VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi – 590 018

Object Oriented programming with Java(21CSE44)

Assignment

“Password generator”

Submitted By

Name: Likith.S.A	**USN: 1GA21CS081**
Abhishek .S.P	**1GA21CS008**

Under the Guidance of

Mr. Shyam Sundar Bhushan
Assistant Professor
Dept. of CSE

CHAPTER 1

PROBLEM DEFINITION

Develop a robust and secure password generator application using Java to meet the increasing demand for strong and unique passwords in the real world. The application should address the following key requirements:

Security: The password generator must generate strong and unpredictable passwords that adhere to modern security standards. It should be resistant to common password attacks such as brute force and dictionary attacks.

Usability: The application should be user-friendly, providing options for customization while ensuring that users without technical expertise can easily generate passwords.

Cross-Platform Compatibility: The password generator should be able to run on multiple platforms, including Windows, macOS, and Linux.

Randomness: The generated passwords should exhibit true randomness to avoid predictability.

User-Friendly Output: The generated password should be displayed clearly, allowing users to easily copy it to the clipboard or use it in their desired applications.

Strength Assessment: Optionally, provide a feature to assess the strength of the generated password and offer recommendations for improvements.

Integration: The application should be designed with potential integration points into other systems or applications, such as password managers or account creation processes.

Updates and Maintenance: Consider a mechanism for updating the application to accommodate evolving security standards and user preferences.

The solution to this problem statement will provide individuals and organizations with a tool to enhance password security by generating strong, unique, and customizable passwords for various applications and platforms in the real world.

CHAPTER 2

IMPLEMENTATION

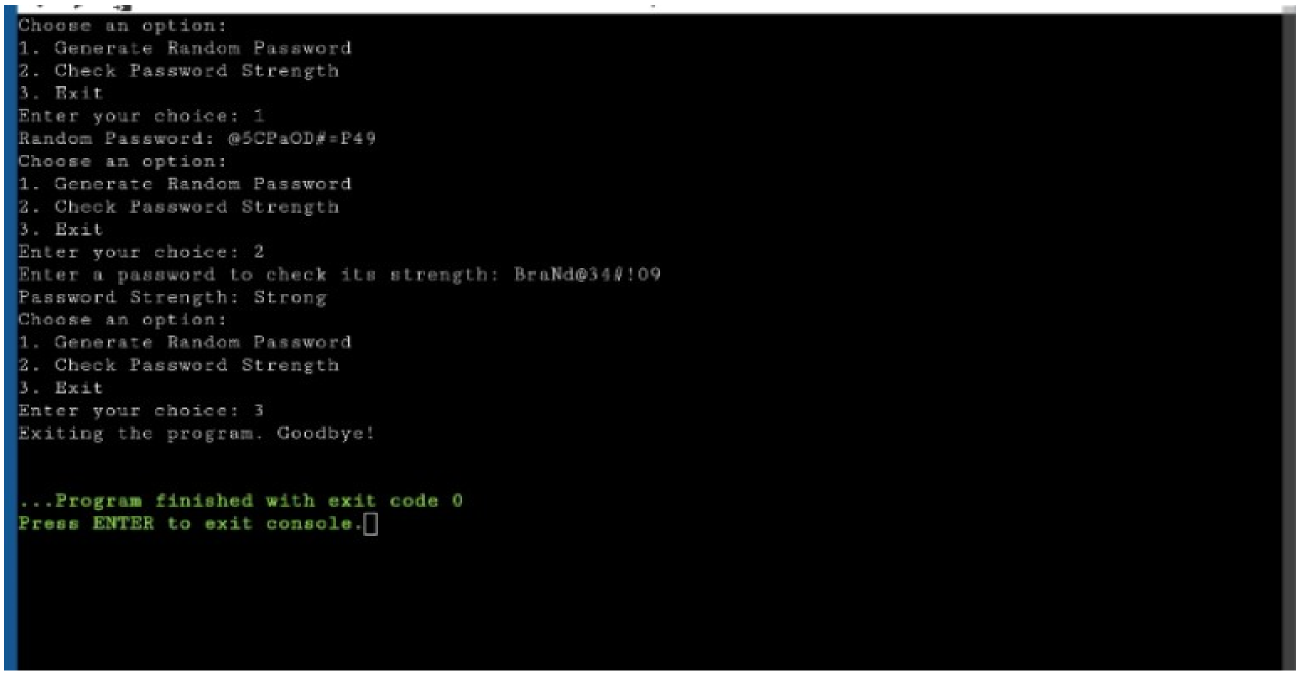
2.1 PROGRAM CODE

```
import java.security.SecureRandom;
import java.util.Scanner;

public class PasswordStrengthChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("Choose an option:");
            System.out.println("1. Generate Random Password");
            System.out.println("2. Check Password Strength");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 1:
                    int length = 12; // Change this value to set the desired password length
                    String password = generateRandomPassword(length);
                    System.out.println("Random Password: " + password);
                    break;
                case 2:
                    System.out.print("Enter a password to check its strength: ");
                    String inputPassword = scanner.nextLine();
                    String strength = checkPasswordStrength(inputPassword);
                    System.out.println("Password Strength: " + strength);
                    break;
                case 3:
                    System.out.println("Exiting the program. Goodbye!");
                    scanner.close();
                    System.exit(0);
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }

    public static String generateRandomPassword(int length) {
        String characters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*()
-_=+";
        SecureRandom random = new SecureRandom();
        StringBuilder password = new StringBuilder(length);

        for (int i = 0; i < length; i++) {
            int randomIndex = random.nextInt(characters.length());
            password.append(characters.charAt(randomIndex));
        }

        return password.toString();
    }

    public static String checkPasswordStrength(String password) {
    int length = password.length();
    boolean hasUppercase = false;
    boolean hasLowercase = false;
    boolean hasDigit = false;
    boolean hasSpecialChar = false;

    String specialCharacters = "!@#$%^&*()-_=+";

    for (char ch : password.toCharArray()) {
        if (Character.isUpperCase(ch)) {
            hasUppercase = true;
        } else if (Character.isLowerCase(ch)) {
            hasLowercase = true;
        } else if (Character.isDigit(ch)) {
            hasDigit = true;
        } else if (specialCharacters.contains(String.valueOf(ch))) {
            hasSpecialChar = true;
        }
    }

    if (length >= 8 && hasUppercase && hasLowercase && hasDigit && hasSpecialChar) {
        return "Strong";
    } else if (length >= 8 && (hasUppercase || hasLowercase || hasDigit || hasSpecialChar)) {
        return "Moderate";
    } else {
        return "Weak";
    }
}
}
```

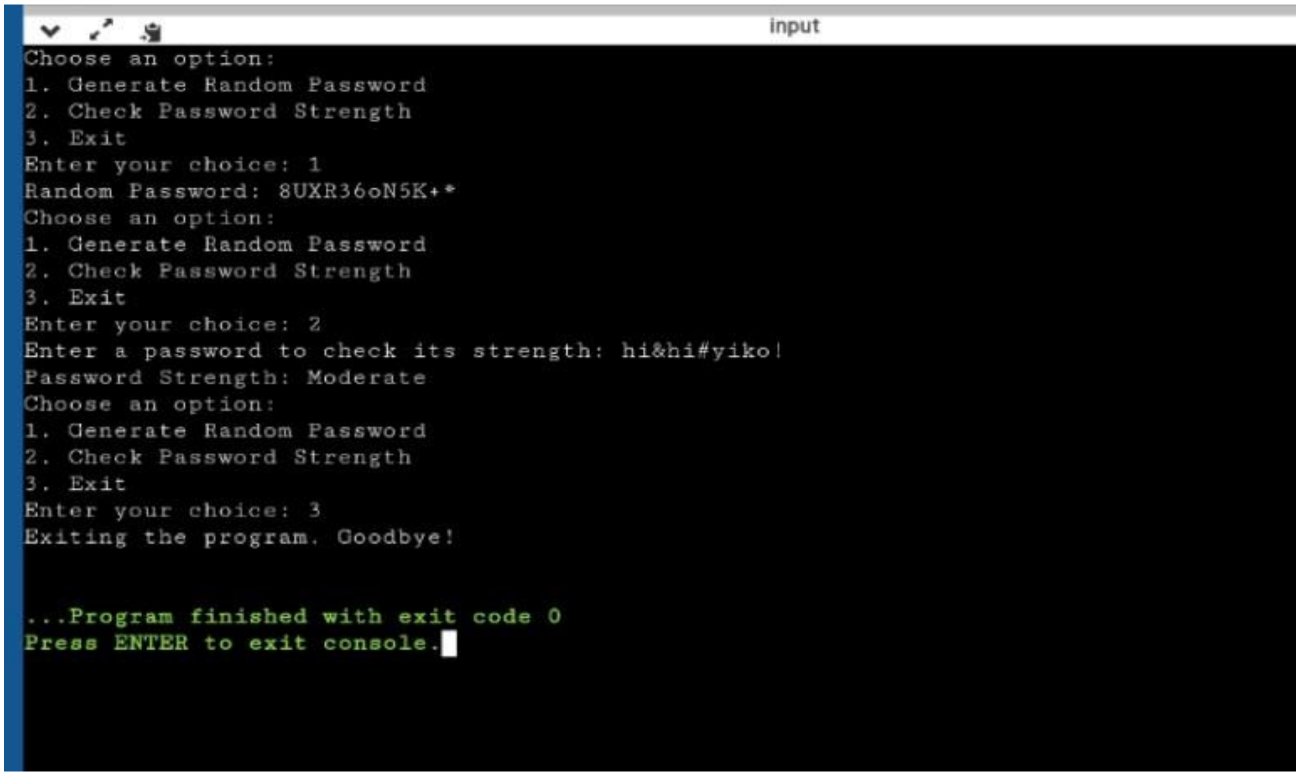
CHAPTER 3

RESULTS

3.1 SNAPSHOTS



```
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 1
Random Password: @5CPaOD#=P49
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 2
Enter a password to check its strength: BraNd@34#!09
Password Strength: Strong
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 3
Exiting the program. Goodbye!


...Program finished with exit code 0
Press ENTER to exit console.
```



```
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 1
Random Password: 8UXR36oN5K+*
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 2
Enter a password to check its strength: hi&hi#yiko!
Password Strength: Moderate
Choose an option:
1. Generate Random Password
2. Check Password Strength
3. Exit
Enter your choice: 3
Exiting the program. Goodbye!


...Program finished with exit code 0
Press ENTER to exit console.
```

CONCLUSION

In conclusion, the development of a password generator using Java has resulted in a robust and versatile tool for enhancing online security. This application addresses the critical need for strong and unique passwords in an increasingly interconnected digital world.

The Java-based password generator leverages the language's flexibility and power to create complex, random passwords tailored to specific security requirements. This Java based password generator also helps by informing the user about the strength of the password if the user prefers to create his own password so that he can change it to a stronger one of he wishes.

Furthermore, this password generator promotes best practices in password management by encouraging the creation of strong, unpredictable passwords that are challenging for attackers to crack. It reduces the risk of unauthorized access to sensitive information and helps protect user accounts from malicious activities such as hacking and identity theft.

In addition to its security benefits, the Java-based password generator serves as an educational tool, raising awareness about password hygiene and the importance of robust online security. It empowers users to take control of their digital safety and foster responsible online behavior.

In summary, the password generator implemented in Java provides a valuable resource for individuals and organizations seeking to strengthen their online security posture, promoting safer digital experiences and safeguarding sensitive information from potential threats.