COLLEGE OF ENGINEERING
Since 1982
An Autonomous Institute
(Affiliated to Visvesvaraya Technological University, Belagavi
Approved By AICTE, New Delhi,
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA and NAAC)

A MINI PROJECT REPORT
ON
PASSWORD AND NOTES MANAGER

Submitted in partial fulfillment of requirements for the award of 6th Sem degree,

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:
ARCHANA S.
1MJ20CS028

Under the Guidance of
Mr. Vinay Raj AS
Assistant Professor, Department of Computer science and Engineering,

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MVJ COLLEGE OF ENGINEERING
BANGALORE-67
ACADEMIC YEAR 2022-23

MVJ COLLEGE OF ENGINEERING

Near ITPB, Whitefield, Bangalore-67

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COLLEGE OF ENGINEERING
Since 1982
An Autonomous Institute
(Affiliated to Visvesvaraya Technological University, Belagavi
Approved By AICTE, New Delhi,
Recognized by UGC under 2(f) & 12(B)
Accredited by NBA and NAAC)

CERTIFICATE

This is to certify that the mini- project work, entitled "**PASSWORD AND NOTES MANAGER"** is a bonafide work carried out by ARCHANA S(1MJ20CS028) in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science & Engineering during the academic year 2022-23. It is certified that all the corrections/suggestions indicated for Internal Assessment have been incorporated in the Report. The mini project report has been approved as it satisfies the academic requirements.

Signature of Guide
Mr. Vinay Raj AS

Signature of the HOD
Dr.Kiran Babu

Name of examiners:
1.
2.

Signature with date:

MVJ COLLEGE OF ENGINEERING

Whitefield, Near ITPB, Bangalore-67

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I, **ARCHANA S** hereby declare that the entire work titled "**PASSWORD AND NOTES MANAGER"** embodied in this mini project report has been carried out by us during the6th semester of BE degree at MVJCE, Bangalore under the esteemed guidance of **Mr. Vinay Raj AS**, Assistant Prof, Dept. of CSE, MVJCE. The work embodied in this dissertation work is original and it has not been submitted in part of full for any other degree in any University.

ARCHANA S
(1MJ20CS028)

Place:

Date:

ABSTRACT

The “Password and notes manager" project application can be helpful in storing and managing Password and notes for an individual. It is a GUI-based project used with the swing library to organize all the elements that work under the password and notes manager. The Language used is Java. The java version is Java SE 18.0.2.1.The features that this project has are Generate a random Password of a length greater than 4,then Store and retrieve passwords, then delete a password, and then Encrypt a plain text with a secret key and decrypt the encrypted text with the secret key, And add a note and view the note added. In this project we have used an Array list to store the notes and a HashMap structure to store the name and password associated with the name. For encryption, we have used the PBEWithMD5AndDES algorithm. Using this project we can generate passwords and encrypt means we can change the given password into cipher text, decrypt the text means the cipher text can be converted into it’s original text and also we can store, search, delete the passwords and then adding and getting the note everything we can do.

AOWLEDGEMENT

With gratitude I acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

I am thankful to the **Management of MVJ College of Engineering Bangalore** for their continuous support and encouragement for carrying out the mini-project work

I am thankful to our **Dr. Mahabaleswarappa,** Principal, MVJCE, Bengaluru for being a constant inspiration and providing all the facilities that was needed throughout the mini- project work.

I like to express our gratitude to our **Dr. M. Brindha**, Vice Principal, MVJCE, Bengaluru, for constant encouragement throughout the course.

I also like to express our sincere gratitude to our **Dr. M A Lourdu Antony Raj** , Registrar and Controller of Examinations, MVJCE, Bengaluru, for persistent guidance.

I am thankful to our **Dr. Kiran Babu ,** HOD of Computer Science Engineering, MVJCE, Bengaluru, for being a constant support and providing all the facilities that was needed throughout the mini-project work.

I consider it as a privilege and honor to express our sincere gratitude to our guide **Mr. Vinay Raj AS, Assistant Professor,** Dept. of CSE, MVJCE, for his encouragement that has been a constant source of motivation to us for successful completion of our project.

It's also a great pleasure to express our deepest gratitude to all the other faculty members of our department for their cooperation and constructive criticism offered, which helped us a lot during our mini-project work.

Finally, I would like to thank all our family members and friends whose encouragement and support was invaluable.

Thanking you

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

A password manager is a software application designed to store and manage online credentials. Usually, these passwords are stored in an encrypted database and locked behind a master password. Read below all about Password Managers and secure your devices with our virus and malware protection here. The purpose of a password manager is to safely store online credentials, help you log in into any account automatically and generate strong and unique passwords. A master password is used to lock down an encrypted vault where these passwords are kept.

In the digital age, our lives revolve around the use of passwords and notes for various online and offline activities. Remembering multiple passwords and keeping track of important notes can be quite tedious and overwhelming. This is where a Password and Notes Manager can come in handy. Our Password and Notes Manager project aims to provide a secure and efficient way to manage passwords and notes in one place. Users can store their passwords and notes, access them easily, and keep them organized while maintaining the highest level of security. In this project, we will be utilizing advanced encryption techniques and secure authentication methods to ensure the protection of sensitive user information. With this Password and Notes Manager, users will have peace of mind knowing that their data is secure and accessible with just a few clicks.

CHAPTER 2

LITERATURE SURVEY

1.The Quest to Replace Passwords

Published in : 2012 IEEE Symposium on Security and Privacy.

Authors : Bonneau, J. et al.

2. Password Authentication from a Human Factors Perspective

Published in : Oct. 2009

Authors : Hoonakker, P., N. Bornoe, and C. Pascale

3. A Large-Scale Study of Web Pass-word Habits

Published in : 2007

Authors : Florencio, D. and C. Herley.

4. The Password Life Cycle

Published in : 10th Symposium On Usable Privacy and Security (SOUPS 2014).

Authors : Stobert, E. and R. Biddle.

5. A Convenient Method for Securely Managing Passwords

Published in : 2005

Authors : Halderman, J., B. Waters, and E. W. Felten.

A literature survey for the project "Password and Notes Manager in Java" would involve researching existing literature, studies, and articles related to password and notes management systems. The goal would be to understand the current state-of-the-art solutions and technologies related to password and notes management systems. This would include examining different types of authentication systems, encryption methods, and secure storage options that could be used in the development of the project. The literature survey would also involve looking at other existing password and notes management systems to identify potential features and functionalities that could be useful in the development of the project. The findings of the literature survey would be used to inform the design and development of the project, ensuring that it incorporates the latest technologies and best practices in password and notes management.

The first concern we will address is the use of the computer memory as discussed in [1]. This paper discusses the benefits of using TPM over memory to store passwords. According to Howtogeek.com, “TPM stands for Trusted Platform Module and is a very helpful tool for encryption. It is basically a chip on the motherboard that helps in encryption without the requirement for extremely long passwords and makes it tamper-resistant. The TPM generates encryption keys, keeping a part of the key to itself”. So, a section of the key is stored in the TPM itself instead of just on the disk, if you’re using encryption with a TPM. This means an attacker cannot remove the drive and attempt to access its files on any other computer. An attacker can’t remove the chip and place it on another motherboard, or tamper with it to attempt to bypass the encryption because the chip provides hardware-based authentication and tamper detection.

CHAPTER 3

PROBLEM ANALYSIS

There are the unauthorized persons who will access our passwords so here this project helps like , The role of password management comes in handy there. Password management is a set of principles and best practices to be followed by users while storing and managing passwords in an efficient manner to secure passwords as much as they can to prevent unauthorized access. Passwords are important when it comes to privacy, online security, and protecting your data. Enter the password manager: a tool that stores one strong master password that gives you easy access to all of your accounts while helping to keep cybercriminals at bay.

The problem statement of the project "Password and Notes Manager in Java" is to develop a secure and user-friendly application that can store and manage passwords and notes for multiple user accounts.

The application should have the following features:

- Allow users to create an account with a unique username and password
- Enable users to store their passwords and notes securely
- Provide the option to add, edit, delete and view passwords and notes
- Ensure that passwords and notes are encrypted to prevent unauthorized access
- Allow users to search for specific passwords and notes
- Ensure that the application is user-friendly and easy to navigate

The goal of the project is to create a robust and reliable password and notes manager that can help users keep track of their sensitive information securely.

CHAPTER 4

IMPLEMENTATION

The following requirements are required for the implementation of the project:

Hardware configuration

- Processor:1 GHz or faster
- RAM:2 GB or more
- Storage:At least 100MB of free disk space

Software configuration

- Visual Studio code with
- Java Development Kit(JDK) version 8 or higher

4.1 Implementation Code

MainActivity.java

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.security.SecureRandom;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.*;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
```

```
                // This class is used to create a loading screen
class SplashScreen {
   JFrame frame;
   JLabel image=new JLabel(new ImageIcon("key-lock.png"));
   JLabel text=new JLabel("PASSWORD & NOTES MANAGER");
   JProgressBar progressBar=new JProgressBar();
   JLabel message=new JLabel();
   SplashScreen()
   {
      createGUI();
      addImage();
      addText();
      addProgressBar();
      runningPBar();
   }

   public void createGUI(){
      frame=new JFrame(); // to create a frame
      frame.getContentPane().setLayout(null); // to set the layout of the frame
      frame.setUndecorated(true);
      frame.setSize(400,400); // to set the size of the frame
      frame.setLocationRelativeTo(null);
      frame.getContentPane().setBackground(new Color(0XFF8787)); // to set the background color of the
frame
      frame.setVisible(true);
   }

   public void addImage(){
      image.setSize(400,200); // to set the size of the image
      frame.add(image);
   }

   public void addText()
   {
      text.setFont(new Font("MV Boli",Font.BOLD,20)); // to set the font of the text
      text.setBounds(30,200,400,30);
      text.setForeground(Color.black);
           frame.add(text);
   }

   public void addProgressBar(){
      progressBar.setBounds(100,280,200,30); // to set the size of the progress bar
      progressBar.setBorderPainted(true);
      progressBar.setStringPainted(true);
      progressBar.setBackground(Color.black);
      progressBar.setForeground(new Color(0X38E54D));
      progressBar.setValue(0);
```

```
 frame.add(progressBar);
   }
   public void runningPBar(){
      int i=0;//Creating an integer variable and initializing it to 0
      while( i<=100)
      {
         try{
            Thread.sleep(40);   //Pausing execution for 50 milliseconds
            progressBar.setValue(i);    //Setting value of Progress Bar
            i++;
            if(i==100)
               frame.dispose();
         }catch(Exception e){
            e.printStackTrace();
         }
      }
   }
}

//Linear Probing Implementation
class HashtablePassword implements hashTableMap {
   private final int useProbe;    //0 = Linear Probing, 1 = Quadratic Probing
   private Entry[] entries;       //The array of entries
   private final float loadFactor;     //The load factor
   private int size, used;         //used acquires

       frame.add(text);
      }

      public void addProgressBar(){
         progressBar.setBounds(100,280,200,30); // to set the size of the progress bar
         progressBar.setBorderPainted(true);
         progressBar.setStringPainted(true);
         progressBar.setBackground(Color.black);
         progressBar.setForeground(new Color(0X38E54D));
         progressBar.setValue(0);
         frame.add(progressBar);
      }
      public void runningPBar(){
         int i=0;//Creating an integer variable and initializing it to 0
         while( i<=100)
         {
            try{
               Thread.sleep(40);   //Pausing execution for 50 milliseconds
               progressBar.setValue(i);    //Setting value of Progress Bar
               i++;
               if(i==100)
                  frame.dispose();
            }catch(Exception e){
               e.printStackTrace();
```

```
            }
        }
    }
}

//Linear Probing Implementation
class HashtablePassword implements hashTableMap {
    private final int useProbe;    //0 = Linear Probing, 1 = Quadratic Probing
    private Entry[] entries;       //The array of entries
    private final float loadFactor;     //The load factor
    private int size, used;         //used acquires

            }
            used = size;
        }
    }

    @Override
    public int add_Acc(Object Account, Object passwd) {
        if(used > (loadFactor*entries.length))rehash();
        int h = hash(Account);
        for (int i = 0; i < entries.length; i++){
            int j = (h+i) % entries.length;
            Entry entry = entries[j];
            if(entry==null){
                entries[j]= new Entry(Account, passwd);
                ++size;
                ++used;
                return h;
            }
            if(entry == NIL)continue;
            if(entry.key.equals(Account)){
                Object oldValue = entry.value;
                entries[j].value = passwd;
                return (int) oldValue;
            }
        }
        return h;
    }

    @Override
    public Object get_Acc(Object Account) {
        int h = hash(Account);
        for(int i = 0; i < entries.length; i++){
            int j = nextProbe(h , i);
            Entry entry = entries[j];
            if(entry == null)break;
            if(entry == NIL)continue;
            if(entry.key.equals(Account)) return entry.value;
        }
        return null;
    }
```

```
    @Override
    public Object remove_Acc(Object Account) {
        int h = hash(Account);
        for(int i = 0; i < entries.length; i++){
            int j = nextProbe(h,i);
            Entry entry = entries[j];
            if(entry == NIL)continue;
            if(entry.key.equals(Account)){
                Object Value = entry.value;
                entries[j] = NIL;
                size--;
                return Value;
            }
        }
        return null;
    }
}
class CryptoUtil
{

    Cipher ecipher;
    Cipher dcipher;
    // 8-byte Salt
    byte[] salt = {

     (byte) 0xA9, (byte) 0x9B, (byte) 0xC8, (byte) 0x32,
        (byte) 0x56, (byte) 0x35, (byte) 0xE3, (byte) 0x03
    };
    // Iteration count
    int iterationCount = 19;

    public CryptoUtil() {

    }

  public String encrypt(String secretKey, String plainText)
            throws NoSuchAlgorithmException,
            InvalidKeySpecException,
            NoSuchPaddingException,
            InvalidKeyException,
            InvalidAlgorithmParameterException,
            UnsupportedEncodingException,
            IllegalBlockSizeException,
            BadPaddingException {
        //Key generation for enc and desc
        KeySpec keySpec = new PBEKeySpec(secretKey.toCharArray(), salt, iterationCount);
        SecretKey key = SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);
        // Prepare the parameter to the ciphers
        AlgorithmParameterSpec paramSpec = new PBEParameterSpec(salt, iterationCount);

        //Enc process
        ecipher = Cipher.getInstance(key.getAlgorithm());
```

```
        ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
        String charSet = "UTF-8";
        byte[] in = plainText.getBytes(charSet);
        byte[] out = ecipher.doFinal(in);
        String encStr = new String(Base64.getEncoder().encode(out));
        return encStr;
    }

    public String decrypt(String secretKey, String encryptedText)
            throws NoSuchAlgorithmException,
            InvalidKeySpecException,
            NoSuchPaddingException,
            InvalidKeyException,
            InvalidAlgorithmParameterException,
            UnsupportedEncodingException,
            IllegalBlockSizeException,
            BadPaddingException,
            IOException {
        //Key generation for enc and desc
        KeySpec keySpec = new PBEKeySpec(secretKey.toCharArray(), salt, iterationCount);
        SecretKey key = SecretKeyFactory.getInstance("PBEWithMD5AndDES").generateSecret(keySpec);
        // Prepare the parameter to the ciphers
        AlgorithmParameterSpec paramSpec = new PBEParameterSpec(salt, iterationCount);
        //Decryption process; same key will be used for decr
        dcipher = Cipher.getInstance(key.getAlgorithm());
        dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        byte[] enc = Base64.getDecoder().decode(encryptedText);
        byte[] utf8 = dcipher.doFinal(enc);
        String charSet = "UTF-8";
        String plainStr = new String(utf8, charSet);
        return plainStr;
    }

}

class PasswordGenerator {
    private static final SecureRandom random = new SecureRandom();
    private static final String caps="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static      final String small_caps="abcdefghijklmnopqrstuvwxyz";
    private static final String Numeric="1234567890";
    private static final String special_char="~!@#$%^&*(_+{}|:_[?]>=<";
    private static final String dic = caps + small_caps + Numeric + special_char;

    public String generatePassword(int len) {
        StringBuilder password= new StringBuilder();
        for (int i = 0; i <len ; i++) {
            int index = random.nextInt(dic.length());
            password.append(dic.charAt(index));
        }
        return password.toString();
    }
```

```
}


interface hashTableMap {

    Object get_Acc(Object Account);
    int add_Acc(Object Account, Object passwd);
    Object remove_Acc(Object Account);
}

class PasswordManager implements ActionListener {

    //Store password class reference
    HashtablePassword data = new HashtablePassword(15,0.5F,0);

    // GUI variables declaration
    JFrame frame;
    JFrame frame2;
    JLabel background;
    Container conn1,conn2;
    JLabel lAcc,lPass;
    JTextArea encryptPasswdArea, genePassArea, searchPassArea;
    JButton PassGeneBtn,PassEncryptBtn, PassStoreBtn, PassSearchBtn;

   JTextField tAcc,tPass;
    JButton addNoteBtn;
    JLabel addNoteLabel;
    JTextArea tNote;
    JButton addNote;
    JFrame conn3;

    ArrayList<String> notes = new ArrayList<>(); // to store the notes in an array list of string type

    @Override
    public void actionPerformed(ActionEvent e) { }

    //Frame settings
    public static void FrameGUI(JFrame frame){
        frame.setVisible(true);
        frame.setLayout(null);
        frame.setLocationRelativeTo(null);
    }


    //container settings
    public static void ContainerGUI(Container conn){
        conn.setVisible(true);
        conn.setBackground(Color.getHSBColor(20.4f, 10.5f, 12.9f));
        conn.setLayout(null);
    }
```

```
// buttons settings
    public void GUIButtonsSetting(JButton btn){
        btn.setBackground(new Color(0XFB2576));
        btn.setForeground(Color.WHITE);
        btn.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));
        btn.setFocusable(false);
     Cursor crs = new Cursor(Cursor.HAND_CURSOR);
        btn.setCursor(crs);
        Font fn = new Font("MV Boli", Font.BOLD, 15);
        btn.setFont(fn);
    }

    //GUI of Store password
    public void StoringGUI()
    {
        frame2 = new JFrame("Store your passwords");
        frame2.setBounds(1400, 300, 800, 500);
        frame2.setSize(400,400);
        FrameGUI(frame2);
        conn2 = frame2.getContentPane();
        ContainerGUI(conn2);
        Font fn = new Font("MV Boli", Font.BOLD, 20);

        //Account textFiled and label
        lAcc = new JLabel("ACCOUNT NAME");
        lAcc.setBounds(90, 23, 380, 20);
        lAcc.setFont(fn);
        conn2.add(lAcc);

        tAcc = new JTextField();
        tAcc.setBounds(90,70,200,50);
        tAcc.setFont(fn);
        tAcc.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));
        tAcc.setForeground(Color.DARK_GRAY);
        conn2.add(tAcc);

        //Account password textField and label
        lPass = new JLabel("ACCOUNT PASSWORD");
        lPass.setBounds(90, 160, 380, 20);
        lPass.setFont(fn);
        conn2.add(lPass);
            tPass = new JTextField();
        tPass.setBounds(90,200,200,50);
        tPass.setFont(fn);
        tPass.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));
        tPass.setForeground(Color.DARK_GRAY);
        conn2.add(tPass);

        AccAddBtn = new JButton("STORE");
        AccAddBtn.setBounds(120, 290, 150, 50);
        conn2.add(AccAddBtn);
        GUIButtonsSetting(AccAddBtn);
```

```
}

  //for password generator and encryption
  public void textArea(String Pass,JTextArea TA){
     TA.setText(Pass);
     Font fn = new Font("MV Boli", Font.BOLD, 20);
     TA.setWrapStyleWord(true);
     TA.setLineWrap(true);
     TA.setCaretPosition(0);
     TA.setEditable(false);
     TA.setFont(fn);

  }

  //GUI of Password Manager
  PasswordManager() {

     frame = new JFrame("Password Manager");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setSize(400,650);
     frame.setResizable(false);
     ImageIcon img = new ImageIcon("background.png");
     background = new JLabel("",img,JLabel.CENTER);
         background.setBounds(0,0,400,650);
     background.setVisible(true);
     frame.add(background);

     FrameGUI(frame);

     conn1 = frame.getContentPane();
     ContainerGUI(conn1);

     //Generator buttons settings
     PassGeneBtn = new JButton("GENERATE PASSWORD");
     PassGeneBtn.setBounds(90, 20, 220, 40);
     conn1.add(PassGeneBtn);
     GUIButtonsSetting(PassGeneBtn);

     //generating password
     PassGeneBtn.addActionListener(e -> {
     if(PassGeneBtn ==e.getSource())
     {
        try{
           int len = Integer.parseInt(JOptionPane.showInputDialog("Enter the password length"));
           if(len>4)
           {
              //  password generator class reference
              PasswordGenerator pass = new PasswordGenerator();
              String passwd = pass.generatePassword(len);
              genePassArea = new JTextArea(5,4);
              textArea(passwd,genePassArea);
              JOptionPane.showMessageDialog(conn1,new JScrollPane(genePassArea),"Copy your
```

```
password",JOptionPane.INFORMATION_MESSAGE);

            }
            else JOptionPane.showMessageDialog (conn1,"Password length must be greater than 8!","Invalid Input
Error",JOptionPane.WARNING_MESSAGE);

          }
          catch(Exception ex){JOptionPane.showMessageDialog(conn1,"Write
something","EXIT!",JOptionPane.ERROR_MESSAGE);}
       }

   }
   );

      // add a encryption button and action
      JButton EncryptBtn = new JButton("ENCRYPT Text");
      EncryptBtn.setBounds(90, 90, 220, 40);
      conn1.add(EncryptBtn);
      GUIButtonsSetting(EncryptBtn);
      EncryptBtn.addActionListener(e -> {
         if(EncryptBtn ==e.getSource())
         {
            try{
               String text = JOptionPane.showInputDialog("Enter the text to encrypt");
               String secretKey = JOptionPane.showInputDialog("Enter the secret key");
               if(text.length()>0 && secretKey.length()>0)
               {
                  //  password generator class reference
                  CryptoUtil pass1 = new CryptoUtil();
                  String passwd = pass1.encrypt(secretKey, text); // encrypting the text
                  genePassArea = new JTextArea(5,4); // text area for the encrypted text
                  textArea(passwd,genePassArea); // setting the text area
                  JOptionPane.showMessageDialog(conn1,new JScrollPane(genePassArea),"Copy your
password",JOptionPane.INFORMATION_MESSAGE); // showing the encrypted text

               }
               else JOptionPane.showMessageDialog (conn1,"Write something","Invalid Input
Error",JOptionPane.WARNING_MESSAGE);

            }
            catch(Exception ex){JOptionPane.showMessageDialog(conn1,"Write
something","EXIT!",JOptionPane.ERROR_MESSAGE);}
         }
      }
      );

     // add a decryption button and action
      JButton DecryptBtn = new JButton("DECRYPT Text");
      DecryptBtn.setBounds(90, 160, 220, 40);
      conn1.add(DecryptBtn);
      GUIButtonsSetting(DecryptBtn);
      DecryptBtn.addActionListener(e -> {
```

```
 if(DecryptBtn ==e.getSource())
        {
           try{
              String text = JOptionPane.showInputDialog("Enter the text to decrypt"); // getting the encrypted text
              String secretKey = JOptionPane.showInputDialog("Enter the secret key"); // getting the secret key
              if(text.length()>0 && secretKey.length()>0) // checking if the text and secret key is not empty
              {
                 //  password generator class reference
                 CryptoUtil pass1 = new CryptoUtil(); // creating a object of the CryptoUtil class
                 String passwd = pass1.decrypt(secretKey, text); // decrypting the text
                 genePassArea = new JTextArea(5,4); // text area for the decrypted text
                 textArea(passwd,genePassArea); // setting the text area
                 JOptionPane.showMessageDialog(conn1,new JScrollPane(genePassArea),"Decrypted
text",JOptionPane.INFORMATION_MESSAGE); // showing the decrypted text

              }
              else JOptionPane.showMessageDialog (conn1,"Password length must be greater than 8!","Invalid Input
Error",JOptionPane.WARNING_MESSAGE);

           }
           catch(Exception ex){JOptionPane.showMessageDialog(conn1,"Write
something","EXIT!",JOptionPane.ERROR_MESSAGE);}
        }
     }
     );

     //storing password using hashtable
     PassStoreBtn = new JButton("STORE PASSWORD");
     PassStoreBtn.setBounds(90, 230, 220, 40);
     conn1.add(PassStoreBtn);
     GUIButtonsSetting(PassStoreBtn);
     //Store password action

   PassStoreBtn.addActionListener(e -> {
        if(PassStoreBtn ==e.getSource())
        {
           try{
              StoringGUI();
              // action on the Store btn
              AccAddBtn.addActionListener(e4 -> {
                 if (AccAddBtn == e4.getSource()) {
                    String account_name = tAcc.getText(); // getting the account name
                    String acc_pass = tPass.getText(); // getting the password
                    if (account_name.isEmpty() && acc_pass.isEmpty()) {
                       JOptionPane.showMessageDialog(conn2,"unable to store your
password!","ERROR",JOptionPane.ERROR_MESSAGE);
                    }
                    else{
                       //calling put method of the hashtablePassword class
                       data.add_Acc(account_name,acc_pass); // adding the account name and password to the hashtable
                       JOptionPane.showMessageDialog(conn2, "Account added Successfully !");
                       tAcc.setText(null);
```

```
                    tPass.setText(null);
                  }
                }
              }
            );
          }
        catch(Exception ex) {JOptionPane.showMessageDialog(conn2,"Write
something","EXIT",JOptionPane.ERROR_MESSAGE);}
        }
      }
      );

      //searching password
      PassSearchBtn = new JButton("SEARCH PASSWORD");
      GUIButtonsSetting(PassSearchBtn);
      PassSearchBtn.setBounds(90, 300, 220, 40);
      conn1.add(PassSearchBtn);
      PassSearchBtn.addActionListener(e ->{
      if (PassSearchBtn ==e.getSource()){
            try{
               String acc_name = JOptionPane.showInputDialog("Enter your Account Name"); // getting the account
name
               if (!acc_name.isBlank()) { // checking if the account name is not empty
                  Object pass = data.get_Acc(acc_name.toLowerCase()); // getting the password of the account name
                  if(pass!=null) { // checking if the password is not null
                     searchPassArea = new JTextArea(4,5); // text area for the password
                     textArea(String.valueOf(pass), searchPassArea); // setting the text area
                     JOptionPane.showMessageDialog(conn1, new JScrollPane(searchPassArea), "Copy your
password", JOptionPane.INFORMATION_MESSAGE);
                  }
                  else JOptionPane.showMessageDialog(conn1, "Account not Found!");
               }
            }
            catch (Exception ex){
               JOptionPane.showMessageDialog(conn1,"Write
something","EXIT",JOptionPane.ERROR_MESSAGE);
               }
            }
         }
      );

      // deleting password
      PassDeleteBtn = new JButton("DELETE PASSWORD");
      GUIButtonsSetting(PassDeleteBtn);
      PassDeleteBtn.setBounds(90, 370, 220, 40);
      conn1.add(PassDeleteBtn);
      PassDeleteBtn.addActionListener(e -> {
         if (PassDeleteBtn == e.getSource()) {
            try {
               String acc_name = JOptionPane.showInputDialog("Enter the Account Name"); // getting the account
name
               if (!acc_name.isBlank()) {
```

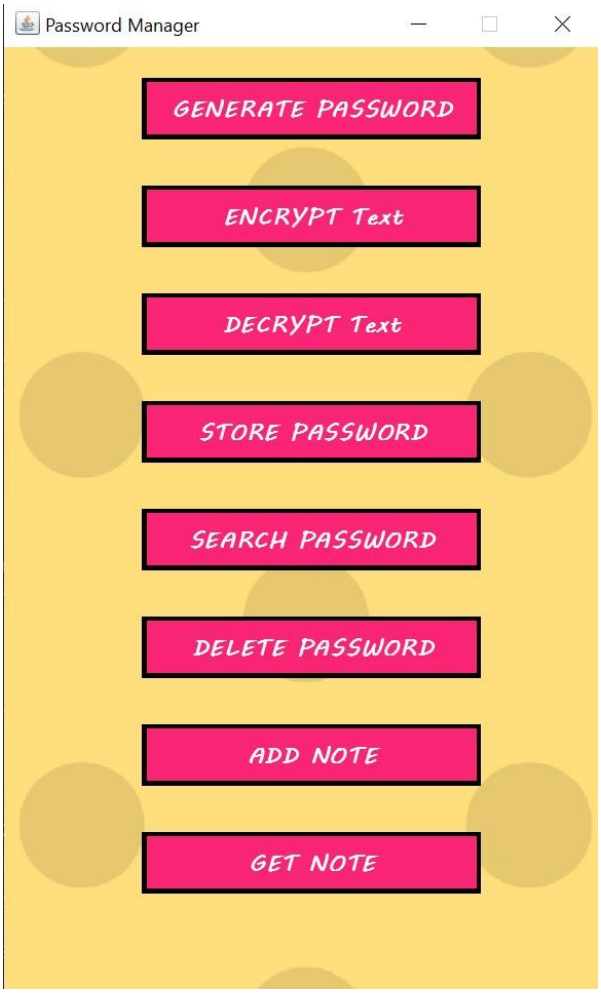
```
data.remove_Acc(acc_name.toLowerCase()); // removing the account name and password from the hashtable
                    JOptionPane.showMessageDialog(conn1, "Delete successfully!"); // showing the message
                }
                else JOptionPane.showMessageDialog(conn1, "Account not found!", "INFO",
JOptionPane.INFORMATION_MESSAGE);
                    } catch (Exception ex) {
                JOptionPane.showMessageDialog(conn1, "Write something", "EXIT",
JOptionPane.ERROR_MESSAGE);
            }
          }

      }
      );

      addNoteBtn = new JButton("ADD NOTE");
      GUIButtonsSetting(addNoteBtn);
      addNoteBtn.setBounds(90, 440, 220, 40);
      conn1.add(addNoteBtn);
      addNoteBtn.addActionListener(e -> {
         if (addNoteBtn == e.getSource()) {
            try {
               NoteGUI();
               // action on the add note btn
               addNote.addActionListener(e4 -> {
                  if (addNote == e4.getSource()) {
                     String note = tNote.getText(); // getting the note
                     if (note.isEmpty()) {
                        JOptionPane.showMessageDialog(conn3, "unable to store your note!", "ERROR",
JOptionPane.ERROR_MESSAGE);
                     } else {
                        //calling put method of the hashtablePassword class
                        notes.add(note); // adding the note to the arraylist
                        JOptionPane.showMessageDialog(conn3, "Note added Successfully !");
                        conn3.setVisible(false);
                        tNote.setText(null);
                     }
                  }
               });
            } catch (Exception ex) {
               JOptionPane.showMessageDialog(conn3, "Write something", "EXIT",
JOptionPane.ERROR_MESSAGE);
            }
      }
      }
      );

      //get all notes
      JButton getNoteBtn = new JButton("GET NOTE");
      GUIButtonsSetting(getNoteBtn);
      getNoteBtn.setBounds(90, 510, 220, 40);
      conn1.add(getNoteBtn);
      getNoteBtn.addActionListener(e -> {
```

```
 if (getNoteBtn == e.getSource()) {
            try {
                String allNotes = notes.get(notes.size() - 1); // getting the last note added
                if (allNotes.isEmpty()) { // checking if the note is empty or not
                    JOptionPane.showMessageDialog(conn1, "No note found!", "INFO",
JOptionPane.INFORMATION_MESSAGE); // showing the message
                } else {
                    searchPassArea = new JTextArea(4, 5); // text area for the note
                    textArea(allNotes, searchPassArea); // setting the text area
                    JOptionPane.showMessageDialog(conn1, new JScrollPane(searchPassArea), "Get your notes",
JOptionPane.INFORMATION_MESSAGE); // showing the message
                }
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(conn1, "Add a note before trying to retrive", "EXIT",
JOptionPane.ERROR_MESSAGE);
            }
        }
    }
    );

}

// method for setting the buttons and GUI for adding notes
private void NoteGUI() {

    conn3 = new JFrame("Add Note");
    conn3.setSize(500, 500);
    conn3.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    conn3.setLocationRelativeTo(null);
    conn3.setLayout(null);
    conn3.setVisible(true);
    conn3.setResizable(false);

    //add note label
     addNoteLabel = new JLabel("Add Note");
    addNoteLabel.setBounds(200, 20, 100, 30);
    conn3.add(addNoteLabel);

    //add note text area
     tNote = new JTextArea(10, 10);
    tNote.setBounds(100, 60, 300, 300);
    conn3.add(tNote);

    //add note button
     addNote = new JButton("ADD NOTE");
    GUIButtonsSetting(addNote);
    addNote.setBounds(140, 380, 220, 30);
    conn3.add(addNote);
}

// main method to run the application
public static void main(String[] args) {
```

```
//loading screen class
        new SplashScreen();
        try {
            new PasswordManager();
        }catch (Exception ex) { ex.printStackTrace(); }
 }
}
```

CHAPTER 5

RESULT



CHAPTER 6

APPLICATIONS & ADVANTAGES AND DISADVANTAGES

6.1 APPLICATIONS

The main benefit of using a password manager to boost your cyber security is that you do not need to have a good memory. That means everyone can incorporate the latest recommendations for secure passwords, including using long phrases, symbols, punctuation, and capitalization. A strong password provides essential protection from financial fraud and identity theft. One of the most common ways that hackers break into computers is by guessing passwords. Simple and commonly used passwords enable intruders to easily gain access and control of a computing device.

6.2 ADVANTAGES & DISADVANTAGES

6.2.1ADVANTAGES

A password and notes manager project in Java can have several

advantages, including:

Enhanced security: With a password and notes manager, users

can securely store their login credentials and important notes,

without the risk of unauthorized access. The project can use

encryption algorithms to protect sensitive information.

Easy access: Users can access their passwords and notes from

anywhere, as long as they have the project installed on their

device. They won't have to worry about forgetting their login

credentials.

3. Time-saving: The password and notes manager can automatically fill in login credentials for websites and applications, saving users time and reducing the risk of typing errors.

4. Organization: Users can organize their passwords and notes in a structured manner, making it easier to find and access the information they need.

5. Cuomization: The project can be customized to suit the specific needs of the user, such as adding custom fields to store additional information or changing the user interface to make it more user-friendly.

6.Cross-platform compatibility: A Java-based project can run on multiple operating systems, including Windows, macOS, and Linux, making it accessible to a wider range of users.

Overall, a password and notes manager project in Java can provide users with a secure, convenient, and organized way to manage their login credentials and important notes.

6.2.2 DISADVANTAGES

While a password and notes manager project in Java has many advantages, there are also some potential disadvantages to consider:

1. Security risks: The project may be vulnerable to security breaches if the encryption algorithms used are weak or if there are any vulnerabilities in the code. This could potentially compromise users' sensitive information.

2. User error: Users may forget the password to access their password and notes manager, which could result in them being locked out of their own account. This could also happen if the user forgets to save changes or accidentally deletes important information.

3. Complexity: A password and notes manager project in Java may be complex to develop, which could lead to bugs and other issues that affect the user experience.

4. Cost: If the project is developed by a professional developer or team, there may be a cost associated with it. This could be a disadvantage for users who are not willing to pay for a password and notes manager.

5. Compatibility issues: While Java is cross-platform compatible, there may still be compatibility issues with different operating systems and devices. This could limit the accessibility of the project for some users.

Overall, while a password and notes manager project in Java has many advantages, it is important to consider these potential disadvantages before deciding to use or develop one.

CONCLUSION

In conclusion, a password and notes manager project in Java can be a useful tool for users who want to securely store and manage their login credentials and important notes. The project can provide enhanced security, easy access, time-saving features, organization, customization, and cross-platform compatibility. However, there are also potential disadvantages to consider, including security risks, user error, complexity, cost, and compatibility issues. Ultimately, the decision to use or develop a password and notes manager project in Java will depend on the specific needs and preferences of the user, as well as the level of security and convenience they require. It is important to weigh the advantages and disadvantages carefully before making a decision.

REFERENCES

- Java Cryptography Architecture (JCA)
- Java Secure Socket Extension (JSSE)
- Apache Commons Codec
- SQLite
- GitHub