**RANDOM PASSWORD GENERATOR**

A MINI PROJECT REPORT

Submitted by

**RA2211003010540 – Pranav Singh**

**RA2211003010545 – Manish Tiwari**

**RA2211003010564 – Swastik Rana**

Under the guidance of   
**Dr.R.Thamizhamuthu** Assistant Professor, Department of Computing Technologies *In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in

**DEPARTMENT OF COMPUTING TECHNOLOGIES**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in "**RANDOM PASSWORD GENERATOR**" is the bonafide work of **Pranav Singh (RA2211003010540), Manish Tiwari (RA2211003010546) and Swastik Rana (RA2211003010564)** who carried out the work under my supervision.

# SIGNATURE SIGNATURE

Dr.R.Thamizhamuthu Dr. M Pushpalatha

Assistant Professor , Professor & Head,

Department of Computing Technologies Department of Computing Technologies

SRM Institute of Science and Technology SRM Institute of Science and Technology

Kattankulathur Kattankulathur

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty **Dr.R.Thamizhamuthu, Assistant Professor, Computing Technologies** for his/herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD, Dr. M Pushpalatha  Professor & Head, Computing Technologies** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

# ABSTRACT

# A random password generator is software program or hardware device that takes input from a random or pseudo-random number generator and automatically generates a password. Random passwords can be generated manually, using simple sources of randomness such as dice or coins, or they can be generated using a computer. Speaking regarding the system, the user can create a random password according to various sizes. After creating a random password, the system presents it in the clipboard where the user can copy and paste easily.This GUI based Password Generator supplies the most basic method for generating a solid password for the individuals. In short, this job just concentrates on producing arbitrary passwords. In order to run the task, you must have set up Python, on your PC. This is a basic GUI Based system, specially composed for the beginners. Password Generator in Python with source code is complementary to download. For the project demo, look at the picture slider listed below.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO** | | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** | | **1** |
|  | 1.1 Motivation | | **1** |
|  | 1.2 Objective | | **1** |
|  | 1.3 Problem Statement | | **2** |
|  | 1.4 Challenges | | **2** |
| **2** | **LITERATURE SURVEY** | | **3** |
| **3** | **REQUIREMENT ANALYSIS** | | **4** |
| **4** | **ARCHITECTURE & DESIGN** | | **5** |
| **5** | **IMPLEMENTATION** | | **6** |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | | **9** |
| **7** | **CONCLUSION** | | **11** |
| **8** | **REFERENCES** | | **12** |

**1. INTRODUCTION**

**1.1 The motivation**:

Creating a random password generator in Python can be motivated by several reasons, including security, convenience, and compliance with best practices. Here are some of the key motivations for building a random password generator:

* **Security:** Randomly generated passwords are generally more secure than easily guessable passwords. They are less susceptible to brute force attacks, dictionary attacks, and other common password cracking methods.
* **Password Strength:** A password generator can ensure that passwords meet certain strength criteria, such as length, complexity (mix of uppercase, lowercase, numbers, and symbols), and avoidance of common patterns or words. This helps users create strong passwords that are difficult to crack.
* **Learning and Skill Development:** Building a password generator is a valuable exercise for learning and practicing Python programming and cryptographic techniques.
* **Customization:** You can tailor a password generator to meet specific requirements, such as creating passwords for different systems, services, or user groups, or generating passwords that are compatible with specific password policies.

**1.2 OBJECTIVE:**

When developing a random password generator in Python, it's important to define clear objectives to guide the development process. Objectives help you understand the purpose and functionality of your password generator and provide a framework for its design and implementation. Here are some common objectives for a random password generator in Python:

* **Password Strength:** Ensure that the generated passwords are strong and resistant to common attacks. This includes using a mix of uppercase letters, lowercase letters, numbers, and special characters.
* **Randomness:** Generate passwords that are truly random and not predictable. Avoid using easily guessable patterns or common words.
* **Customization:** Allow users to customize password parameters, such as length and character set, to meet specific requirements.

**1**

* **Security:** Implement the generator in a secure way to protect the generated passwords from exposure. This includes not storing passwords in plain text and using appropriate cryptographic libraries if needed.
* **User-Friendly:** Make the generator easy to use for non-technical users. Provide clear instructions and, if applicable, a user-friendly interface.

**1.3 PROBLEM STATEMENTS:**

Problem statements are a way to articulate the specific challenges or needs that a random password generator in Python aims to address. These problem statements can help you define the scope and purpose of your project. Here are some problem statements you might consider for a random password generator:

1. **Password Security Enhancement:** Develop a Python password generator that can create strong, random passwords to enhance the security of user accounts, preventing common password-based attacks.
2. **User-Friendly Password Creation:** Create a user-friendly Python application that simplifies the process of generating secure, complex passwords for individuals who struggle with creating strong passwords on their own.
3. **Customizable Password Generator:** Build a Python-based password generator that allows users to customize password parameters, including length, character set, and complexity requirements, to meet specific security policies or personal preferences..

**1.4 CHALLENGES**

Creating a random password generator in Python comes with several challenges, especially when you aim to develop a robust and secure tool. Here are some common challenges you may encounter:

* **Password Strength:** Ensuring generated passwords are truly strong and resistant to common attacks, including ensuring the use of a variety of character types (uppercase, lowercase, digits, special characters) and avoiding predictable patterns.
* **Randomness:** Generating truly random passwords can be challenging. Pseudorandom number generators may have limitations, and achieving true randomness often involves using external sources of entropy.
* **Customization:** Allowing users to customize password parameters (e.g., length, character set) while ensuring that user-generated passwords are still strong and secure.

**2**

**2. LITERATURE SURVEY**

This literary survey reviews a Flask-based web application for generating and managing random passwords. The application allows users to generate multiple passwords with specific parameters and store them in an SQLite database. It features a user-friendly web interface and leverages Flask, SQLite, and Python for implementation.

* **Introduction:**

Password management is crucial in today's digital age. This project aims to provide a solution for users to generate and store secure passwords conveniently. The Flask framework is used to create a web interface that facilitates password generation and storage.

* **Flask Web Application:**

The application is built using Flask, a popular Python web framework. Flask simplifies web development by providing tools and libraries for handling HTTP requests and responses. It enables easy routing, template rendering, and interaction with the database.

* **SQLite Database:**

The application utilizes an SQLite database to store generated passwords. SQLite is a lightweight, file-based database system that doesn't require a separate server. The 'project.db' database contains a single table called 'passwords' with columns for ID, name, registration number, and password.

* **Conclusion:**

This Flask-based password generator and manager provide a simple yet effective solution for users to create and store secure passwords. It emphasizes ease of use and leverages Flask and SQLite to handle web requests and manage the database. The application can be further enhanced with features like password strength assessment and password recovery options to improve its utility. Overall, it serves as a practical example of web application development using Flask for password management.

**3**

**3. REQUIREMENTS ANALYSIS:**

The requirements for a random password generator in Python will depend on the specific use case, the target audience, and any compliance or security considerations. However, here's a list of common requirements to consider when developing a random password generator:

* **Security Requirements:**

Generate strong, random passwords that are resistant to common password-cracking techniques, including a mix of uppercase letters, lowercase letters, numbers, and special characters.

Avoid common password patterns and easily guessable sequences.

Use a secure random number generator or entropy source to ensure true randomness.

* **Customization:**

Allow users to customize password parameters, such as length and character set (e.g., exclude certain special characters).

* **User-Friendly Interface:**

Provide an easy-to-use interface for users to generate passwords.

Offer clear instructions and guidance on how to use the generator effectively**.**

* **Password History and Rotation:**

Implement password history management to prevent users from reusing recent passwords.

Support password rotation by generating new passwords at regular intervals.

* **User Education:**

Provide educational materials or tips on password security best practices to help users understand the importance of strong, unique passwords.

* **Documentation:**

Create clear and concise documentation that explains how to use the password generator, its features, and any customization options.

**4**

**4. ARCHITECTURE AND DESIGN:**

USER INPUT

(NAME, REG NO.)

RANDOM PASSWORD GENERATOR

FLASK APPLICATION

(WEB FRAMEWORK)

SQL DATABASE

DATABASE OPERATIONS

(INSERTION, SELECT)

WEB PAGE

(HTML TEMPLATES)

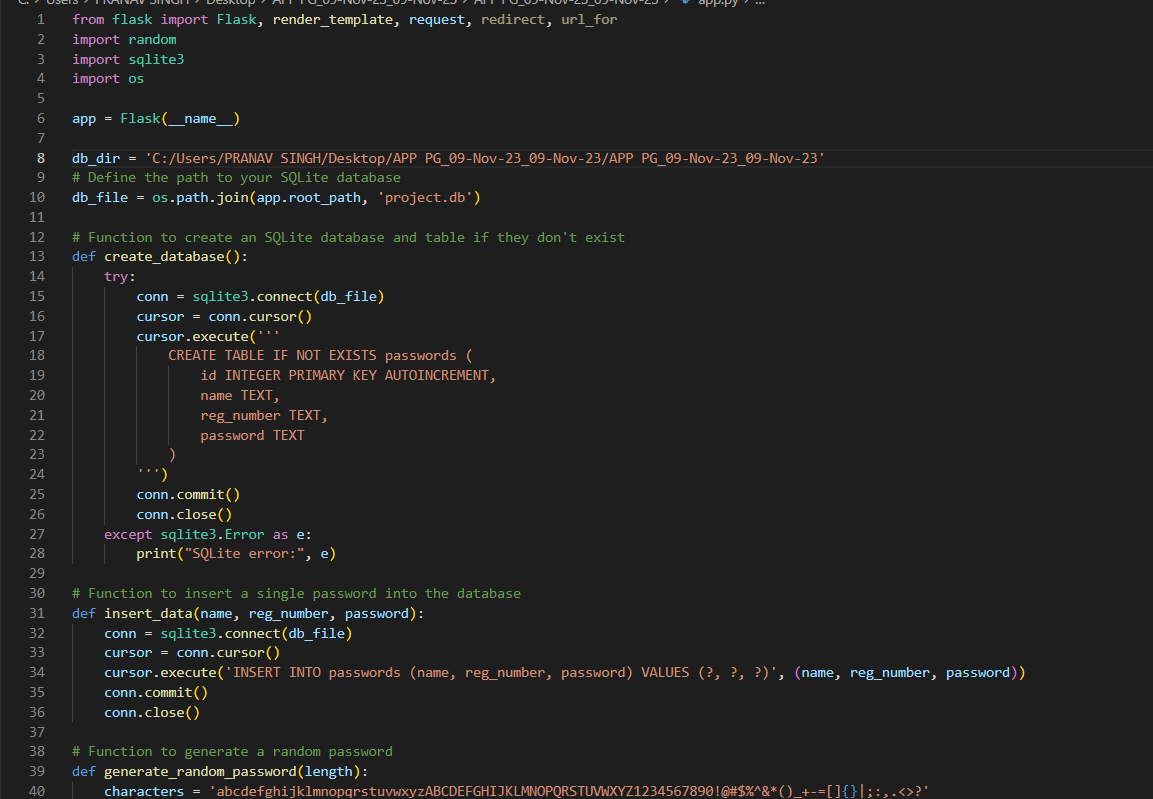
USER OUTPUT

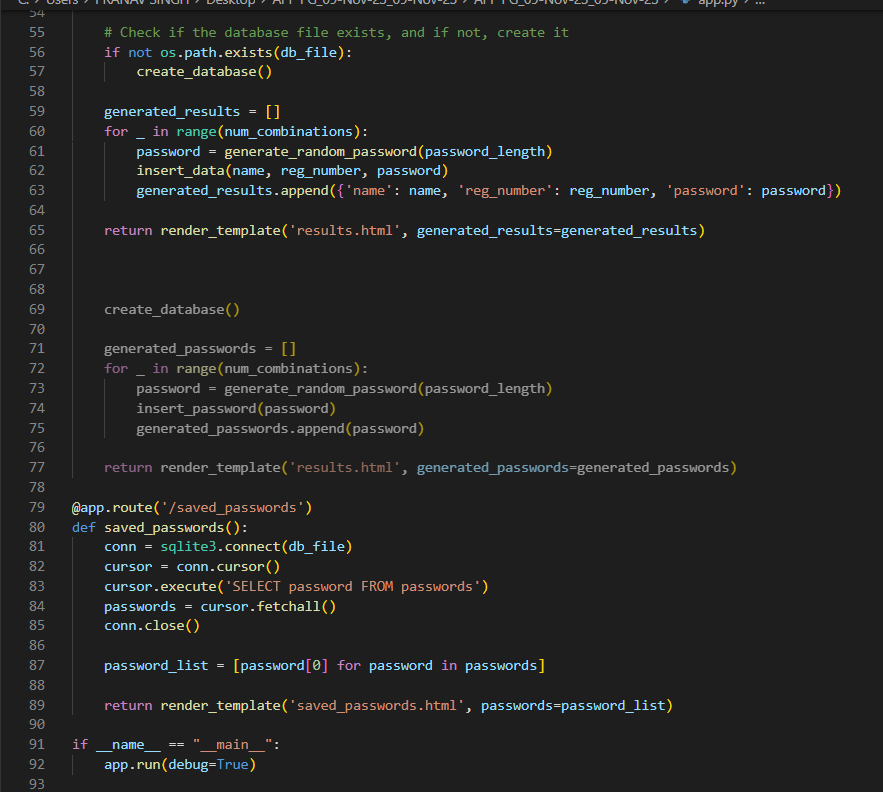
(GENERATOR PASSWORD)

**5**

**5. IMPLEMENTATION:**

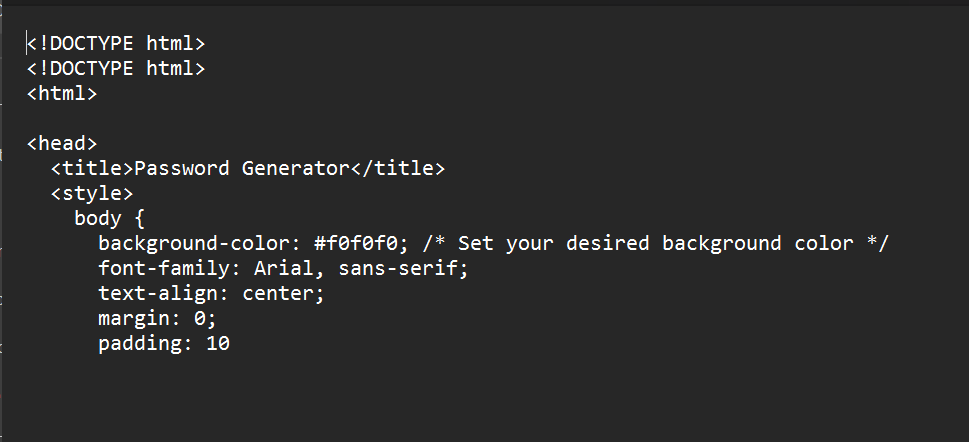
**5.1 Code:**

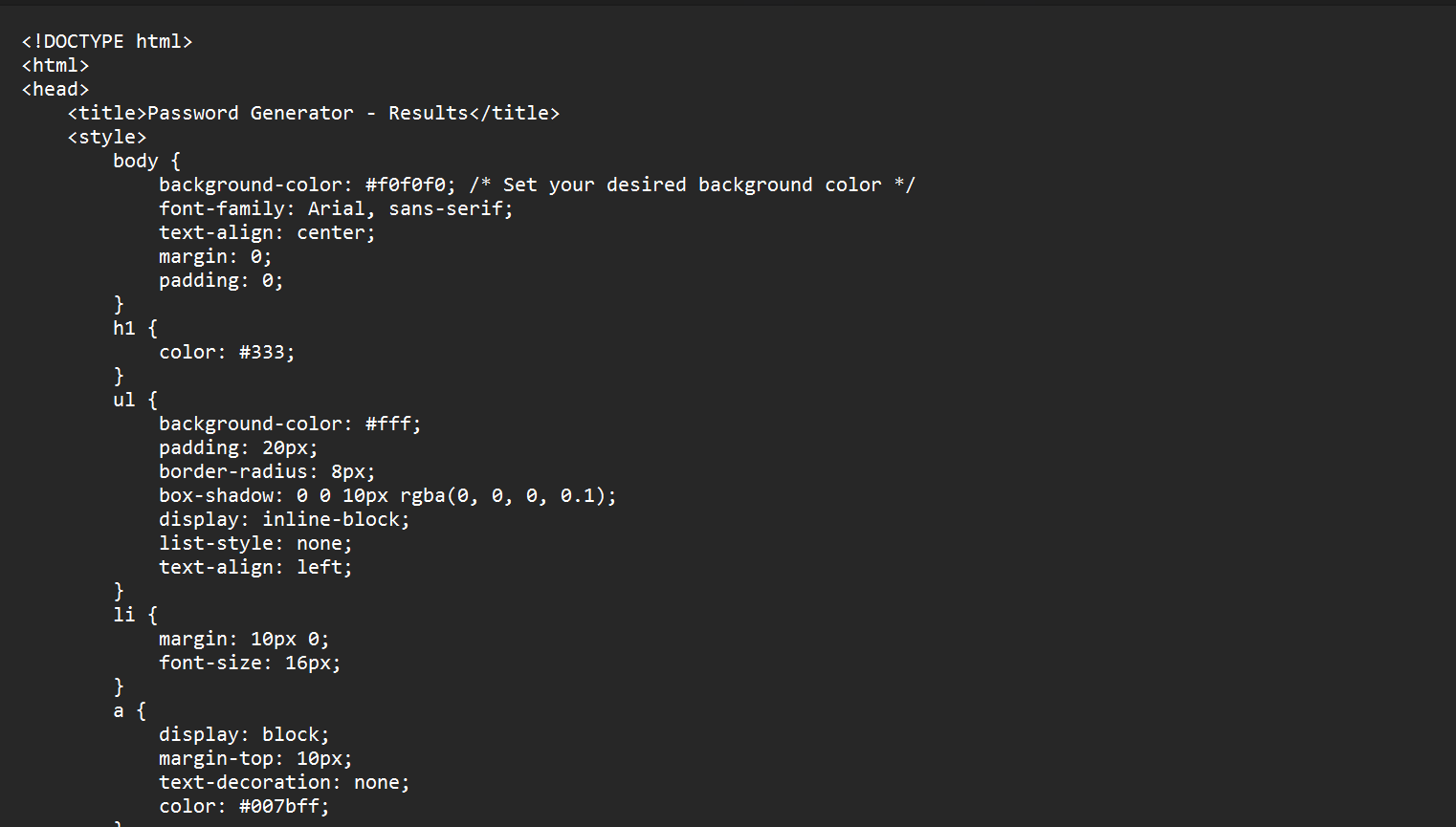
****

****

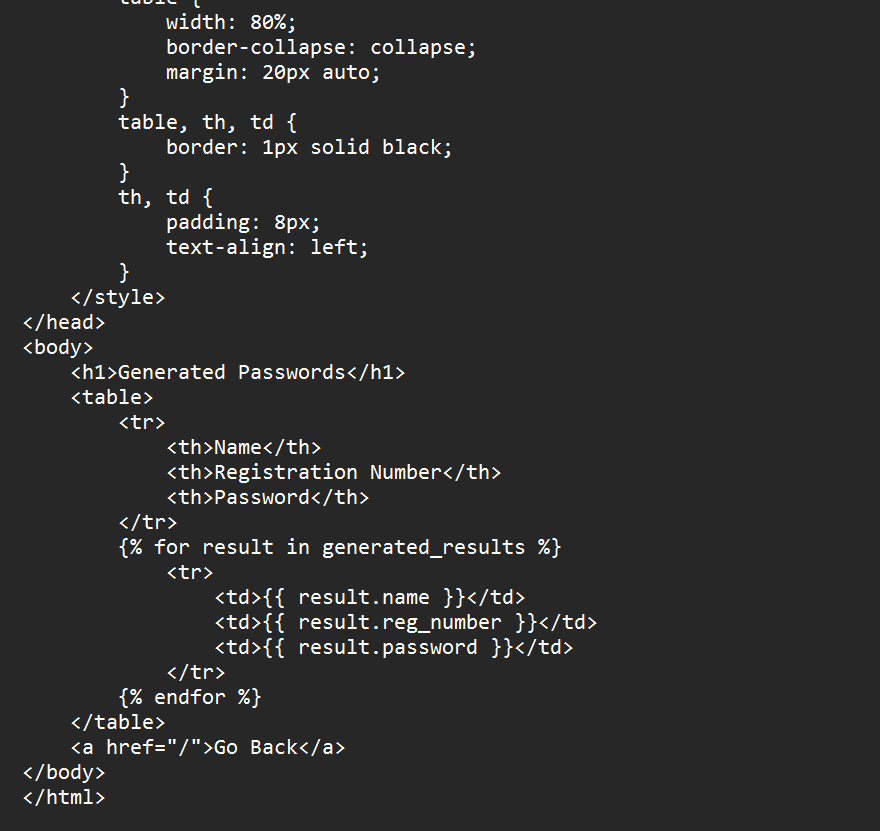
**6**

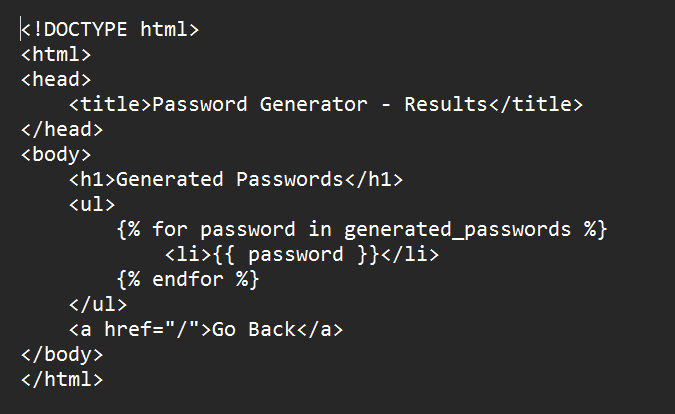
**5.2 HTML:**

****

****

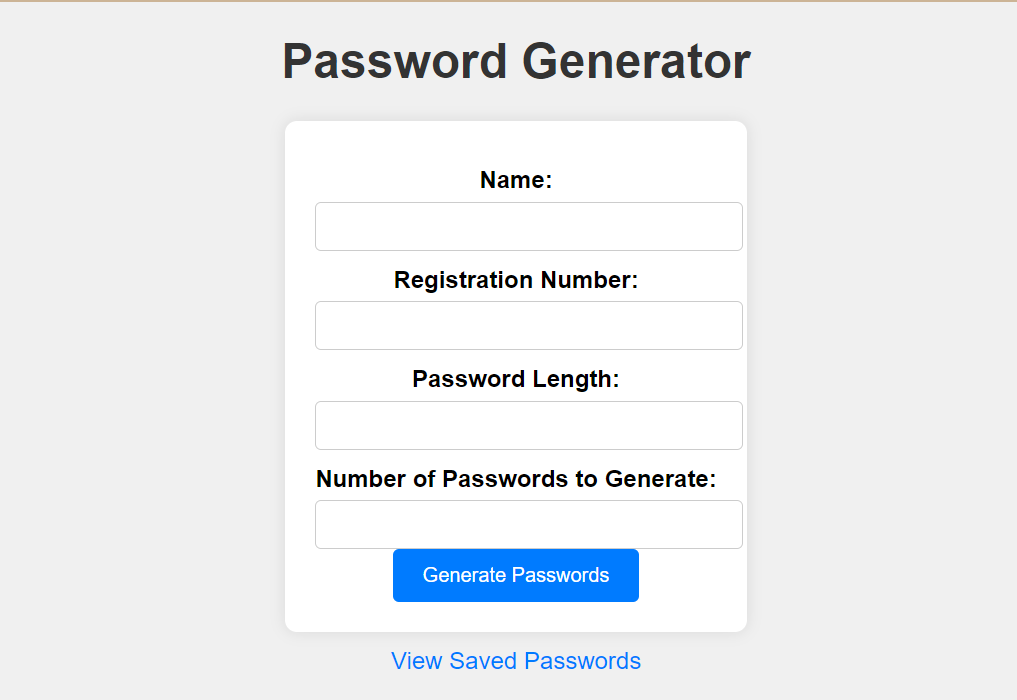
**7**

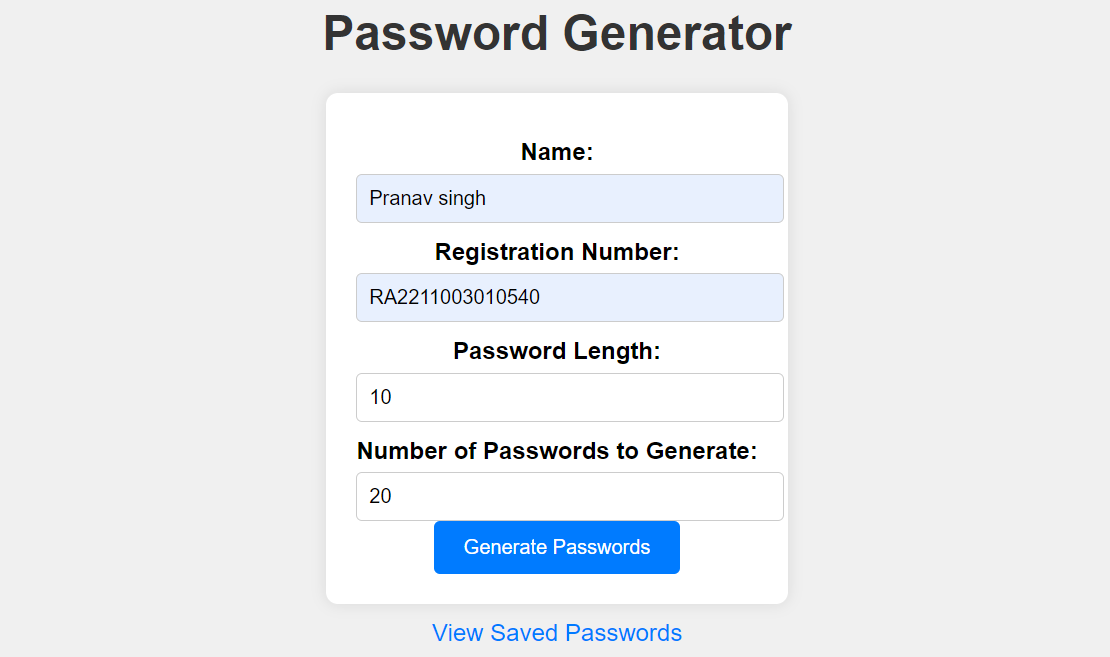
****

****

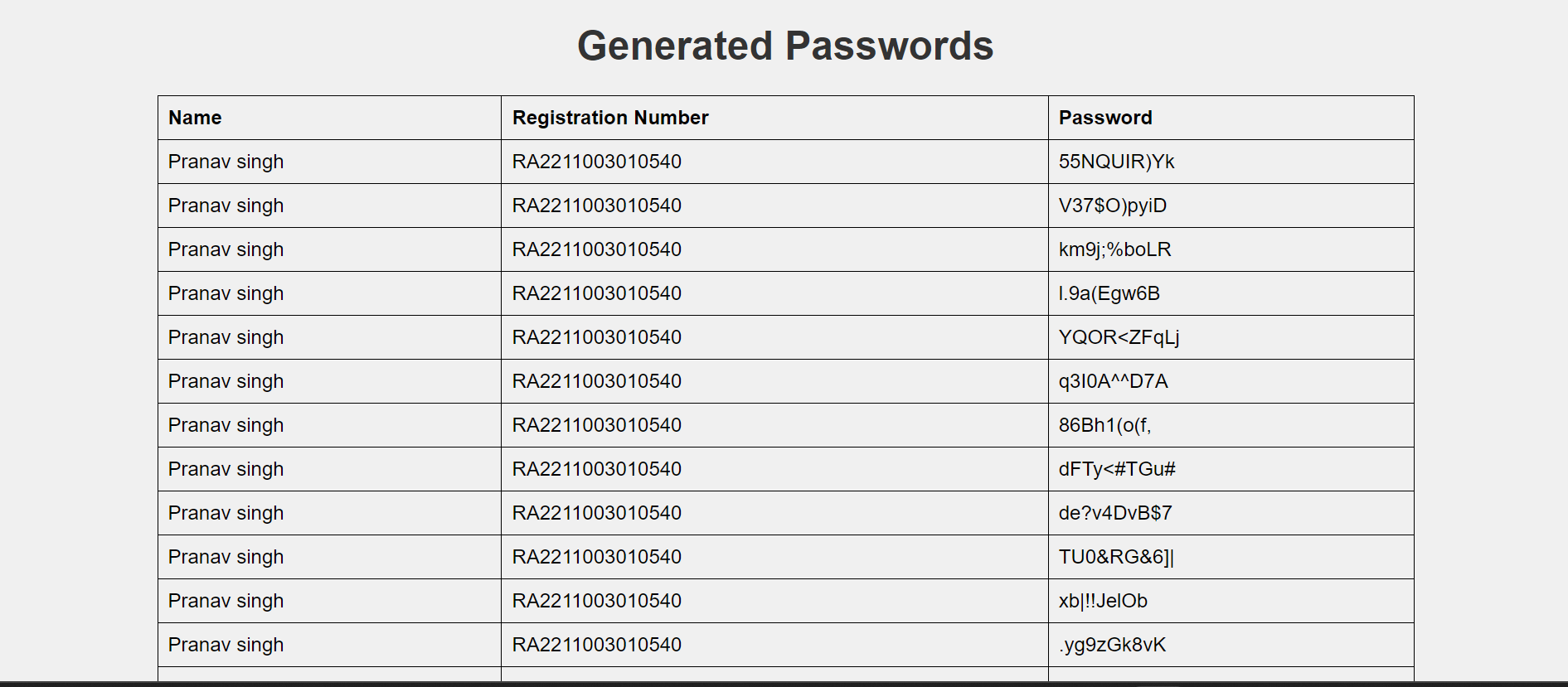
**8**

**6. RESULTS AND DISCUSSION:**

****

****

**9**



**10**

**7. CONCLUSION:**

In conclusion, a random password generator in Python is a valuable tool for enhancing security and user convenience in various contexts, from personal password management to enterprise-level security practices. It serves as a critical component of password security, helping users create strong, unique passwords and adhere to best practices. Here are the key takeaways:

* **Enhanced Security:** A random password generator creates strong, complex passwords that are resistant to common attacks, significantly improving the security of user accounts and systems.
* **User Convenience:** By simplifying the process of generating secure passwords, a password generator reduces the cognitive burden on users and encourages the use of strong, unique credentials.
* **Customization:** The ability to customize password parameters allows users to meet specific security requirements and preferences while maintaining password strength.

Ultimately, a well-designed and well-implemented random password generator in Python contributes to a stronger and more secure digital environment. It empowers users to protect their accounts and data effectively, reducing the risk of security breaches and unauthorized access. As technology evolves and security threats continue to evolve, the role of password generators remains crucial in safeguarding our digital lives.

**11**

**8. REFERENCES:**

**8.1 Geeks for Geeks:**

https://www.geeksforgeeks.org/create-a-random-password-generator-using-python/

**8.2 ChatGPT:**

https://chat.openai.com/

**8.3 Geek Flare:**

https://geekflare.com/password-generator-python-code/

**8.4 Quora:**

https://www.quora.com

**12**