**DEVELOPMENT OF A RANDOM PASSWORD GENERATOR**

**USING (PYTHON, HTML)**

**BY**

**OLADOKUN PRAISE MICHAEL**

**MATRIC No: 206131**

**SUBMITTED TO**

**THE DEPARTMENT OF COMPUTER SCIENCE**

**FACULTY OF COMPUTING AND INFOMATICS**

**LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY, OGBOMOSO**

**OYO STATE NIGERIA**

**JUNE 2023**

ABSTRACT

In today's digital era, where data breaches and online security threats are prevalent, it is crucial to use strong, unique passwords for each online account. However, generating and managing such passwords can be challenging for individuals. This project aims to develop a random password generator that can create strong, secure passwords to enhance online security.

The random password generator will be designed as a software application or a web-based tool, providing a user-friendly interface for generating passwords. The application will utilize a combination of cryptographic algorithms, random number generation techniques, and best practices in password security to ensure the generated passwords are highly secure.

TABLE OF CONTENT

TITLE PAGE

ABSTRACT

TABLE OF CONTENTS

INTRODUCTION

PROJECT OVERVIEW

PROJECT OBJECTIVE

IMPLEMENTATION

SOURCE CODE AND OUTPUT

CONCLUSION

*Chapter 1:* INTRODUCTION

* 1. Background Study:

The background study of a random password generator involves understanding the principles and requirements of password security, as well as the techniques used to generate random and secure passwords.

In today’s digital age, ensuring the security of our online accounts has become more important than ever. One of the fundamental aspects of maintaining a secure online presence is by using strong, unique passwords for each account we create. However, coming up with such passwords can be a daunting task for many individuals.

To address this challenge, a random password generator is an invaluable tool. A random password generator is a program or algorithm that generates complex, unique passwords based on predefined criteria. These criteria typically include password length, character types (e.g., uppercase letters, lowercase letters, numbers, special characters), and other customizable options.

The purpose of this project is to develop a random password generator that can provide users with strong, secure, and easily customizable passwords. By using this tool, users will no longer need to worry about the vulnerability of their online accounts due to weak passwords or the difficulty of coming up with unique combinations on their own.

In this project, we will explore various programming techniques and algorithms to create a robust random password generator. We will focus on developing a user-friendly interface that allows users to specify their desired password length, character types, and any other relevant options. The generated passwords will adhere to commonly accepted password security best practices, such as avoiding common patterns, incorporating a mix of character types, and avoiding easily guessable combinations.

* 1. Aims and Objectives of the Study

The objective of a random password generator is to create strong, secure, and unpredictable passwords for various online accounts. The primary goals of a random password generator include:

1. Password Strength: The generator aims to produce passwords that meet or exceed recommended security standards. This involves generating passwords with sufficient complexity, length, and diversity of characters to make them resistant to guessing and brute-force attacks.
2. Security Enhancement: By generating strong passwords, the generator helps enhance the overall security of online accounts. Strong passwords make it significantly more difficult for attackers to gain unauthorized access to user accounts, protecting sensitive personal information and preventing potential security breaches.
3. Convenience and Efficiency: The generator aims to provide an efficient and convenient solution for users to create strong passwords. It eliminates the need for users to come up with passwords themselves, ensuring that the generated passwords are truly random and not easily guessable
4. User-Friendly Experience: A random password generator should offer a user-friendly experience, making it accessible to a wide range of users, including those with limited technical expertise. It should be intuitive and easy to use, allowing users to quickly generate secure passwords without complications.
5. Customization and Flexibility: Depending on the implementation, the generator may offer customization options, allowing users to specify password length, character sets, or additional requirements. This customization helps cater to specific password policies or user preferences.

1

1. Integration and Compatibility: The generator may be integrated into various platforms, such as web-based tools, mobile apps, or password management applications, to provide seamless integration and compatibility across different devices and systems.

Overall, the objective of a random password generator is to promote stronger password security practices, mitigate the risks of password-related vulnerabilities, and enhance the protection of user accounts and sensitive information in an increasingly interconnected digital environment.

* 1. Statement of Problem

The problem at hand is the need for a reliable and secure random password generator. In today’s digital age, where online accounts and sensitive information are prevalent, the importance of strong passwords cannot be overstated. Many individuals struggle to create strong and unique passwords, often resorting to using easily guessable or repetitive combinations, which puts their personal data and online security at risk.

Existing password generators available online or as built-in features in certain applications often lack key features or suffer from vulnerabilities that compromise their effectiveness. These limitations include generating passwords that are not truly random, insufficient complexity options, lack of customization, limited password length, or potential security flaws that could expose the generated passwords to malicious actors.

Therefore, there is a need to develop a robust and user-friendly random password generator that addresses these shortcomings and provides individuals and organizations with a reliable tool for generating strong, unique, and secure passwords. The password generator should adhere to industry best practices and standards for password security, offering a range of customizable options such as password length, character types (lowercase, uppercase, numbers, symbols), and exclusion of ambiguous or easily confusable characters.

The solution should prioritize the following objectives:

1. Generate truly random and unpredictable passwords to ensure maximum security.

2. Allow customization of password criteria to accommodate different security requirements.

3. Provide user-friendly interfaces, whether through a standalone application, web-based service, or integration into existing software.

4. Ensure the generated passwords meet recommended complexity standards, such as minimum length and inclusion of various character types.

5. Implement secure coding practices and protocols to protect the generated passwords from potential attacks or breaches.

6. Consider cross-platform compatibility, ensuring accessibility across a range of devices and operating systems.

7. Offer additional features, such as password strength evaluation, password history, and secure storage options, to enhance user experience and password management capabilities.

By addressing these objectives, the development of a reliable and secure random password generator will contribute to strengthening online security practices and empower individuals and organizations to protect their digital identities and sensitive information.

2

* 1. Significance of Study

The study of the development of a random password generator holds significant importance in the realm of cybersecurity and user privacy. Here are some key reasons why this area of study is significant:

1. Security Enhancement: Passwords are essential for protecting sensitive information, such as personal data, financial accounts, and online identities. However, weak or easily guessable passwords can compromise security. A random password generator ensures the creation of strong and unique passwords, enhancing overall security.
2. Protection Against Brute-Force Attacks: Brute-force attacks involve systematically attempting various combinations of passwords to gain unauthorized access. By generating complex and random passwords, the likelihood of successful brute-force attacks is significantly reduced, thereby enhancing the security of user accounts and systems.
3. Mitigation of Human Error: Users tend to select weak passwords that are easily guessable or reuse the same passwords across multiple accounts. A random password generator eliminates the reliance on human-generated passwords, reducing the risk of human error and enhancing security by generating strong and unique passwords.
4. Privacy Preservation: A random password generator contributes to maintaining user privacy by ensuring that passwords are not derived from personal information or easily associated with individuals. This reduces the risk of unauthorized access to accounts through social engineering techniques or data breaches that expose personal information.
5. Compliance with Security Standards: Many industries and organizations have specific security standards and regulations that require the use of strong passwords. By implementing a random password generator, businesses can ensure compliance with these standards and protect sensitive data, avoiding potential legal and financial consequences.
6. User Convenience: While generating complex passwords manually can be challenging and time-consuming, a random password generator simplifies the process for users. It allows them to quickly create strong passwords without the need to remember or invent complex combinations themselves, thereby promoting good security practices.

Overall, the study and development of a random password generator have significant implications for strengthening cybersecurity, protecting user privacy, and promoting secure online practices. By leveraging advanced algorithms and encryption techniques, password generators can enhance security measures and provide a valuable defence against unauthorized access and data breaches.

* 1. History of Study

The development of random password generators can be traced back to the early days of computer security. Here is a brief history of their development:

1. Pre-Internet Era:

In the early days of computing, security concerns were relatively low, and password requirements were often weak. Users would often choose simple and easily guessable passwords. However, as computer networks expanded and became more interconnected, the need for stronger passwords became evident.

1. First Password Generators:

In the late 1970s and early 1980s, the first password generators emerged. These early generators were usually simple programs or scripts that randomly combined letters, numbers, and special characters to create passwords. However, they were often limited in terms of the randomness and complexity of the passwords they generated.

1. Password Complexity Requirements:

3

As computer systems became more sophisticated, security standards started to evolve, and organizations

began implementing password complexity requirements. These requirements mandated the use of a

combination of uppercase and lowercase letters, numbers, and special characters. Password generators had to adapt to these requirements and ensure the passwords they generated met the specified complexity rules.

1. Cryptographically Secure Randomness:

To enhance password security, the use of cryptographically secure random number generators (CSRG) became essential. CSRGs generate random numbers that are highly unpredictable and suitable for cryptographic purposes. Password generators started incorporating CSRGs to ensure the passwords they produced were not easily guessable.

1. Online Password Generators:

With the rise of the internet and the proliferation of online services, web-based password generators became popular. These generators provided a convenient way for users to create strong passwords without having to install any software. Online password generators often included additional features like password strength meters and the ability to customize password length and character sets.

1. Two-Factor Authentication:

As security threats increased, two-factor authentication (2FA) became a common practice for enhancing account security. Some password generators started integrating 2FA capabilities, enabling users to generate both a strong password and a corresponding one-time verification code.

1. Password Managers:

The development of password manager applications further revolutionized password generation. Password managers not only generate strong passwords but also store and manage them securely. They often have built-in password generators that can generate complex and unique passwords for each account, reducing the burden on users to remember them.

1. Ongoing Enhancements:

The development of random password generators continues to this day. Developers constantly refine the algorithms used to generate passwords, improve the user experience, and adapt to evolving security threats. Features like password entropy estimation, user-friendly interfaces, and integration with other security tools are among the ongoing advancements in this field.

Today, random password generators are widely used and considered an essential tool for maintaining online security. They play a crucial role in protecting sensitive data and preventing unauthorized access to accounts and systems.

4

*Chapter 2:*LITERATURE OVERVIEW

The Random Password Generator is a software tool designed to create strong and secure passwords. It generates random combinations of characters, including uppercase and lowercase letters, digits, and special symbols. This project overview will provide a high-level description of the development process and key features of the Random Password Generator.

1. Objectives:
   * Develop a user-friendly application that generates random passwords.
   * Ensure the generated passwords are strong, secure, and difficult to guess.
   * Provide customization options for password length and character types.
   * Implement proper error handling and validation to enhance usability.
   * Deliver a cross-platform solution that can run on various operating systems.
2. Key Features: a. Password Generation:
   * The application will generate passwords with a combination of uppercase letters, lowercase letters, digits, and special symbols.
   * Users can specify the length of the generated password.

b. Customization Options:

* + The application will allow users to choose which character types to include in the generated password.
  + Users can enable or disable uppercase letters, lowercase letters, digits, and special symbols based on their requirements.

c. Strength Indicator:

* + The application will provide a visual indicator of the password strength.
  + The strength indicator can show weak, medium, or strong password ratings based on complexity criteria.

d. Copy to Clipboard:

* + The generated password can be copied to the clipboard with a single click for easy use.

e. Error Handling:

* + Proper error handling will be implemented to handle invalid inputs and edge cases gracefully.
  + Users will be notified of any errors or exceptional conditions encountered during password generation.

f. Cross-platform Compatibility:

* + The application will be developed to run on major operating systems such as Windows, macOS, and Linux.

5

* + It will be designed to have a responsive and intuitive user interface across different platforms.

1. Technologies and Tools:
   * Programming Language: Python, JavaScript (for web-based version)
   * User Interface: Command Line Interface (CLI), Graphical User Interface (GUI) using Tkinter, or Web Interface using HTML/CSS/JavaScript.
   * Development Tools: Integrated Development Environment (IDE) of choice (e.g., Visual Studio Code, PyCharm).
   * Version Control: Git for source code management.
   * Packaging and Distribution: PyInstaller, Py2exe, or Py2app for creating standalone executables.
2. Development Steps:
   * Define project requirements and specifications.
   * Choose the development platform and programming language.
   * Design the user interface and overall application flow.
   * Implement the password generation algorithm.
   * Incorporate customization options and strength indicator.
   * Add copy to clipboard functionality.
   * Handle error cases and implement appropriate validation.
   * Test the application thoroughly, including edge cases and user inputs.
   * Package the application for distribution.
   * Deploy the application on desired platforms.

6

*Chapter 3:* METHODLOGY

The development of an online food ordering system typically follows a structured methodology to ensure a successful and efficient implementation. Here is a general outline of the methodology:

Requirements Gathering: The first step is to gather and document the requirements for the online food ordering system. This involves understanding the business objectives, user needs, and desired features. This can be done through meetings, interviews, and surveys with stakeholders, including restaurant owners, managers, and potential customers.

System Design: Once the requirements are gathered, the system design phase begins. This includes creating a high-level architecture and designing the user interface (UI) and user experience (UX) for both customers and restaurant owners. The design should consider aspects such as menu management, order processing, payment integration, and delivery logistics.

Technology Selection: Based on the system requirements, the appropriate technologies and platforms are selected. This includes choosing a programming language, web framework, database management system, and any necessary third-party integrations such as payment gateways or mapping services.

Database Design: The database design phase involves creating a data model to represent the system's entities, such as customers, restaurants, menu items, orders, and payments. The database schema should be optimized for efficient data retrieval and storage.

Development: In this phase, the actual coding and development of the online food ordering system take place. It is recommended to follow an agile development approach, breaking down the development into smaller sprints or iterations. This allows for incremental development, testing, and feedback from stakeholders throughout the process.

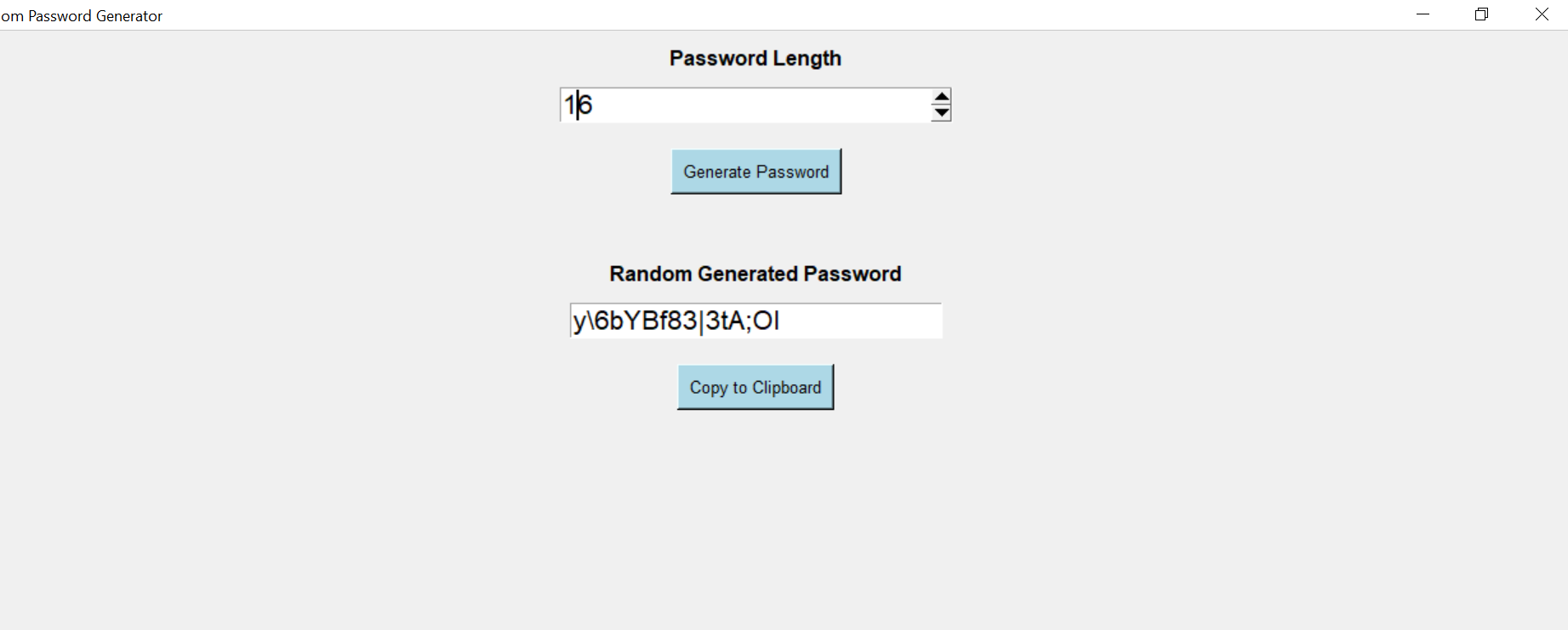
Integration and Testing: As the development progresses, integration with third-party services, such as payment gateways and SMS notifications, should be implemented and thoroughly tested. Additionally, comprehensive testing should be conducted at various levels, including unit testing, integration testing, and system testing, to ensure the system functions as expected.

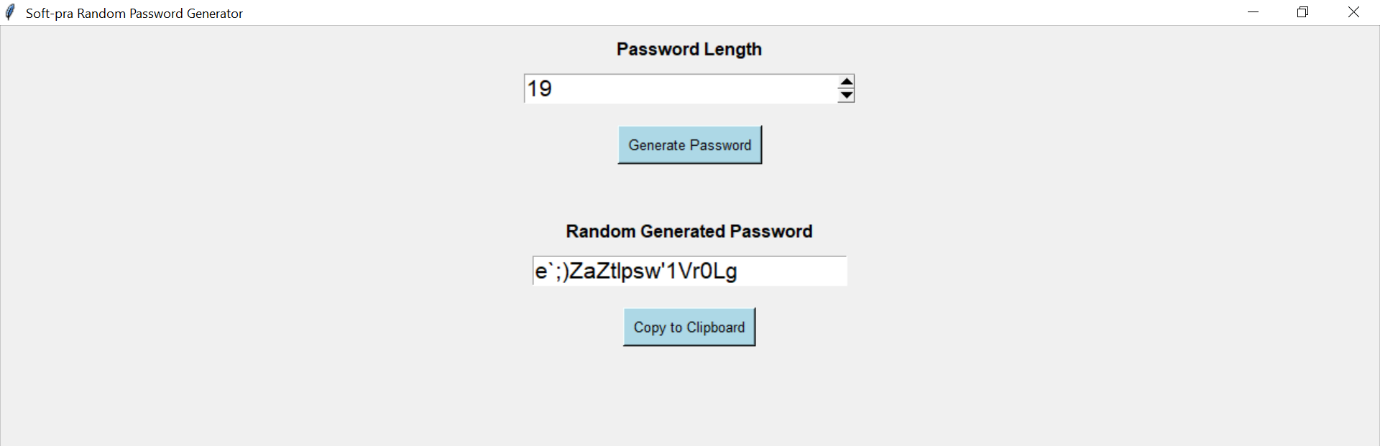
Deployment: Once the development and testing are complete, the system is ready for deployment. This involves setting up servers, configuring the necessary infrastructure, and ensuring proper security measures are in place. The deployment can be done on-premises or using cloud-based services.

User Training and Acceptance: Before the system is launched to the public, it is important to train the users, including restaurant owners and staff, on how to use the online food ordering system effectively. User acceptance testing should also be conducted to validate that the system meets the requirements and to address any remaining issues or feedback.

Launch and Maintenance: After successful user acceptance testing, the online food ordering system can be launched. It is important to monitor the system closely during the initial period to ensure it performs well and address any issues promptly. Ongoing maintenance, updates, and support should be provided to ensure the system remains secure, stable, and up-to-date with changing business needs and technology advancements.

7

*****Chapter 4:* RESULT AND CONCLUSION

****

Discussion:

Random password generators are powerful tools for enhancing online security. They help create strong, unique passwords that are difficult for hackers to guess or crack. In this discussion, we can explore the development of random password generators, their importance, and some best practices for using them effectively.

Random password generators work by generating strings of characters that include a combination of letters (uppercase and lowercase), numbers, and special symbols. The length of the password and the character set used can often be customized to meet specific requirements or security standards.

One of the primary benefits of using a random password generator is the ability to create strong, complex passwords that are challenging to guess. These passwords are typically long and include a mix of different character types, making them resistant to brute-force attacks. By generating random passwords, users can avoid using easily guessable information such as their names, birthdates, or commonly used words, which are common targets for hackers.

8

CONCLUSION

In conclusion, the development of a random password generator is a valuable tool for enhancing online security and protecting sensitive information. This project involves creating a program or application that generates strong, unique, and unpredictable passwords for users.

Throughout the development process, several key considerations should be taken into account. First and foremost, the random password generator should prioritize security by employing cryptographic algorithms and techniques to generate passwords that are resistant to brute-force attacks and other common hacking methods. It should also adhere to best practices for password strength, such as including a combination of uppercase and lowercase letters, numbers, and special characters.

Additionally, the random password generator should be user-friendly and easily accessible. It can be implemented as a standalone application, a web-based tool, or integrated into existing software systems. The user interface should be intuitive, allowing users to specify password length, character sets, and any additional requirements they may have.

Furthermore, it is crucial to consider the storage and handling of generated passwords. Storing passwords in a secure manner, such as using strong encryption and following industry-standard practices, is essential to prevent unauthorized access and protect user privacy

9

SOURCE CODE

import random, string

from tkinter import \*

root =Tk()

root.geometry("400x400")

root.title(" Soft-pra Random Password Generator")

output\_pass = StringVar()

all\_combi = [string.punctuation, string.ascii\_uppercase, string.digits, string.ascii\_lowercase]

print('Soft-pra Codez')

def randPassGen():

password = ""

for y in range(pass\_len.get()):

char\_type = random.choice(all\_combi)

password = password + random.choice(char\_type)

output\_pass.set(password)

def copyPass():

pyperclip.copy(output\_pass.get())

pass\_head = Label(root, text = 'Password Length', font = 'arial 12 bold').pack(pady=10)

pass\_len = IntVar()

length = Spinbox(root, from\_ = 4, to\_ = 32 , textvariable = pass\_len , width = 24, font='arial 16').pack()

Button(root, command = randPassGen, text = "Generate Password", font="Arial 10", bg='lightblue', fg='black', activebackground="teal", padx=5, pady=5 ).pack(pady= 20)

pass\_label = Label(root, text = 'Random Generated Password', font = 'arial 12 bold').pack(pady="30 10")

Entry(root , textvariable = output\_pass, width = 24, font='arial 16').pack()

Button(root, text = 'Copy to Clipboard', command = copyPass, font="Arial 10", bg='lightblue', fg='black', activebackground="teal", padx=5, pady=5 ).pack(pady= 20)

root.mainloop()

10