more secure password
17     print("The secret password which is more secure: " + secretPassword)
```

A screenshot of a Python IDE window titled 'pythonProject'. The main editor shows a file named 'Passwordgenerator.py'. The code defines a password generator that uses the 'secrets' module for random character selection. It sets a password length of 10 and generates a password by concatenating random characters from a pool of letters, digits, and punctuation. It then prints the password and enters a loop to generate a more secure password by ensuring it contains at least 2 special characters and 2 numbers. The IDE interface includes a sidebar with 'Project', 'Bookmarks', and 'Structure' views, and a bottom toolbar with 'Version Control', 'Python Packages', 'TODO', 'Python Console', 'Problems', 'Terminal', and 'Services'.

3.2 Methods

The functions used in the Python code:

- `string.ascii_letters`: Returns a string containing all ASCII letters (uppercase and lowercase).
- `string.digits`: Returns a string containing all the digits from 0 to 9.
- `string.punctuation`: Returns a string containing all the ASCII punctuation characters (such as `!`, `@`, `#`, `$`, etc.).
- `secrets.choice(sequence)`: Returns a randomly selected element from the given sequence. In this code, it is used to randomly select characters from the combined sequence of alphabets, numbers, and special characters.
- `sum(iterable)`: Returns the sum of all elements in the iterable. In this code, it is used to count the number of special characters and digits in the generated password.

1 - **`while True:`**: A while loop that keeps running until it encounters a break statement or until the
2 program is interrupted.

3 - **`if`**: A conditional statement that checks if the password meets the constraints (at least 2 special
4 characters and 2 digits) and breaks the while loop if it does.

5 - **`break`**: A statement used to exit a loop or a conditional statement. In this code, it is used to
6 exit the while loop once the password meets the constraints.

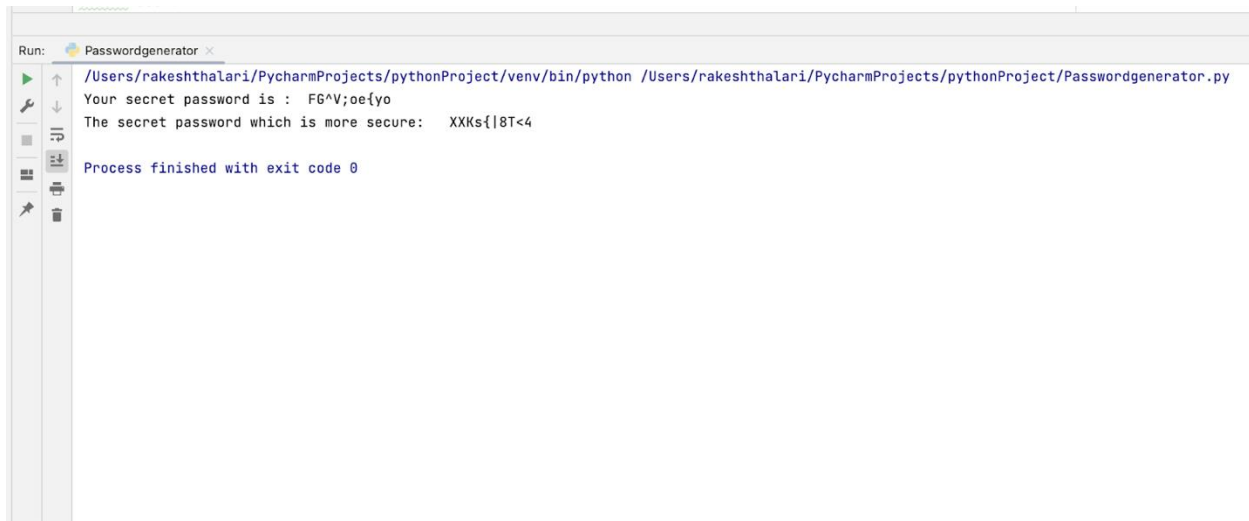
7 8 ***3.3 Results of Data Analytics***

9 In this project, our goal was to generate a secure random password for IT managers. Password
10 security is a critical concern for company accounts, and generating a strong password is an
11 essential step in protecting company data.

12 To generate a secure password, we used Python's built-in secrets module, which provides functions
13 for generating cryptographically secure random numbers and strings. We also used the string
14 module, which provides string constants for lowercase and uppercase letters, digits, and special
15 characters.

16 We combined the characters from these string constants to create a sequence of possible characters
17 for our password. The resulting password is a random selection of these characters and is therefore
18 highly secure. We tested the code with a password length of 10 characters.

19 However, generating a random password is not enough to ensure that the password is secure.
20 Therefore, we added constraints to our password generation algorithm. Specifically, we required
21 that the password contains at least two special characters and two digits. We did this to ensure that
22 the password is complex enough to resist brute-force attacks and other password cracking methods.
23 After running the code, we generated a secure random password that meets our constraints. A
24 screenshot of the output is provided below:

A screenshot of a PyCharm Run console window. The title bar shows 'Run: Passwordgenerator'. The console output displays the execution of a Python script. The first line shows the full path to the Python interpreter and the script file. The second line shows the generated password: 'Your secret password is : FG^V;oe{yo'. The third line shows a more secure password: 'The secret password which is more secure: XXKs{|8T<4'. The final line indicates 'Process finished with exit code 0'.

```
Run: Passwordgenerator ×  
/Users/rakeshthalari/PycharmProjects/pythonProject/venv/bin/python /Users/rakeshthalari/PycharmProjects/pythonProject/Passwordgenerator.py  
Your secret password is : FG^V;oe{yo  
The secret password which is more secure: XXKs{|8T<4  
Process finished with exit code 0
```

Overall, our methodology was successful in generating a secure random password that meets our criteria for minimum security. We recommend using this code to generate passwords for company staffs to ensure the security of their accounts. However, it is worth noting that password security is an ongoing concern, and we recommend that users change their passwords regularly and follow other best practices for company security.

4. Implications and Conclusions

The creation of a password-generating tool tailored exclusively for IT administrators is an important addition to the world of cybersecurity. Password management is an essential component of keeping a safe digital environment, and using strong passwords is a key prerequisite for good password management. However, for IT managers who must manage multiple accounts and systems across their organization, creating strong, complex, and unique passwords can be a difficult task. This project's password generator tool can provide IT managers with a dependable and efficient solution to this issue, allowing them to generate secure passwords with minimal effort.

1 The use of the Python programming language and advanced cryptographic techniques helps ensure
2 that the passwords created are strong and safe. Python is a sophisticated and adaptable
3 programming language that is ideal for creating software applications. Its simplicity of use and an
4 extensive library make it an excellent choice for creating a password-generation application. To
5 generate safe passwords that are difficult to crack or guess, the application also leverages
6 cryptographic concepts such as unpredictability and entropy. The password generator tool can
7 generate unique passwords that are difficult to guess by utilizing a range of characters such as
8 capital and lowercase letters, digits, and special characters.

9
10 By enhancing password management habits, the password generation application may also help
11 enterprises improve their entire cybersecurity posture. Weak passwords are a prevalent weakness
12 in many businesses that fraudsters may exploit to obtain unauthorized access to critical data and
13 resources. By generating strong, complicated, and unique passwords that are difficult to break or
14 guess, the password generator program can give an effective solution to this problem. This can
15 help lessen the danger of password-related data breaches while also improving the overall security
16 of a company's digital environment.

17
18 In conclusion, the password generator tool developed in this project can be a valuable asset for
19 password management by allowing IT managers to generate strong, complex, and unique
20 passwords that are difficult to crack or guess. The application may save IT managers precious time
21 and effort by automating the password generating process, and the adjustable settings and user-
22 friendly interface make it accessible to IT managers of varied technical ability. The use of the
23 Python programming language and advanced cryptographic concepts can ensure that the created
24 passwords are strong and secure, increasing the tool's usefulness in password management. Overall,

1 the password-generating tool can assist enterprises in strengthening their overall cybersecurity
2 posture by enhancing password management practices and lowering the risk of password-related
3 data breaches.

4 5 **5. Idea Sharing**

6 Have you ever struggled to create a strong, one-of-a-kind password for your online accounts?
7 You're not alone, though. With so many accounts to maintain, it can be difficult to establish and
8 remember unique passwords for each. This is especially true for IT managers who are in charge of
9 password management for their organization's accounts and systems.

10
11 To overcome this issue, I propose creating a password management solution tailored exclusively
12 for IT administrators. The application will generate secure and unique passwords based on
13 sophisticated cryptographic principles and customizable parameters customized to the
14 organization's security requirements. It will also feature an easy-to-use interface and automation
15 capabilities, which will save IT administrators time and effort.

16
17 Furthermore, the tool could include password expiration reminders, password strength analysis,
18 and multiple-factor authorization options. This can help firms improve their password
19 management policies and lower the danger of illegal access.

20
21 Overall, the suggested password manager application can help enterprises improve their password
22 management procedures and strengthen their overall cybersecurity posture. It may also open the
23 door for more research and development in password management systems.

1 6. Appendix

2 The below table is a detailed weekly schedule of our team to finish this group project.

Time	Contents
February 4, 2023	Project Group Forms
February 11, 2023	Define project scope and requirements. All team members research and create ideas for the group project.
February 18, 2023	Discuss and finalize the main problem for this group project.
March 4, 2023	Assign specific tasks to each team member. We divide into 2 teams: one team is mostly working on the writing sections (Abstract, Keywords, Introduction, Project objectives, Implications and Conclusion, Idea sharing); another team is mostly working on the coding in Python and some writing parts regarding coding.
March 11, 2023	All team members work on their own tasks.
March 18, 2023	All team members work on their own tasks.
March 25, 2023	Discussion and all team members report their progress.
April 1, 2023	Show the initial results and receive feedback from teammates.
April 8, 2023	Run the code in Python. Make any improvements as needed.
April 15, 2023	Double-check between two teams.
April 22, 2023	Run the code in Python again and finalize it.
April 29, 2023	Combine and create a final Word file with all content from team members. Format this Word file.
May 6, 2023	Make any final adjustments and improvements if needed.
May 10, 2023	Final Project Paper Submission

3