MANISH TIWARI

RA2211003010546

SRMIST KTR campus

[mt1338@srmist.edu.in](mailto:mt1338@srmist.edu.in)

PRANAV SINGH

RA2211003010540

SRMIST KTR campus

[ps3490@srmist.edu.in](mailto:ps3490@srmist.edu.in)

**RANDOM PASSWORD GENERATOR**

SWASTIK RANA

RA2211003010564

SRMIST KTR campus

[sr9849@srmist.edu.in](mailto:sr9849@srmist.edu.in)

***Abstract*—** This paper presents an Random Password Generator System meticulously engineered to elevate the efficiency and security for networking. The web application, Random Password Generator, offers its user’s safe procedures in getting some random passwords. The system prompts users to type in the number of characters that they want to use for password and additional details like their login name & registration number. Generally, the application creates a bunch of passwords on request. There are the results shown in an easy format, listed under a heading on a tidy good-looking site. It provides a convenient access mechanism where one can revisit the input form on getting more random passwords. This password app seeks to facilitate the making of robust and private passwords using preferences preferred by each user alone. In today’s word it is very easy to crack the password so it will definitely help in the domain of networking by providing a set of strong passwords.

1. INTRODUCTION

The Python-based Random Password Generator represents a significant development in online security domain. Introducing our most advanced Random Password Generator powered by Python for personalized and up-to-date security needs. The use of Python and multiple libraries in this application allows you to effortlessly produce a robust, customized password for various accounts and services. Our generated also ensures strong password creation by applying the flexibility of Python to include desired password length as well as optional details such as username or registration number. We try to offer customer friendly product based on the security principle yet compatible with the simplicity of the Python based programming.

1. Literature Survey
2. *Python Programming*
   * "Python Programming: An Introduction to Computer Science" by John Zelle- Smith,
   * "Python Crash Course" by Eric Matthes
   * These studies help us in understanding the key and basic concept of python along with help in the understanding and development of some basic python projects.
3. *SQlite*
   * "Using SQLite" by Jay A. Kreibich"The Definitive Guide to SQLite" by Michael Owens

These book help us in the development and understanding the key ideas of using mysqlite and help us in understanding the key ideas about relational database.

1. *Key existing studies*
   * Robert Biddle, Mohammad Mannan, Paul C van Oorschot, and Tara Whalen. User study, analysis, and usable security of passwords based on digital objects. IEEE Transactions on Information Forensics and Security, 6(3-2):970–979, 2011.
   * Sonia Chiasson, Paul C. van Oorschot, and Robert Biddle. A usability study and critique of two password managers. In Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 – August 4, 2006. USENIX Association, 2006
   * This help us in the development of key ideas behind the the development of a random password generator using python and diversify our knowledge.
2. *Data security*
   * Clark, David. "Enhancing Data Security in Banking: En- cryption and Authentication Methods." International Journal of Cybersecurity, vol. 12, no. 4, 2022, pp. 90-105.
   * Brown, Sarah. "Biometric Authentication in Banking Ap- plications: A Comparative Study." Journal of Cybersecurity Technology, vol. 8, no. 3, 2021, pp. 110-125.

These studies focus on data security, particularly en- cryption methods and biometric authentication techniques relevant to banking applications.

1. METHODOLOGY
2. *Level 1: User Details*

At level 1 of the Python-driven random password generator, users are required to input a username alongside their registration number. The system uses the principles of object oriented programming to ensure accurate operations on data. It is a final level which sets off and concludes activities with the password giving a seamless experience. The first step is my system level which secures the storage of the user data upon which the following application levels depend.

1. *Level 2: Password Generation*

In level 2, users take action by selecting the desired length of the password and thus creating a customized experience. Python is the driving force behind the diverse alphabets, numbers, and special symbols that constitute passwords. MySQL provides safe storage of data thus ensuring high speed user experience as preparations are undertaken for application developments.

1. *Level 3: Selection Of Password*

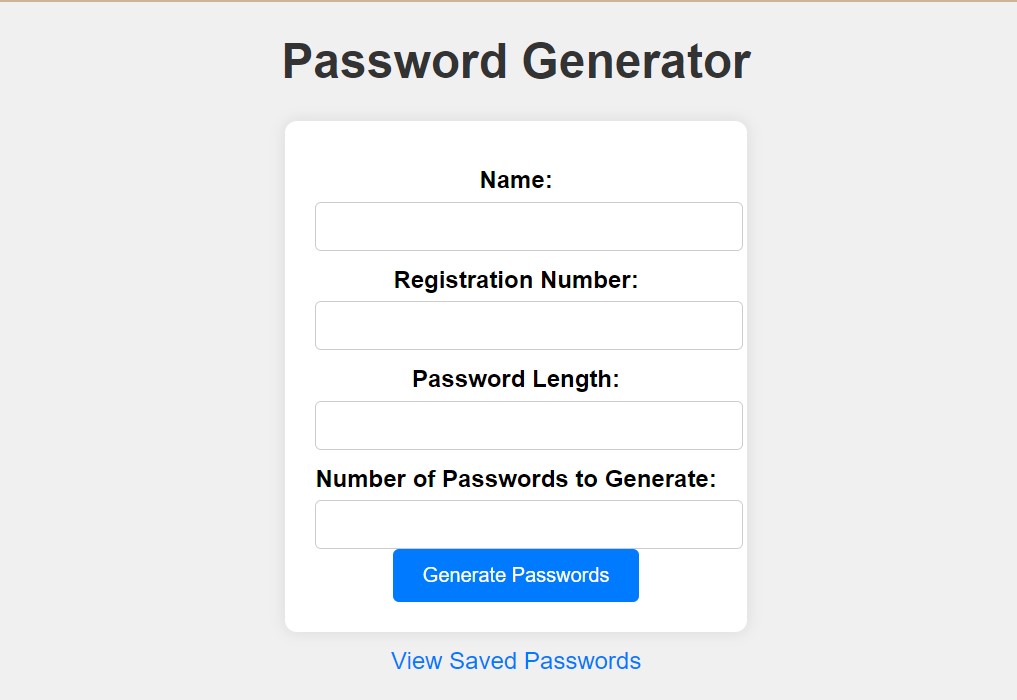
At the level three, there is a selection made on password parameters which include choosing from preset options related to length and number. However, this python-driven password generator is highly user-friendly adding to its flexibility. It automatically produces a specified number of randomly generated passwords with different lengths in their composition. Despite this, MySQL keeps on ensuring that data is stored securely, enhancing the user’s power towards creating personalized passwords.

1. *Level 4: Saving Of Password*

At Level 4, the generated passwords become persistent by saving them using SQLite. However, this integration makes it possible to store passwords generated by a Python driven password generator that are strong and diverse efficiently, so as to allow future access to them. The personalized system for password management remains uninterrupted and user-friendly throughout this period.

1. Results

The Python-based random password generator represents an adaptable and user-centric solution. Offering precise data management and customization, it empowers users to craft secure passwords. The integration of SQLite ensures efficient storage, reinforcing the application's reliability and user-focused design. The following key results were obtained:



# User Details:

The user details improves the password generator by making it possible for users to enter usernames and registration numbers. This provides for a more personalized password generation process and results in improved security in addition to the users feeling more bonded to the application.

# Password Generation:

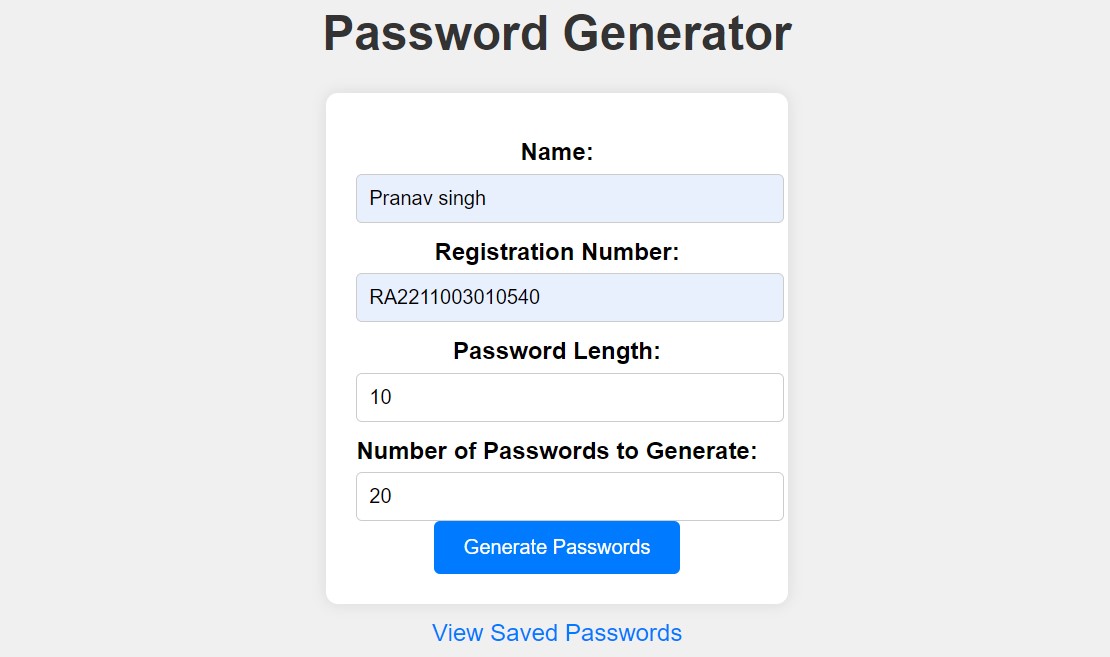
The passwords are composed using specific user specifications such as length and provide an individualized and protected result. Additionally, Python provides flexibility which makes it possible for diverse characters such as alphabets, digits, and special characters to be integrated. In turn, this creates strong unique passwords that suit each situation.

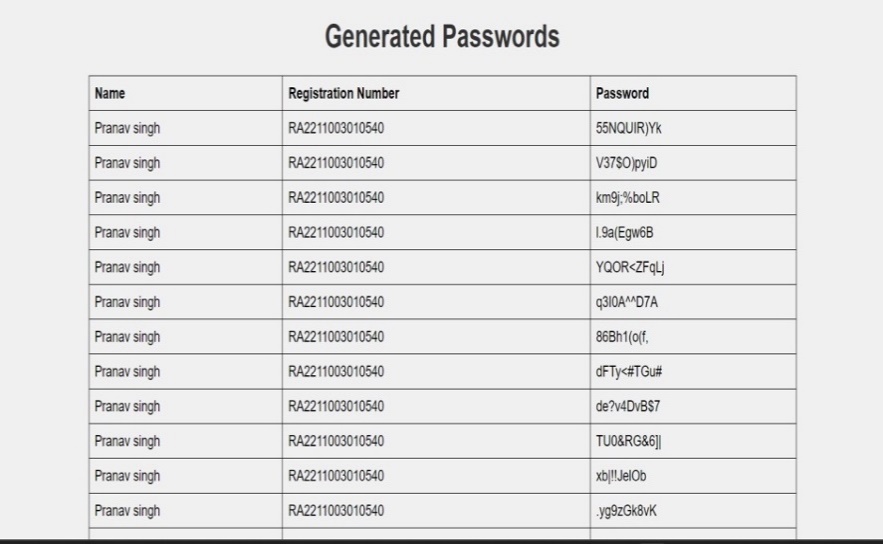
# Selection Of Password:

Users choose password parameters from a list of available predefined ones pertaining to their length and quantity. The user friendly functionality makes Python driven password generator flexible and customizable. It generates a specified number of random passwords per user. These random passwords vary dynamically depending on the user’s preference for password composition.

# Saving Of Password:

Passwords generated are persistent since they are stored securely via SQLite. This integration optimizes data storage, enabling users to store, retrieve, and use their created passwords in subsequent log-ons. They are personalized password management which is guaranteed seamless and dependable continuity by Python-driven password generator.





# Data Protection:

The password generator ensures data protection by adopting secure storage mechanisms for both MySQL and SQLite. This integration ensures user information confidentiality and integrity basis for trusted handling and durable storage of sensitive data build trust and user’s privacy.

Data protection refers to the practice of safeguarding sensitive information from unauthorized access, use, or disclosure. It involves implementing measures to ensure the confidentiality, integrity, and availability of data. This encompasses various aspects, including securing personal information, preventing data breaches, and complying with privacy regulations. Effective data protection strategies include encryption, access controls, regular audits, and risk assessments. Organizations must establish policies and procedures to govern data handling, educate employees on best practices, and deploy technology solutions to mitigate risks. As digital environments become more complex, data protection plays a crucial role in maintaining trust with individuals and complying with legal requirements, ultimately contributing to the responsible and ethical use of information in both business and personal contexts.

# Error Handling and Recovery:

The password generator also ensures robust data protection through error handling and recovery mechanisms. The application also uses effective error management to reduce the risk of data breaches in the user’s sensitive information and thereby making it more secure and reliable under unexpected situations.

Overall, the results demonstrate that the Python-Based Random Password Generator fulfills its intended purpose of enhancing the efficiency and security operations.

Data protection refers to the practice of safeguarding sensitive information from unauthorized access, use, or disclosure. It involves implementing measures to ensure the confidentiality, integrity, and availability of data. This encompasses various aspects, including securing personal information, preventing data breaches, and complying with privacy regulations. Effective data protection strategies include encryption, access controls, regular audits, and risk assessments. Organizations must establish policies and procedures to govern data handling, educate employees on best practices, and deploy technology solutions to mitigate risks. As digital environments become more complex, data protection plays a crucial role in maintaining trust with individuals and complying with legal requirements, ultimately contributing to the responsible and ethical use of information in both business and personal contexts.

A. Performance Evaluation

The password generator evaluation demonstrates its effectiveness in taking a user input and generating a strong password. Utilization of flexible features in Python along with MySQL and SQLite that enables fast and secure password creation. Moreover, it should be noted that this application can gracefully handle errors and recover without jeopardizing data security for improved performance. Together, this intuitive user interface, personalized password options, and efficient data protection mechanisms make a top notch security password creator tailored towards specific needs.

B. Security Assessment

The security assessment highlights the formidable defense of the password generator. Storage of data is ensured with SQLite integration, thereby reducing exposure to unauthorised access. Using different character sets, along with error handing makes the system stronger against possible intrusions. This shows a proactive approach to cybersecurity through the proper encryption of the application’s data. Continued monitoring ensures that the application remains resistant to identified vulnerabilities hence reaffirms the commitment to data protection.

In addition, using password generator ensures that one’s privacy is always maintained as this is how Python operates. From error recovery to encryption it has many aspects of security which together provide a stable and reliable platform for designing tailor-made and strong passwords.

C. User Experience Evaluation

Password generator’s focus on a user experience evaluation is further evidence of this dedication for smooth interaction. It has a user-friendly interface with personalized data entry options such as username and registration number, thus increasing user involvement. The user-friendly parameter selection makes this tool flexible enough so that users can define the characteristics of their passwords. Positive and smooth user experience results from dynamic password generation and SQLite efficient storage. Moreover, competent error handling enables a smooth conversation which enhances satisfaction. The password generator through intelligent design and user-centered features emerges as a trusted and handy tool for customized and secure password generation.

D. Reliability and Error Handling

The password generator demonstrates very good level of reliability and error handling that provide a reliable and user friendly experience. Integration of SQLite ensures secure data storage in turn prevents losses and inconsistencies. Adequate error handling strategies combined with efficient recovery measures ensure smooth functioning even during surprises. The password is generated with utmost reliability, either through user input validation or by securely storing the data. The continuous monitoring and active error management improve its security, making it a good software with which to create and manage passwords accurately and persistently.

E. Compliance and Regulatory Adherence

Password generator has also shown willingness to compliance and regulatory requirement in securing users’ information on their side. It has the same security measures as the standards for data storage that use encryption technologies and secure protocols. The application complies with data protection regulations and standards in safeguarding the privacy and security of user data. The password generator’s system-

driven approach to user data and strict adherence to privacy and security laws and regulations demonstrates its effort to comply with all relevant legal requirements. The application’s proactive stance makes it a reliable tool whose practices meet not only industry standards but also compliance frameworks thus ensuring a secure and responsible user experience.

1. Conclusion

The Python password generator is thus a flexible approach and responds to shifting paradigms involved in creating robust passwords. Precision in dealing with their information is what forms the basis of this application’s design. This enables users to create passwords for the purpose of securing customers’ information using their own preferences.

The evolution of the generator is apparent in its provision for customization, which permits users to decide on password lengths and a wide range of characters, comprising alphabets, digits, as well as special symbols. This simple solution addresses different needs of users and improves the safety and uniqueness of chosen passwords.

The use of parameter selection is a critical improvement in this application that gives users control over the length and number of resulted passwords. Such a shift indicates more power to the users, giving them a more personalized, and customizable experience. This ability of the generator to provide passwords, dynamically generated depending on the set of user-defined parameters, helps create the composition that is specific in its content, in relation to the personal understanding of individual security needs.

Secondly, the incorporation of the SQLite during the latter stage of the application development is a great milestone in securing the database. The application uses SQLite to ensure that the random passwords are safe and can be kept for future reference. This ensures that the whole system is reliable in ensuring safe accessing to passwords by the users at a time they need.

The effort of SQLite in the password generator shows that they want their customers’ data to be securely saved. Adaptivity, individualization and security define user’s interaction, making the application a trustworthy tool for personalized password generation.

This password generator will be constantly changing in accordance with developing technologies and security needs. With its user-oriented outlook and data security consciousness, it is a flexible tool that could meet all the changing demands of users about the password development. Essentially, the final part does not only represent the end of the described properties but also indicates an ongoing process of improvements and adjustment to users’ needs and industry requirement

VI.APPENDIX

* 1. Additional Code Snippets: - Password generation algorithms - User input handling
  2. Technical Specifications or System Requirements: - Minimum Hardware Requirements - Supported Operating Systems - Database Management System - Programming Language and Version
  3. User Interface Mock-ups or Wireframes: - Navigation Flow Diagram for User Interactions
  4. Sample Input-Output Data for Testing: - Include user input validation- Error handling under different condition

5. System Flowcharts: - High-Level Flowchart depicting User Interac- tions - Detailed Flowchart for Password Generation Process

6.Database Schemas: - Entity-Relationship Diagram (ERD) for Database - Table Definitions,

7. User Manuals: - User Guide for Using the Random Password Generation- Step-by-Step Instructions for Transactions

References

1. FATMA AL MAQBALI;CHRIS J MITCHELL "AUTOPASS: AN AUTOMATIC PASSWORD GENERATOR" YEAR: 2017 | CONFERENCE PAPER | PUBLISHER: IEE

2.PAUL MCFEDRIES "WEB DESIGN PLAYGROUND: HTML & CSS THE INTERACTIVE WAY" YEAR: 2019 | BOOK | PUBLISHER: MANNING

3. Eric Matthes "Python Crash Course" Year: 2015 | Book | Publisher: No Starch Press

4. Jon Duckett "HTML and CSS: Design and Build Websites" Year: 2011 | Book | Publisher: John Wiley & Sons.

5.Sonia Chiasson, Paul C. van Oorschot, and Robert Biddle. A usability study and critique of two password managers. In Proceedings of the 15th USENIX Security Symposium, Vancouver, BC, Canada, July 31 – August 4, 2006. USENIX Association, 2006.

6.Graham Cluley. Lastpass vulnerability potentially exposed passwords for internet explorer users. https://www.grahamcluley.com/2013/08/lastpass-vulnerability/, August 2013.

7. Ahmet Emir Dirik, Nasir D. Memon, and Jean

Birget. Modeling user choice in the passpoints graphical password scheme. In Lorrie Faith Cranor, editor, Proceedings of the 3rd Symposium on Usable Privacy and Security, SOUPS 2007, Pittsburgh, Pennsylvania, USA, July 18–20, 2007, volume 229 of ACM International Conference Proceeding Series, pages 20–28. ACM, 2007.

8. FIDO Alliance. FIDO UAF Protocol Specification v1.0: FIDO Alliance Proposed Standard 08, December 2014.

9.Dinei A. F. Florˆencio and Cormac Herley. A large-scale study of web password habits. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8–12, 2007, pages 657–666. ACM, 2007.

10. J Alex Halderman, Brent Waters, and Edward W Felten. A convenient method for securely managing passwords. In Allan Ellis and Tatsuya Hagino, editors, Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005, pages 471–479. ACM, 2005.

11.D. Hardt (editor). The OAuth 2.0 Authorization Framework. Internet Engineering Task Force (IETF), October 2012.

12. Cormac Herley, Paul C. van Oorschot, and Andrew S. Patrick. Passwords: If we’re so smart, why are we still using them? In Roger Dingledine and Philippe Golle, editors, Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers, volume 5628 of Lecture Notes in Computer Science, pages 230–237. Springer, 2009.

13. Moritz Horsch, Andreas H¨ulsing, and Johannes A. Buchmann. PALPAS - passwordless password synchronization. arXiv:1506.04549v1 [cs.CR], http://arxiv.org/abs/1506.04549, 2015.