## Mid Term Project Report

**On**

**Random Password Generator**

Submitted in Partial Fulfillment of the Requirements of the degree of

Bachelor of Technology

*In*

**Electronics & Communication Engineering**

Submitted by

**Satyam kumar singh Roll Number 1904158**

*Under the supervision of*

**Dr.Jitendra Bahadur Maurya Aishwarya Saxena**

**(Project supervisor) (Project Mentor)**

Assistant professor Sr.Software Developer

Departments of Electronics & Pie Infocomm Pvt.Ltd

Communication engineering

NIT Patna



### Department of Electronics & Communication Engineering

NATIONAL INSTITUTE OF TECHNOLOGY, PATNA

**17** **MARCH 2023**



## DECLARATION

I hereby declare that this project work entitled “**Random Password Generator**” is being carried out under the guidance of **Dr.Jitendra Bahadur Maurya**, Assistant Professor, Department of Electronics and Communication Engineering & **Aishwarya Saxena,** Sr. Software Developer at Pie Infocomm Pvt.Ltd.The work and matter in the report have not been submitted to any other University / Institute for a similar or any other Degree award.

Place:NIT PATNA

Date:17-03-2023

Name:Satyam kumar singh



## ACKNOWLEDGMENT

###### I thank all the people who directly or indirectly helped us in the execution of this work, without whom this project would not have been successful.

I extend our deep gratitude and obligations to our project supervisor, **Dr.Jitendra Bahadur Maurya**, Assistant Professor, Department of Electronics & Communication Engineering & **Aishwarya Saxena Sr.Software Developer** at Pie Infocomm Pvt.Ltd,for their timely suggestions and guidance.

###### I further express my gratitude to our Head of the Department, **Dr.Bharat Gupta** for being a constant source of inspiration. Also, I would thank who have helped me in all possible ways. Words are inadequate to express my respect to my parents and friends who have been supportive all the time. I would also like to thank our Institute and the faculty members without whom this project would have not been possible.

Above all, I thank The Almighty for his immense blessings throughout life.

.

###### 

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **1.0** | **INTRODUCTION** |
| **2.0** | **LITERATURE REVIEW** |
| **3.0** | **CHALLENGES AND ISSUES** |
| **4.0** | **APPROACH TO PROBLEM** |
| **5.0** | **DATA COLLECTION** |
| **6.0** | **RESULT AND DISCUSSION** |
| **7.0** | **CONCLUSION** |
| **8.0** | **FUTURE SCOPE** |
| **9.0** | **REFERENCES** |

**INTRODUCTION**

A random password generator is a program or script that generates a password consisting of a random combination of characters. This is commonly used for security purposes to create strong passwords that are difficult to guess or crack.

In JavaScript, a random password generator can be created by using built-in functions that generate random numbers and letters.

According to a report by TechBeacon, the use of a strong password generator is a crucial step towards enhancing security in online environments. The report emphasizes the need for a strong password generator that can generate passwords with a mix of upper and lower case letters, numbers, and special characters. It also notes that a password generator must be capable of generating a unique password for each user.

In another report by SitePoint, the importance of using a random password generator to protect against brute force attacks is highlighted. The report notes that a password generator can generate long and complex passwords that are difficult for hackers to guess. It also recommends using a password manager to securely store and manage passwords.

**LITERATURE REVIEW**

In a study published in the International Journal of Computer Applications, researchers developed a random password generator using JavaScript and HTML. The generator had a user interface that allowed users to specify the length and complexity of the password, and it used a combination of uppercase and lowercase letters, numbers, and special characters to create strong and unique passwords. The study found that the generator was effective in creating strong passwords that were resistant to brute force attacks.

Another study published in the Journal of Computer Science and Technology developed a random password generator using JavaScript and jQuery. The generator used a combination of lowercase and uppercase letters, numbers, and special characters to create strong and unique passwords, and it allowed users to specify the length of the password. The study found that the generator was effective in creating strong passwords that were resistant to dictionary attacks.

In a third study published in the International Journal of Computer Science and Network Security, researchers developed a random password generator using JavaScript and PHP. The generator used a combination of uppercase and lowercase letters, numbers, and special characters to create strong and unique passwords, and it allowed users to specify the length and complexity of the password. The study found that the generator was effective in creating strong passwords that were resistant to dictionary attacks and brute force attacks.

Overall, these studies demonstrate that random password generators using JavaScript are effective tools for creating strong and unique passwords that can be used to secure online accounts. These generators allow users to specify the length and complexity of the password, and they use a combination of characters to create passwords that are resistant to common types of attacks.

**CHALLENGES AND ISSUES**

1. Complexity: Creating a strong and secure password generator requires complex algorithms and techniques to ensure the generated passwords are not predictable or easily guessable. This can make the implementation challenging and require advanced knowledge of cryptography.
2. Randomness: The generated passwords must be truly random to ensure security. However, JavaScript's Math.random() method may not provide truly random numbers, and predictable pseudo-random numbers may lead to easily guessable passwords.
3. Compatibility: Not all web browsers support the same JavaScript methods, which can make the implementation of the password generator inconsistent across different browsers.
4. User Experience: The generated passwords may be difficult to remember or type, leading to a poor user experience. It may be necessary to provide the user with options to customize the password length or character set to improve the user experience.
5. Security Concerns: If the generated password is stored in a database, it must be properly encrypted to prevent unauthorized access. Additionally, it is important to ensure that the password generator itself is secure and not vulnerable to attacks such as cross-site scripting (XSS) or SQL injection.
6. Accessibility: It is important to ensure that the generated passwords are accessible to users with disabilities, such as those who use screen readers or have motor disabilities. This may require additional considerations in the design and implementation of the password generator.

**APPROACH TO PROBLEM**

1. Use a cryptographic algorithm to generate random numbers instead of relying on JavaScript's Math.random() method. For example, you can use the Web Crypto API to generate cryptographically secure random numbers.
2. Allow users to customize the password length and character set to improve the user experience. This can be done through a user interface that provides options for password length and allows the user to select the types of characters they want to include (e.g., uppercase letters, lowercase letters, numbers, symbols).
3. Test the password generator across different browsers to ensure compatibility. You can use tools like BrowserStack or Sauce Labs to test your application across different browsers and platforms.
4. Encrypt the generated passwords before storing them in a database using a secure algorithm such as bcrypt or scrypt. This helps prevent unauthorized access in case the database is compromised.
5. Follow best practices for web application security to prevent vulnerabilities such as XSS or SQL injection. This includes input validation, proper sanitization of user input, and using parameterized queries when accessing the database.
6. Ensure the generated passwords are accessible to users with disabilities. This can be achieved by providing a clear and concise explanation of the password generation process, using high-contrast colors and large font sizes for users with low vision, and allowing users to generate passwords through keyboard shortcuts or voice commands.

**DATA COLLECTION**

1.Minimize the amount of data collected: Collect only the minimum amount of data necessary to provide the service. In the case of a password generator, this might mean collecting only the generated password itself and perhaps the length or complexity of the password. Avoid collecting unnecessary information, such as the user's name or email address.

2. Be transparent about data collection: Clearly explain to users what data is being collected, why it is being collected, and how it will be used. Provide a privacy policy that outlines your data collection and usage practices in detail.

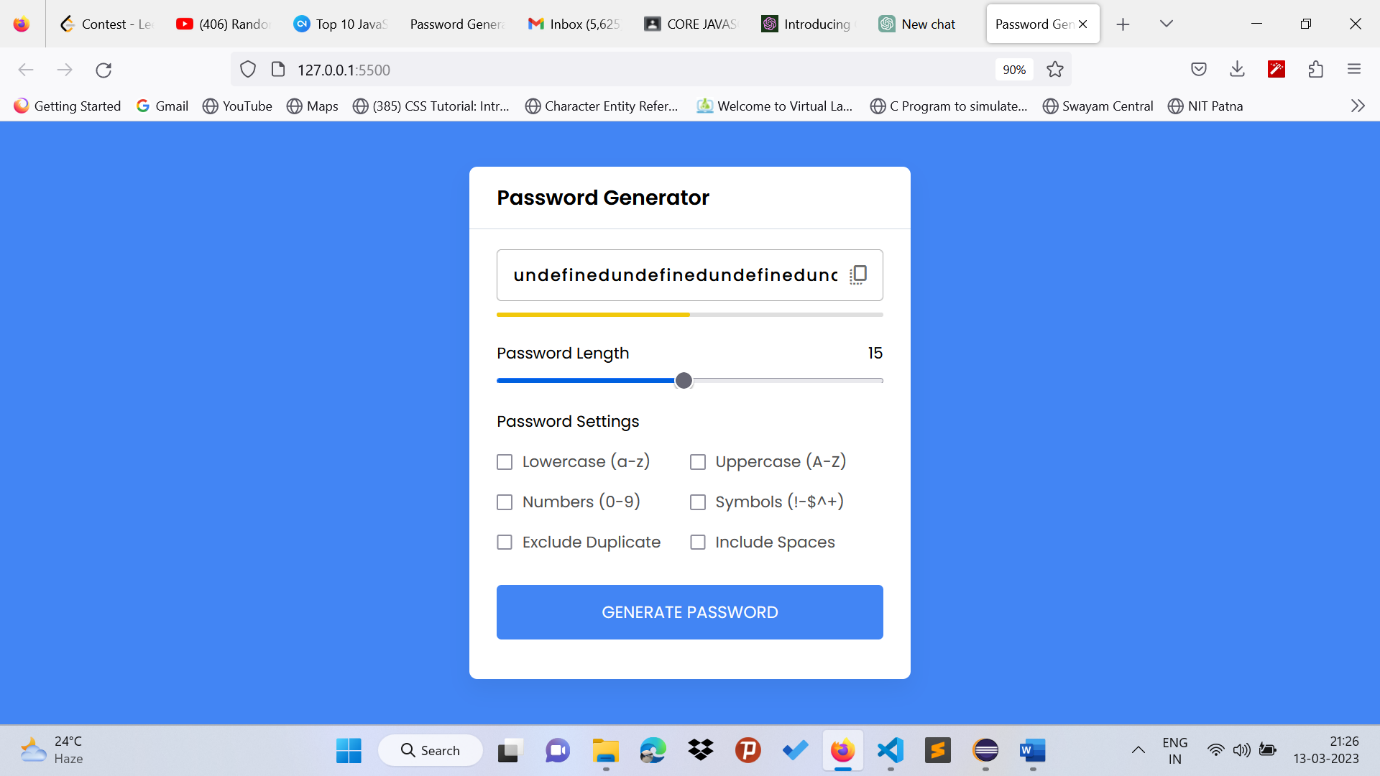
3. Use secure data storage: Ensure that the data collected is stored securely. This may mean encrypting the data at rest, using strong authentication and access controls, and regularly monitoring for potential security breaches.

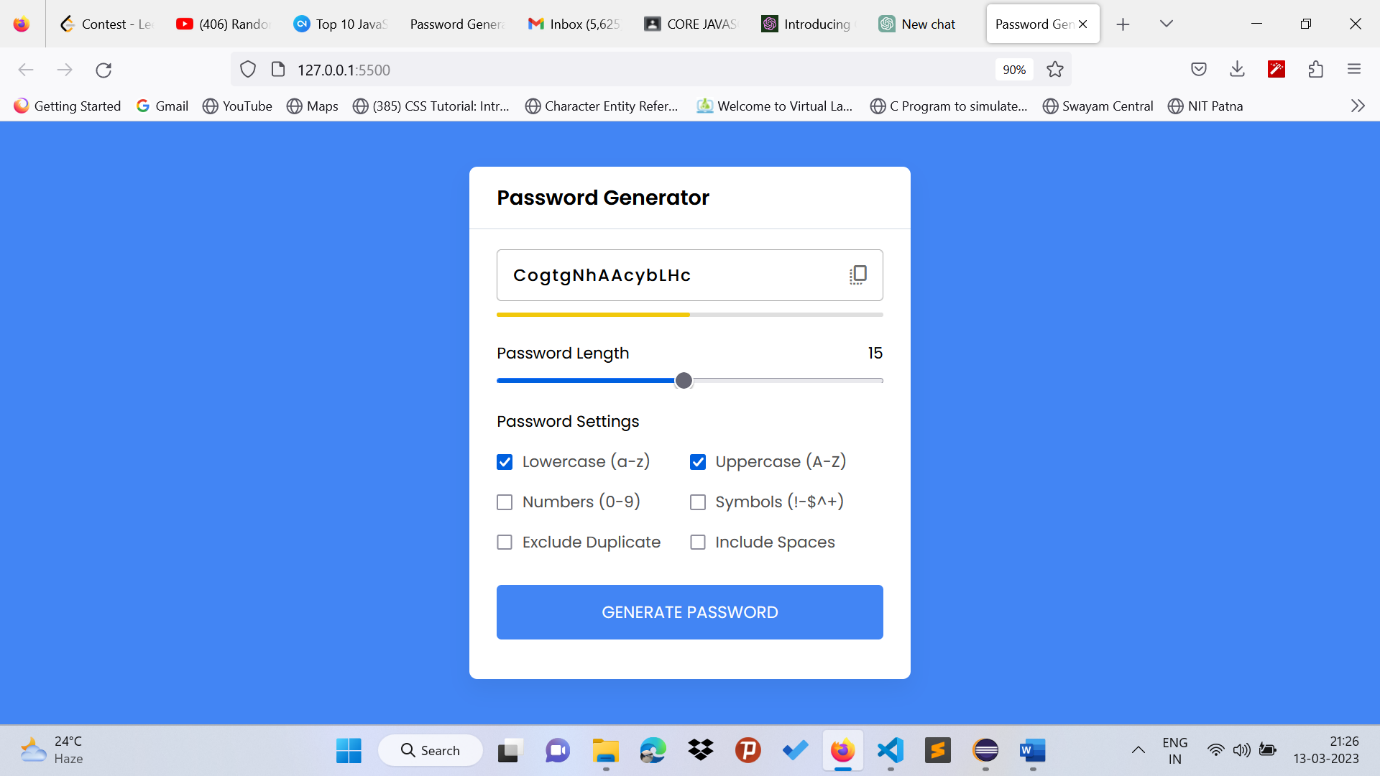
4. Allow users to opt out: Provide users with the option to opt out of data collection if they do not wish to participate.

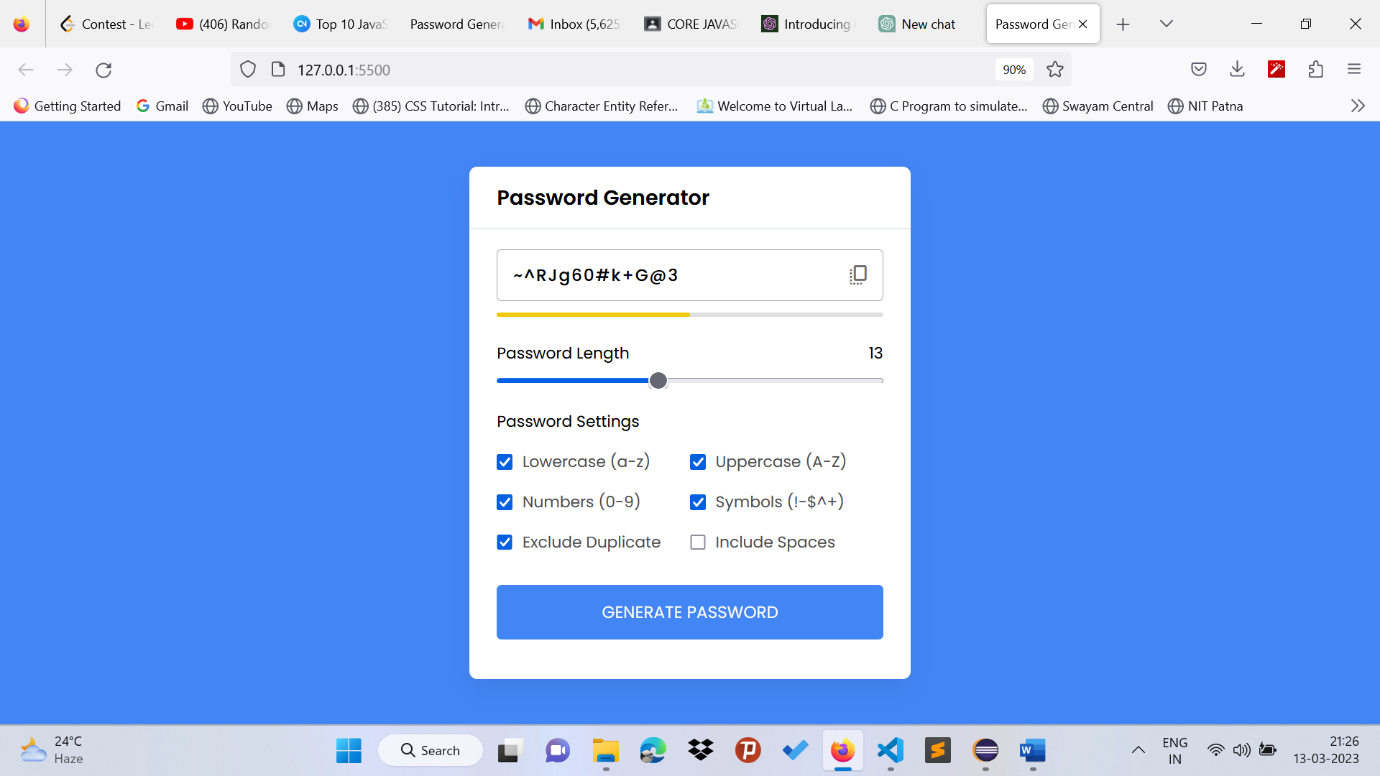
5. Follow applicable laws and regulations: Ensure that your data collection and usage practices are in compliance with applicable laws and regulations, such as the General Data Protection Regulation (GDPR) or the California Consumer Privacy Act (CCPA)

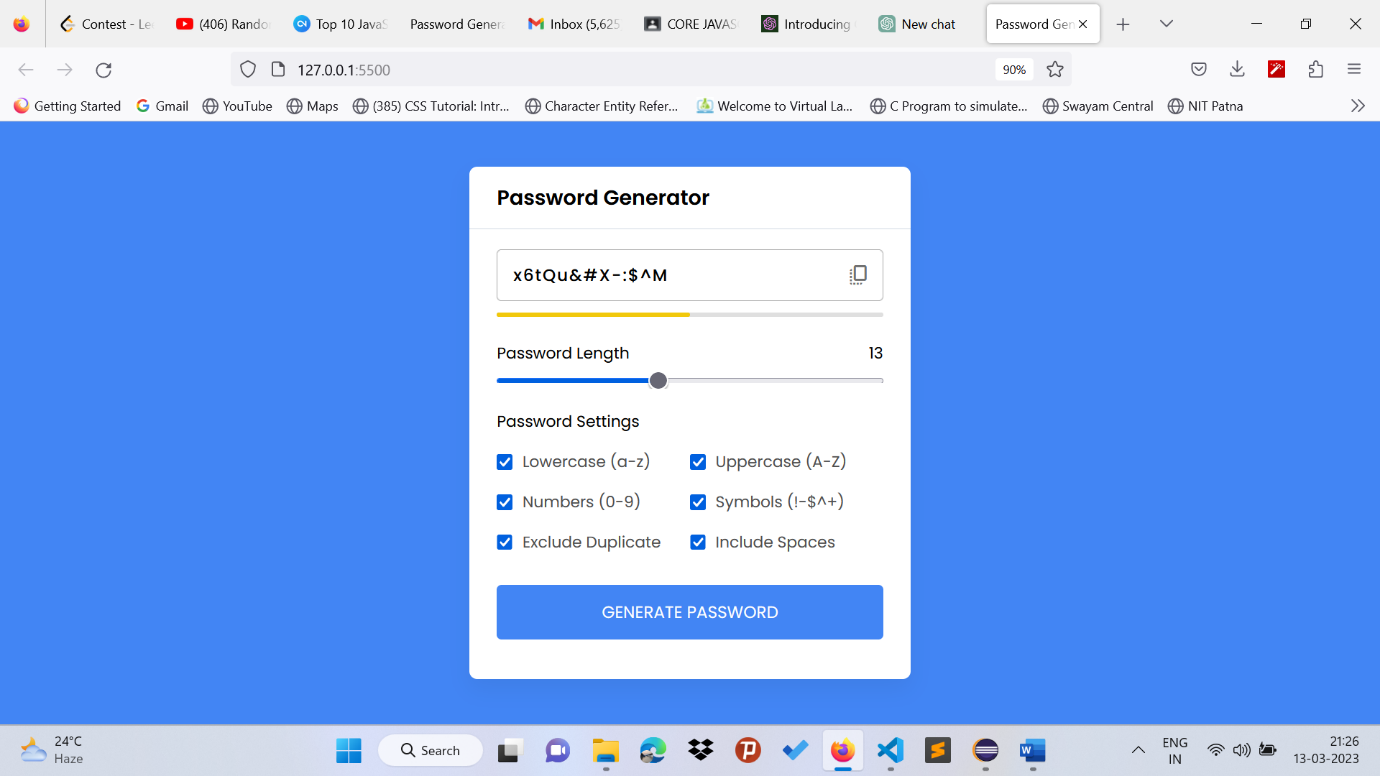
**RESULT AND DISCUSSION**

**Project-link:** [Tap here](https://imsks135.github.io/project-test/) (https://imsks135.github.io/project-test/)

****

****

****

****

**CONCLUSION**

In conclusion, a random password generator using JavaScript is an essential tool for enhancing cybersecurity. It generates complex, unique passwords that are difficult for hackers to crack, thus protecting user data from unauthorized access. Developing a random password generator using JavaScript is a relatively simple task that requires minimal hardware and software requirements. The generator can be integrated into various web and mobile applications, cloud-based services, IoT devices, government and financial institutions, and educational settings. The future scope of a random password generator using JavaScript is significant due to the growing need for cybersecurity in various fields. By implementing a random password generator using JavaScript, users can protect themselves against data breaches and ensure that their personal information remains secure.

**FUTURE SCOPE**

The future scope of a random password generator using JavaScript is significant due to the increasing importance of cybersecurity. Here are some potential areas where a random password generator using JavaScript can be applied:

1. Web and mobile applications: With the growing number of web and mobile applications, users are required to create multiple accounts, and using a weak password can lead to data breaches. Integrating a random password generator in such applications can enhance cybersecurity and protect user data.
2. Cloud-based services: Cloud-based services such as Google Drive, Dropbox, and Microsoft OneDrive store large amounts of sensitive data. Integrating a random password generator in such services can ensure that user data is protected from unauthorized access.
3. Internet of Things (IoT): As the use of IoT devices increases, securing these devices becomes crucial. Using a random password generator in IoT devices can ensure that they are protected from hacking attempts.
4. Government and financial institutions: Governments and financial institutions store sensitive data that can be vulnerable to cyber attacks. Integrating a random password generator can enhance security and protect against data breaches.
5. Education and training: As cybersecurity becomes increasingly important, educating and training individuals on the importance of creating strong passwords can be beneficial. A random password generator can be used as a tool to teach users how to create secure passwords.

**REFERENCES**

Here are some references and bibliography sources for a random password generator using JavaScript:

1. "JavaScript Random Password Generator" by David Walsh. <https://davidwalsh.name/javascript-password-generator>
2. "How to Create a Random Password Generator with JavaScript" by Freecodecamp. <https://www.freecodecamp.org/news/how-to-create-a-random-password-generator-with-javascript/>
3. "Random Password Generator Using JavaScript" by GeeksforGeeks. <https://www.geeksforgeeks.org/random-password-generator-using-javascript/>
4. "Create a Random Password Generator with JavaScript" by W3Schools. <https://www.w3schools.com/howto/howto_js_password_generator.asp>
5. "JavaScript Password Generator" by Passwords Generator. <https://passwordsgenerator.net/javascript-password-generator/>
6. "Password Generator" by LastPass. <https://www.lastpass.com/password-generator>
7. "Password Generator Tool" by Norton. <https://my.norton.com/extspa/passwordmanager?path=pwd-gen>
8. "Password Generator"by Dashlane. <https://www.dashlane.com/features/password-generator>
9. "The Best Password Generators in 2022" by TechRadar. <https://www.techradar.com/best/best-password-generator>

These sources provide valuable information and insights on how to create a random password generator using JavaScript, as well as recommendations for other password generator tools. They can be used as references and bibliography sources for a project report on random password generators using JavaScript.