Secure Password Management System using Java HashMap and SHA-256 Hashing

A PROJECT REPORT

Submitted by

M RAMYA SREE	**BL.EN.U4AIE21072**
M SIDDHARTHA	**BL.EN.U4AIE21080**
POLI VAMSI VARDHAN	**BL.EN.U4AIE21101**

for the course

21AIE203- Data Structures & Algorithms-2

Guided and Evaluated by

Ms. Aiswarya Milan

Dept. of CSE,

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDHYAPEETHAM

BANGALORE-560035

January 2023

ABSTRACT

This project is a password management system implemented in Java. It utilizes a HashMap to store user account information, including usernames, hashed passwords and security questions. The system also includes a password hashing function which uses the SHA-256 algorithm to ensure the security of the stored passwords. Users can create new accounts, log in to existing accounts, change their passwords and recover forgotten passwords through the use of security questions. The system also checks the strength of the entered password, if the password is weak, it prompts the user for password suggestions. The system is designed to be simple and easy to use, while providing a secure method for storing and managing passwords.

CONTENTS

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

The project provides a user-friendly interface for managing account information. When the program is run, users are presented with a menu of options including: creating a new account, logging into an existing account, changing the password for an existing account, and resetting a forgotten password.

When creating a new account, the program prompts the user to enter a username and a password. The password is then checked for strength using a checkPasswordStrength function. If the password is determined to be weak, the program suggests a password to the user. The suggested password is generated using the generatePassword function. If the user is satisfied with the suggested password, it is then hashed using the hashPassword function which implements the SHA-256 algorithm. The hashed password, along with the username, is then added to the accounts HashMap.

For logging in, the program prompts the user for their username and password. The entered password is hashed and then compared to the stored hashed password for the given username. If the passwords match, the user is granted access to their account.

For changing the password, the program prompts the user for their username and current password. The entered password is hashed and then compared to the stored hashed password for the given username. If the passwords match, the user is prompted to enter a new password. The new password is also checked for strength and hashed. The hashed password is then updated in the accounts HashMap.

For resetting a forgotten password, the program prompts the user for their username. The program then retrieves the security question and answer associated with the username from the accounts HashMap. The user is prompted to answer the security question, and if the answer is correct, the user is prompted to enter a new password. The new password is also checked for strength and hashed, and the hashed password is then updated in the accounts HashMap.

For this project we have used SHA-256 (Secure Hash Algorithm 256) is a cryptographic hash function that takes an input (or 'message') and returns a fixed-size string of characters, which is a 'digest' that is unique to the unique input. The same input will always produce the same output, but even a small change to the input will produce a very different output. It is one-way function, it is practically infeasible to generate the original input (or 'message') by knowing the digest. It is widely used in various cryptographic operations and is a part of the SHA-2 family (SHA-224, SHA-256, SHA-384, and SHA-512) of hash functions.

This project also includes a mechanism for preventing brute-force attacks by locking accounts after a certain number of failed login attempts. This is a mechanism for preventing brute-force attacks by locking accounts after a certain number of failed login attempts.

Overall, this project provides a simple, secure and easy to use the password management system, which can be useful for various applications and projects.

CHAPTER-2

LITERATURE SURVEY

In[1] In this paper presented a comprehensive review of the papers on deep hashing, including deep supervised hashing, deep unsupervised hashing and other related topics. Based on how measuring the similarities of hash codes, we divide deep supervised hashing methods into four categories: pairwise methods, ranking-based methods, pointwise methods and quantization. In addition, it categorized deep unsupervised hashing into three classes based on semantics learning manners, i.e., reconstruction-based methods, pseudo-label-based methods and prediction-free self-supervised learning-based methods. We also explore three important topics including semisupervised deep hashing, domain adaption deep hashing and multi-modal deep hashing.

In[2] this paper, The significance of the hardware implementation of the MD5 algorithm has been examined. Two architectures have been studied for both area utilization and speed with FPGAs as the target device. It is clear that both architectures can be easily fitted to a single device. Although the inherent nature of the MD5 structure does not allow parallel hash operations of blocks, hardware implementations can obtain a significant throughput to cater to some of currently available IP bandwidths. FPGA implementations would therefore be suitable as components in cryptographic accelerators.

[3] Secure password storing is essential in systems working based on password authentication. In this paper, SXR algorithm (Split, Exclusive OR, and Replace) was proposed to improve secure password storing and could also be applied to current authentication systems. SXR algorithm consisted of four steps. First, the received password from users was hashed through a general hash function. Second, the ratio and the number of iterations from the secret key (username and password) were calculated. Third, the hashed password and ratio were computed, and the hashed password was divided based on the ratio (Split) into two values. Both the values were applied to XOR equation according to the number of iterations, resulting in two new values. Last, the obtained values were concatenated and stored in the database (Replace). On evaluating, complexity analyses and comparisons has shown that SXR algorithm could provide attack resistance with a stronger hashed password against the aforementioned attacks.

[4] This survey presented details about the Secure Hash Algorithms and their hardware implementations using the FPGAs. Moreover, a comparison of three hash standards (SHA-1, SHA-2, and SHA-3) was presented. Optimization methods provide fair comparisons between various FPGA implementations of hash standards. CSA, pipelining, and Loop Unrolling are used to exploit the FPGA implementations of the secure hash Algorithms. Moreover, FPGA resources are used to mitigate the FPGA optimization techniques.

CHAPTER-3

SYSTEM MODEL

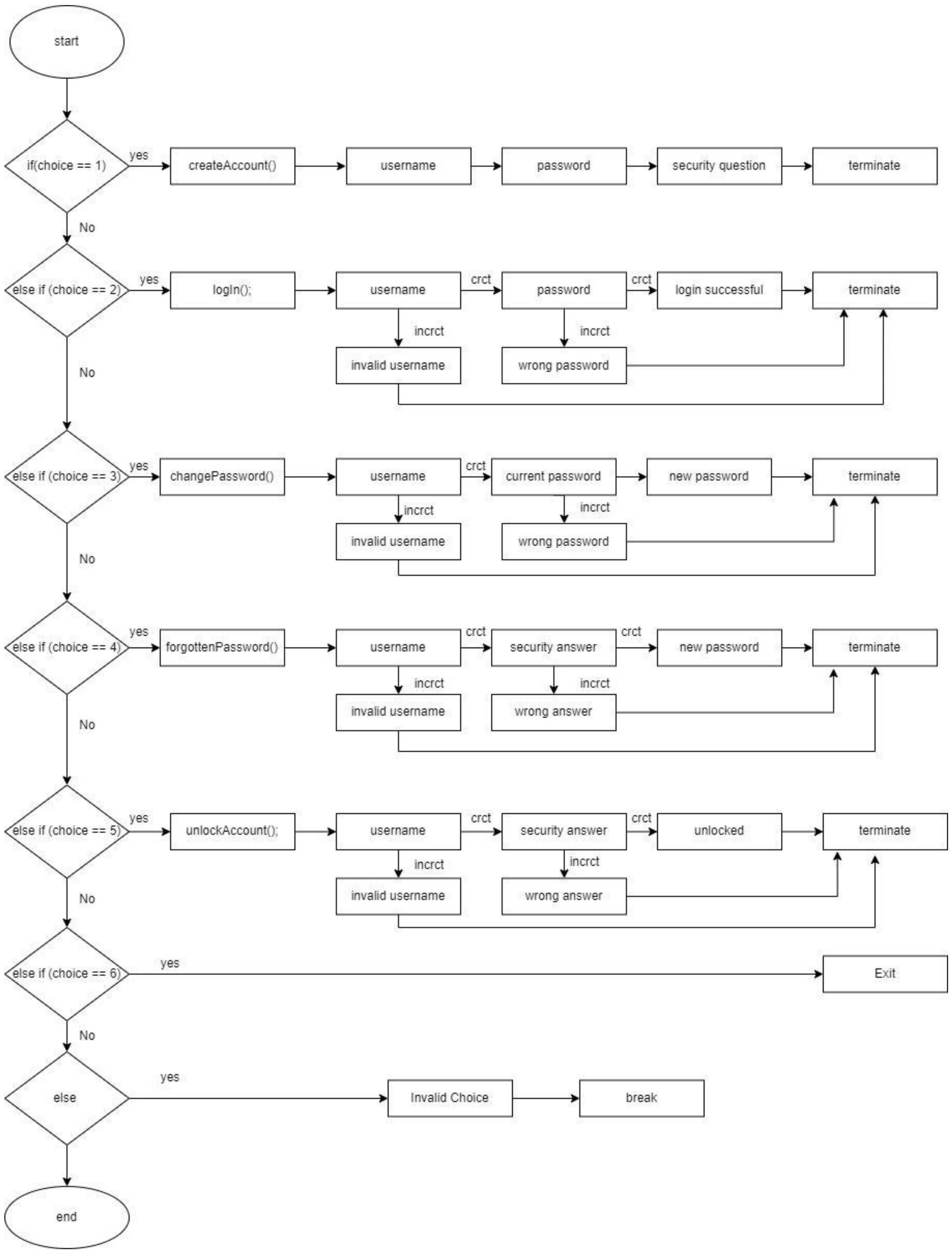
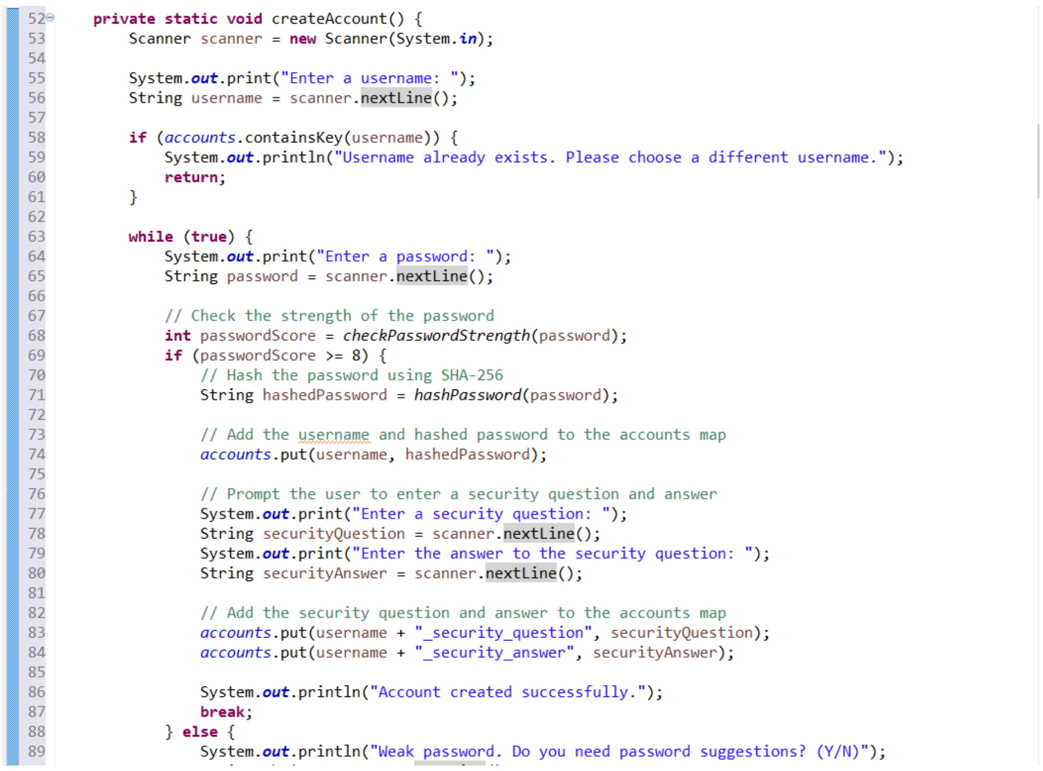


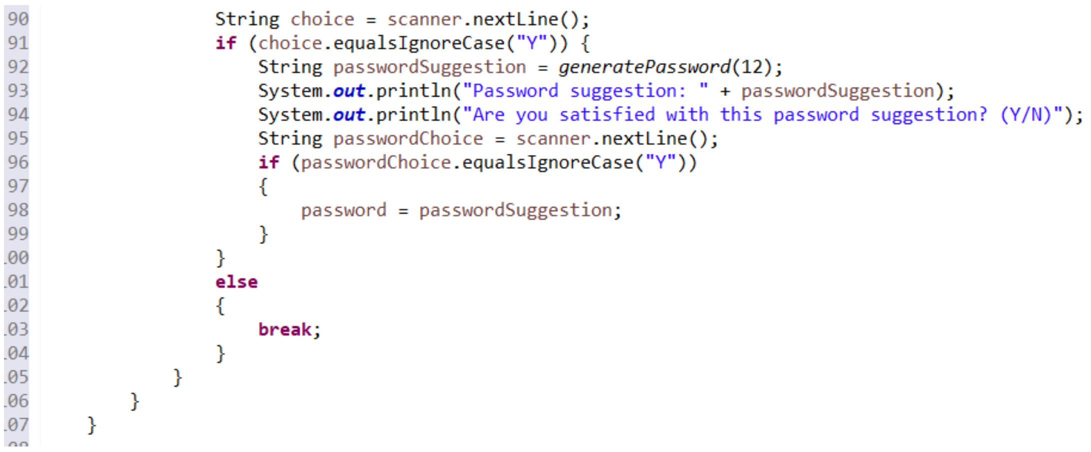
Fig.3

CHAPTER-4

IMPLEMENTATION



```java
private static void createAccount() {
    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter a username: ");
    String username = scanner.nextLine();

    if (accounts.containsKey(username)) {
        System.out.println("Username already exists. Please choose a different username.");
        return;
    }

    while (true) {
        System.out.print("Enter a password: ");
        String password = scanner.nextLine();

        // Check the strength of the password
        int passwordScore = checkPasswordStrength(password);
        if (passwordScore >= 8) {
            // Hash the password using SHA-256
            String hashedPassword = hashPassword(password);

            // Add the username and hashed password to the accounts map
            accounts.put(username, hashedPassword);

            // Prompt the user to enter a security question and answer
            System.out.print("Enter a security question: ");
            String securityQuestion = scanner.nextLine();
            System.out.print("Enter the answer to the security question: ");
            String securityAnswer = scanner.nextLine();

            // Add the security question and answer to the accounts map
            accounts.put(username + "_security_question", securityQuestion);
            accounts.put(username + "_security_answer", securityAnswer);

            System.out.println("Account created successfully.");
            break;
        } else {
            System.out.println("Weak password. Do you need password suggestions? (Y/N)");
```



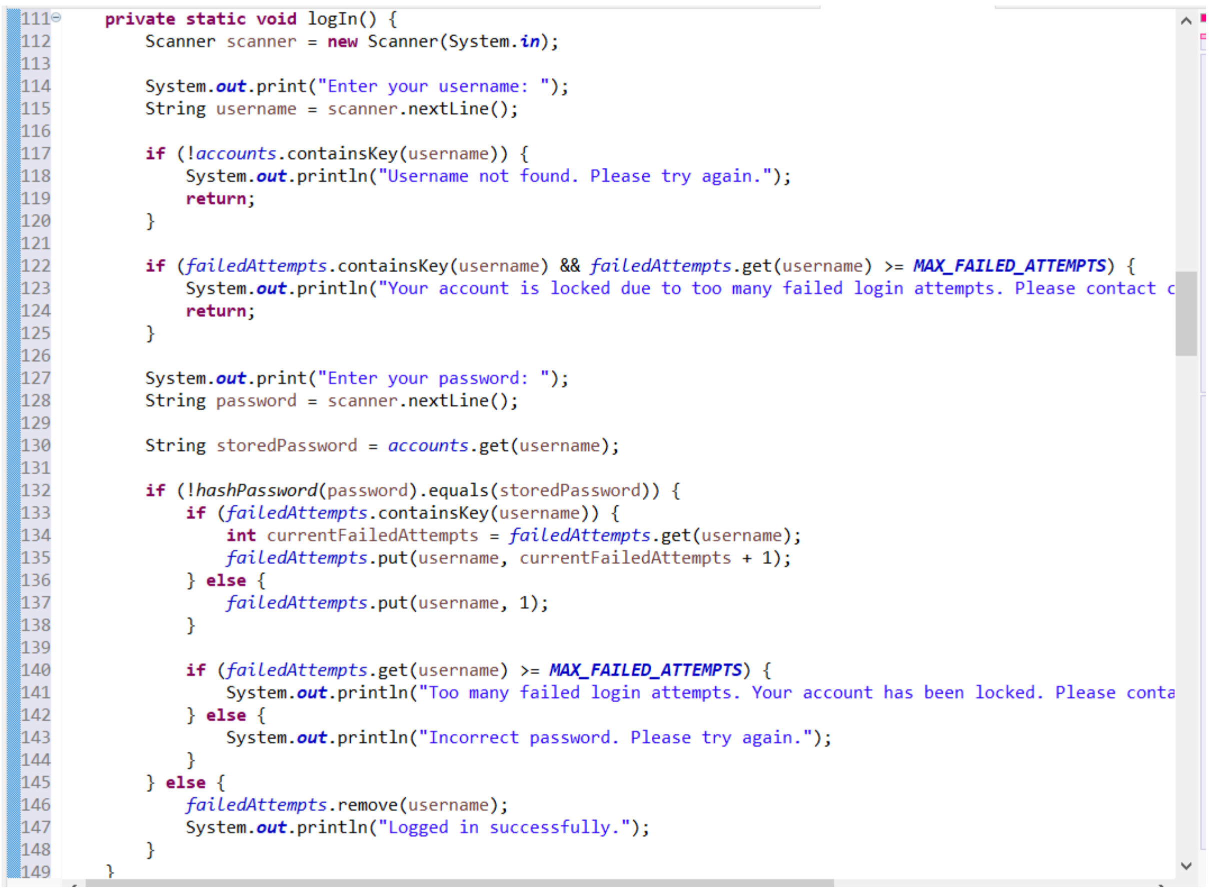
```java
            String choice = scanner.nextLine();
            if (choice.equalsIgnoreCase("Y")) {
                String passwordSuggestion = generatePassword(12);
                System.out.println("Password suggestion: " + passwordSuggestion);
                System.out.println("Are you satisfied with this password suggestion? (Y/N)");
                String passwordChoice = scanner.nextLine();
                if (passwordChoice.equalsIgnoreCase("Y"))
                {
                    password = passwordSuggestion;
                }
            }
            else
            {
                break;
            }
        }
    }
}
```

This code snippet is a method for creating a new account. The method first prompts the user to enter a username and checks if the username already exists in the system. If the username already exists, the user is prompted to choose a different one. If the username is unique, the user is prompted to enter a password.

The code then checks the strength of the password entered by the user by calling a method called "checkPasswordStrength" and passing in the entered password as an argument. The method returns a score which is compared to 8. If the score is greater than or equal to 8, the password is considered strong and the code proceeds to hash the password using SHA-256 by calling the method "hashPassword" and passing in the entered password as an argument. The hashed password is then added to the accounts map along with the entered username.

Then the user is prompted to enter a security question and answer. These are also added to the accounts map along with a unique key for each.

Finally, the user is informed that the account has been created successfully. If the password score is less than 8, the user is prompted to decide if they want to receive password suggestions, if yes the code will generate a password suggestion and check if the user is satisfied with the suggestion if yes the suggestion will be used as password.



```java
    private static void logIn() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your username: ");
        String username = scanner.nextLine();

        if (!accounts.containsKey(username)) {
            System.out.println("Username not found. Please try again.");
            return;
        }

        if (failedAttempts.containsKey(username) && failedAttempts.get(username) >= MAX_FAILED_ATTEMPTS) {
            System.out.println("Your account is locked due to too many failed login attempts. Please contact c
            return;
        }

        System.out.print("Enter your password: ");
        String password = scanner.nextLine();

        String storedPassword = accounts.get(username);

        if (!hashPassword(password).equals(storedPassword)) {
            if (failedAttempts.containsKey(username)) {
                int currentFailedAttempts = failedAttempts.get(username);
                failedAttempts.put(username, currentFailedAttempts + 1);
            } else {
                failedAttempts.put(username, 1);
            }

            if (failedAttempts.get(username) >= MAX_FAILED_ATTEMPTS) {
                System.out.println("Too many failed login attempts. Your account has been locked. Please conta
            } else {
                System.out.println("Incorrect password. Please try again.");
            }
        } else {
            failedAttempts.remove(username);
            System.out.println("Logged in successfully.");
        }
    }
```

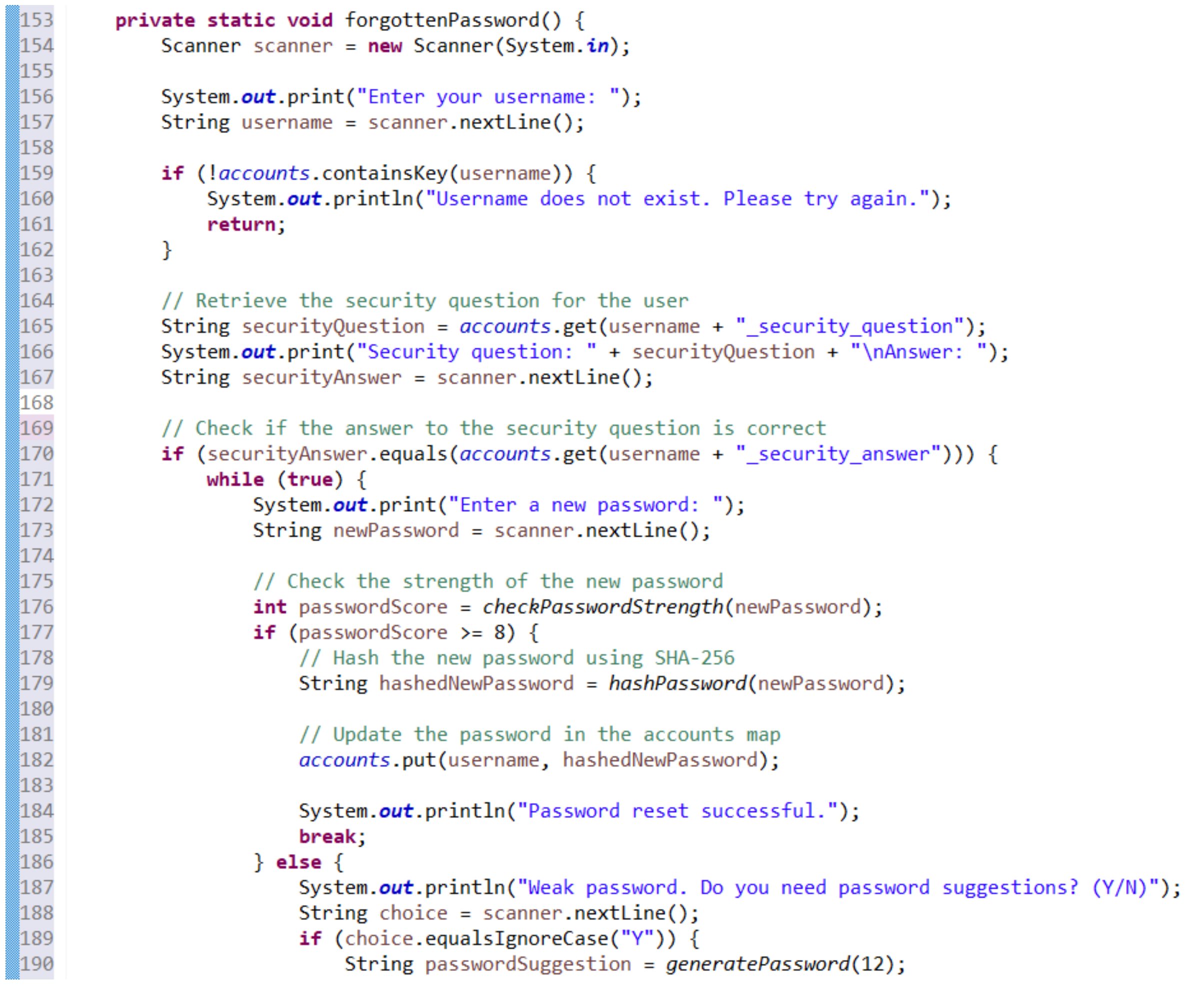
This code snippet is a method for logging into an existing account. The method first prompts the user to enter their username and checks if the username exists in the system. If the username is not found, the user is prompted to try again.

The code then checks if the user's account is locked, by checking if the username is in the failedAttempts map and if the number of failed attempts is greater than or equal to a pre-defined maximum number of attempts. If the account is locked, the user is prompted to contact customer service to unlock the account.

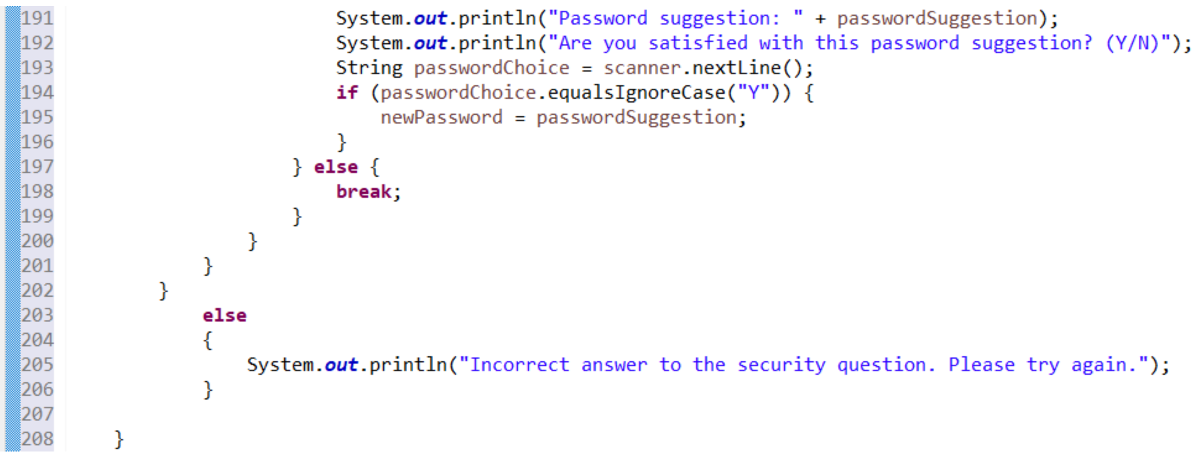
Then the user is prompted to enter their password. The code then retrieves the hashed version of the user's password from the accounts map and compares it to the hashed version of the entered password by calling the method "hashPassword" and passing in the entered password as an argument.

If the entered password does not match the stored password, the user is prompted that the password is incorrect and the code increments the number of failed attempts for this user in the failedAttempts map. If the number of failed attempts exceeds the maximum allowed, the user is prompted that their account has been locked and they should contact customer service to unlock their account.

Otherwise, the user is prompted that they have been logged in successfully and the failedAttempts entry for this user is removed from the map.



```java
    private static void forgottenPassword() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your username: ");
        String username = scanner.nextLine();

        if (!accounts.containsKey(username)) {
            System.out.println("Username does not exist. Please try again.");
            return;
        }

        // Retrieve the security question for the user
        String securityQuestion = accounts.get(username + "_security_question");
        System.out.print("Security question: " + securityQuestion + "\nAnswer: ");
        String securityAnswer = scanner.nextLine();

        // Check if the answer to the security question is correct
        if (securityAnswer.equals(accounts.get(username + "_security_answer"))) {
            while (true) {
                System.out.print("Enter a new password: ");
                String newPassword = scanner.nextLine();

                // Check the strength of the new password
                int passwordScore = checkPasswordStrength(newPassword);
                if (passwordScore >= 8) {
                    // Hash the new password using SHA-256
                    String hashedNewPassword = hashPassword(newPassword);

                    // Update the password in the accounts map
                    accounts.put(username, hashedNewPassword);

                    System.out.println("Password reset successful.");
                    break;
                } else {
                    System.out.println("Weak password. Do you need password suggestions? (Y/N)");
                    String choice = scanner.nextLine();
                    if (choice.equalsIgnoreCase("Y")) {
                        String passwordSuggestion = generatePassword(12);
```



```java
                        System.out.println("Password suggestion: " + passwordSuggestion);
                        System.out.println("Are you satisfied with this password suggestion? (Y/N)");
                        String passwordChoice = scanner.nextLine();
                        if (passwordChoice.equalsIgnoreCase("Y")) {
                            newPassword = passwordSuggestion;
                        }
                    } else {
                        break;
                    }
                }
            }
        }
            else
            {
                System.out.println("Incorrect answer to the security question. Please try again.");
            }

    }
```

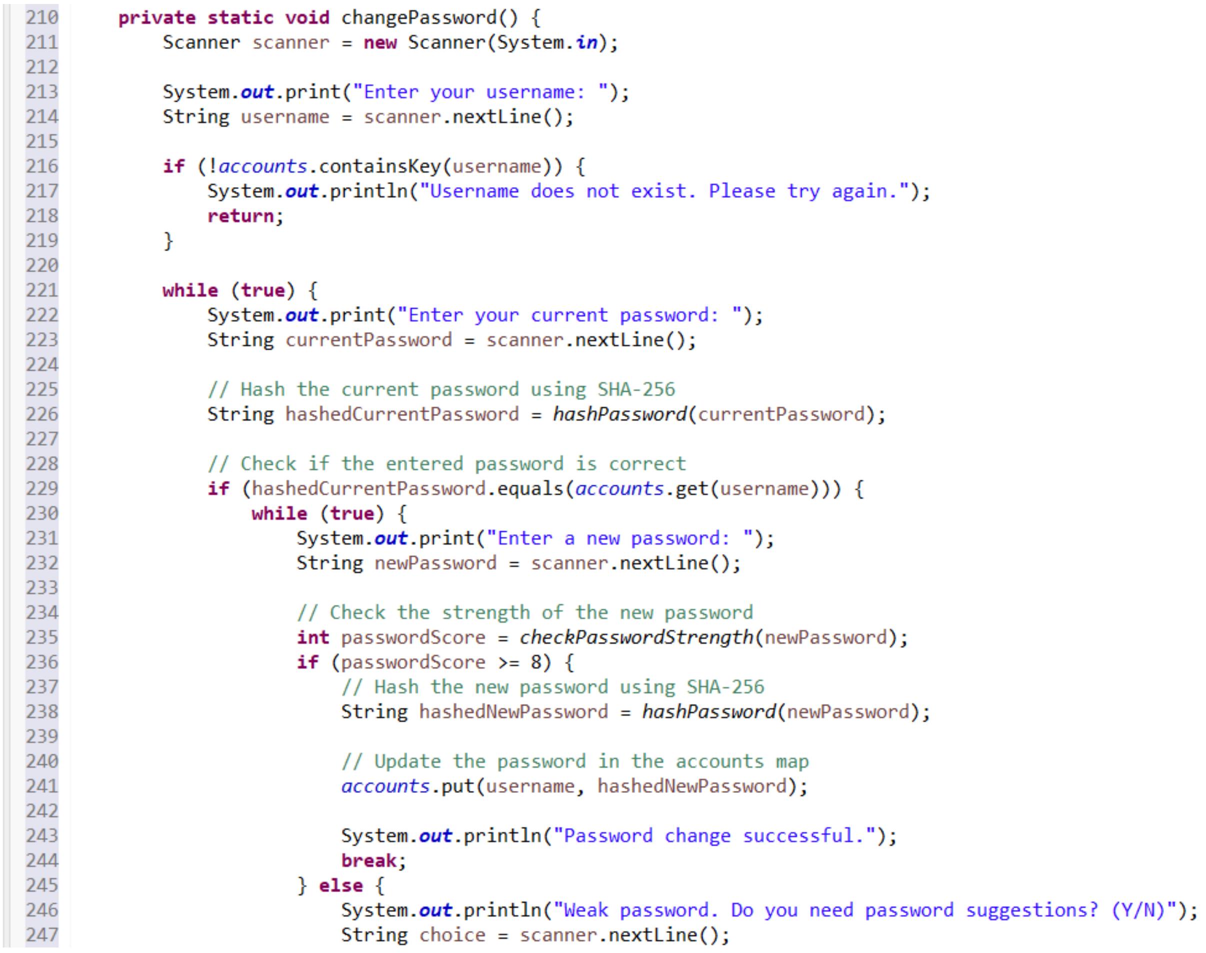
This code snippet is a method for resetting a forgotten password. The method first prompts the user to enter their username and checks if the username exists in the system. If the username does not exist, the user is prompted to try again.

Then the code retrieves the security question for the user from the accounts map and prompts the user to enter the answer. It then checks if the entered answer matches the stored answer and if it does, the user is prompted to enter a new password.

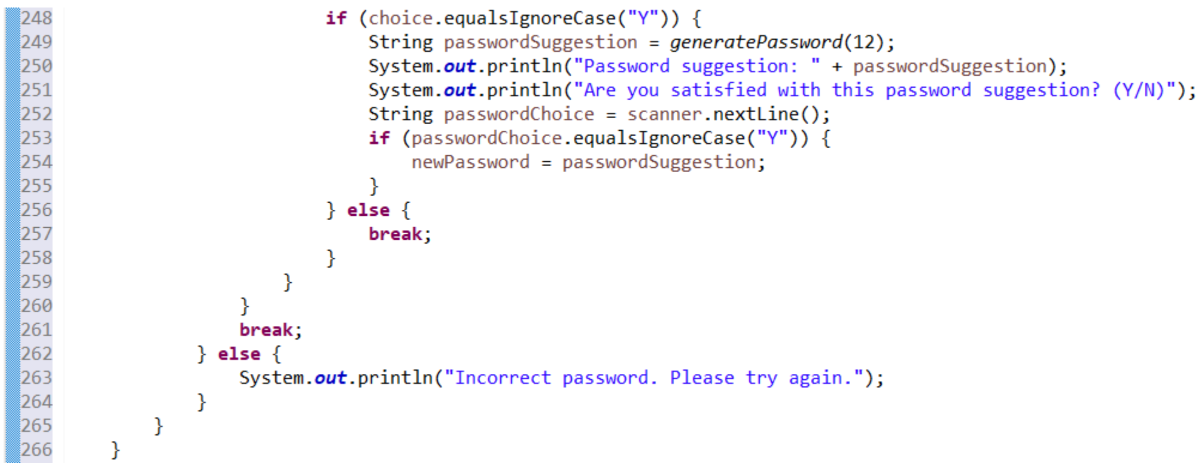
The code then checks the strength of the new password by calling a method called "checkPasswordStrength" and passing in the entered password as an argument. The method returns a score which is compared to 8. If the score is greater than or equal to 8, the password is considered strong and the code proceeds to hash the new password using SHA-256 by calling the method "hashPassword" and passing in the entered password as an argument. The hashed new password is then updated in the accounts map along with the entered username.

If the new password is considered weak, the user is prompted to decide if they want to receive password suggestions, if yes the code will generate a password suggestion and check if the user is satisfied with the suggestion if yes the suggestion will be used as password.

Finally, the user is informed that the password reset was successful. If the answer to the security question was incorrect, the user is prompted to try again.



```java
    private static void changePassword() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your username: ");
        String username = scanner.nextLine();

        if (!accounts.containsKey(username)) {
            System.out.println("Username does not exist. Please try again.");
            return;
        }

        while (true) {
            System.out.print("Enter your current password: ");
            String currentPassword = scanner.nextLine();

            // Hash the current password using SHA-256
            String hashedCurrentPassword = hashPassword(currentPassword);

            // Check if the entered password is correct
            if (hashedCurrentPassword.equals(accounts.get(username))) {
                while (true) {
                    System.out.print("Enter a new password: ");
                    String newPassword = scanner.nextLine();

                    // Check the strength of the new password
                    int passwordScore = checkPasswordStrength(newPassword);
                    if (passwordScore >= 8) {
                        // Hash the new password using SHA-256
                        String hashedNewPassword = hashPassword(newPassword);

                        // Update the password in the accounts map
                        accounts.put(username, hashedNewPassword);

                        System.out.println("Password change successful.");
                        break;
                    } else {
                        System.out.println("Weak password. Do you need password suggestions? (Y/N)");
                        String choice = scanner.nextLine();
```



```java
                    if (choice.equalsIgnoreCase("Y")) {
                        String passwordSuggestion = generatePassword(12);
                        System.out.println("Password suggestion: " + passwordSuggestion);
                        System.out.println("Are you satisfied with this password suggestion? (Y/N)");
                        String passwordChoice = scanner.nextLine();
                        if (passwordChoice.equalsIgnoreCase("Y")) {
                            newPassword = passwordSuggestion;
                        }
                    } else {
                        break;
                    }
                }
            }
            break;
        } else {
            System.out.println("Incorrect password. Please try again.");
        }
    }
}
```

This code snippet is a method for changing an existing password. The method first prompts the user to enter their username and checks if the username exists in the system. If the username does not exist, the user is prompted to try again.

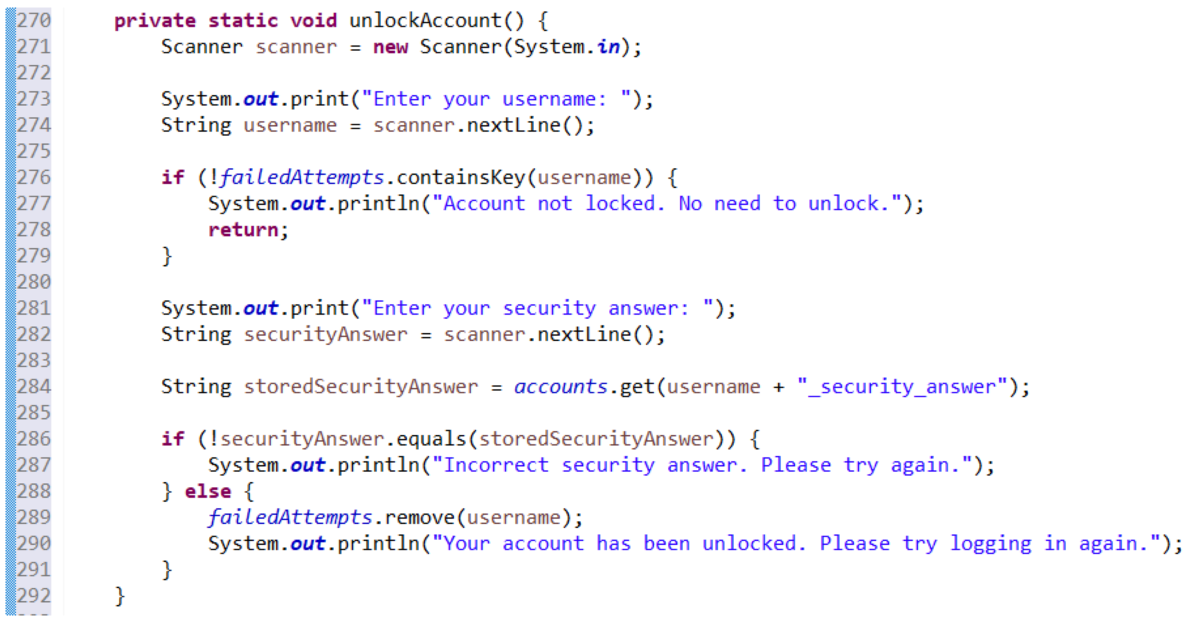
Then the user is prompted to enter their current password. The code then hashes the entered current password using SHA-256 by calling the method "hashPassword" and passing in the entered password as an argument.

The code then compares the hashed entered current password with the stored password from the accounts map, if it matches the user is prompted to enter a new password.

The code then checks the strength of the new password by calling a method called "checkPasswordStrength" and passing in the entered password as an argument. The method returns a score which is compared to 8. If the score is greater than or equal to 8, the password is considered strong and the code proceeds to hash the new password using SHA-256 by calling the method "hashPassword" and passing in the entered password as an argument. The hashed new password is then updated in the accounts map along with the entered username.

If the new password is considered weak, the user is prompted to decide if they want to receive password suggestions, if yes the code will generate a password suggestion and check if the user is satisfied with the suggestion if yes the suggestion will be used as password.

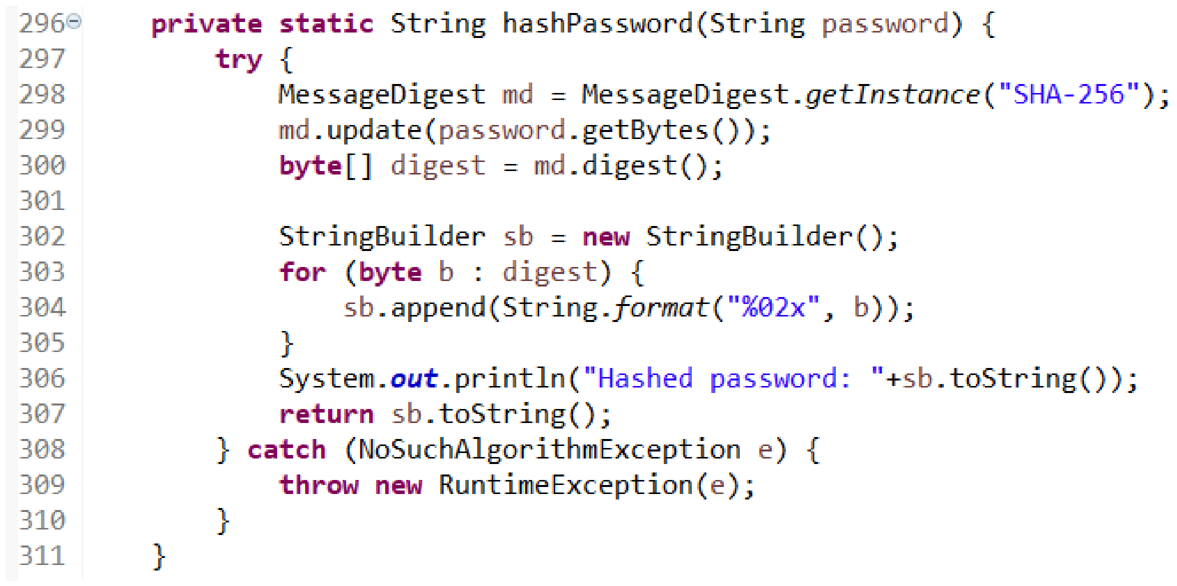
Finally, the user is informed that the password change was successful. If the entered current password is incorrect, the user is prompted to try again.



```java
    private static void unlockAccount() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your username: ");
        String username = scanner.nextLine();

        if (!failedAttempts.containsKey(username)) {
            System.out.println("Account not locked. No need to unlock.");
            return;
        }

        System.out.print("Enter your security answer: ");
        String securityAnswer = scanner.nextLine();

        String storedSecurityAnswer = accounts.get(username + "_security_answer");

        if (!securityAnswer.equals(storedSecurityAnswer)) {
            System.out.println("Incorrect security answer. Please try again.");
        } else {
            failedAttempts.remove(username);
            System.out.println("Your account has been unlocked. Please try logging in again.");
        }
    }
```

This code snippet is a method for unlocking an account that has been locked due to too many failed login attempts. The method first prompts the user to enter their username and checks if the username is in the failedAttempts map. If the username is not found in the failedAttempts map, the user is informed that the account is not locked and there is no need to unlock it.

Then the user is prompted to enter the answer to their security question and the entered answer is compared to the stored answer. If the entered answer matches the stored answer, the failedAttempts entry for this user is removed from the map and the user is informed that their account has been unlocked and they should try logging in again. If the entered answer does not match the stored answer, the user is informed that their answer is incorrect and they should try again.



```java
    private static String hashPassword(String password) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(password.getBytes());
            byte[] digest = md.digest();

            StringBuilder sb = new StringBuilder();
            for (byte b : digest) {
                sb.append(String.format("%02x", b));
            }
            System.out.println("Hashed password: "+sb.toString());
            return sb.toString();
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
```

This code snippet is a method called "hashPassword" that takes in a plain-text password as a parameter and returns the hashed version of the password as a string.

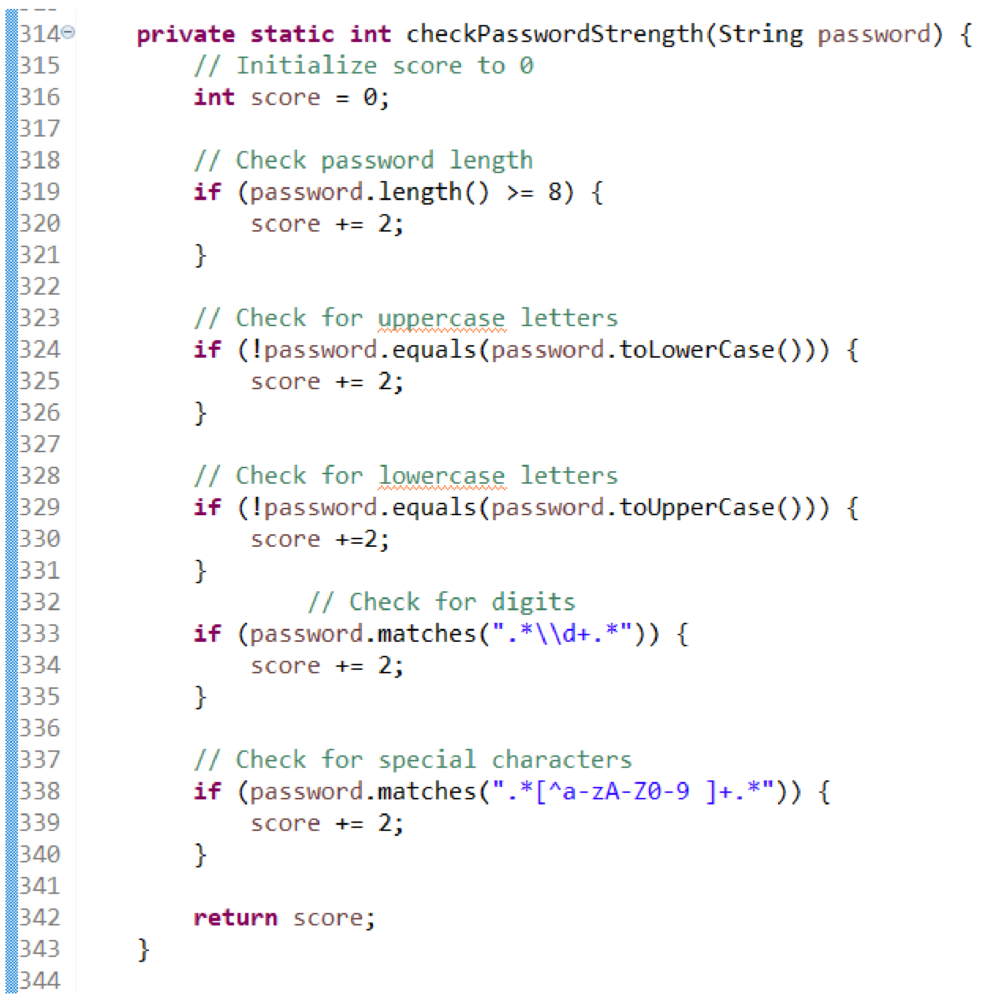
The method uses the SHA-256 hashing algorithm to generate the hash.

The method starts by creating an instance of the MessageDigest class with the algorithm "SHA-256", it then uses the update method of the MessageDigest class to pass in the bytes of the plain-text password.

The digest() method is then called on the MessageDigest object to generate the hash of the password, the resulting hash is stored in a byte array called "digest".

A StringBuilder object is then created and a for loop is used to iterate over the bytes of the digest. For each byte, the method formats it to a string of the format "%02x" using the String.format method and appends it to the StringBuilder object.

The toString() method is then called on the StringBuilder object to convert it to a string and that string is returned as the hashed password.



```java
    private static int checkPasswordStrength(String password) {
        // Initialize score to 0
        int score = 0;

        // Check password length
        if (password.length() >= 8) {
            score += 2;
        }

        // Check for uppercase letters
        if (!password.equals(password.toLowerCase())) {
            score += 2;
        }

        // Check for lowercase letters
        if (!password.equals(password.toUpperCase())) {
            score +=2;
        }
                // Check for digits
        if (password.matches(".*\\d+.*")) {
            score += 2;
        }

        // Check for special characters
        if (password.matches(".*[^a-zA-Z0-9 ]+.*")) {
            score += 2;
        }

        return score;
    }

```

This code snippet is a method called "checkPasswordStrength" that takes in a plain-text password as a parameter and returns an integer value representing the strength of the password.

The method starts by initializing a score variable to zero. It then uses several if statements to check various characteristics of the password and adds points to the score based on these characteristics.

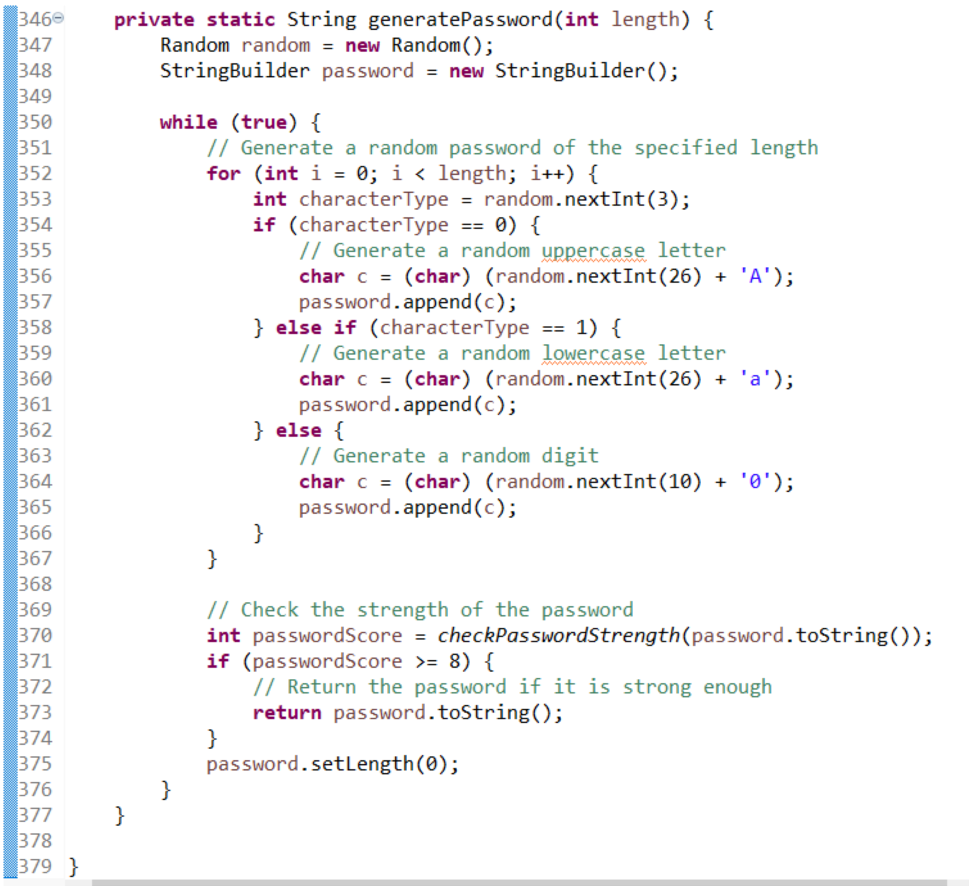
First, the method checks the length of the password. If the password is at least 8 characters long, the score is incremented by 2.

Then it check whether the password contains at least one uppercase letter and one lowercase letter by using the equals method of the String class, if it's true the score is incremented by 2.

Then it checks if the password contains digits by using the matches method of the String class and a regular expression, if it's true the score is incremented by 2.

Lastly, it checks if the password contains special characters by using the matches method of the String class and a regular expression, if it's true the score is incremented by 2.

At the end of the method, the score variable is returned as the strength of the password.



```
    private static String generatePassword(int length) {
        Random random = new Random();
        StringBuilder password = new StringBuilder();

        while (true) {
            // Generate a random password of the specified length
            for (int i = 0; i < length; i++) {
                int characterType = random.nextInt(3);
                if (characterType == 0) {
                    // Generate a random uppercase letter
                    char c = (char) (random.nextInt(26) + 'A');
                    password.append(c);
                } else if (characterType == 1) {
                    // Generate a random lowercase letter
                    char c = (char) (random.nextInt(26) + 'a');
                    password.append(c);
                } else {
                    // Generate a random digit
                    char c = (char) (random.nextInt(10) + '0');
                    password.append(c);
                }
            }

            // Check the strength of the password
            int passwordScore = checkPasswordStrength(password.toString());
            if (passwordScore >= 8) {
                // Return the password if it is strong enough
                return password.toString();
            }
            password.setLength(0);
        }
    }

}
```

This code snippet is a method called "generatePassword" that takes in an integer value representing the length of the password as a parameter and returns a randomly generated password as a string.

The method starts by creating a new instance of the Random class and a StringBuilder object called "password".

It then uses a while loop to repeatedly generate a new password until a strong password is found.

The while loop starts by using a for loop to generate a random password of the specified length.

For each character in the password, the method uses the nextInt method of the Random class to generate a random number between 0 and 2.

Based on the generated number, the method generates a random character of the corresponding type (uppercase letter, lowercase letter, or digit) and appends it to the StringBuilder object.

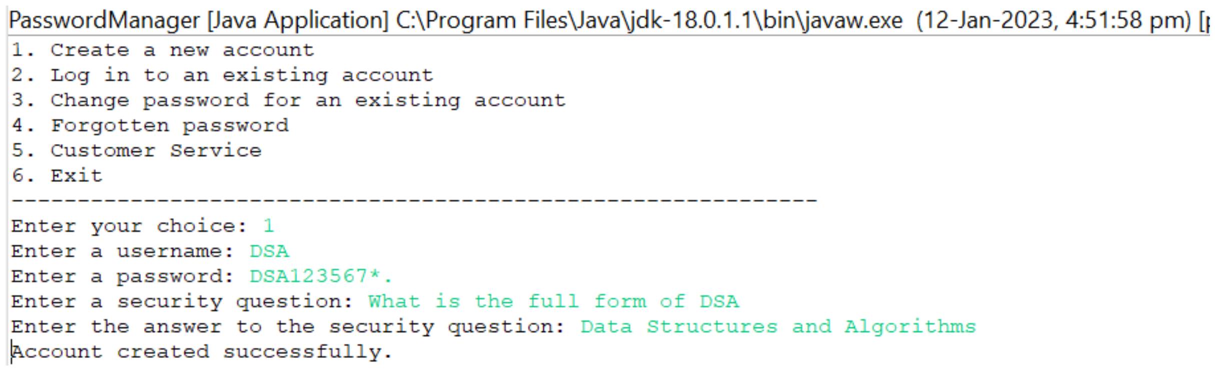
Once the for loop is finished, the method calls the "checkPasswordStrength" method to check the strength of the generated password and compares the score with 8.

If the score is greater than or equal to 8, the method returns the password. Otherwise, it sets the length of the StringBuilder object to zero and starts the while loop again to generate a new password

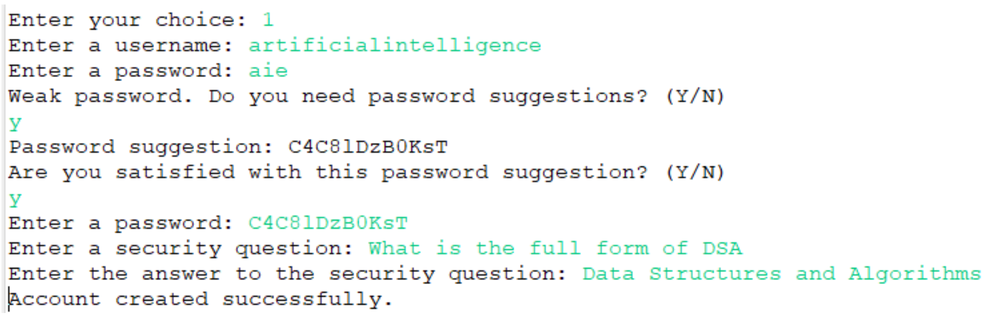
CHAPTER-5

RESULTS

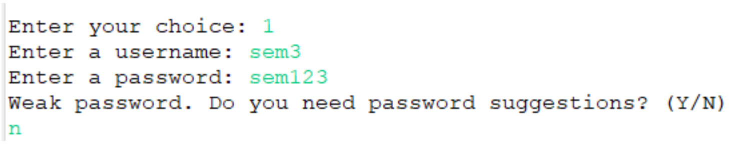
1. Creating account



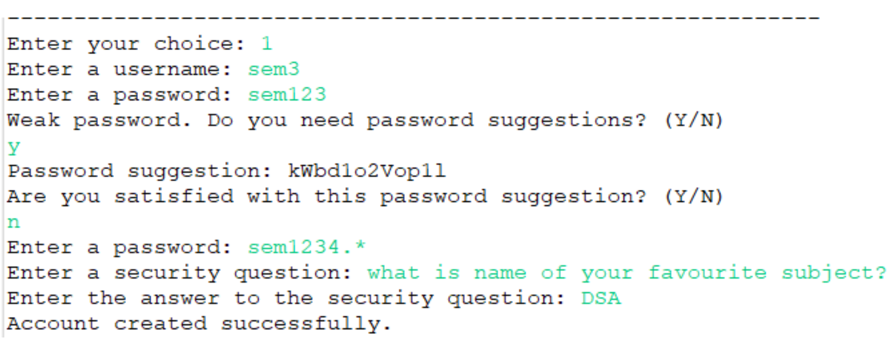
```
PasswordManager [Java Application] C:\Program Files\Java\jdk-18.0.1.1\bin\javaw.exe  (12-Jan-2023, 4:51:58 pm) [
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
-------------------------------------------------------------
Enter your choice: 1
Enter a username: DSA
Enter a password: DSA123567*.
Enter a security question: What is the full form of DSA
Enter the answer to the security question: Data Structures and Algorithms
Account created successfully.
```



```
Enter your choice: 1
Enter a username: artificialintelligence
Enter a password: aie
Weak password. Do you need password suggestions? (Y/N)
y
Password suggestion: C4C8lDzB0KsT
Are you satisfied with this password suggestion? (Y/N)
y
Enter a password: C4C8lDzB0KsT
Enter a security question: What is the full form of DSA
Enter the answer to the security question: Data Structures and Algorithms
Account created successfully.
```

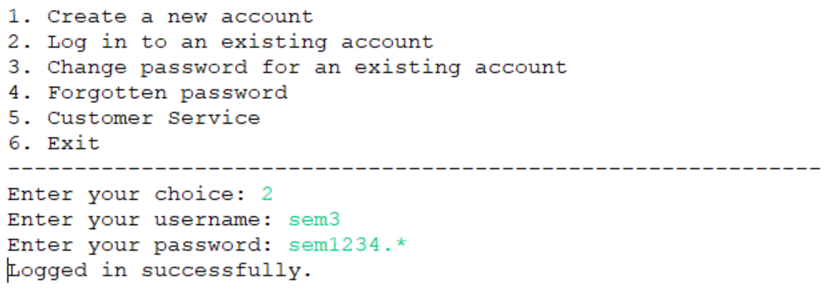


```
Enter your choice: 1
Enter a username: sem3
Enter a password: sem123
Weak password. Do you need password suggestions? (Y/N)
n
```

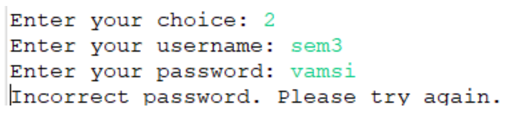


```
-------------------------------------------------------------
Enter your choice: 1
Enter a username: sem3
Enter a password: sem123
Weak password. Do you need password suggestions? (Y/N)
y
Password suggestion: kWbd1o2Vop1l
Are you satisfied with this password suggestion? (Y/N)
n
Enter a password: sem1234.*
Enter a security question: what is name of your favourite subject?
Enter the answer to the security question: DSA
Account created successfully.
```

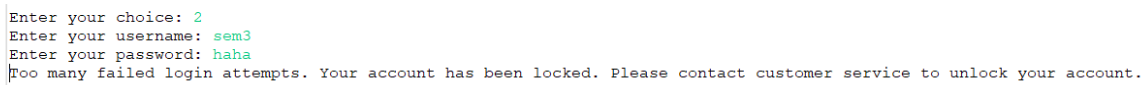
2. Logging in



```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
------------------------------------------------------------
Enter your choice: 2
Enter your username: sem3
Enter your password: sem1234.*
Logged in successfully.
```

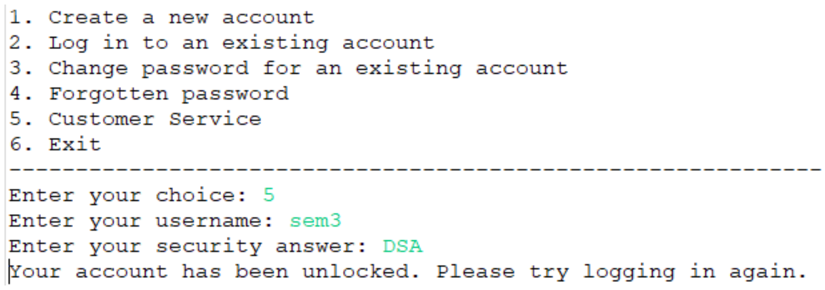


```
Enter your choice: 2
Enter your username: sem3
Enter your password: vamsi
Incorrect password. Please try again.
```



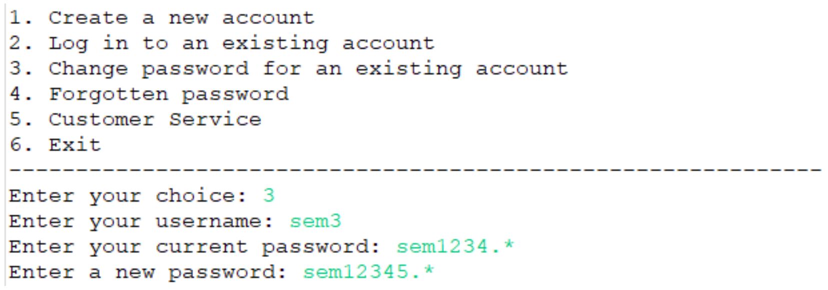
```
Enter your choice: 2
Enter your username: sem3
Enter your password: haha
Too many failed login attempts. Your account has been locked. Please contact customer service to unlock your account.
```

3. Contacting customer service



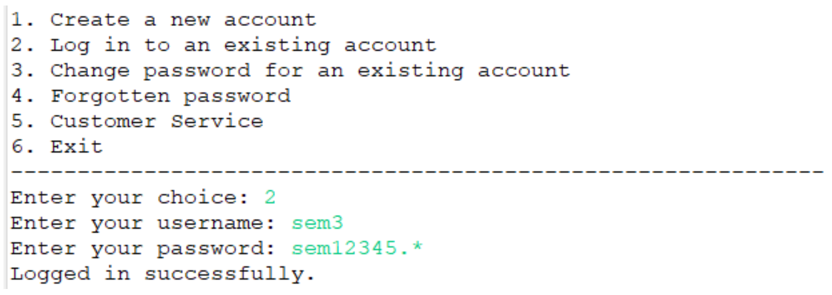
```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
------------------------------------------------------------
Enter your choice: 5
Enter your username: sem3
Enter your security answer: DSA
Your account has been unlocked. Please try logging in again.
```

4. Changing password



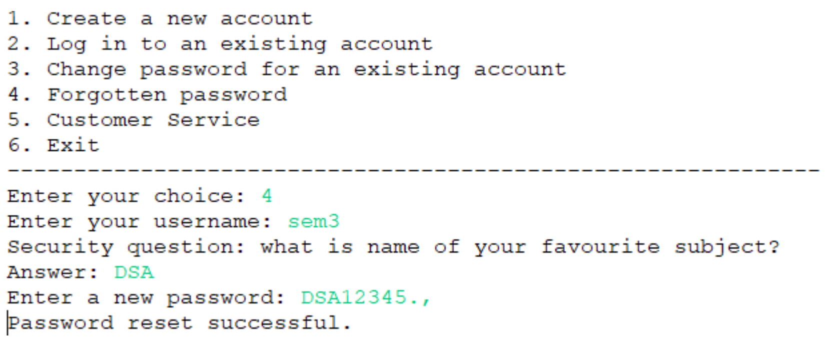
```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
------------------------------------------------------------
Enter your choice: 3
Enter your username: sem3
Enter your current password: sem1234.*
Enter a new password: sem12345.*
```

5. Logging in again after changing the password



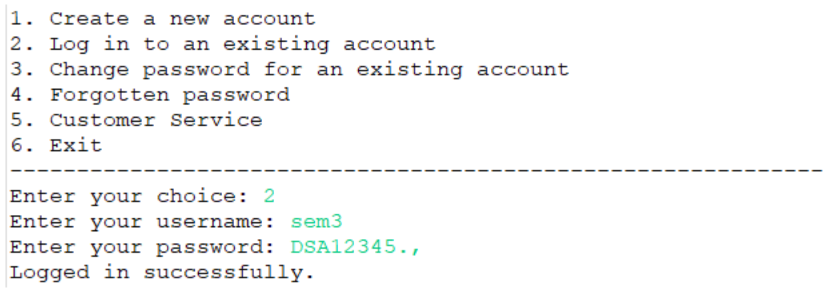
```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
-------------------------------------------------------------
Enter your choice: 2
Enter your username: sem3
Enter your password: sem12345.*
Logged in successfully.
```

6. Resetting when we forgot our password



```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
-------------------------------------------------------------
Enter your choice: 4
Enter your username: sem3
Security question: what is name of your favourite subject?
Answer: DSA
Enter a new password: DSA12345.,
Password reset successful.
```

7. Logging in again after resetting



```
1. Create a new account
2. Log in to an existing account
3. Change password for an existing account
4. Forgotten password
5. Customer Service
6. Exit
-------------------------------------------------------------
Enter your choice: 2
Enter your username: sem3
Enter your password: DSA12345.,
Logged in successfully.
```

CHAPTER-6

CONCLUSION

The above code implemented in this project is password manager program that provides several functionalities to help users manage their passwords securely. The program uses HashMap to store the username and hashed password pairs, and uses SHA-256 to hash the passwords for added security. The program also includes a feature to check the strength of the entered password and provide password suggestions if it is considered weak. Additionally, the program includes a feature that locks an account after a certain number of failed login attempts.

The use of a HashMap to store the accounts, along with the SHA-256 algorithm for hashing passwords, makes the program secure by preventing the passwords from being stored in plain text. Additionally, the program includes a feature for checking the strength of passwords and providing suggestions for stronger passwords, which makes it more difficult for attackers to gain unauthorized access to user accounts. The program also includes a mechanism for preventing brute-force attacks by locking accounts after a certain number of failed login attempts. This is a mechanism for preventing brute-force attacks by locking accounts after a certain number of failed login attempts.

Overall, it is a basic password manager that can be used to store and manage passwords securely.

CHAPTER-7

REFERENCES

[1] Xiao Luo, Haixin Wang, Daqing Wu,Chong Chen, Minghua Deng, Jianqiang Huang, Xian-Sheng Hua (2022). A Survey on Deep Hashing Methods.

https://dl.acm.org/doi/abs/10.1145/3532624

[2] J. Deepakumara, H.M. Heys, R. Venkatesan FPGA implementation of MD5 hash algorithm

https://ieeexplore.ieee.org/abstract/document/933564

[3] Jakkapong Polpong, Pongpisit Wuttidittachotti (2020). Authentication and password storing improvement using SXR algorithm with a hash function. International Journal of Electrical and Computer Engineering (IJECE) Vol. 10, No. 6, December 2020, pp. 6582~6591 ISSN: 2088-8708, DOI: 10.11591/ijece.v10i6.pp6582-6591.

https://pdfs.semanticscholar.org/27f6/b55b87028fa90bff40a0d0373122260b44df.pdf

[4] ZEYAD A. AL-ODAT, MAZHAR ALI , ASSAD ABBAS, SAMEE U. KHAN(2020). Secure Hash Algorithms and the Corresponding FPGA Optimization Techniques.

https://dl.acm.org/doi/pdf/10.1145/3311724