SZABIST

Course Name: CSC2102 - Data Structures and Algorithms
Course Instructor: Muhammad Talha Javaid

Course Name: CSCL2102 - Data Structures and Algorithms
Course Instructor: Syed Muhammad Hassan

PROJECT REPORT

Password Management System

Group Member	Registration Number
Mustan Ali	2112121
Umer Amir	2112241
Rohail Rathore	2012362

Table of Contents

1. Introduction & Problem Statement

The problem that the password management system aims to solve is the issue of password security and management. Many individuals and organizations struggle with creating and remembering strong passwords, and often end up using weak or easily guessable passwords. This can lead to security breaches and data loss, as weak passwords are more susceptible to being hacked or guessed.In addition, managing multiple passwords across various online accounts can be a time-consuming and confusing task. Users may forget their passwords, leading to frustration and inconvenience, or they may reuse the same password across multiple accounts, which can also increase the risk of a security breach. The password management system aims to solve these problems by providing a central location for storing and managing passwords, as well as tools for generating strong passwords. By using this system, individuals and organizations can improve their password security and reduce the risk of data loss or security breaches.

2. Features

- Generate Password
- Update Password
- Delete Password
- Saved Passwords
- Unsafe Passwords list
- Create Memo
- Delete Memo
- View Memos

3.0. Function Description

3.1. Password Generator

The generatePassword method takes an integer size as input and returns a random password of the given length. The password is created by selecting random characters from a string of characters that includes upper and lower case letters, numbers, and special characters. The password is generated by looping through a loop and selecting a random character from the string of characters for each iteration until the desired password length is reached. The Random class is used to generate random numbers to select the characters.

3.2. Password Management

addPassword

This function prompts the user to enter a service name and a username, and then either generates a password or prompts the user to enter a password. If the user chooses to generate a password, they are prompted to enter a password length and a password of that length is generated using the passwordGenerator object. If the user chooses to enter their own password, the password is checked for various criteria such as length, presence of special characters, numbers, and upper and lower case letters. The password is also checked to make

sure it is not the same as the service name or the username. If the entered password meets all the criteria, it is accepted, otherwise the user is prompted to enter a new password. The password strength is also displayed to the user using unicode characters. After the password is generated or entered, a new NodePassword object is created and initialized with the service name, username, and password. The new node is then added to the list of passwords.

updatePassword

This code is a method in a Java program that updates the password for a given service name and username. The method prompts the user to enter the service name and username, and then searches the list of passwords for a match. If a match is found, the user is given the option to either enter a new password manually or to generate a new password using the passwordGenerator object. If the user chooses to enter a new password manually, the password is accepted and assigned to the matching node in the list of passwords. If the user chooses to generate a new password, they are prompted to enter the length of the password and a new password of that length is generated using the passwordGenerator object. The new password is then assigned to the matching node in the list of passwords. If the service name and username are not found in the list of passwords, a message is displayed indicating that the data does not exist. Finally, the updated list of passwords is written to the file "passwordDB.txt".

deletePassword

The method prompts the user to enter the service name and username, and then searches the list of passwords for a match. If the service name and username match the first node in the list, the first node is removed by updating the first reference to point to the second node. If the service name and username match the last node in the list, the last node is removed by updating the next reference of the second to last node to null and the last reference to the second to last node. If the service name and username do not match the first or last node, the method searches the list for a match and removes the matching node by updating the next and previous references of the nodes before and after the matching node. Finally, the updated list of passwords is written to the file "passwordDB.txt". If the service name and username are not found in the list, the method does nothing and the list remains unchanged.

getAllPassoword

This method reads in a file called "passwordDB.txt" and parses the contents of the file to create a linked list of NodePassword objects. Each line of the file represents a single NodePassword object and is formatted as serviceName<->username<->password. The method splits each line by the <-> delimiter and assigns the resulting strings to the corresponding fields of a new NodePassword object. The new object is then added to the end of the linked list.

3.3. Memo Management

weakPassword
It looks like the weakPassword method is designed to print the service name and username for any password in the linked list that is found in a file called hackedPassword.txt.The method starts by looping through each element in the linked list and storing the password for that element in a variable called pass. It then initializes a boolean flag called flag to false. This flag will be used to track whether the password was found in the hackedPassword.txt file.Next, the method creates a Scanner object to read the hackedPassword.txt file, and then it reads each line of the file until it either reaches the end of the file or it finds a line that matches the password. If a matching line is found, the method sets the flag variable to true and breaks out of the loop.Finally, if the flag variable is true, the method prints the service name and username for the current element in the linked list. The method then moves on to the next element in the linked list and repeats the process until it has checked all the elements.

3.3. Memo Management

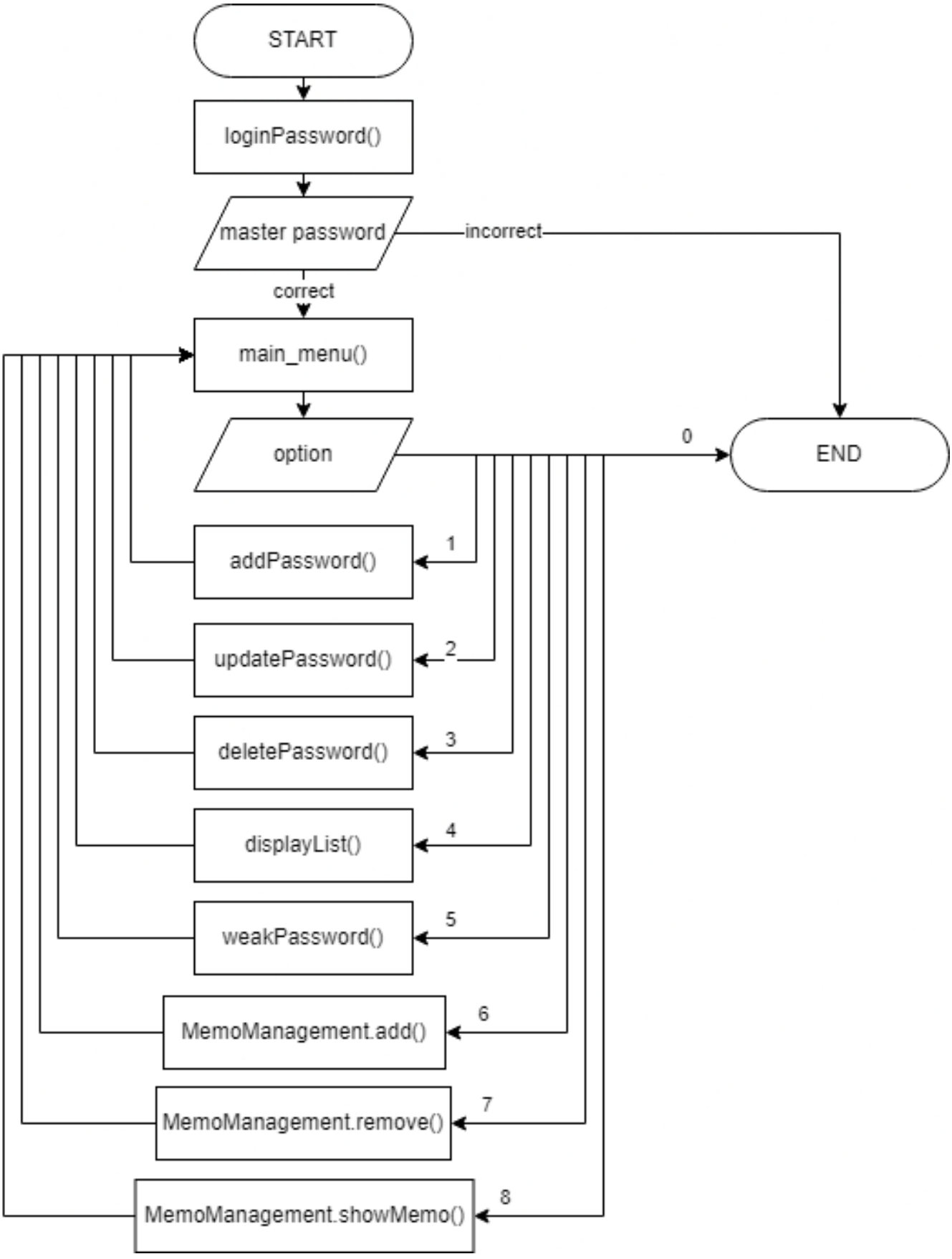
addMemo
This is a Java method that adds a new memo to a list of memos and writes it to a file called "Memo.txt". The method takes a single parameter called "memo", which is a string that represents the memo to be added.

getAllMemo
The getAllMemo() method retrieves all the memos from the text file and adds them to the Memos list. It does this by creating a Scanner object to read the text file, and then reads each line of the file and adds it to the Memos list.

updateMemo
The updateMemo() method updates the text file with the current state of the Memos list. It does this by creating a FileWriter object to write to the text file, and then iterating through the Memos list and writing each memo to the file. It appends a newline character to each memo before writing it to the file.

showMemo
The showMemo() method displays the memos in the Memos list to the user. It first checks if the Memos list is empty, and if it is, it prints a message indicating that there are no memos. If the Memos list is not empty, it iterates through the list and prints each memo, along with a counter that increments for each memo. The method uses System.out.printf() to format the output, with the counter being printed using a fixed width of 5 characters and the memo being printed using a fixed width of 10 characters.

4.0. Flowchart



5.0. Program Code

5.1. Main

```
import java.io.IOException;
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
       MemoManagement.initialize();
       print_header();
       loginPassword();
   }

   private static void loginPassword() {
       Scanner input = new Scanner(System.in);
       int tries = 0;

       while (tries < 3) {
           System.out.print("Enter master password: ");
           String password = input.nextLine();
           if (password.equals("admin")) {
               System.out.println("Login Successful");
               main_menu();
               break;
           } else {
               System.out.println("Login Failed");
               tries++;
               System.out.println("Remaining tries: " + (3 - tries));
           }
       }
       if (tries == 3) {
           System.out.println("You have exceeded the number of tries");
       }
       input.close();
   }

   private static void print_header() {
       System.out.println("*----------------------------------------*");
       System.out.println("|       Password Management System       |");
       System.out.println("*----------------------------------------*");
       System.out.println();
   }

   private static void main_menu() {
       System.out.println("--------> Main Menu <--------");
       System.out.println("[1]- Generate Password");
       System.out.println("[2]- Update Password");
       System.out.println("[3]- Delete Password");
       System.out.println("[4]- Saved Passwords");
       System.out.println("[5]- Unsafe Passwords list");
       System.out.println("[6]- Create Memo");
       System.out.println("[7]- Delete Memo");
       System.out.println("[8]- View Memos");
       System.out.println("[0]- Exit System");
       System.out.println("-----------------------------");
       String choice;
       PasswordManagement pm = new PasswordManagement();
       MemoManagement mm = new MemoManagement();
       pm.getAllPasswords();
```

```
Scanner Buffer = new Scanner(System.in);
Scanner input = new Scanner(System.in);
Scanner sc = new Scanner(System.in);
char c;
do {
    System.out.print("Choice: ");
    choice = input.nextLine();
    switch (choice) {
        case "0":
            System.exit(0);
            break;
        case "1":
            pm.addPassword();
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "2":
            pm.updatePassword();
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "3":
            pm.deletePassword();
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "4":
            pm.displayList();
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "5":
            pm.weakPassword();
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "6":
            System.out.println("Your Memo Start from Here: ");
            String memo = Buffer.nextLine();
            try {
                MemoManagement.add(memo);
            } catch (IOException ex) {
                System.out.println("ERROR: File not Found!");
            }
            System.out.println("\nPress any key to return\n");
            c = sc.next().charAt(0);
            main_menu();
            break;
        case "7":
            MemoManagement.showMemo();
            int index;
            do {
                System.out.print("Customer Index to remove : ");
                index = input.nextInt();
```

```
                    } while (index < 1 || index > MemoManagement.Memos.size());
                    MemoManagement.Memos.remove(MemoManagement.Memos.get(index - 1));

                    try {
                        MemoManagement.updateMemo();
                    } catch (IOException ex) {
                        System.out.println("ERROR: File not Found!");
                    }

                    System.out.println("Removed Successfully!\n");
                    System.out.println("\nPress any key to return\n");
                    c = sc.next().charAt(0);
                    main_menu();
                    break;
                case "8":
                    MemoManagement.showMemo();
                    System.out.println("\nPress any key to return\n");
                    c = sc.next().charAt(0);
                    main_menu();
                    break;
                default:
                    System.out.println("ERROR: Choice not valid");
            }
        } while (choice != "0" && choice != "1" && choice != "2" && choice != "3" &&
choice != "4" && choice != "5" && choice != "6" && choice != "7" && choice != "8");
    }
}
```

5.2. Password Generator

```
import java.util.Random;
public class PasswordGenerator {
    public String generatePassword(int size) {
        String chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&*()_+-/.,<>?;':[
]{}\"\\|`~";

        String password = "";
        Random r = new Random();

        while (password.length() < size) {
            int index = (int) (r.nextFloat() * chars.length());
            password += chars.charAt(index);
        }
        return password;
    }
}
```

5.3. NodePassword

```
public class NodePassword {
    public String serviceName;
    public String userName;
    public String password;
    public NodePassword next;
    public NodePassword previous;

    public void displayNode() {
        System.out.printf("| %-8s | %-8s | %-8s |\n", serviceName, userName, password);
    }
```

```
}
```

5.4. PasswordManagement

```
import java.io.*;
import java.util.Scanner;
import java.util.ArrayList;

public class PasswordManagement {
   Scanner sc = new Scanner(System.in);
   PasswordGenerator passwordGenerator = new PasswordGenerator();

   private NodePassword first;
   private NodePassword last;

   public PasswordManagement() {
       this.first = null;
       this.last = null;
   }

   public void addPassword() {
       System.out.println("==> ADD <==");

       System.out.println("Enter Service Name:");
       String ServiceName = sc.nextLine();

       System.out.println("Enter Username:");
       String username = sc.nextLine();

       String option = "";
       Scanner sc = new Scanner(System.in);

       do {
           System.out.println("Do you want to generate password? (y/n) ");
           option = sc.nextLine();

           if (!option.equalsIgnoreCase("Y") && !option.equalsIgnoreCase("N")) {
               System.out.println("Invalid choice");
           }
       } while (!option.equalsIgnoreCase("Y") && !option.equalsIgnoreCase("N"));


       String password = "";

       int length;
       if (option.equals("y") || option.equals("Y")) {
           do {
               System.out.println("Enter size:");
               length = sc.nextInt();

               if (length < 8) {
                   System.out.println("Minimum Password length = 8");
               }
           } while (length < 8);

           sc = new Scanner(System.in); // Clearing Input Buffer
           password = passwordGenerator.generatePassword(length);
```

```
        } else {
            Boolean weak;
            do {
                ArrayList<String> errorMessages = new ArrayList<>();
                int strength = 5;
                weak = false;
                System.out.println("Enter Password: ");
                password = sc.nextLine();
                if (password == ServiceName) {
                    errorMessages.add("Password cannot be same as the Service name");
                    weak = true;

                }

                if (password == username) {
                    errorMessages.add("Password cannot be same as your Username");
                    weak = true;

                }

                if (password.length() < 8) {
                    errorMessages.add("Password should contain atleast 8 characters");
                    weak = true;
                    strength--;
                }

                if (!password.matches(".*\\d.*")) {
                    errorMessages.add("Password must contain at least one number.");
                    weak = true;
                    strength--;
                }

                if
(!password.matches(".*[!@#$%^&*()_+\\-=\\[\\]{};':\"\\\\|,.<>\\/?].*")) {
                    errorMessages.add("Password must contain at least one special
character");
                    weak = true;
                    strength--;
                }

                if (!password.matches(".*[A-Z].*")) {
                    errorMessages.add("Password must contain at least one Uppercase
character");
                    weak = true;
                    strength--;
                }

                if (!password.matches(".*[a-z].*")) {
                    errorMessages.add("Password must contain atleast one Lowercase
character");
                    weak = true;
                    strength--;
                }
                System.out.println("Password Strength:" + strength + " out of 5");

                for (int i = 0; i < 5; i++) {
                    if (strength > 0) {
                        System.out.print('\u25A0');
                        strength--;
```

```
                } else {
                    System.out.print('\u25A1');
                }
            }

            System.out.println();

            if (weak) {
                System.out.println("Errors:");
                for (String errorMessage : errorMessages) {
                    System.out.println(errorMessage);
                }
            }
            System.out.println();

        } while (weak == true);
    }

    NodePassword newNode = new NodePassword();
    newNode.serviceName = ServiceName;
    newNode.userName = username;
    newNode.password = password;
    if (first == null) {
        first = newNode;
    } else {
        last.next = newNode;
        newNode.previous = last;
    }
    this.last = newNode;

    String data = ServiceName + "<->" + username + "<->" + password + "\n";

    File file = new File("passwordDB.txt");
    FileWriter writer = null;
    try {
        writer = new FileWriter(file, true);
        writer.write(data);
        writer.close();
    } catch (IOException e) {
        System.out.println("Error writing to file");
        throw new RuntimeException(e);
    }
}

public void updatePassword() {
    Scanner input = new Scanner(System.in);
    String sn, un;

    System.out.println("Enter Service Name:");
    sn = input.nextLine();

    System.out.println("Enter Username:");
    un = input.nextLine();

    sc = new Scanner(System.in); // Clearing Input Buffer

    int flag = 0;
    NodePassword current = first;
```

```
        while (current != null && flag == 0) {
            if (sn.equals(current.serviceName) && un.equals((current.userName))) {
                System.out.println("Data Found");
                System.out.println("[1] To change password manually");
                System.out.println("[2] To change generate new password");
                String pass = "";
                short choice;
                do {
                    System.out.print("Choice: ");
                    choice = input.nextShort();
                    switch (choice) {
                        case 1:
                            System.out.println();
                            input = new Scanner(System.in); // refresh scanner to avoid
errors
                            System.out.print("Enter Password: ");
                            pass = input.nextLine();
                            current.password = pass;
                            System.out.println();
                            break;
                        case 2:
                            int l;
                            do {
                                System.out.println();
                                input = new Scanner(System.in); // refresh scanner to
avoid errors
                                System.out.println("Enter Length of you Password");
                                l = input.nextInt();
                                if (l < 8) {
                                    System.out.println("Minimum Password length = 8");
                                }
                            } while (l < 8);
                            pass = passwordGenerator.generatePassword(l);
                            current.password = pass;
                            break;
                        default:
                            System.out.println("ERROR: Choice not valid");
                    }
                } while (choice < 1 || choice > 2);
                flag = 1;
            }
            current = current.next;
        }

        if (flag == 0) {
            System.out.println("Data does not exist!");
        } else {
            PrintWriter writer = null;
            try {
                writer = new PrintWriter("passwordDB.txt");
                writer.print("");
                writer.close();
            } catch (FileNotFoundException e) {
                System.out.println("File not found");
                e.printStackTrace();
            }

            NodePassword c = first;
            while (c != null) {
```

```
                String data = c.serviceName + "<->" + c.userName + "<->" + c.password +
"\n";

                File file = new File("passwordDB.txt");
                FileWriter fileWriter = null;
                try {
                    fileWriter = new FileWriter(file, true);
                    fileWriter.write(data);
                    fileWriter.close();
                } catch (IOException e) {
                    System.out.println("Error");
                    throw new RuntimeException(e);
                }
                c = c.next;
            }
        }
    }

    public void deletePassword() {
        System.out.println("==> DELETE <==");

        Scanner input = new Scanner(System.in);
        String ServiceName, UserName;

        System.out.println("Enter Service Name:");
        ServiceName = input.nextLine();

        System.out.println("Enter Username:");
        UserName = input.nextLine();

        if (first == null) {
            System.out.println("The list is empty");
            return;
        }

        if (ServiceName.equals(first.serviceName) && UserName.equals(first.userName)) {
            first = first.next;
        } else if (ServiceName.equals(last.serviceName) &&
UserName.equals(last.userName)) {
            last.previous.next = null;
            last = last.previous;
        } else {
            NodePassword current = first.next;
            while (current != null) {
                if (ServiceName.equals(current.serviceName) &&
UserName.equals(current.userName)) {
                    current.previous.next = current.next;
                    current.next.previous = current.previous;
                    break;
                }
                current = current.next;
            }
        }

        PrintWriter printWriter = null;
        try {
            printWriter = new PrintWriter("passwordDB.txt");
            printWriter.print("");
            printWriter.close();
```

```
        } catch (FileNotFoundException e) {
            System.out.println("File not found");
            e.printStackTrace();
        }

        NodePassword current = first;
        while (current != null) {
            String data = current.serviceName + "<->" + current.userName + "<->" +
current.password + "\n";

            File file = new File("passwordDB.txt");
            FileWriter fileWriter = null;
            try {
                fileWriter = new FileWriter(file, true);
                fileWriter.write(data);
                fileWriter.close();
            } catch (IOException e) {
                System.out.println("Error");
                throw new RuntimeException(e);
            }
            current = current.next;
        }
        System.out.println("Password successfully deleted!");
    }


    public void weakPassword() {
        NodePassword current = first;
        while (current != null) {
            String pass = current.password;
            boolean flag = false;
            Scanner sc = null;
            try {
                sc = new Scanner(new FileInputStream("hackedPassword.txt"));
            } catch (FileNotFoundException e) {
                System.out.println("File not found");
            }
            while (sc.hasNextLine()) {
                String line = sc.nextLine();
                if (line.equals(pass)) {
                    flag = true;
                    break;
                }
            }
            if (flag) {
                System.out.println(current.serviceName + " " + current.userName);
            }
            current = current.next;
        }
    }

    public void displayList() {
        System.out.printf("| %-8s | %-8s | %-8s |\n", "Service", "Username",
"Password");
        System.out.printf("-----------------------------------\n");
        NodePassword current = first;
        while (current != null) {
            current.displayNode();
            current = current.next;
```

```
        }
    }

    public void getAllPasswords() {
        File file = new File("passwordDB.txt");
        try {
            Scanner reader = new Scanner(file);
            String line;
            String[] stringArray;

            while (reader.hasNext()) {
                line = reader.nextLine();
                stringArray = line.split("<->");
                NodePassword newNode = new NodePassword();
                newNode.serviceName = stringArray[0];
                newNode.userName = stringArray[1];
                newNode.password = stringArray[2];

                if (first == null) {
                    first = newNode;
                } else {
                    last.next = newNode;
                    newNode.previous = last;
                }
                this.last = newNode;
            }
            reader.close();
        } catch (FileNotFoundException ex) {
            System.out.println("File not found");
        }
    }
}
```

5.5. MemoManagement

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

public class MemoManagement {
    Scanner input = new Scanner(System.in);

    public int mNo;
    public static ArrayList<String> Memos=new ArrayList<String>();

    public static void initialize(){
        getAllMemo();
    }

    public static void add(String memo) throws IOException{
        Memos.add(memo);
        String data=memo+"\n";
        File file=new File("Memo.txt");
        FileWriter writer=new FileWriter(file,true);
        writer.write(data);
        writer.close();
```

```
    }

    private static void getAllMemo(){
        File file=new File("Memo.txt");
        try {
            Scanner reader = new Scanner(file);
            String line;
            while(reader.hasNext()){
                line=reader.nextLine();
                Memos.add(line);
            }
        } catch (FileNotFoundException ex) {
            System.out.println("Memo File Not Found !");
        }
    }

    public static void updateMemo() throws IOException{
        File file=new File("Memo.txt");
        try (FileWriter writer = new FileWriter(file)) {
            String data;
            for (String m : Memos){
                data=m+"\n";
                writer.write(data);
            }
        }
    }

    public static void showMemo(){
        int counter=0;

        if (Memos.isEmpty()){
            System.out.println("\t==> No Memos added yet <==");
        }
        for (String m :Memos){
            System.out.printf("%5d | %-10s \n",++counter,m);
        }
    }

}
```

6.0.Output

```
*--------------------------------------*
|      Password Management System      |
*--------------------------------------*

Enter master password: admin1
Login Failed
Remaining tries: 2
Enter master password:
```

```
Enter master password: admin1
Login Failed
Remaining tries: 2
Enter master password: admin1
Login Failed
Remaining tries: 1
Enter master password: admin1
Login Failed
Remaining tries: 0
You have exceeded the number of tries
```

```
*---------------------------------------*
|       Password Management System      |
*---------------------------------------*

Enter master password: admin
Login Successful
--------->  Main Menu  <--------
[1]- Generate Password
[2]- Update Password
[3]- Delete Password
[4]- Saved Passwords
[5]- Unsafe Passwords list
[6]- Create Memo
[7]- Delete Memo
[8]- View Memos
[0]- Exit System
-----------------------------
Choice:
```

```
Choice: 1
==> ADD <==
Enter Service Name:
netflix
Enter Username:
user1
Do you want to generate password? (y/n)
n
Enter Password:
hello
Password Strength:1 out of 5
■□□□□
Errors:
Password should contain atleast 8 characters
Password must contain at least one number.
Password must contain at least one special character
Password must contain at least one Uppercase character

Enter Password:
H3ll0.w0rld
Password Strength:5 out of 5
■■■■■
```

```
Choice: 2
Enter Service Name:
netflix
Enter Username:
user1
Data Found
[1] To change password manually
[2] To change generate new password
Choice: 2

Enter Length of you Password
9
```

```
Choice: 3
==> DELETE <==
Enter Service Name:
netflix
Enter Username:
user1
Password successfully deleted!
```

```
Choice: 4
| Service  | Username | Password |
-----------------------------------
| hello    | hello    | "ACR7EBQ |
| facebook | johncena | admin    |
| twitter  | patrick  | Abcksak123* |
| netflix  | user1    | \qd>"|{t? |
```

```
Choice: 5
facebook johncena
```

```
Choice: 6
Your Memo Start from Here:
test memo
```

```
Choice: 7
    1 | HELLOWORLD
    2 | HELLO
    3 | h
    4 | qweqweqwF
    5 | Dog
    6 | test memo
Customer Index to remove : 4
Removed Successfully!
```

```
Choice: 8
    1 | HELLOWORLD
    2 | HELLO
    3 | h
    4 | Dog
    5 | test memo
```

```
Choice: 0

Process finished with exit code 0
```

7.0. Future Scope

A PMS is a tool that helps users in managing and securely storing their passwords. For each of their accounts, it enables users to create strong, unique passwords, which are then saved in a secure database. A single master password, which is the only one users need to know, may then be used to access the PMS. Additional features offered by some password management applications include the ability to generate safe, random passwords and the ability to automatically fill out login forms with the appropriate passwords. Users may be able to avoid the typical security issue of repeating passwords as a result. For individuals who have a lot of online accounts, using a PMS can be a more easy and safe approach to handle passwords.

8.0. Conclusion

This project is a password management system that allows users to generate, update, and view saved passwords, as well as create and view memos.The project’s goal of creating a password management system has been successfully achieved. It allows both users and administrators to easily handle and secure their passwords. In today's digital age, this project can greatly streamline processes and improve efficiency by eliminating the need for manual password management. Overall, the password management system is a valuable addition to any organization.