PASSWORD GENERATOR

A PROJECT REPORT

Submitted by

NAGARAJ S (2303811710421100)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on " **PASSWORD GENERATOR"** is the bonafide work of **NAGARAJ S (2303811710421100)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.)
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE	**SIGNATURE**
Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,	Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.,),
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Technology (Autonomous)	K.Ramakrishnan College of Technology (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on 03-12-2024.

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

CGB1201-JAVA PROGRAMMING
Ms.R.ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

[illegible]

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **"PASSWORD GENERATOR"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING.**

Signature

[illegible]

NAGARAJ S

Place: Samayapuram

Date:03-12-2024.

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.,),** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

In the modern digital era, online security is paramount, with passwords serving as the primary line of defense for personal and organizational data. Weak or easily guessable passwords are a significant vulnerability, often exploited by attackers. To address this issue, this project aims to design and develop a **Password Generator Application** using Java and AWT (Abstract Window Toolkit).

The application provides an intuitive graphical user interface (GUI) that enables users to create strong, secure, and random passwords based on customizable criteria. Users can specify parameters such as password length, inclusion of uppercase letters, lowercase letters, numbers, and special characters. The application ensures that the generated passwords adhere to contemporary security standards, making them resistant to brute force and dictionary attacks.

By leveraging the Java AWT library, the application delivers a lightweight and platform-independent solution. The project demonstrates the practical implementation of randomization techniques, GUI design principles, and user-interaction management.

This Password Generator Application is not only a useful tool for individual users but also a stepping stone for students and developers to understand the intersection of usability and security in software design.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Password Generator Application** simplifies the complex process of creating secure passwords through an intuitive interface. By abstracting the technical details of password generation, such as randomization algorithms and character set management, the application allows users to focus solely on defining their preferences, like password length and character types. This abstraction ensures that users without technical expertise can still generate strong, secure passwords effortlessly, thereby bridging the gap between usability and robust cybersecurity practices.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the Password Generator Application is to enhance online security by providing users with a reliable tool to create strong and secure passwords. The specific goals include:

1. **Facilitating Secure Password Creation**: Generate passwords that are resistant to common attacks such as brute force, dictionary, and phishing attacks.
2. **Customizability**: Allow users to specify criteria for password generation, such as length and character types (uppercase, lowercase, numbers, special characters).
3. **Promoting Cybersecurity Awareness**: Educate users about the importance of strong passwords in safeguarding personal and sensitive information.
4. **User-Friendly Interface**: Develop a simple, intuitive, and accessible interface using Java AWT, enabling users of all technical backgrounds to use the application effectively.
5. **Efficiency and Randomness**: Ensure that passwords are generated quickly and incorporate randomness to maximize security.

These objectives aim to address the growing need for strong password management tools in an increasingly digital world.

1.2 Overview

The Password Generator Application is designed to assist users in creating secure and robust passwords to safeguard their online accounts and sensitive information. With the increasing prevalence of cyberattacks, having strong passwords is critical to minimizing vulnerabilities. The application utilizes Java's AWT library to provide a platform-independent graphical interface that ensures ease of use and accessibility.

1.3 Java Programming Concepts

The development of the Password Generator Application leverages various programming concepts in Java, with a strong foundation in object-oriented programming (OOPS). These concepts include:

1.3.1 Object-Oriented Programming (OOPs) Concepts

1. **Encapsulation**:

   The application uses encapsulation to bind the data (such as user-defined criteria) and methods (such as password generation logic) together into cohesive classes. This ensures that sensitive data remains secure and accessible only through controlled methods.

   - Example: The criteria for password generation (length, character types) are encapsulated within a class, ensuring modularity and data protection.

2. **Inheritance**:

   Inheritance is utilized to extend functionalities where necessary. For instance, the GUI design may extend AWT classes like Frame or Panel to incorporate custom behaviors while reusing existing components.

3. **Polymorphism**:

   Polymorphism is applied to provide flexibility in handling different operations, such as validating user input or generating random characters. Methods may be overridden or overloaded to cater to specific needs.

4. **Abstraction**:

   The application hides complex implementation details of password generation algorithms and GUI interactions, exposing only the functionalities needed by the user. This makes the application intuitive and easy to use.

5. **Classes and Objects**:

   The application uses multiple classes, such as a class for handling user input, another for password generation logic, and a separate class for managing the GUI. Objects are instantiated from these classes to perform specific tasks.

1.3.2 Additional Java Concepts

1. **Randomization**:

   Java's Random or SecureRandom class is used to generate unpredictable and secure passwords.

2. **Exception Handling**:

   Exception handling ensures that the application runs smoothly and provides user-friendly error messages in case of invalid inputs or unexpected errors.

3. **Collections Framework**:
   Lists and character sets may be used to manage different categories of characters (uppercase, lowercase, digits, symbols) during password generation.
4. **AWT Components**:
   Java AWT components like Button, TextField, Label, and Checkbox are used to create an interactive user interface for customizing and generating passwords.

By combining these concepts, the Password Generator Application demonstrates the power and versatility of Java programming in creating efficient, secure, and user-friendly software solutions.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed architecture for the Password Generator Application follows a modular design to ensure scalability, maintainability, and ease of development. The architecture incorporates three main layers:

1. Presentation Layer (User Interface)

- **Purpose**: To provide an interactive and user-friendly graphical interface for users to specify password criteria and generate passwords.
- **Components**:
    - Java AWT components such as Frame, Button, Label, Checkbox, and TextField.
    - Input fields for user-defined parameters like password length, and checkboxes for selecting character types (uppercase, lowercase, digits, special characters).
    - Buttons for actions like "Generate Password" and "Reset".

2. Application Logic Layer (Core Logic)

- **Purpose**: To implement the core functionality of password generation based on user-defined criteria.
- **Components**:
    - **PasswordGenerator Class**:
        - Generates random passwords using Java’s SecureRandom class for enhanced security.
        - Ensures compliance with user-defined rules, such as including specific character types and meeting the desired length.
    - **Validation Module**:
        - Validates user input (e.g., ensures password length is within acceptable limits).
        - Handles errors like invalid or incomplete criteria.

3. Data Layer (Character Sets)

- **Purpose**: To manage the data required for password generation.
- **Components**:
  - Predefined character sets for uppercase letters, lowercase letters, numbers, and special characters.
  - These character sets are stored as arrays or lists and accessed dynamically during password generation.

4.Data Flow

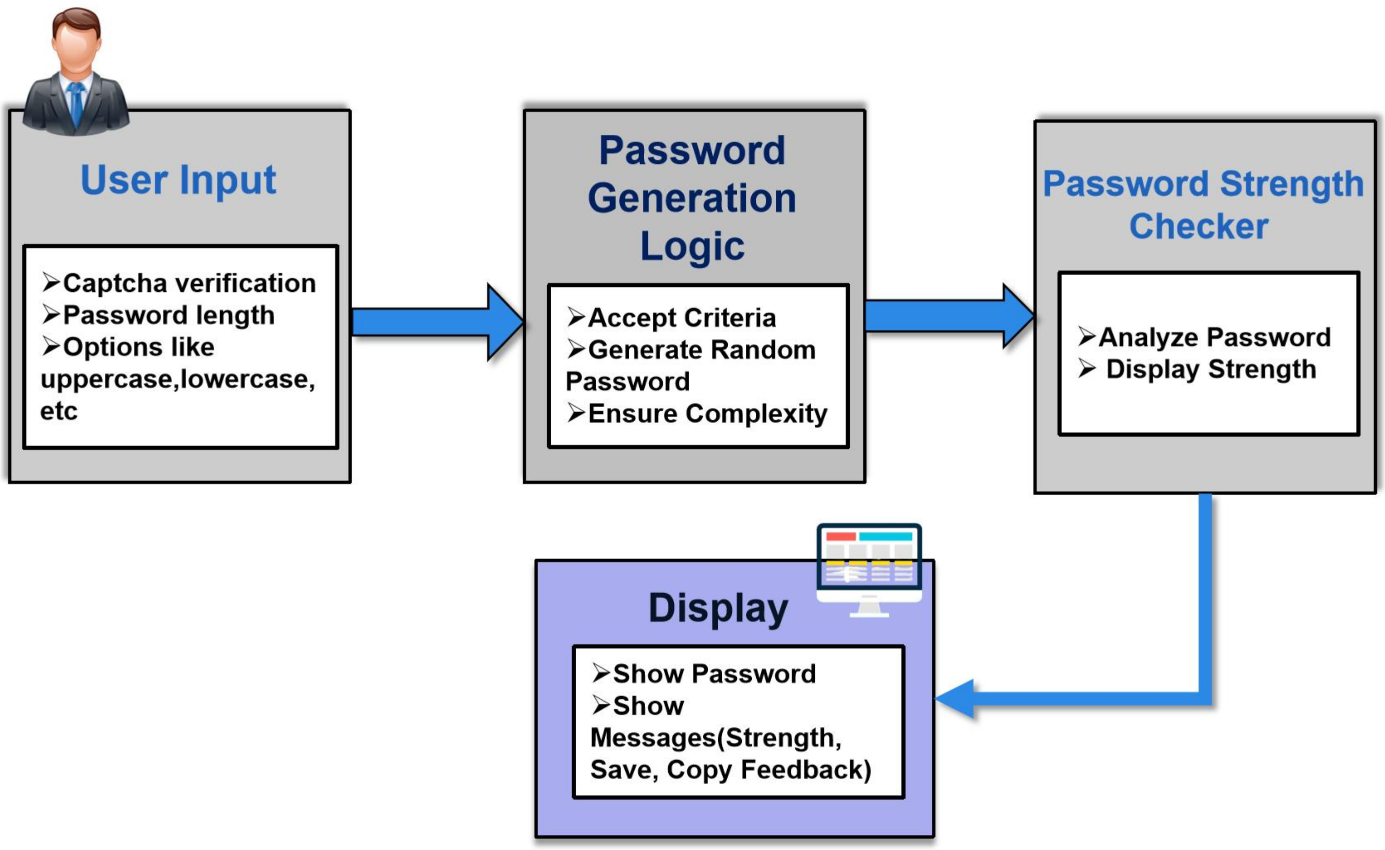
1. **User Input**:
   Users interact with the GUI to define password parameters. Input is validated and passed to the core logic.
2. **Core Processing**:
   The PasswordGenerator class processes the input, selects characters from the relevant sets, and generates a random password.
3. **Output Display**:
   The generated password is displayed in the user interface for copying or further use.

5.Advantages of the Proposed Architecture

- **Modularity**: Separate layers for UI, logic, and data enhance maintainability and facilitate debugging.
- **Scalability**: New features, such as storing generated passwords or adding advanced security checks, can be integrated easily.
- **Reusability**: The core logic can be reused or extended for similar applications.
- **Security**: Use of SecureRandom ensures that the generated passwords are cryptographically secure.

This architecture ensures that the application meets its objectives while maintaining a clean and efficient design structure.

2.2 Block Diagram


User Input
➢Captcha verification
➢Password length
➢Options like uppercase,lowercase, etc
Password Generation Logic
➢Accept Criteria
➢Generate Random Password
➢Ensure Complexity
Password Strength Checker
➢Analyze Password
➢ Display Strength
Display
➢Show Password
➢Show Messages(Strength, Save, Copy Feedback)

CHAPTER 3
MODULE DESCRIPTION

3.1 CAPTCHA Verification Module

Purpose:

This module ensures that only legitimate human users can access the application by solving a CAPTCHA. It serves as a security measure against automated bots.

Key Features:

- Generates a CAPTCHA by creating a simple math problem (e.g., addition of two random numbers).
- Validates the user's input against the expected result.
- Grants access to the Password Generator only upon successful CAPTCHA verification.
- Allows refreshing the CAPTCHA for generating a new challenge.

Relevance:

This module adds an extra layer of security to prevent misuse or unauthorized access.

3.2 Password Generation Module

Purpose:

This module allows users to generate random passwords based on specific criteria to ensure strong, secure passwords.

Key Features:

- Accepts the desired password length (between 5 and 15).
- Provides options for including uppercase letters, lowercase letters, digits, special characters, and spaces in the password.
- Randomly constructs a password from the selected character pool using a secure random generator.
- Ensures that the password adheres to user-defined criteria.

Relevance:
This module is the core functionality of the application, offering flexibility and customization for password creation.

3.3 Password Strength Evaluation Module

Purpose:
This module evaluates the complexity of the generated password and provides feedback to the user regarding its security level.

Key Features:

- Analyzes the password for the presence of uppercase letters, lowercase letters, digits, special characters, and spaces.
- Assigns a strength score based on the complexity of the password.
- Displays the password strength visually using a progress bar (e.g., weak, moderate, strong).

Relevance:
Helps users understand how secure their passwords are and encourages the creation of stronger, more secure passwords.

3.4 Utility Module

Purpose:
This module adds additional functionalities that enhance user convenience and usability.

Key Features:

- Allows users to save the generated password to a file (`passwords.txt`) for later use.
- Provides the ability to copy the password directly to the clipboard for easy use in other applications.
- Includes feedback dialogs to inform users about successful actions, such as saving or copying the password.

Relevance:

Improves the overall user experience by offering tools for managing and utilizing generated passwords efficiently.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Password Generator Application successfully addresses the need for creating strong and secure passwords in today's digital era. With its CAPTCHA verification module, the application ensures access only to legitimate users, adding an extra layer of security. The password generation module provides users with the flexibility to create highly customizable passwords, tailored to specific security needs. The password strength evaluation module offers real-time feedback, enabling users to make informed decisions about their password complexity. Additionally, the utility module enhances usability through convenient features like saving and copying passwords. Overall, this application promotes better password practices, thereby improving online security and reducing the risks associated with weak passwords.

4.2 FUTURE SCOPE

1. **Advanced CAPTCHA Mechanisms:**
   - Incorporate image-based or audio CAPTCHAs to enhance human verification and improve accessibility.
2. **Password Policies:**
   - Add support for predefined password policies, such as organizational standards or compliance with frameworks like GDPR or NIST.
3. **Multi-Language Support:**
   - Allow users to use the application in different languages to make it accessible globally.
4. **Password Management Integration:**
   - Extend the application to include password storage and retrieval functionality, acting as a basic password manager.
5. **Real-Time Password Validation:**
   - Integrate APIs or algorithms to check the generated passwords against common password breach databases.
6. **Mobile and Web Versions:**

    - Develop mobile and web-based versions of the application for greater portability and accessibility.
7. **Improved User Interface:**
    - Introduce modern UI components and themes for an enhanced visual and user experience.
8. **Machine Learning for Password Analysis:**
    - Implement AI models to predict and improve password security based on patterns and user behavior.

These enhancements will make the Password Generator Application more robust, versatile, and appealing to a broader audience, ensuring it stays relevant in an evolving digital security landscape.

APPENDIX A
(SOURCE CODE)

```java
import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.io.*;
import java.util.Random;

public class PasswordGeneratorApp {

    public static void main(String[] args) {
        new CaptchaFrame();  // Start with CAPTCHA Frame
    }

    // CAPTCHA Verification Frame
    static class CaptchaFrame extends Frame implements ActionListener {
        private Label labelCaptcha, labelWelcome;
        private TextField textCaptcha;
        private Button verifyButton, refreshButton;
        private int captchaResult;

        public CaptchaFrame() {
            setTitle("Captcha Verification");
            setLayout(new FlowLayout());

            // Welcome message
            labelWelcome = new Label("Welcome to the Password Generator!");
            labelWelcome.setFont(new Font("Arial", Font.BOLD, 16));
            add(labelWelcome);

            // CAPTCHA Label
            labelCaptcha = new Label("Solve this CAPTCHA: ");
            add(labelCaptcha);
            // Generate CAPTCHA
            generateCaptcha();
            textCaptcha = new TextField(5);
            add(textCaptcha);

            // Buttons for verification and refresh
            verifyButton = new Button("Verify");
            verifyButton.addActionListener(this);
            add(verifyButton);

            refreshButton = new Button("Refresh CAPTCHA");
            refreshButton.addActionListener(this);
            add(refreshButton);

            setSize(300, 200);
            setVisible(true);

            addWindowListener(new WindowAdapter() {
                public void windowClosing(WindowEvent e) {
                    dispose();
                }
            });
        }

        // Generate CAPTCHA
        private void generateCaptcha() {
            Random random = new Random();
            int num1 = random.nextInt(10) + 1;
            int num2 = random.nextInt(10) + 1;
            captchaResult = num1 + num2;
            labelCaptcha.setText("Solve: " + num1 + " + " + num2);
        }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == verifyButton) {
                try {
                    int userInput = Integer.parseInt(textCaptcha.getText());
                    if (userInput == captchaResult) {
                        labelCaptcha.setText("Captcha Verified!");
                        new PasswordGeneratorFrame();  // Open Password Generator Frame after captcha
verification
                        dispose();
                    } else {
                        labelCaptcha.setText("Incorrect! Try again.");
                    }
                } catch (Exception ex) {
                    labelCaptcha.setText("Invalid input! Try again.");
                }
            } else if (e.getSource() == refreshButton) {
                generateCaptcha();  // Refresh CAPTCHA
            }
        }
    }

    // Password Generator Frame
    static class PasswordGeneratorFrame extends Frame implements ActionListener {
        private Label labelLength, labelPassword, labelStrength;
        private TextField textLength, textPassword;
        private Button generateButton, okButton, saveButton, copyButton;
        private Checkbox includeUpperCase, includeLowerCase, includeDigits, includeSpecialChars,
includeSpaces;
        private ProgressBar strengthBar;
        private Random random;

        public PasswordGeneratorFrame() {
            setTitle("Password Generator");
```

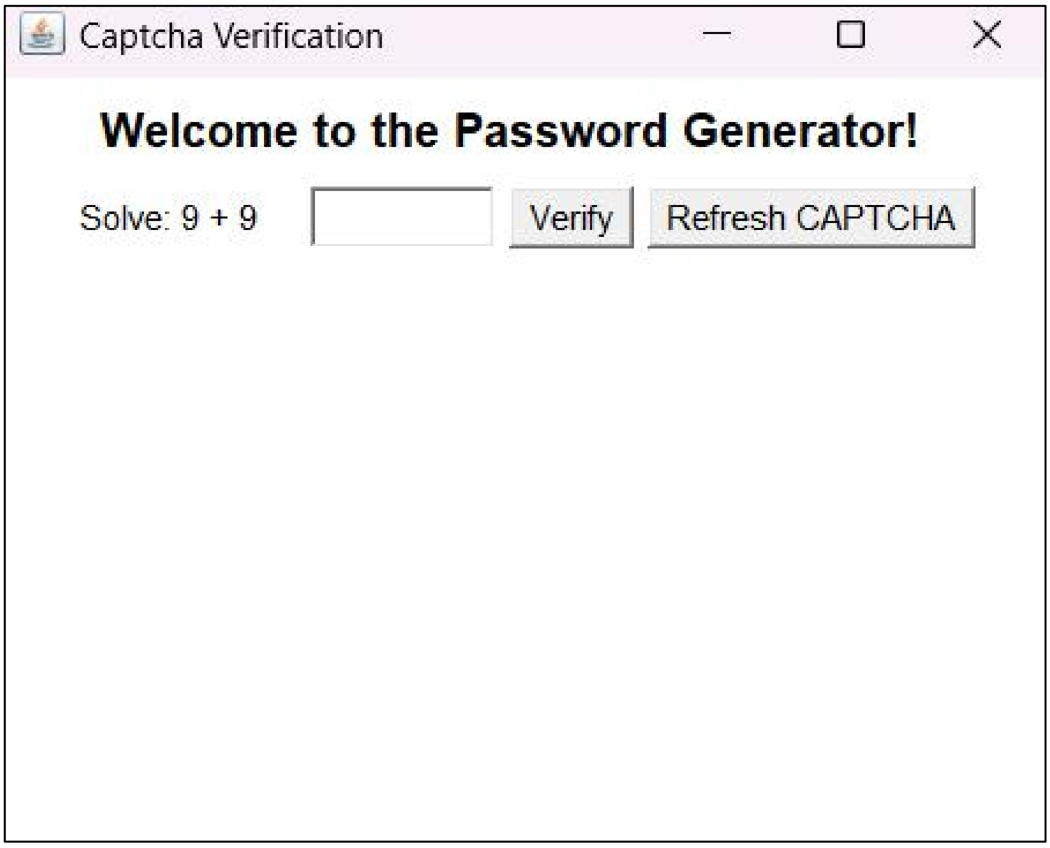
```
setLayout(new FlowLayout());

labelLength = new Label("Enter Password Length (5-15):");
add(labelLength);

textLength = new TextField(10);
add(textLength);

// Character options
includeUpperCase = new Checkbox("Include Uppercase Letters", true);
includeLowerCase = new Checkbox("Include Lowercase Letters", true);
includeDigits = new Checkbox("Include Digits", true);
includeSpecialChars = new Checkbox("Include Special Characters", true);
includeSpaces = new Checkbox("Include Spaces", false);  // Initially no spaces included

add(includeUpperCase);
add(includeLowerCase);
add(includeDigits);
add(includeSpecialChars);
add(includeSpaces);

generateButton = new Button("Generate Password");
generateButton.addActionListener(this);
add(generateButton);

labelPassword = new Label("Generated Password:");
add(labelPassword);

textPassword = new TextField(20);
textPassword.setEditable(false);
add(textPassword);

labelStrength = new Label("Password Strength:");
add(labelStrength);
```

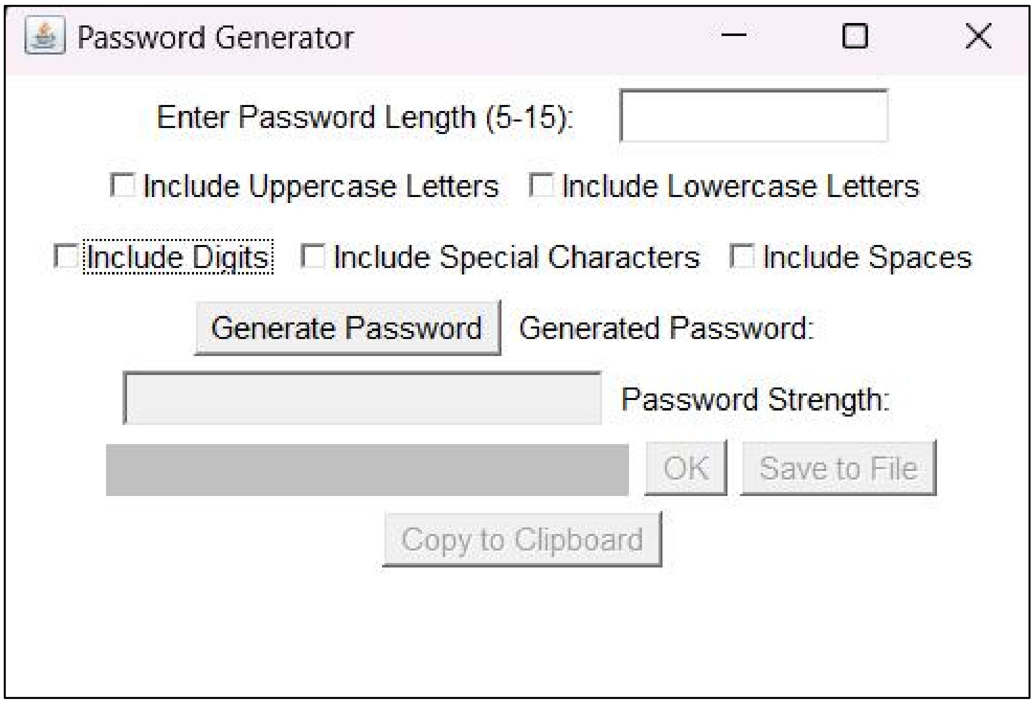
```
        strengthBar = new ProgressBar(200, 20);
        add(strengthBar);

        okButton = new Button("OK");
        okButton.addActionListener(this);
        okButton.setEnabled(false);  // Disable initially
        add(okButton);

        saveButton = new Button("Save to File");
        saveButton.addActionListener(this);
        saveButton.setEnabled(false);  // Disable initially
        add(saveButton);

        copyButton = new Button("Copy to Clipboard");
        copyButton.addActionListener(this);
        copyButton.setEnabled(false);  // Disable initially
        add(copyButton);

        setSize(350, 250);
        setVisible(true);

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                dispose();
            }
        });

        random = new Random();
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == generateButton) {
            try {
                int length = Integer.parseInt(textLength.getText());
                if (length < 5 || length > 15) {
                    textPassword.setText("Length must be between 5 and 15.");
                } else {
                    String password = generatePassword(length);
                    textPassword.setText(password);
                    updateStrengthIndicator(password);  // Update strength bar
                    okButton.setEnabled(true);  // Enable OK button once password is generated
                }
            } catch (NumberFormatException ex) {
                textPassword.setText("Invalid number! Enter a valid length.");
            }
        } else if (e.getSource() == okButton) {
            saveButton.setEnabled(true);
            copyButton.setEnabled(true);
            okButton.setEnabled(false);
        } else if (e.getSource() == saveButton) {
            savePasswordToFile();
        } else if (e.getSource() == copyButton) {
            copyToClipboard();
        }
    }

    // Method to generate random password
    private String generatePassword(int length) {
        StringBuilder charPool = new StringBuilder();

        // Add selected character types to pool
        if (includeUpperCase.getState())
charPool.append("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        if (includeLowerCase.getState()) charPool.append("abcdefghijklmnopqrstuvwxyz");
        if (includeDigits.getState()) charPool.append("0123456789");
        if (includeSpecialChars.getState()) charPool.append("!@#$%^&*()-_=+[]{}|;:,.<>?/\\");
    if (includeSpaces.getState()) charPool.append(" ");  // Add space if selected

    // If no character type is selected, return error message
    if (charPool.length() == 0) return "Error: Select at least one character type!";

    StringBuilder password = new StringBuilder();
    for (int i = 0; i < length; i++) {
        int index = random.nextInt(charPool.length());
        password.append(charPool.charAt(index));
    }

    return password.toString();
}

// Method to update password strength indicator
private void updateStrengthIndicator(String password) {
    int strength = calculateStrength(password);
    strengthBar.setValue(strength);

    if (strength <= 50) {
        labelStrength.setText("Password Strength: Weak");
    } else if (strength <= 75) {
        labelStrength.setText("Password Strength: Moderate");
    } else {
        labelStrength.setText("Password Strength: Strong");
    }
}

// Method to calculate password strength
private int calculateStrength(String password) {
    int strength = 0;

    if (password.matches(".*[A-Z].*")) strength += 25;  // Uppercase Letters
    if (password.matches(".*[a-z].*")) strength += 25;  // Lowercase Letters
```

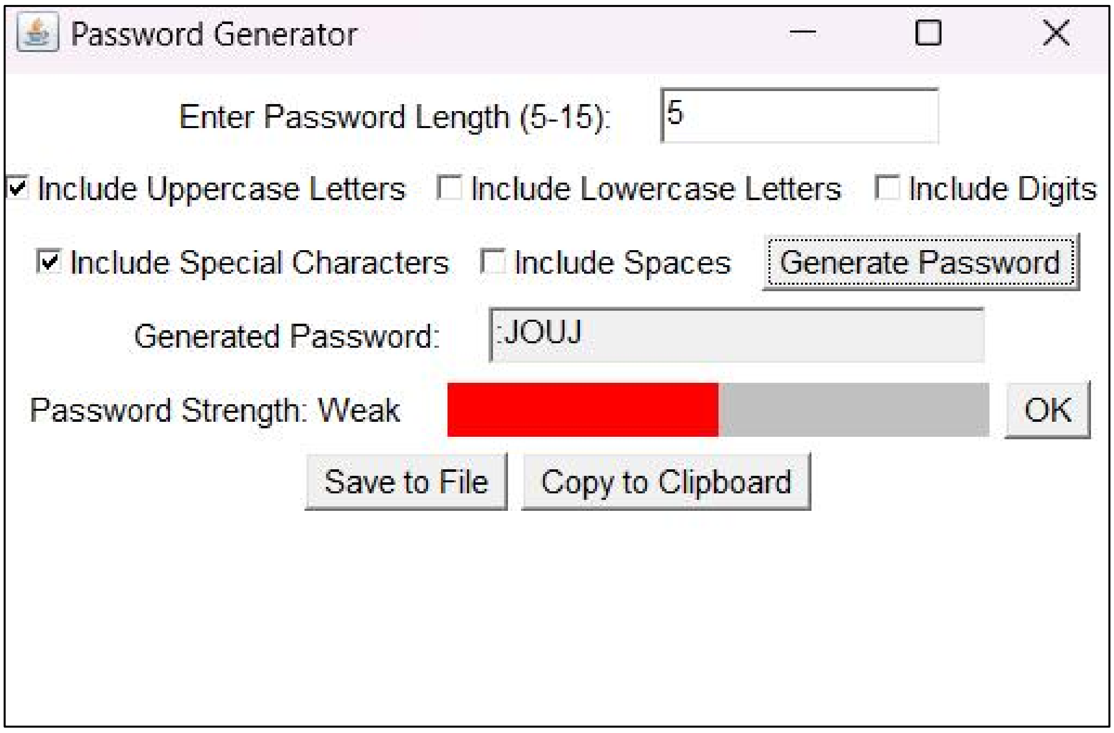
```
        if (password.matches(".*[0-9].*")) strength += 25;  // Digits
        if (password.matches(".*[!@#$%^&*(),.?\":{}|<>].*")) strength += 25;  // Special
Characters
        if (password.contains(" ")) strength += 10;  // Spaces

        return strength;
    }

    // Method to save password to a file
    private void savePasswordToFile() {
        String password = textPassword.getText();
        try (BufferedWriter writer = new BufferedWriter(new FileWriter("passwords.txt", true))) {
            writer.write(password);
            writer.newLine();
            writer.close();
            showPasswordSavedDialog();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    // Show dialog for successful password saving
    private void showPasswordSavedDialog() {
        Dialog dialog = new Dialog(this);
        dialog.setLayout(new FlowLayout());
        dialog.setSize(250, 100);
        dialog.add(new Label("Password saved to file!"));
        Button okButton = new Button("OK");
        okButton.addActionListener(e -> dialog.dispose());
        dialog.add(okButton);
        dialog.setVisible(true);
    }

    // Copy password to clipboard
    private void copyToClipboard() {
        String password = textPassword.getText();
        StringSelection selection = new StringSelection(password);
        Clipboard clipboard = Toolkit.getDefaultToolkit().getSystemClipboard();
        clipboard.setContents(selection, null);
        showPasswordCopiedDialog();
    }

    // Show dialog for password copied to clipboard
    private void showPasswordCopiedDialog() {
        Dialog dialog = new Dialog(this);
        dialog.setLayout(new FlowLayout());
        dialog.setSize(250, 100);
        dialog.add(new Label("Password copied to clipboard!"));
        Button okButton = new Button("OK");
        okButton.addActionListener(e -> dialog.dispose());
        dialog.add(okButton);
        dialog.setVisible(true);
    }
}

// Custom Progress Bar to show password strength
static class ProgressBar extends Canvas {
    private int width, height, value;

    public ProgressBar(int width, int height) {
        this.width = width;
        this.height = height;
        this.value = 0;
        setSize(width, height);
    }

    public void setValue(int value) {
        this.value = value;
```

```
            repaint();
        }

        @Override
        public void paint(Graphics g) {
            g.setColor(Color.LIGHT_GRAY);
            g.fillRect(0, 0, width, height);
            g.setColor(getColorBasedOnStrength(value));
            g.fillRect(0, 0, (int) (width * (value / 100.0)), height);
        }

        private Color getColorBasedOnStrength(int value) {
            if (value <= 50) return Color.RED;
            else if (value <= 75) return Color.ORANGE;
            else return Color.GREEN;
        }
    }
}
```

APPENDIX B
(SCREENSHOTS)







Password Generator

Enter Password Length (5-15): 5

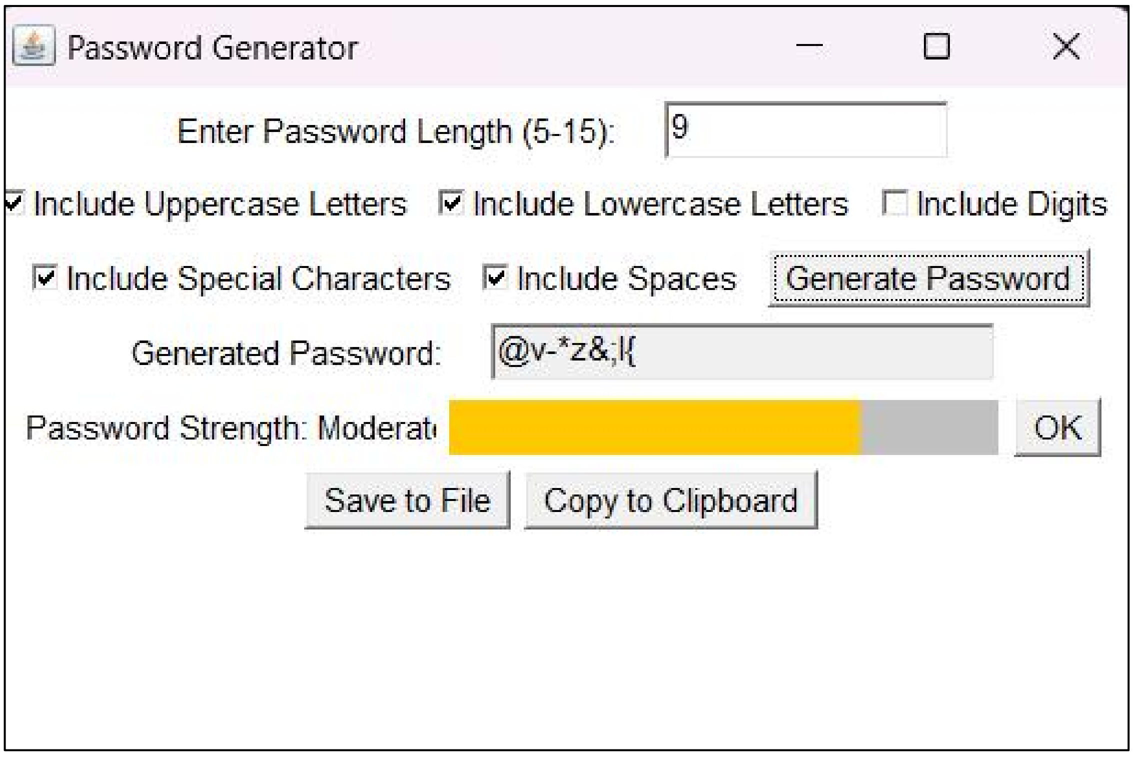
[✓ Include Uppercase Letters] Include Lowercase Letters Include Digits

[✓ Include Special Characters] Include Spaces Generate Password

Generated Password: :JOUJ

Password Strength: Weak OK

Save to File Copy to Clipboard



Password Generator

Enter Password Length (5-15): 9

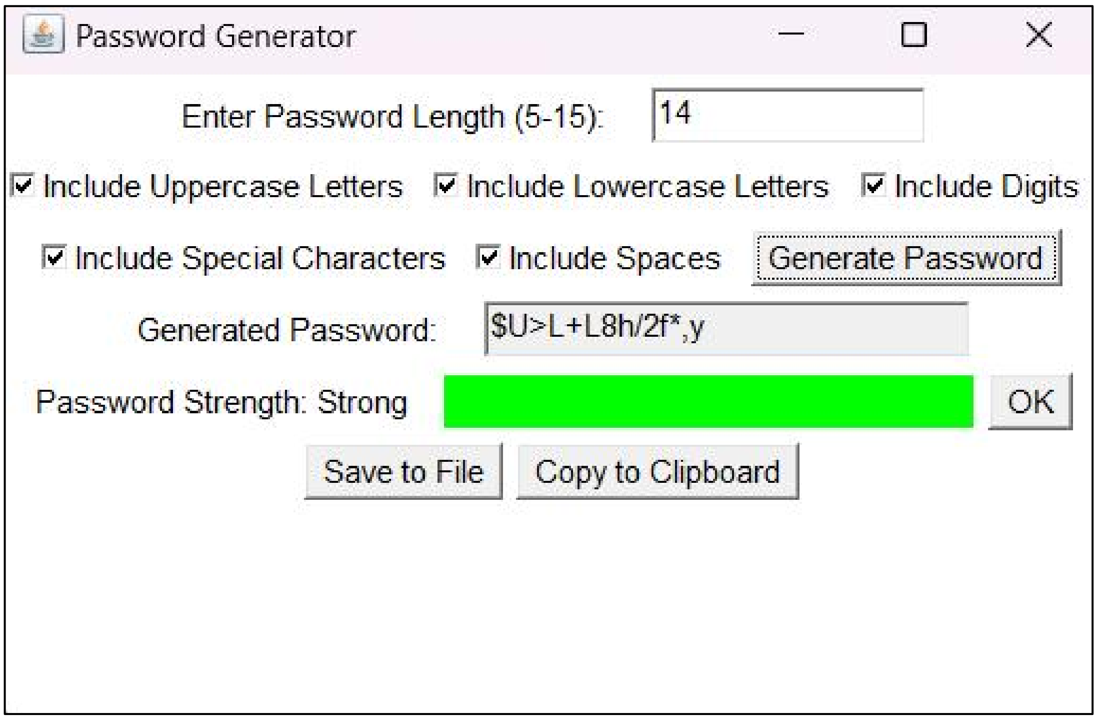
[✓ Include Uppercase Letters] [✓ Include Lowercase Letters] Include Digits

[✓ Include Special Characters] [✓ Include Spaces] Generate Password

Generated Password: @v-*z&;l{

Password Strength: Moderat OK

Save to File Copy to Clipboard



Password Generator

Enter Password Length (5-15): 14

☑ Include Uppercase Letters ☑ Include Lowercase Letters ☑ Include Digits

☑ Include Special Characters ☑ Include Spaces

Generated Password: $U>L+L8h/2f*,y

Password Strength: Strong 🟩

REFERENCES

YOUTUBE LINKS

- **https://youtu.be/niay2OKGmlM?feature=shared**
- **https://youtu.be/oBU1Mr1rZag?feature=shared**

WEBSITES

- **https://github.com/KZarzour/Password-Generator**
- **https://chatgpt.com/share/674b5386-1de0-8004-a478-f9d8f1aa95f4**

BOOKS

- **"Core Java Volume I - Fundamentals" by Cay S. Horstmann and Gary Cornell**
  A classic book that covers Java fundamentals, including GUI development.
- **"Java: A Beginner's Guide" by Herbert Schildt**
  A beginner-friendly guide to Java, with sections on GUI and event-driven programming.
- **"Head First Java" by Kathy Sierra and Bert Bates**
  A fun and engaging way to learn Java, including GUI programming basics.