PASSWORD GENERATOR

A PROJECT REPORT

Submitted by

VAISHALI A (2303811724322118)

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

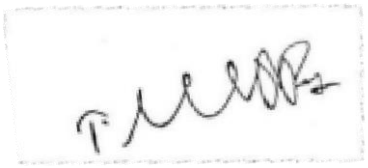
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"PASSWORD GENERATOR"** is the bonafide work of **VAISHALI A (2303811724322118)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

Dr. T. AVUDAIAPPAN M.E., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy -621 112.

Signature

Mrs. S. GEETHA M.E.,

SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Engineering,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24

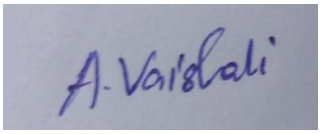
INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on "**PASSWORD GENERATOR**" is the result of original work done by me and best of my knowledge, similar work has not been submitted to "**ANNA UNIVERSITY CHENNAI**" for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING.**

Signature



VAISHALI A

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

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

I extend our sincere acknowledgment and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.,** for having provided 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 our project and offering an adequate duration to complete it.

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

I thank **Dr. T. AVUDAIAPPAN, M.E.,Ph.D**., Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs. S. GEETHA, M.E**. Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project

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

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conductive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics.

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

In today's digital era, online security is of utmost importance, and a strong password is a critical component of maintaining this security. This project focuses on the design and development of a Password Generator application in Java, aimed at assisting users in creating robust and secure passwords that are resistant to hacking attempts. The application provides an intuitive interface for users to define specific criteria for password generation, including length, character types (uppercase, lowercase, numbers, and special symbols), and complexity levels. It employs a combination of randomization algorithms and security best practices to ensure the generated passwords are both unique and unpredictable. Through this application, users can enhance their online security and mitigate risks associated with weak or reused passwords. The final product is a lightweight, efficient, and user-friendly tool, showcasing the potential of Java for secure application development.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	**vii**
1	**INTRODUCTION**	**B**
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
2	**PROJECT METHODOLOGY**	
	2.1 PROPOSED WORK	3
	2.2 BLOCK DIAGRAM	4
3	**JAVA PROGRAMMING CONCEPTS**	
	3.1 OBJECT-ORIENTED PROGRAMMING	5
	3.2 CONTROL STRUCTURES	5
4	**MODULE DESCRIPTION**	
	4.1 USER MODULE	6
	4.2 CHARACTER POOL MODULE	6
	4.3 PASSWORD GENERATOR MODULE	6
	4.4 OUTPUT MODULE	6
	4.5 VALIDATION MODULE	6
5	**CONCLUSION**	7
	REFERENCES	8
	APPENDICES	
	Appendix A – Source code	9
	Appendix B – Screen shots	13

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

A **Password Generator** is a program designed to create secure, random, and strong passwords. In today's digital world, where security breaches are common, using strong passwords is essential to protect personal and sensitive information. Building a password generator in Java is a great way to learn about randomness, string manipulation, and security practices.

The core functionality of a password generator includes:

- **Randomization**: Generating characters in a way that avoids patterns and ensures randomness.
- **Customization**: Allowing users to specify requirements such as length, inclusion of uppercase letters, numbers, symbols, and more.
- **Security**: Using techniques that produce unpredictable results to enhance password strength.

2. OBJECTIVE

- **Enhance Security**: Enable users to generate strong and secure passwords to reduce the risk of unauthorized Access.
- **Customization**: Allow users to define specific password parameters such as length, inclusion of special, characters, numbers, uppercase, and lowercase letters.
- **Ease of Use:** Provide a simple and user-friendly interface for generating passwords.

- **Standards Compliance**: Ensure the generated passwords meet modern cybersecurity standards like NIST guideliness.
- **Efficiency**: Develop a lightweight application with fast password generation capabilities.

CHAPTER 2 PROJECT METHODOLOGY

2.1 PROPOSED WORK

- **Requirement Analysis**

Identify the features: password length, character types (uppercase, lowercase, numbers, symbols), and randomness.

Research security guidelines (e.g., NIST standards).

Understand user needs for customization and ease of use.

- **System Design**

Plan the application structure with clear modules (input handling, password generation, output display).

Use diagrams (flowcharts or class diagrams) to map the workflow.

Choose secure algorithms for random password generation.

- **Implementation**

Set up the project in a Java IDE (e.g., IntelliJ, Eclipse).

Code the modules:

Character pool setup.

Password generation logic.

User input validation.

Optional: Add a graphical interface (GUI) using JavaFX or Swing.

- **Testing**

Test individual components (e.g., random password generation).

Check if modules work together smoothly.

•Handle edge cases like invalid inputs or extreme password lengths.

•Ensure passwords meet security standards.

5. Deployment

•Package the application for users as a .jar or .exe file.

•Provide simple installation and usage instructions.

6. Maintenance and Updates

•Collect feedback for improvements or new features.

•Update the app to fix bugs or enhance security.

2.2 BLOCK DIAGRAM

CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 OBJECT-ORIENTED PROGRAMMING (OOP):

Encapsulation: Organizing the password generator logic within dedicated classes.

Abstraction: Hiding the implementation details of the password generation algorithm.

Inheritance and Polymorphism (if extending functionalities in future versions).

- **Randomization**:

Use of the Secure Random class for generating cryptographically secure random values.

- **String Manipulation**:

Use of StringBuilder for constructing passwords efficiently. Handling character sets using arrays and lists for inclusion/exclusion.

3.2 CONTROL STRUCTURES:

Loops to iterate through user-defined password length.

Conditional Statements to include specific character types based on user input.

- **Collections Framework:**

Use of Array List or HashSet for managing and shuffling character pools.

- **Error Handling:**

Exception handling to validate user inputs and handle edge cases gracefully.

CHAPTER 4

MODULE DESCRIPTION

4.1 USER MODULE

Lets users set password length and choose character types (uppercase, lowercase, numbers, symbols).

Validates inputs to meet security rules.

4.2 CHARACTER POOL MODULE

Creates a pool of characters based on user choices.

Combines selected character types into one list.

4.3 PASSWORD GENERATOR MODULE

Randomly selects characters from the pool to create a password.

Ensures randomness and meets user-defined criteria.

4.4 OUTPUT MODULE

Displays the generated password. Allows

copying or saving the password.

4.5 VALIDATION MODULE

Ensures the password is secure and meets required rules.

CHAPTER 5

CONCLUSION

The **Password Generator** application effectively addresses the need for creating secure and strong passwords, enhancing online safety. By integrating user-defined criteria, randomization, and validation, it ensures that generated passwords meet modern security standards while remaining easy to use. The modular design simplifies maintenance and scalability, allowing for future enhancements, such as integrating password storage or advanced GUI features. Written in Java, the application leverages robust libraries like Secure Random to guarantee cryptographically secure password generation. This project demonstrates the practical application of Java programming concepts while contributing to the critical goal of improving cybersecurity.

REFERENCES:

- https://github.com/Baeldung/java-password-generator
- https://www.geeksforgeeks.org/generating-password-otp-java/
- https://www.tutorialspoint.com/java/index.htm

APPENDICES

APPENDIX A – SOURCE CODE

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

public class PasswordGeneratorAWT extends Frame implements ActionListener {
    Label labelLength, labelOptions, labelResult;
    TextField textLength, textResult;
    Checkbox checkboxUppercase, checkboxLowercase, checkboxNumbers, checkboxSymbols;
    Button generateButton, closeButton;
    SecureRandom random = new SecureRandom();

    public PasswordGeneratorAWT() {
        // Frame settings
        setTitle("Password Generator");
        setSize(500, 300);
        setLayout(new FlowLayout());
        setVisible(true);

        // Input fields and labels
        labelLength = new Label("Password Length:");
        textLength = new TextField(10);
        labelOptions = new Label("Include:");
        checkboxUppercase = new Checkbox("Uppercase Letters");
        checkboxLowercase = new Checkbox("Lowercase Letters");
        checkboxNumbers = new Checkbox("Numbers");
        checkboxSymbols = new Checkbox("Symbols");

        labelResult = new Label("Generated Password:");
```

```
    textResult = new TextField(30);
    textResult.setEditable(false);

    // Buttons
    generateButton = new Button("Generate"); closeButton
    = new Button("Close");

    // Adding components to Frame
    add(labelLength);  add(textLength);
    add(labelOptions);
    add(checkboxUppercase);
    add(checkboxLowercase);
    add(checkboxNumbers);
    add(checkboxSymbols);  add(labelResult);
    add(textResult);  add(generateButton);
    add(closeButton);

    // Action Listeners
    generateButton.addActionListener(this);
    closeButton.addActionListener(this);

    // Closing the window  addWindowListener(new
    WindowAdapter() {
        public void windowClosing(WindowEvent e) { dispose();
                        }
                    });
}

@Override
public void actionPerformed(ActionEvent e) {
```

```
if (e.getSource() == generateButton) {
    try {
        // Step 1: Get length input
        int length = Integer.parseInt(textLength.getText());

        // Step 2: Validate length
        if (length < 8) {
            textResult.setText("Length must be at least 8.");
            return;
        }

        // Step 3: Get character pool options
        boolean[] options = {
            checkboxUppercase.getState(),
            checkboxLowercase.getState(),
            checkboxNumbers.getState(),
            checkboxSymbols.getState()
        };

        ArrayList<Character> pool = createCharacterPool(options);

        if (pool.isEmpty()) {
            textResult.setText("Select at least one character type.");
            return;
        }

        // Step 4: Generate password
        String password = generatePassword(length, pool);

        // Step 5: Display password
        textResult.setText(password);

    } catch (NumberFormatException ex) {
        textResult.setText("Enter a valid number for length.");
```

```
        }
    } else if (e.getSource() == closeButton) { dispose();

// Module 1: Character Pool Creation
public ArrayList<Character> createCharacterPool(boolean[] options) { ArrayList<Character> pool =
    new ArrayList<>();
    if (options[0]) {
        for (char c = 'A'; c <= 'Z'; c++) pool.add(c);
    }
    if (options[1]) {
        for (char c = 'a'; c <= 'z'; c++) pool.add(c);
    }
    if (options[2]) {
        for (char c = '0'; c <= '9'; c++) pool.add(c);
    }
    if (options[3]) {
        String symbols = "!@#$%^&*()-_=+[]{}|;:'\",.<>?/`~"; for (char c :
        symbols.toCharArray()) pool.add(c);
    }
    return pool;
}

// Module 2: Password Generator
public String generatePassword(int length, ArrayList<Character> pool) { StringBuilder password =
    new StringBuilder(length);
    for (int i = 0; i < length; i++) {  password.append(pool.get(random.nextInt(pool.size())));
    }       return password.toString();
}

public static void main(String[] args) { new
    PasswordGeneratorAWT();
}
```

APPENDIX B - SCREENSHOTS