DATA STRUCTURE PROJECT REPORT

Simple Password Strength Checker

Submitted by

Yashashree Ganesh Borkar

Registration Number: 12409867

Course Code: CSM-228

Under the Guidance of

Mr. Aman Kumar

Delivery and Student Success

School of Computer Science and Engineering

ACKNOWLEDGEMENT

I wish to express my Sincere gratitude to **Mr. Aman Kumar** for his unwavering support and assistance throughout my project. I also extend my thanks to our friends for providing me with the opportunity to work on a project titled " Shortest Job First Scheduling " . Their guidance and insights were instrumental in the successful completion of this project.

Name: **Yashashree Ganesh Borkar**
Registration Number: **12409867**

Table of Contents

1.INTRODUCTION

In the modern digital era, passwords play a critical role in protecting personal and organizational data from unauthorized access. Weak passwords are one of the major reasons for security breaches and cyberattacks. Therefore, it is essential to create strong passwords that follow standard security guidelines.

The **Simple Password Strength Checker** project is a **command-line based Java application** that analyzes the strength of a user-entered password based on predefined rules such as length, usage of uppercase and lowercase letters, digits, and special characters. The system continuously prompts the user to enter a password until a **strong password** is provided.

This project demonstrates the **practical application of Data Structures and Algorithms (DSA)** concepts such as strings, linear traversal, conditional logic, loops, and time–space complexity analysis. The project is purely console-based and is developed for academic and learning purposes.

The project has been implemented using:

1. Java Programming Language
2. Core Java Libraries

2.OBJECTIVES AND SCOPE OF THE PROJECT

2.1 Objectives

The primary objectives of the Simple Password Strength Checker project are:

1. To design a Java-based console application that evaluates password strength.

2. To educate users about strong password guidelines before password entry.

3. To identify weak and medium passwords and provide clear improvement suggestions.

4. To repeatedly prompt the user until a strong password is entered.

5. To demonstrate the application of basic DSA concepts such as strings, loops, and conditional statements.

6. To use dummy password examples to help users understand password strength levels.

2.2 Scope of the Project

The scope of the project includes:

- Password validation based on predefined security rules.
- Displaying guidelines for creating strong passwords before user input.
- Classifying passwords into **Weak, Medium, and Strong** categories.
- Providing suggestions to improve password strength.
- Demonstrating DSA concepts in a real-world cybersecurity-related problem.
- Command-line based execution without any graphical user interface.

2.3 Out of Scope

The following features are not included in the project:

- Password encryption or storage
- Database integration
- Graphical User Interface (GUI)
- User authentication systems

- Network-based password validation

2.4 Limitations

- The application works only in a console environment.
- Password rules are predefined and static.
- Designed only for academic demonstration purposes.

3.APPLICATION TOOLS

3.1 Programming Language

- **Java** – Used to implement the complete application logic.

3.2 Development Environment

- Visual Studio Code / Terminal
- macOS / Windows Operating System

3.3 Data Structures Used

- **String** – To store and analyze passwords.

- **ArrayList** – To store missing password requirements.

3.4 Algorithms Used

- Linear traversal algorithm
- Counting technique
- Conditional decision-making
- Looping mechanism (do–while loop)

4.METHODOLOGY/ALGORITHM IMPLEMENTATION

The project follows a structured methodology to evaluate password strength.

4.1 Password Strength Evaluation Algorithm

Steps involved:

1. Display password guidelines to the user.
2. Accept password input from the user.
3. Check password length.

4. Verify presence of uppercase, lowercase, digits, and special characters.

5. Calculate password score.

6. Identify missing security rules.

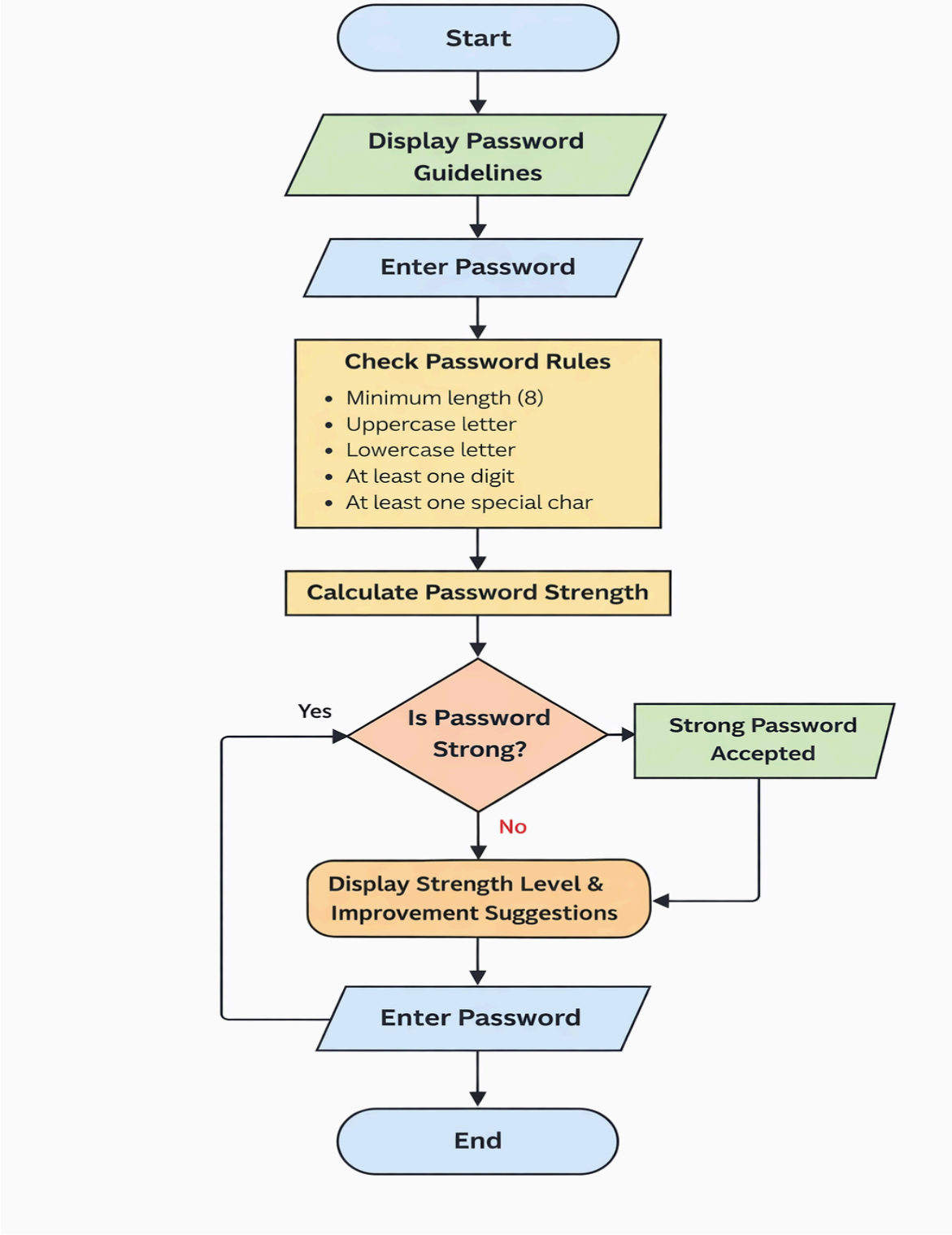
7. Display strength result and improvement suggestions.

8. Repeat steps until password is strong.

4.2 Time and Space Complexity Analysis

- **Time Complexity:** O(n), where n is the length of the password.

- **Space Complexity:** O(1), as only a limited number of variables are used.



Flowchart of Password Strength Checker

5. SCREENSHOT OF EXECUTION

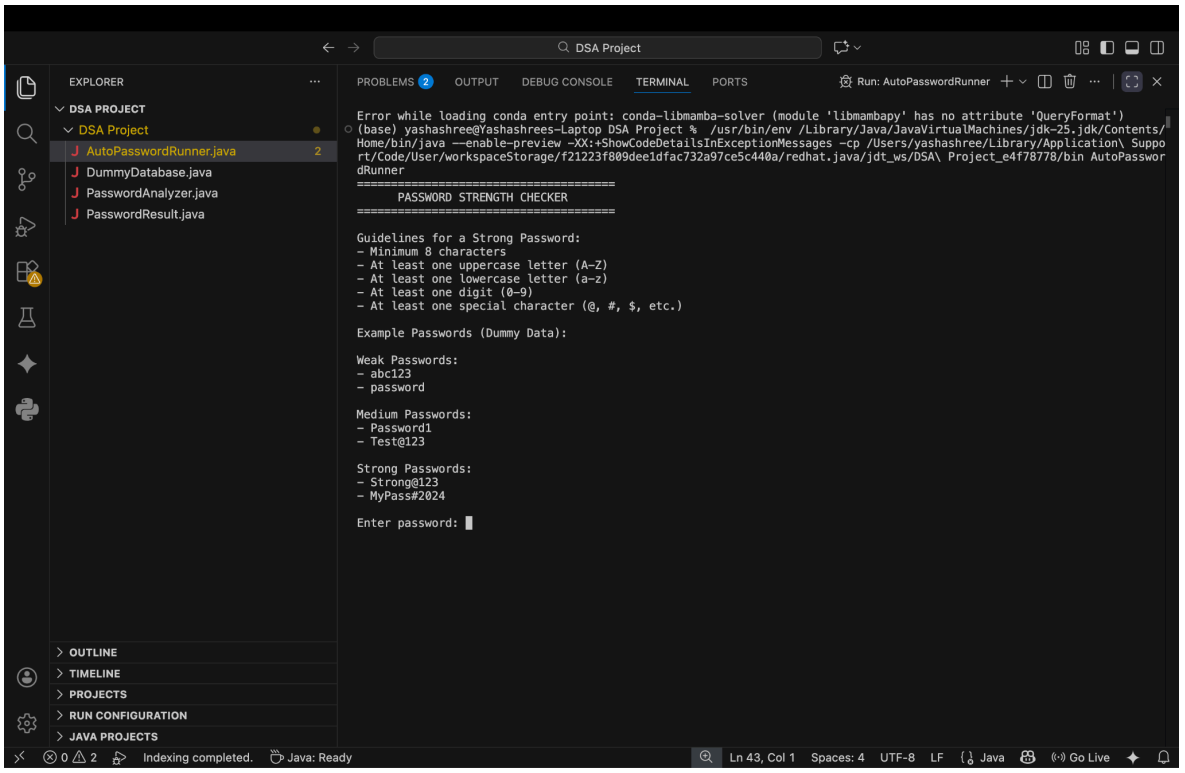


Fig. 1: Program start screen showing guidelines

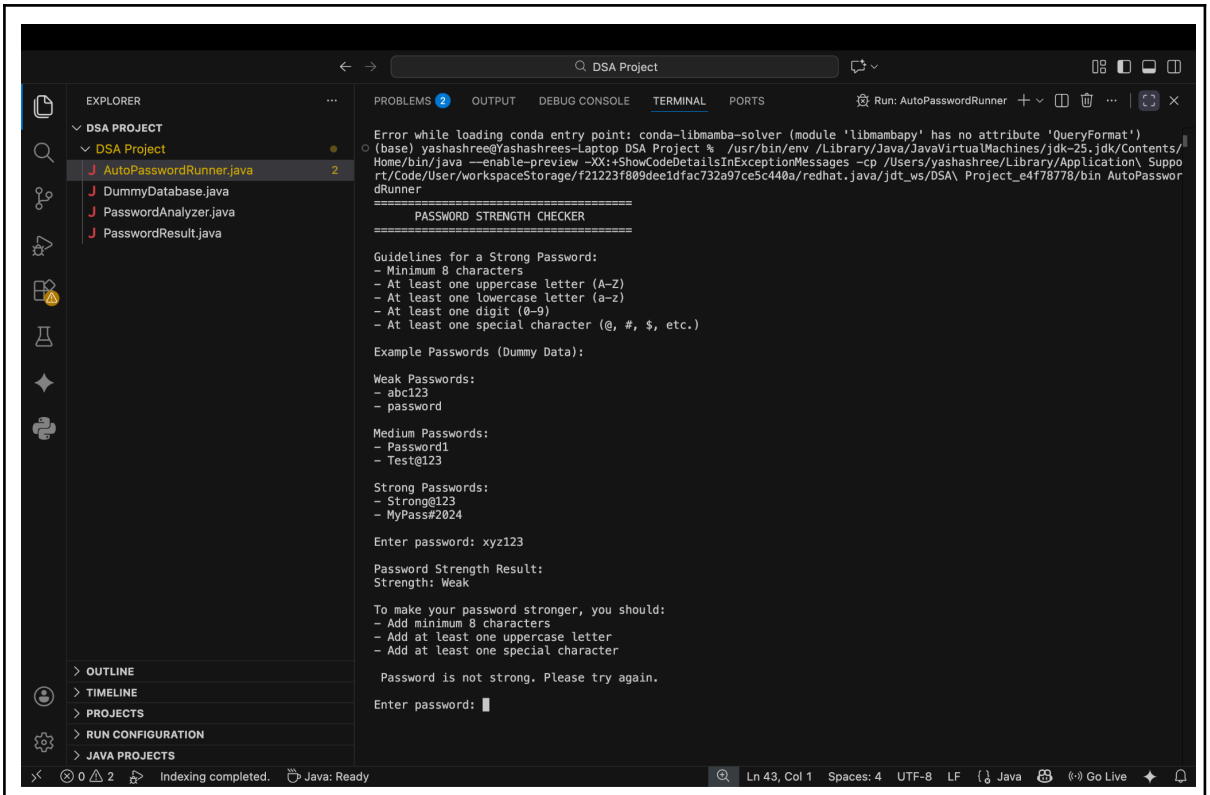


Fig. 2: Password entered as weak or medium

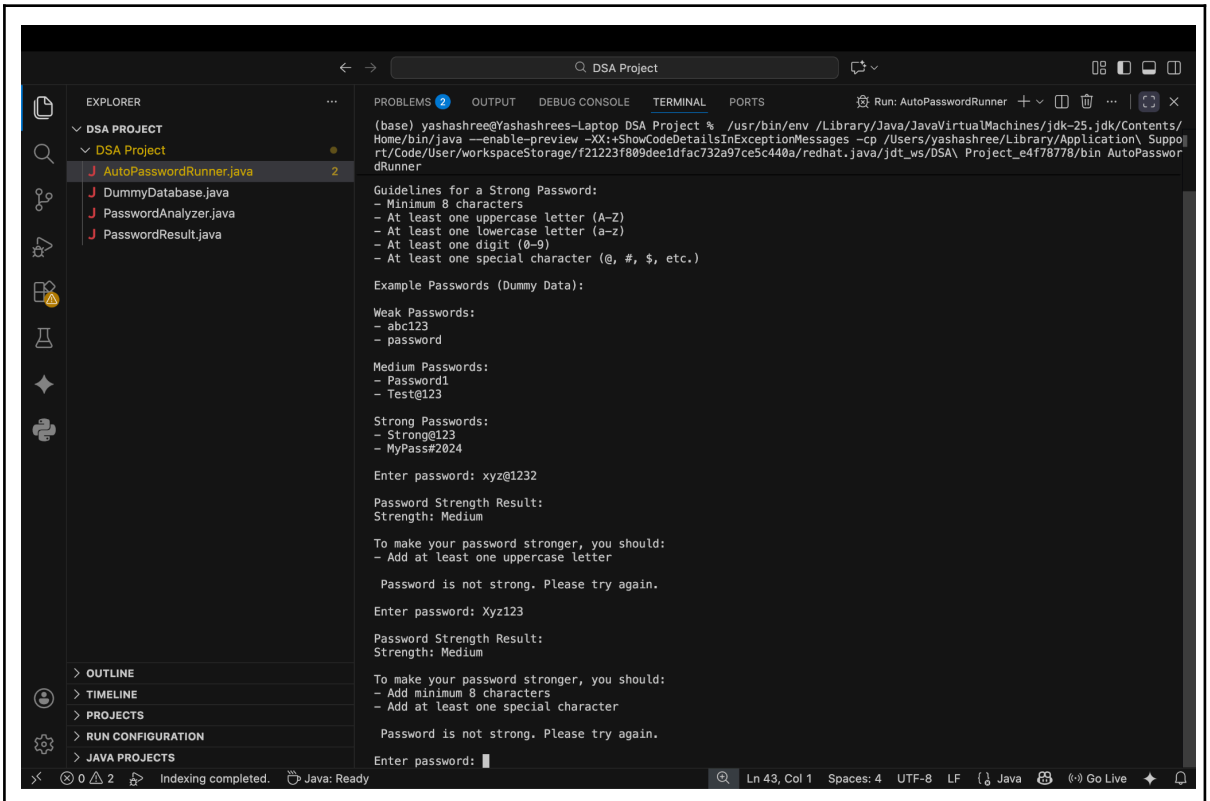


Fig. 3: Suggestions displayed

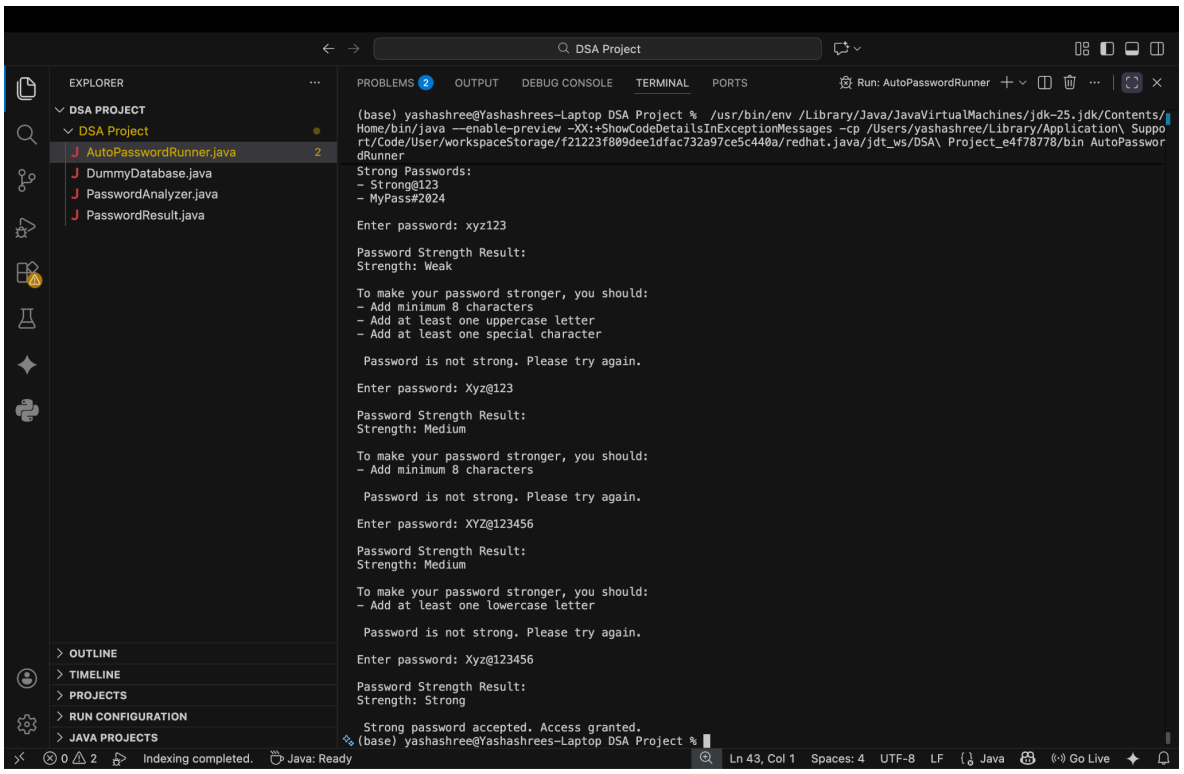


Fig. 4: Final strong password accepted

6.SUMMARY

The Simple Password Strength Checker project successfully demonstrates how Data Structures and Algorithms can be applied to solve a real-world cybersecurity problem. The system ensures that users understand password security guidelines and encourages the creation of strong passwords by repeatedly prompting until all security conditions are met.

This project effectively uses strings, loops, and conditional logic while maintaining low time and space complexity. It serves as a foundational

academic project that bridges theoretical DSA concepts with practical implementation.

7.BIBLIOGRAPHY

- Oracle Java Documentation
- GeeksforGeeks – Password Strength Algorithms
- Class Notes
- Java Official Tutorials

8.ANNEXURE

8.1 Source Code Files

1. AutoPasswordRunner.java

```
import java.util.Scanner;


public class AutoPasswordRunner {


   public static void main(String[] args) {
```

```
        Scanner sc = new Scanner(System.in);
        PasswordAnalyzer analyzer = new PasswordAnalyzer();



System.out.println("==========================================
=====");
        System.out.println("      PASSWORD STRENGTH
CHECKER");

System.out.println("==========================================
=====");


        showGuidelines();
        DummyDatabase.showExamples();


        boolean isStrong = false;


        do {
            System.out.print("\nEnter password: ");
            String password = sc.nextLine();
```

```
        isStrong = analyzer.analyze(password);

        if (!isStrong) {
            System.out.println("\n Password is not strong. Please try again.");
        }

    } while (!isStrong);

    System.out.println("\n Strong password accepted. Access granted.");
    sc.close();
  }

  static void showGuidelines() {
    System.out.println("\nGuidelines for a Strong Password:");
    System.out.println("- Minimum 8 characters");
    System.out.println("- At least one uppercase letter (A-Z)");
```

```
        System.out.println("- At least one lowercase letter (a-z)");
        System.out.println("- At least one digit (0-9)");
        System.out.println("- At least one special character (@, #, $, etc.)");
    }
}
```

2. PasswordAnalyzer.java

```
import java.util.ArrayList;

public class PasswordAnalyzer {

    public boolean analyze(String password) {

        ArrayList<String> missing = new ArrayList<>();
        int score = 0;

        if (password.length() >= 8) score++;
        else missing.add("Add minimum 8 characters");
```

```
        if (password.matches(".*[A-Z].*")) score++;
        else missing.add("Add at least one uppercase letter");

        if (password.matches(".*[a-z].*")) score++;
        else missing.add("Add at least one lowercase letter");

        if (password.matches(".*[0-9].*")) score++;
        else missing.add("Add at least one digit");

        if (password.matches(".*[^a-zA-Z0-9].*")) score++;
        else missing.add("Add at least one special character");

        return PasswordResult.display(score, missing);
    }
}
```

3. PasswordResult.java

```
import java.util.ArrayList;
```

```
public class PasswordResult {


  public static boolean display(int score, ArrayList<String> missing)
{


      System.out.println("\nPassword Strength Result:");


      if (score <= 2) {
          System.out.println("Strength: Weak");
      } else if (score <= 4) {
          System.out.println("Strength: Medium");
      } else {
          System.out.println("Strength: Strong");
          return true;
      }
      System.out.println("\nTo make your password stronger, you
should:");
      for (String m : missing) {
          System.out.println("- " + m);
```

```
        }


        return false;
    }
}
```

4. DummyDatabase.java

```
public class DummyDatabase {


    public static void showExamples() {
        System.out.println("\nExample Passwords (Dummy Data):");


        System.out.println("\nWeak Passwords:");
        System.out.println("- abc123");
        System.out.println("- password");


        System.out.println("\nMedium Passwords:");
        System.out.println("- Password1");
        System.out.println("- Test@123");
```

```
        System.out.println("\nStrong Passwords:");
        System.out.println("- Strong@123");
        System.out.println("- MyPass#2024");
    }
}
```