PROJECT REPORT

ON

# **PASSWORD GENERATOR AND MANAGER**

REPORT SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR

THE PBL OF FUNDAMENTAL OF PROGRAMMING IN

INFORMATION TECHNOLOGY DEPARTMENT

BY

|  |  |  |
| --- | --- | --- |
| NAME | ROLL NO. | PRN |
| Mandar Dakhorkar | 232034 | 22211540 |
| Chinmay Mhaske | 232037 | 22211431 |
| Jayesh Patil | 232046 | 22210862 |
| Shriraj Patil | 232052 | 22211449 |

## Class: S.Y Division: B Batch: B2

Batch Teacher : Mrs. Sonali Bhoite Ma’am /

Mr.Manohar Kodmelwar

INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | Contents | Page No. |
| 1 | ABSTRACT | 1 |
| 2 | INTRODUCTION AND THEORY | 2,3 |
| 3 | EXPERIMENTAL DETAILS  Algorithm  Flowchart  Program |  |
| 4 | RESULTS | 13,14 |
| 5 | ER DIAGRAM OF SYSTEM | 14,15 |
| 6 | REFERENCE | 15 |

**ABSTRACT**

The "SecuGen" project addresses the imperative need for robust password management in the digital age, presenting a Java-based solution with a sophisticated password generation engine. This feature-rich system allows users to customize password length and composition, incorporating uppercase and lowercase letters, digits, and special characters for enhanced security. The user-friendly Graphical User Interface (GUI) facilitates seamless interaction with password generation and management functionalities. SecuGen ensures secure password storage, enabling users to save passwords with personalized naming conventions. Developed in Java, it offers cross-platform compatibility and leverages Java's SecureRandom class for cryptographically secure randomization. The project emphasizes secure file handling and export capabilities, allowing users to efficiently manage and access generated passwords.

**INTRODUCTION**

The "Password Generator and Saver" project, developed in Java, stands as a sophisticated response to the pressing need for heightened cybersecurity in our digitally connected world. With an emphasis on password management, this project offers a holistic solution, combining cutting-edge technology and user-centric design. The endeavor recognizes the increasing vulnerability of digital identities amid the proliferation of online platforms and evolving cyber threats.

Driven by multifaceted objectives, the project excels in advanced password generation through intricate algorithms, providing users with customization options for tailoring password attributes. The intuitive Graphical User Interface (GUI) ensures accessibility for users of diverse technical backgrounds, emphasizing a visually appealing and user-friendly design. Additionally, the integration of robust password management, complete with encryption mechanisms, enhances data security. Leveraging the Java programming language, the project achieves cross-platform compatibility, ensuring seamless operation across various operating systems.

The scope of the Password Generator and Saver project extends beyond traditional password management boundaries. It strives to simplify password creation while instilling confidence in users regarding the secure storage and retrieval of their credentials. With a wide-ranging applicability, the project caters to both individual users and organizations with complex security requirements. In a world where digital interactions are integral, this project assumes paramount significance, contributing to the broader discourse on cybersecurity best practices. Ultimately, it empowers users to navigate the digital landscape with resilience and confidence against emerging threats..

**THEORY**

**Password Generation:**

The foundation of the SecuGen project lies in the robust algorithm employed for password generation. The primary goal is to create passwords that are not only complex but also resistant to brute-force attacks and other common hacking techniques. The algorithm considers several factors, including:

**Customization Options:**

Users can specify the desired length of the password.

The algorithm allows users to choose whether to include uppercase letters, lowercase letters, digits, and special characters in the generated password.

Cryptographically Secure Randomization:

The project leverages Java's SecureRandom class to ensure the generation of cryptographically secure random numbers.This enhances the unpredictability and randomness of the generated passwords, mitigating the risk of predictability-based attacks.

**Balanced Composition:**

The algorithm ensures a balanced distribution of character types to create passwords that are not skewed towards a specific set of characters.

This balanced composition contributes to the overall strength and resilience of the passwords.

**Password Management:**

While password generation is a critical aspect of cybersecurity, effective password management is equally vital. The SecuGen project incorporates a user-friendly Graphical User Interface (GUI) for seamless interaction. The password management component includes:

**Intuitive User Interface:**

The GUI is designed with simplicity and clarity to cater to users of all technical backgrounds.

Users can easily navigate the interface to generate passwords, view them, and save them securely.

**Secure Storage:**

The password manager allows users to securely store generated passwords.

Passwords can be saved to a file using secure file handling techniques, ensuring the confidentiality and integrity of stored data.

**Export and Flexibility:**

Users have the flexibility to export generated passwords to a file of their choice, facilitating easy access and integration with other tools.

**EXPERIMENTAL DETAILS**

**Objective:**

The primary objective of the experiment is to implement a Java-based Password Manager and Generator (SecuGen) that includes a robust password generation algorithm and an intuitive user interface for password management.

**Environment:**

Development Environment: Java SE Development Kit (JDK)

Integrated Development Environment (IDE): IntelliJ IDEA or Eclipse

Libraries: Java Swing for GUI components

**Algorithm for Password Generation:**

The password generation algorithm involves the following steps:

Accept user input for password length and composition preferences.

Utilize Java's SecureRandom class for cryptographically secure randomization.

Generate random characters based on user preferences (uppercase, lowercase, digits, special characters).

Assemble the characters to form the final password.

**Experimental Procedure:**

Compile and run the PasswordGenerator class.

Input password length and composition preferences.

Observe the generated password.

**Flowchart Description:**

Start: Initiates the password generation process.

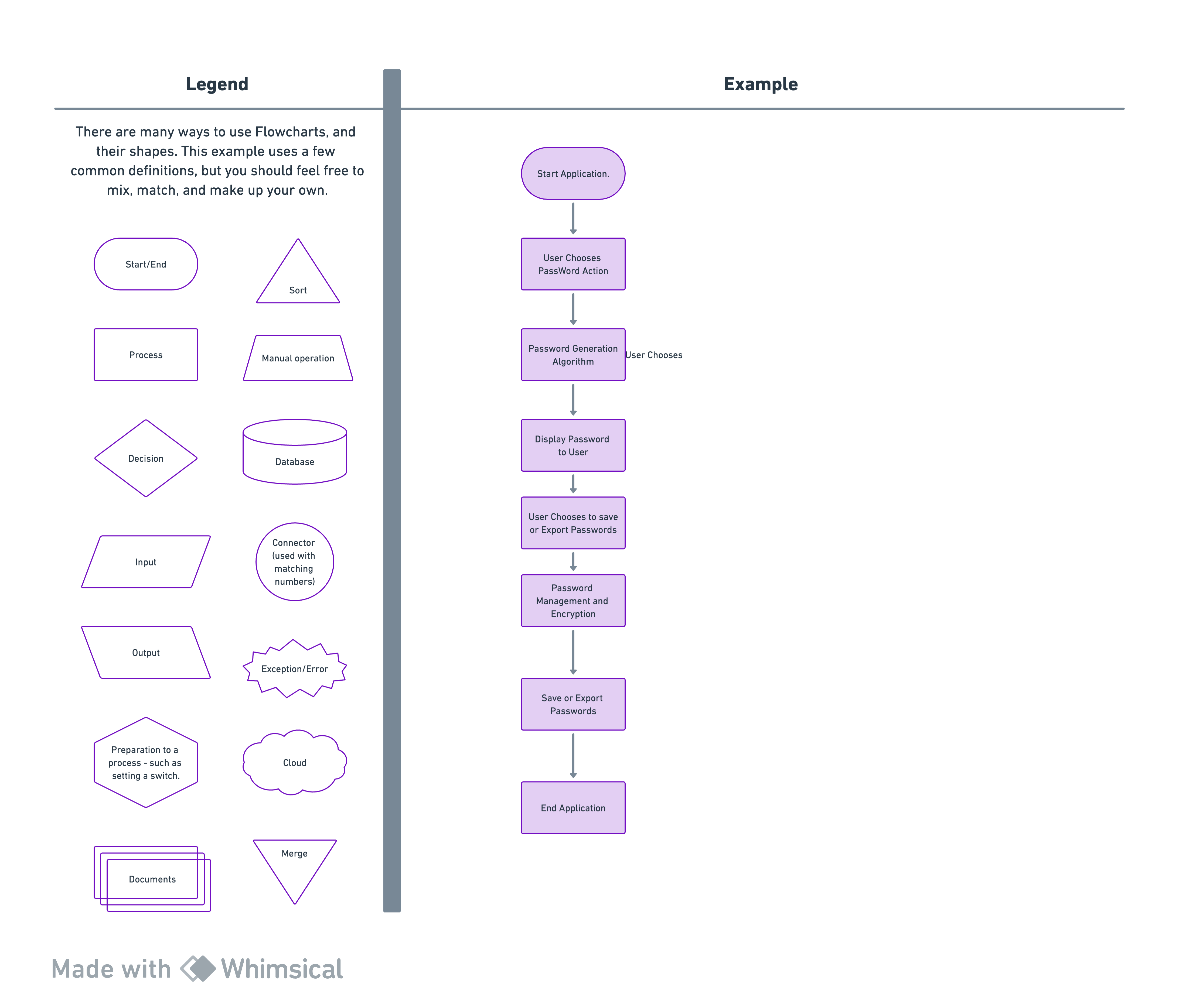
Input: Receives user preferences for password length and composition.

Randomization: Utilizes SecureRandom to generate random characters.

Assembling: Assembles characters based on user preferences.

End: Completes the password generation process.

**FlowChart :**



**PROGRAM**

**import javax.swing.*\**;**

**import java.awt.*\**;**

**import java.awt.event.ActionEvent;**

**import java.awt.event.ActionListener;**

**import java.io.BufferedWriter;**

**import java.io.File;**

**import java.io.FileWriter;**

**import java.io.IOException;**

**import java.security.SecureRandom;**

**public class PasswordGeneratorGUI extends JFrame {**

**private static final String UPPER\_CASE = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";**

**private static final String LOWER\_CASE = "abcdefghijklmnopqrstuvwxyz";**

**private static final String DIGITS = "0123456789";**

**private static final String SPECIAL\_CHARACTERS = "!@#$%^&\*()-\_=+[]{}|;:'\",.<>/?";**

**private JTextField lengthField;**

**private JTextArea passwordArea;**

**public PasswordGeneratorGUI() {**

**setTitle("Password Generator");**

**setSize(500, 300);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**JPanel mainPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10, 10));**

***// mainPanel.setLayout(new GridLayout(5, 3, 10, 10));***

**mainPanel.setBackground(Color.LIGHT\_GRAY);**

**JLabel instructionsLabel = new JLabel("Password Manager:");**

**instructionsLabel.setFont(new Font("Calibri", Font.PLAIN, 16));**

**instructionsLabel.setForeground(Color.BLUE);**

**JLabel lengthLabel = new JLabel("Password Length:");**

**lengthLabel.setFont(new Font("Calibri", Font.PLAIN, 16));**

**lengthLabel.setForeground(Color.BLACK);**

**lengthField = new JTextField("12");**

**lengthField.setFont(new Font("Calibri", Font.PLAIN, 16));**

**lengthField.setBackground(Color.WHITE);**

**JButton generateButton = new JButton("Generate Password");**

**generateButton.setFont(new Font("Calibri", Font.PLAIN, 16));**

**generateButton.setBackground(Color.GREEN);**

**generateButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**generatePassword();**

**}**

**});**

**JButton saveButton = new JButton("Save to File");**

**saveButton.setFont(new Font("Calibri", Font.PLAIN, 16));**

**saveButton.setBackground(Color.ORANGE);**

**saveButton.addActionListener(new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**saveToFile();**

**}**

**});**

**JLabel passwordLabel = new JLabel("Generated Password:");**

**passwordLabel.setFont(new Font("Calibri", Font.PLAIN, 16));**

**passwordLabel.setForeground(Color.BLACK);**

**passwordArea = new JTextArea();**

**passwordArea.setEditable(false);**

**passwordArea.setFont(new Font("Monospaced", Font.PLAIN, 16));**

**passwordArea.setBackground(Color.WHITE);**

**mainPanel.add(instructionsLabel);**

**mainPanel.add(new JLabel());**

**mainPanel.add(lengthLabel);**

**mainPanel.add(lengthField);**

**mainPanel.add(generateButton);**

**mainPanel.add(saveButton);**

**mainPanel.add(passwordLabel);**

**mainPanel.add(passwordArea);**

**add(mainPanel);**

**setVisible(true);**

**}**

**private void generatePassword() {**

**try {**

**int passwordLength = Integer.parseInt(lengthField.getText());**

**SecureRandom random = new SecureRandom();**

**StringBuilder password = new StringBuilder();**

**String allCharacters = UPPER\_CASE + LOWER\_CASE + DIGITS + SPECIAL\_CHARACTERS;**

**for (int i = 0; i < passwordLength; i++) {**

**int randomIndex = random.nextInt(allCharacters.length());**

**password.append(allCharacters.charAt(randomIndex));**

**}**

**passwordArea.setText(password.toString());**

**} catch (NumberFormatException ex) {**

**JOptionPane.showMessageDialog(*this*, "Please enter a valid number for password length.", "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**private void saveToFile() {**

**JFileChooser fileChooser = new JFileChooser();**

**fileChooser.setDialogTitle("Save Password to File");**

**int userSelection = fileChooser.showSaveDialog(*this*);**

**if (userSelection == JFileChooser.APPROVE\_OPTION) {**

**File fileToSave = fileChooser.getSelectedFile();**

**try (BufferedWriter writer = new BufferedWriter(new FileWriter(fileToSave))) {**

**writer.write(passwordArea.getText());**

**JOptionPane.showMessageDialog(*this*, "Password saved to: " + fileToSave.getAbsolutePath(), "Save Successful", JOptionPane.INFORMATION\_MESSAGE);**

**} catch (IOException e) {**

**JOptionPane.showMessageDialog(*this*, "Error saving password to file.", "Error", JOptionPane.ERROR\_MESSAGE);**

**}**

**}**

**}**

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

**SwingUtilities.invokeLater(new Runnable() {**

**@Override**

**public void run() {**

**new PasswordGeneratorGUI();**

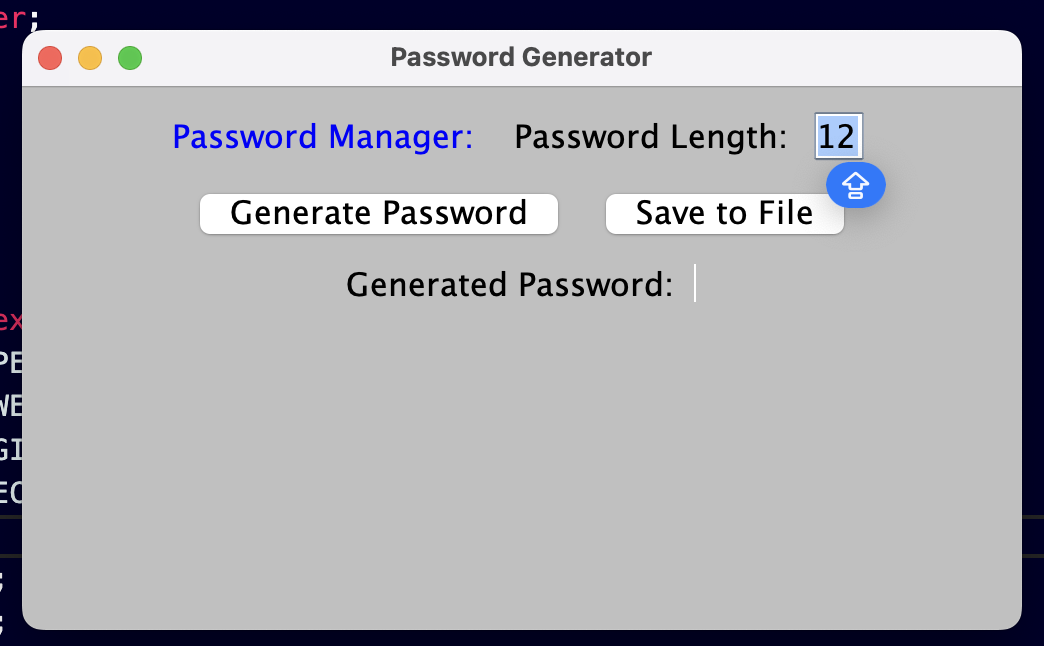
**}**

**});**

**}**

**}**

**RESULT**



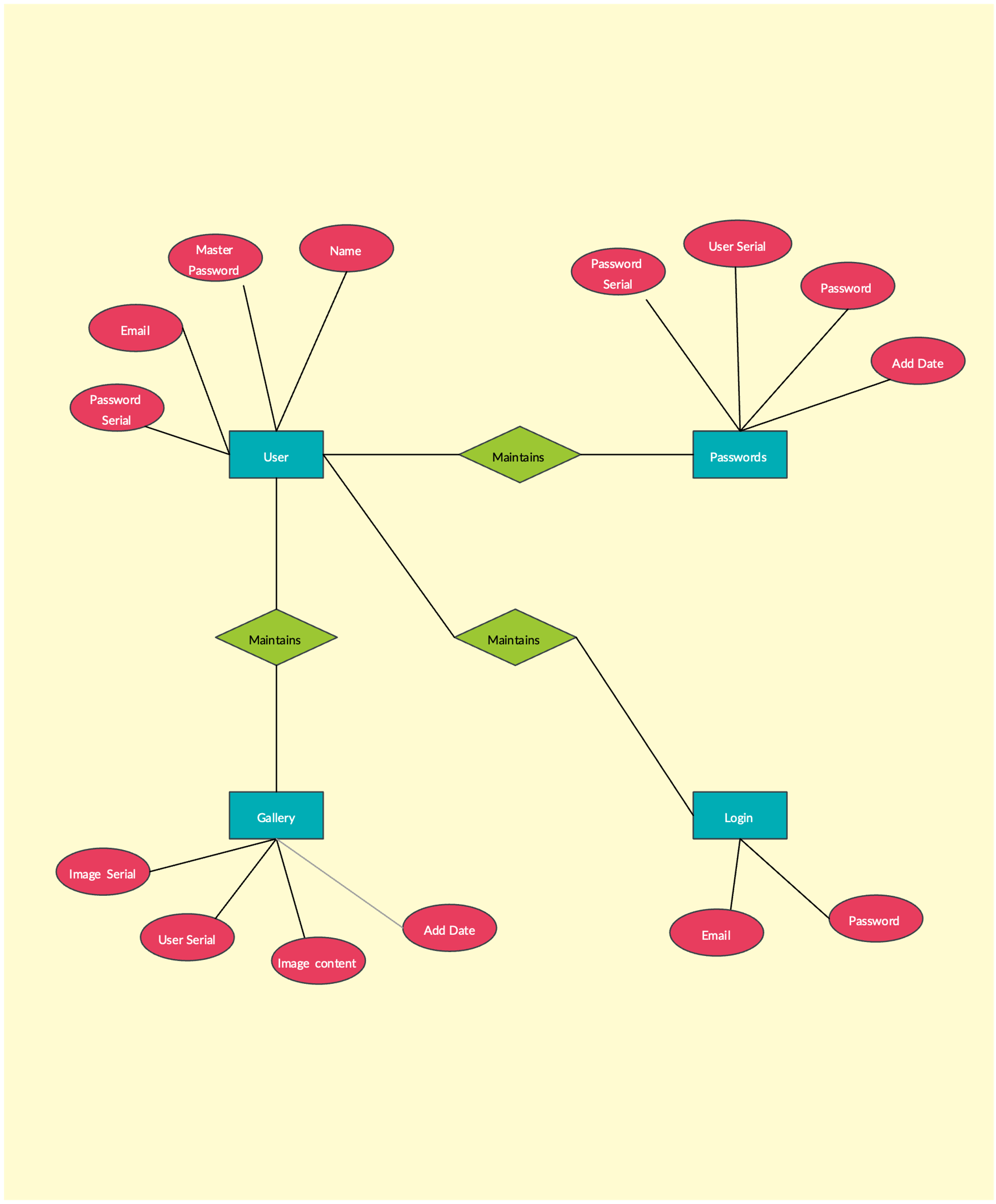
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**ER DIAGRAM OF SYSTEM**



**REFERENCES**

1. **Java Programming:**
   * Horstmann, C. S. (2019). Core Java Volume I - Fundamentals (11th ed.). Pearson.
   * Bloch, J. (2008). Effective Java (2nd ed.). Addison-Wesley.
2. **Java SecureRandom Class:**
   * [Java SecureRandom Class Documentation](https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/security/SecureRandom.html)
3. **Password Security Best Practices:**
   * [NIST Special Publication 800-63B](https://pages.nist.gov/800-63-3/sp800-63b.html) - Digital Identity Guidelines: Authentication and Lifecycle Management
   * FIPS PUB 202 - [National Institute of Standards and Technology](https://csrc.nist.gov/publications/detail/fips/202/final)
4. **Secure Password Hashing in Java:**
   * [Java Cryptography Architecture (JCA) Reference Guide](https://docs.oracle.com/en/java/javase/11/security/java-cryptography-architecture-jca-reference-guide.html)
5. **GUI Development in Java (for the password generator GUI):**
   * Eckel, B. (2004). Thinking in Java (4th ed.). Prentice Hall.
6. **Java Swing (for GUI development):**
   * [Java Swing Tutorial](https://docs.oracle.com/javase/tutorial/uiswing/)