

PASSWORD GENERATOR



A PROJECT REPORT

Submitted by

SAGANA S (2303811710422134)

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 **SAGANA S (2303811710422134)** 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

SIGNATURE

Mrs.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

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

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 6/12/2024.

CGB1201-JAVA PROGRAMMING
Mr. M. S. SAGANA S, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Mr.R. KARUNHIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

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



SAGANA S

Place: Samayapuram

Date:6/12/2024

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 **MR. A. MALARMANNAN, M.E.**, 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

A password generator is a tool designed to create secure, random, and strong passwords, essential for protecting sensitive digital information and preventing unauthorized access to systems, accounts, and data. It uses algorithms to generate passwords that are difficult to guess or crack, often incorporating a mix of uppercase and lowercase letters, numbers, and special characters. This randomness significantly increases the complexity and security of passwords compared to user-created ones, which are often predictable.

Password generators can be standalone software, integrated features in password management systems, or online services. Many modern generators allow customization, enabling users to define parameters such as password length, inclusion or exclusion of certain characters, and specific security requirements. Advanced versions can generate passwords that comply with organizational policies, such as adherence to regulatory standards or compatibility with multi-factor authentication systems.

The use of password generators is increasingly critical in the digital age, where weak or reused passwords are a leading cause of cyber breaches. By generating unique passwords for each account or system, these tools minimize risks such as credential stuffing, brute force attacks, and phishing exploits. They are particularly valuable in environments where users need to manage multiple accounts or access high-security systems. In combination with password managers, password generators contribute to robust cybersecurity practices, fostering safe and

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
A password generator is a tool designed to create strong, secure, and unique passwords to enhance cybersecurity. It employs algorithms to produce passwords that are difficult to guess or crack, often combining random characters, numbers, and symbols. Advanced generators may include user-defined criteria, such as length, complexity, or exclusions, to balance security and usability. These tools are essential for protecting sensitive information in personal and professional settings, reducing the risk of unauthorized access and data breaches.	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 NO.	TITLE	PAGE NO.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	3
	2.1 Proposed Work	3
	2.2 Block Diagram	4
3	MODULE DESCRIPTION	5
	3.1 User Input And Customization Module	5
	3.2 Password Generation Engine	6
	3.3 Randomization And Shuffling Module	6
	3.4 Randomization And Shuffling Module	7
	3.5 Password Output And Validation Module	7
4	CONCLUSION & FUTURE SCOPE	8
	4.1 Conclusion	8
	4.2 Future Scope	8
	APPENDIX A (SOURCE CODE)	9
	APPENDIX B (SCREENSHOT)	14
	REFERENCES	15

CHAPTER 1

INTRODUCTION

1.1 OBJECTIVE

In today's digital landscape, where cyber threats and security breaches are becoming increasingly sophisticated, password security remains one of the most critical defenses against unauthorized access. Despite this, many individuals continue to rely on weak passwords, simple patterns, or the same password across multiple accounts, leaving their digital identities exposed to potential attackers. The Custom Password Generator App seeks to tackle this issue head-on by offering a robust, intuitive, and secure solution for creating strong, customized passwords.

The goal of this project is to empower users to generate passwords that are not only strong but tailored to meet their specific needs. By providing flexibility in the number of uppercase letters, lowercase letters, numbers, and symbols, the app allows users to craft passwords that suit the complexity requirements of different online services, while also ensuring each password is unique and highly secure. Unlike traditional password managers, which often rely on pre-set templates, this tool emphasizes user control and customization, offering them the ability to create a password that aligns with their security standards.

A key feature of the app is its use of SecureRandom, which ensures that the passwords generated are cryptographically secure and resistant to common brute-force or dictionary-based attack methods.

1.2 OVERVIEW

The Custom Password Generator App is a dynamic and intuitive tool designed to simplify the process of creating strong, secure, and personalized passwords. With the rapid increase in online services and the subsequent need for robust digital security, this Java-based desktop application offers a straightforward solution for individuals and businesses looking to strengthen their password protection.

This application allows users to generate unique passwords tailored to their specific requirements by customizing the number of uppercase letters, lowercase letters, numbers, and special characters. Unlike traditional password managers, which automatically generate random passwords without user input, the Custom Password Generator provides a hands-on approach, allowing users to actively define the structure of their password. This flexibility ensures that the passwords meet the varying complexity rules set by different websites or applications, making it highly adaptable to individual needs.

1.3 JAVA PROGRAMMING CONCEPTS

The Custom Password Generator App leverages several key concepts of Java programming, ensuring not only functionality but also security, efficiency, and user interactivity. Here's a deeper dive into some of the core Java programming concepts applied throughout the development of this app:

- **Encapsulation:** The app encapsulates user inputs and password generation logic within the CustomPasswordGeneratorApp class. The user interface components such as text fields, labels, and buttons are instantiated as objects and interact through defined methods, keeping the implementation neat and isolated from other components.
- **Inheritance:** In this project, the app extends the Frame class to create a custom window frame for the graphical user interface (GUI). This inheritance allows the app to inherit properties and methods from Frame while customizing it to suit the app's requirements.
- **Polymorphism:** By implementing ActionListener and WindowListener interfaces, the app handles multiple types of user interactions (such as button clicks and window closing events) with polymorphism. This allows the app to respond dynamically based on the event type.
- **Abstraction:** The complexity of password generation (randomness, secure shuffling, and character pool creation) is abstracted into reusable methods, which simplifies the app's main workflow and enhances readability.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The Custom Password Generator App can be significantly improved with several enhancements to its functionality, user experience, and security features.

1. Enhanced User Interface (UI) and User Experience (UX)

- Transition to JavaFX for a more modern, visually appealing interface with features like smoother animations, interactive form validation, and a dark mode option for improved accessibility.
- Password Strength Indicator to evaluate and display the complexity of the generated password in real-time.

2. Multi-Language Support

- Internationalization (i18n) and Localization (L10n) to offer the app in multiple languages, catering to a global user base with region-specific formatting.

3. Password Storage and Encryption

- Local Password Storage with secure encryption for storing generated passwords.
- Master Password Protection to ensure only authorized users can access saved passwords.

4. Enhanced Security Features

- Customizable Password Rules to meet specific security standards for different platforms.
- Two-Factor Authentication (2FA) support for generating OTPs as an added layer of security.
- Password Expiration feature to remind users to periodically update passwords.

5. Cloud Integration and Cross-Platform Support

- Cloud Sync to allow password access across multiple devices.
- Cross-Platform Compatibility for mobile (iOS/Android) or web applications, providing users with flexibility on the go.

2.2 BLOCK DIAGRAM

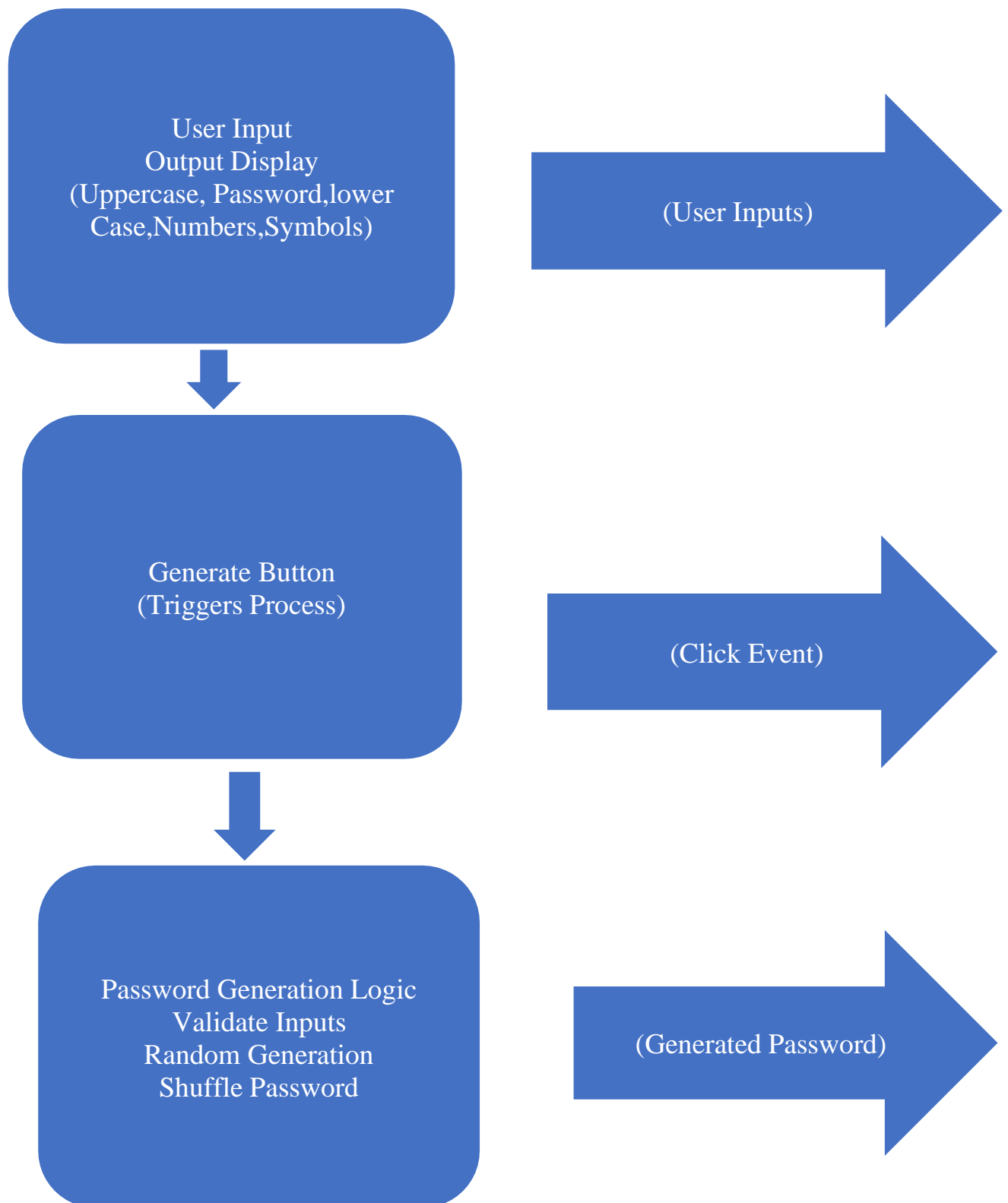


Figure 2.2.1 CustomPasswordGenerator

CHAPTER 3

MODULE DESCRIPTION

3.1 USER INPUT AND CUSTOMIZATION MODULE

This module allows users to specify the number of characters in each category for their password (uppercase letters, lowercase letters, numbers, and symbols). The app provides input fields where users can enter their desired quantities for each character type. It ensures flexibility, letting users generate passwords with the exact mix of characters that meet their security needs. The input validation prevents invalid entries, ensuring the app only generates passwords when the user inputs correct values.

INTERACTIVE SLIDERS FOR CHARACTER COUNTS:

- Replace or supplement the text fields with sliders for selecting the number of uppercase, lowercase, numbers, and symbols.
- Add real-time feedback to display the total character count dynamically.

PASSWORD STRENGTH INDICATOR:

- Show a strength indicator (e.g., "Weak", "Moderate", "Strong") based on the user's input.
- Consider factors like total length, diversity of character types, etc.

CUSTOMIZATION OPTIONS:

- Allow users to define their own character pool (e.g., specific symbols or additional characters).
- Include a toggle for excluding ambiguous characters like l, I, O, 0.

PREVIEW MODE:

- Show a live preview of the password as the inputs are adjusted.

CHARACTER DISTRIBUTION VISUALIZATION:

- Add a pie chart or bar graph to visually represent the percentage of uppercase, lowercase, numbers, and symbols in the generated password.

THEMES FOR UI:

- Offer multiple color themes or dark mode support for the application.

3.2 PASSWORD GENERATION ENGINE

At the core of this application lies the password generation engine. This engine utilizes a `SecureRandom` instance to randomly select characters from predefined pools of uppercase letters, lowercase letters, numbers, and symbols. This ensures that the generated password is both unpredictable and secure. By selecting characters randomly, the engine ensures that the passwords meet cryptographic standards, making them strong and difficult to guess or crack. Once the characters are randomly selected from the pools, the password generation logic proceeds to shuffle the characters to ensure that the final password does not have any predictable patterns, such as all uppercase letters followed by all numbers. This shuffling step adds an extra layer of randomness to the generated password, ensuring that it is more secure.

The password generation logic is separated into its own module to make the process easier to understand and modify. If the password generation logic needs to be updated, such as introducing a new character pool or modifying how characters are selected, these changes can be made here without disrupting the rest of the program. This makes the application more flexible and adaptable to future requirements, such as adding support for additional character types or changing how the password is shuffled.

3.3 RANDOMIZATION AND SHUFFLING MODULE

To further enhance password security, the app includes a shuffling module. After selecting the characters for the password, this module randomly shuffles the order of characters to eliminate any predictable patterns. This adds an extra layer of randomness, making the final output a more secure password, ensuring that even if certain patterns are used, their positioning is entirely random. This validation is important because it helps to ensure that the password generation logic only receives valid and meaningful data, thereby preventing errors and maintaining the overall integrity of the application. If the validation fails, the application will display an error message, guiding the user to correct their input.

Having input validation as a separate module allows for easy expansion and modification. For example, if the requirements for password generation change, and new validation rules are introduced, it can be done in this module without

affecting the rest of the program. This separation of concerns enhances the maintainability of the program.

3.4 GRAPHICAL USER INTERFACE (GUI) MODULE

The application is equipped with a simple AWT-based GUI (Abstract Window Toolkit) that provides a user-friendly interface for interaction. Users can easily input the number of characters for each category, click a button to generate the password, and view the result in a display field. This GUI ensures that the app remains intuitive and easy to use for users of all technical backgrounds. The interface is designed to be lightweight and straightforward, making it simple to navigate. In this module, the program starts by creating an instance of the User Interface Module, displaying the UI to the user, and setting the application's visibility. Then, it initializes the event handling module, which listens for user actions and processes them accordingly. The main module ties everything together by calling other modules as needed and ensuring the application runs correctly.

By separating the main application logic from other components, this module focuses on high-level coordination and orchestration. This allows for easier future changes, as the logic for integrating the various components of the application is kept in one place. Additionally, this modularity makes the program more adaptable to future expansions, such as adding new features or integrating the password generator into other applications.

3.5 PASSWORD OUTPUT AND VALIDATION MODULE

Password Output and Validation Module Once the password is generated, the app displays the result in a non-editable text field, ensuring that the user can see their newly created password clearly. This module also handles error messages in case the user inputs invalid data, such as negative numbers or non-numeric values. If the inputs are incorrect, the app provides a message indicating the error. Additionally, the module validates the character counts to ensure they are non-negative and that the total number of characters is greater than zero before proceeding with the password generation.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Custom Password Generator App serves as a robust and secure tool for generating strong, customizable passwords. It allows users to define the complexity of their passwords while ensuring that the generated passwords are secure and random. By offering a simple GUI, users can easily interact with the app to create secure passwords suited to their needs. The integration of secure randomization techniques and shuffling ensures that the passwords generated are resistant to brute force and other forms of attacks.

4.2 FUTURE SCOPE

While the current implementation provides basic functionality, there are several enhancements that could make the Custom Password Generator App even more useful:

1. **Multi-Language Support:** Future versions could include support for different languages, expanding its global usability.
2. **Password Storage Integration:** The app could incorporate local password storage or integrate with cloud-based services, allowing users to store and manage their passwords securely.
3. **Advanced Security Features:** Adding features like two-factor authentication (2FA) code generation or password expiration reminders could further enhance the security offerings of the app.
4. **Mobile and Web Versions:** The app could be adapted for mobile devices (Android and iOS) or as a web-based application, broadening accessibility and allowing users to generate passwords on the go.
5. **Password Strength Analyzer:** A feature that evaluates the strength of the generated password based on various factors such as length, character diversity, and randomness could provide real-time feedback to help users generate stronger passwords.

APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.security.SecureRandom;

class CustomPasswordGeneratorApp extends Frame {
    private TextField upperField, lowerField, numberField, symbolField,
passwordField;

    public CustomPasswordGeneratorApp() {
        // Set up the frame
        setTitle("Custom Password Generator");
        setSize(500, 300);
        setLayout(new FlowLayout());
        setResizable(false);
        setBackground(new Color(255, 228, 225)); // Light pink background

        // Add input fields and labels for character counts
        add(new Label("Uppercase Letters:"));
        upperField = new TextField(5);
        add(upperField);

        add(new Label("Lowercase Letters:"));
        lowerField = new TextField(5);
        add(lowerField);
```

```

add(new Label("Numbers:"));
numberField = new TextField(5);
add(numberField);

add(new Label("Symbols:"));
symbolField = new TextField(5);
add(symbolField);

// Add a button to generate the password
Button generateButton = new Button("Generate Password");
generateButton.setBackground(new Color(173, 216, 230)); // Light blue button
generateButton.setForeground(Color.BLACK); // Black text
add(generateButton);

// Add a text field to display the generated password
add(new Label("Generated Password:"));
passwordField = new TextField(30);
passwordField.setEditable(false);
add(passwordField);

// Add an action listener to the button
generateButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            int upper = Integer.parseInt(upperField.getText());
            int lower = Integer.parseInt(lowerField.getText());
            int numbers = Integer.parseInt(numberField.getText());
            int symbols = Integer.parseInt(symbolField.getText());

```

```

        String password = generateCustomPassword(upper, lower, numbers,
symbols);
        passwordField.setText(password);
    } catch (NumberFormatException ex) {
        passwordField.setText("Invalid input! Enter numbers only.");
    } catch (IllegalArgumentException ex) {
        passwordField.setText(ex.getMessage());
    }
}
});

```

```

// Add a window listener to close the application
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        dispose();
    }
});
}

```

```

public static void main(String[] args) {
    CustomPasswordGeneratorApp app = new CustomPasswordGeneratorApp();
    app.setVisible(true);
}

```

```

public static String generateCustomPassword(int upper, int lower, int numbers,
int symbols) {
    // Validate the inputs
    if (upper < 0 || lower < 0 || numbers < 0 || symbols < 0) {
        throw new IllegalArgumentException("Counts must not be negative.");
    }
}

```

```

    }
    int total = upper + lower + numbers + symbols;
    if (total == 0) {
        throw new IllegalArgumentException("Total characters must be greater than
zero.");
    }

    // Define character pools
    String upperCase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String lowerCase = "abcdefghijklmnopqrstuvwxyz";
    String numPool = "0123456789";
    String symbolPool = "!@#$%^&*()-_+=<>?";

    SecureRandom random = new SecureRandom();
    StringBuilder password = new StringBuilder();

    // Generate specified counts of characters
    for (int i = 0; i < upper; i++) {
        password.append(upperCase.charAt(random.nextInt(upperCase.length())));
    }
    for (int i = 0; i < lower; i++) {
        password.append(lowerCase.charAt(random.nextInt(lowerCase.length())));
    }
    for (int i = 0; i < numbers; i++) {
        password.append(numPool.charAt(random.nextInt(numPool.length())));
    }
    for (int i = 0; i < symbols; i++) {
        password.append(symbolPool.charAt(random.nextInt(symbolPool.length())));
    }

```

```

    }

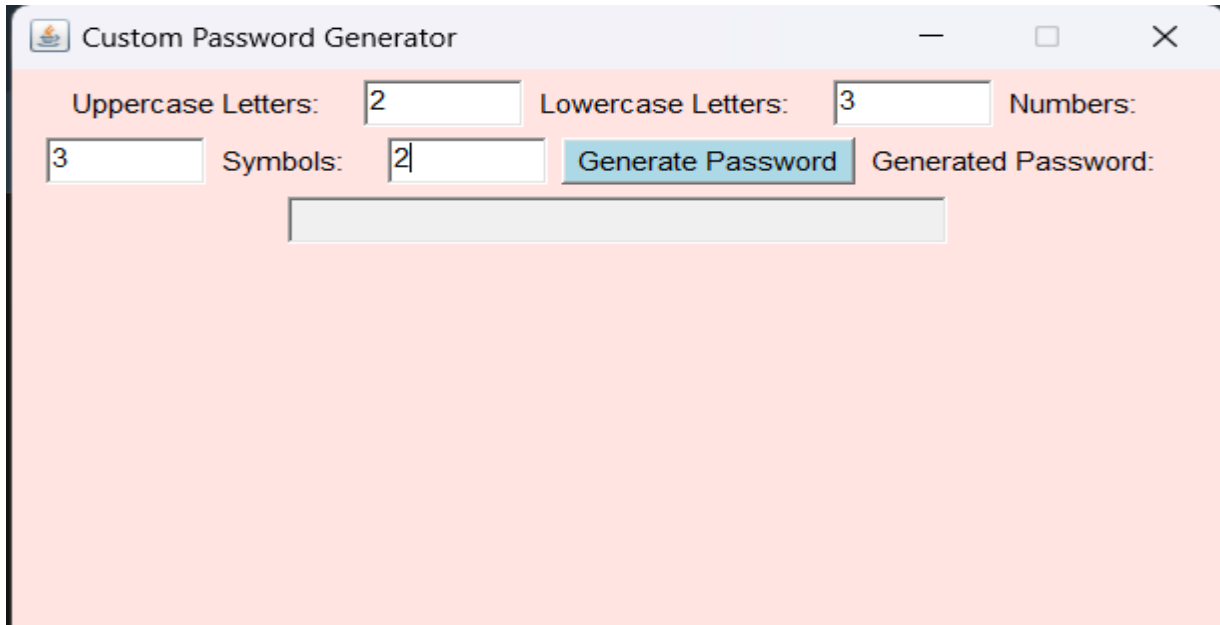
    // Shuffle the password to ensure randomness
    return shuffleString(password.toString(), random);
}

public static String shuffleString(String input, SecureRandom random) {
    char[] characters = input.toCharArray();
    for (int i = characters.length - 1; i > 0; i--) {
        int j = random.nextInt(i + 1);
        // Swap characters[i] with characters[j]
        char temp = characters[i];
        characters[i] = characters[j];
        characters[j] = temp;
    }
    return new String(characters);
}
}

```

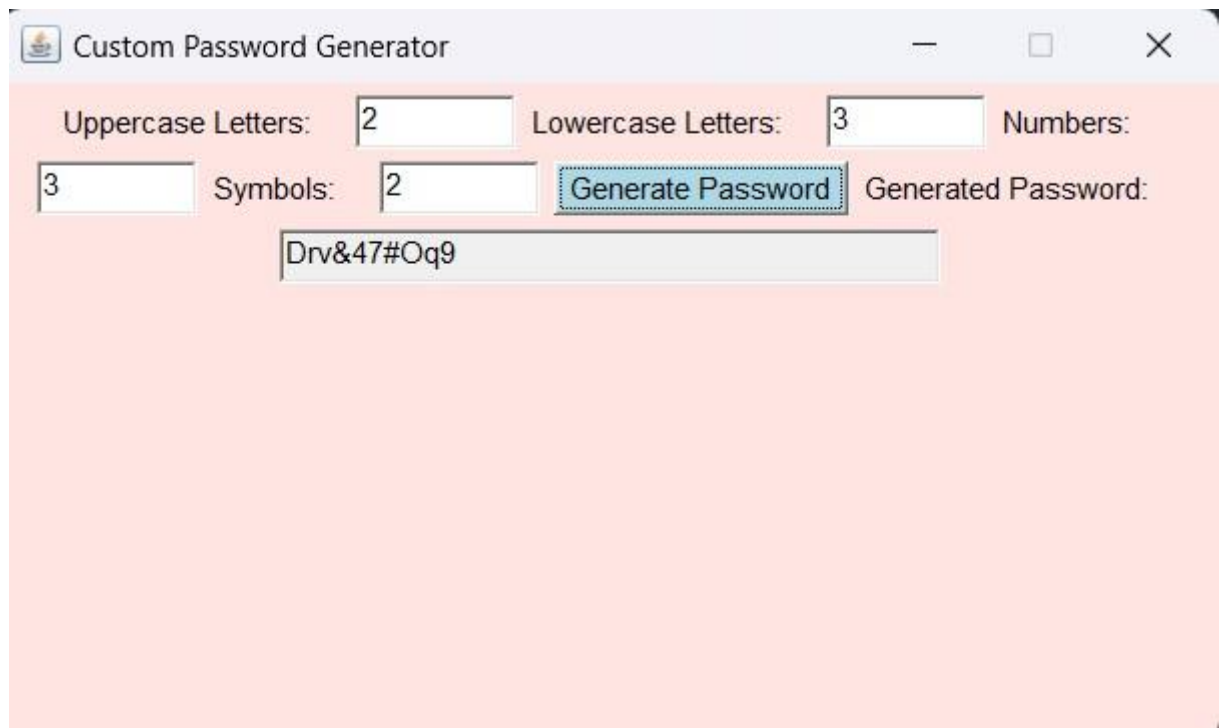
APPENDIX B

(SCREENSHOTS)



Custom Password Generator

Uppercase Letters: Lowercase Letters: Numbers: Symbols: Generated Password:



Custom Password Generator

Uppercase Letters: Lowercase Letters: Numbers: Symbols: Generated Password:

REFERENCES

1. Sharma, Dharmendra Kumar, and Vineet Tiwari. "Small and medium range wireless electronic notice Bluetooth and ZigBee." IEEE, 2015. Digital Notice Board (A Mobile Application)
2. Khandagale, A.S., et al. "Design E - Notice Board using Effective web Technologies for Educational Organizations." International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), vol. 9, no. 3, 2020.35_Design_JP (1).PDF
3. "Java: A Beginner's Guide" by Herbert Schildt ,8th Edition, 2018. Covers basic Java programming and includes chapters on GUI development with AWT and Swing.
4. "Head First Java" by Kathy Sierra and Bert Bates ,2nd Edition, 2005.A beginner-friendly book that of object-oriented programming and GUI development in Java.