SECURE FILE TRANSFER APPLICATION

Technical Manual

Author: Aoife O'Brien

Student ID: C00214279

Project Supervisor: Patrick Tobin

Recipient: Institute of Technology Carlow

Date: Monday 20th April 2020

Contents

Introduction

The following document contains all the code for the secure file transfer application, along with the tools and technologies used throughout the development process.

Tools and Technologies

The following technologies and tools were used to create the application:

Technologies:

- Java
- MySQL

Tools:

- Java runtime environment (https://java.com/en/download/)
- Eclipse IDE (https://www.eclipse.org/downloads/)
- MySQL Workbench (https://www.mysql.com/products/workbench/)

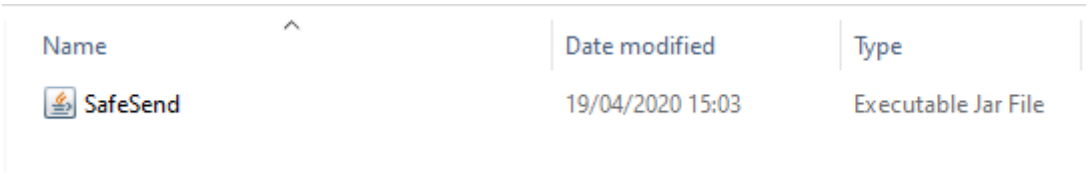
Versions

The versions of the languages and tools used are as follows:

- Java Version 8 Update 201
- Eclipse IDE Version 2020-03
- MySQL Workbench Version 8.0.19

How To Install

In order to run this application, users must have a Java runtime environment installed on their machine. The application was designed for Windows based machines, and currently only operates on Windows operating systems. The application can be launched simply from running the executable jar file of the application, as pictured below, which contains the complete application in one package. The application is centrally hosted and would need to be installed individually on users machines and configured for their specific needs.



Name	Date modified	Type
SafeSend	19/04/2020 15:03	Executable Jar File

Dependencies

Dependencies are the libraries that are required by an application to successfully build and run, along with the source code.

The following dependencies are used by the application:

```
java.sql.*;

javax.swing.*;
javax.swing.SwingUtilities;
javax.swing.JFileChooser;
javax.swing.filechooser.FileSystemView;

java.awt.*;
java.awt.event.*;

javax.crypto.Cipher;
javax.crypto.SecretKey;
javax.crypto.spec.SecretKeySpec;
javax.crypto.spec.IvParameterSpec;
javax.crypto.BadPaddingException;
javax.crypto.Cipher;
javax.crypto.IllegalBlockSizeException;
javax.crypto.KeyGenerator;
javax.crypto.NoSuchPaddingException;

java.nio.file.FileSystems;
java.nio.file.Files;
java.nio.file.Path;
java.nio.file.Paths;
java.nio.file.WatchEvent;
java.nio.file.WatchKey;
java.nio.file.WatchService;
java.nio.file.StandardWatchEventKinds.*;
```

```
java.nio.channels.FileLock;

org.bouncycastle.util.encoders.Base64;

java.io.ByteArrayInputStream;
java.io.File;
java.io.FileInputStream;
java.io.FileOutputStream;
java.io.InputStream;
java.io.OutputStream;
java.io.BufferedInputStream;
java.io.BufferedOutputStream;
java.io.DataInputStream;
java.io.DataOutputStream;
java.io.IOException;

java.security.InvalidKeyException;
java.security.KeyFactory;
java.security.KeyPair;
java.security.KeyPairGenerator;
java.security.NoSuchAlgorithmException;
java.security.PrivateKey;
java.security.PublicKey;
java.security.SecureRandom;
java.security.spec.InvalidKeySpecException;
java.security.spec.PKCS8EncodedKeySpec;
java.security.spec.X509EncodedKeySpec;
java.security.MessageDigest;

java.net.ServerSocket;
java.net.Socket;

java.util.UUID;
```

```
java.util.stream.Stream;
java.util.zip.ZipEntry;
java.util.zip.ZipInputStream;
java.util.zip.ZipOutputStream;
```

External JAR Files

commons-io-2.4.jar	(Source Website)
commons-io-2.4-javadoc.jar	(Source Website)
commons-io-2.4-sources.jar	(Source Website)
commons-io-2.4-tests.jar	(Source Website)
commons-io-2.4-test-sources.jar	(Source Website)
mysql-connector-java-8.0.19.jar	(Source Website)
bcprov-ext-jdk15on-165.jar	(Source Website)

Code

Following is all the code for the application listed by class;

Main.java

```
public class Main
{
    public static void main(String[] args)
    {
        SwingUtilities.invokeLater(new Runnable()
        {
             @Override
            public void run()
            {
                    new RegistrationPage();
            }
        });
    }
}
```

RegistrationPage.java

```
public class RegistrationPage extends JFrame implements ActionListener
{
    JFrame frame;
    JLabel firstnameLabel=new JLabel("FIRST NAME");
    JLabel lastnameLabel=new JLabel("LAST NAME");
    JLabel usernameLabel=new JLabel("USERNAME");
```

```
    JLabel passwordLabel=new JLabel("PASSWORD");
    JLabel confirmPasswordLabel=new JLabel("CONFIRM PASSWORD");
    JLabel emailLabel=new JLabel("WORK EMAIL");
    JTextField firstnameTextField=new JTextField();
    JTextField lastnameTextField=new JTextField();
    JTextField usernameTextField=new JTextField();
    static JPasswordField passwordField=new JPasswordField();
    JPasswordField confirmPasswordField=new JPasswordField();
    JTextField emailTextField=new JTextField();
    JButton registerButton=new JButton("REGISTER");
    JButton resetButton=new JButton("RESET");
    JButton goToLogin=new JButton("Go to login page");
    Color myTextColor;

    RegistrationPage()
    {
       createWindow();
       addComponentsToFrame();
       setLocationAndSize();
        actionEvent();
    }


public void createWindow()
    {
        frame=new JFrame();
        frame.setTitle("Registration Page");
        frame.setFont(new Font("Dialog", Font.PLAIN, 96));
       frame.setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Project\\src\\
       Socket\\secureft.png"));

    }

    public void setLocationAndSize()
    {
        firstnameLabel.setBounds(20,20,80,70);
        firstnameLabel.setForeground(Color.white);
        lastnameLabel.setBounds(20,70,80,70);
        lastnameLabel.setForeground(Color.white);
        usernameLabel.setBounds(20,120,100,70);
        usernameLabel.setForeground(Color.white);
        passwordLabel.setBounds(20,170,100,70);
        passwordLabel.setForeground(Color.white);
        confirmPasswordLabel.setBounds(20,220,140,70);
        confirmPasswordLabel.setForeground(Color.white);
        emailLabel.setBounds(20,270,100,70);
        emailLabel.setForeground(Color.white);
        firstnameTextField.setBounds(180,43,165,23);
        lastnameTextField.setBounds(180,93,165,23);
        usernameTextField.setBounds(180,143,165,23);
        passwordField.setBounds(180,193,165,23);
        confirmPasswordField.setBounds(180,243,165,23);
        emailTextField.setBounds(180,293,165,23);
        registerButton.setBounds(70,370,100,35);
        resetButton.setBounds(220,370,100,35);
        goToLogin.setBounds(105,440,165,23);
    }
```

```
public void addComponentsToFrame()
{
    frame.add(firstnameLabel);
    frame.add(lastnameLabel);
    frame.add(usernameLabel);
    frame.add(passwordLabel);
    frame.add(confirmPasswordLabel);
    frame.add(emailLabel);
    frame.add(firstnameTextField);
    frame.add(lastnameTextField);
    frame.add(usernameTextField);
    frame.add(passwordField);
    frame.add(confirmPasswordField);
    frame.add(emailTextField);
    frame.add(goToLogin);
    frame.add(registerButton);
    frame.add(resetButton);
    ImageIcon background=new
   ImageIcon("C:\\Project\\src\\Socket\\background.jpg");
    Image img=background.getImage();
    Image temp=img.getScaledInstance(380,500,Image.SCALE_SMOOTH);
    background=new ImageIcon(temp);
    JLabel back=new JLabel(background);
    frame.add(back);
    frame.pack();
   frame.setLocationRelativeTo(null);
   frame.getContentPane().setLayout(null);
   frame.setSize(380,500);
   frame.setVisible(true);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setResizable(false);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setResizable(false);
}

public void actionEvent()
{
    registerButton.addActionListener(this);
    resetButton.addActionListener(this);
    goToLogin.addActionListener(this);
    goToLogin.setActionCommand("Open");
}

@Override
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==registerButton)
    {
         String username = usernameTextField.getText();
         try {
         Connection
         connection=DriverManager.getConnection("jdbc:mysql://*redacted*:3308
         /safesend?useTimezone=true&serverTimezone=UTC","root",
         "*redacted*");
         PreparedStatement Pstatement=connection.prepareStatement("insert
         into users values(?,?,?,?,?,?,?,?,?,?)");
         Pstatement.setString(1,firstnameTextField.getText());
         Pstatement.setString(2,lastnameTextField.getText());
```

```
        Pstatement.setString(3,usernameTextField.getText());
        Pstatement.setString(5,emailTextField.getText());
        Pstatement.setNull(7,java.sql.Types.NULL);
        Pstatement.setNull(8,java.sql.Types.NULL);
        String password=passwordField.getText();
        String confirmPassword=confirmPasswordField.getText();
        if(password.equalsIgnoreCase(confirmPassword))
        {
                String salt = passwordFunction.getSalt(45);
                String hashedPassword =
                passwordFunction.generateSecurePassword(password, salt);
                Pstatement.setString(6,salt);
                Pstatement.setString(4,hashedPassword);
                KeyPair keyPair = RSAEncryptionWithAES.genRSAKeys();
                KeyFactory fact = KeyFactory.getInstance("RSA");
                PublicKey publicKey = keyPair.getPublic();
                String publicK =
                Base64.toBase64String(publicKey.getEncoded());
                PrivateKey privateKey  = keyPair.getPrivate();
                String privateK =
                Base64.toBase64String(privateKey.getEncoded());
                byte[] decodedKey = "*redacted*".getBytes();
                SecretKey originalKey = new SecretKeySpec(decodedKey, 0,
                decodedKey.length, "AES");
                Cipher aesCipher = Cipher.getInstance("AES");
                aesCipher.init(Cipher.ENCRYPT_MODE, originalKey);
                byte[] CipherText = aesCipher.doFinal(publicK.getBytes());
                publicK = Base64.toBase64String(CipherText);
                byte[] input =privateK.getBytes();
                byte[] CipherText2 = aesCipher.doFinal(input);
                privateK = Base64.toBase64String(CipherText2);
                Pstatement.setString(9, publicK);
                Pstatement.setString(10, privateK);
                Pstatement.executeUpdate();
                JOptionPane.showMessageDialog(null,"Registration Successful");

                frame.dispose();
                LoginPage lp = new LoginPage();
                lp.setTitle("Login");
            }
            else
            {

                JOptionPane.showMessageDialog(null,"Passwords do not match,
                please try again");
            }

        } catch (Exception e1)
        {
            e1.printStackTrace();
        }
    }

    if(e.getSource()==resetButton)
    {
        firstnameTextField.setText("");
        lastnameTextField.setText("");
        usernameTextField.setText("");
```

```
            passwordField.setText("");
            confirmPasswordField.setText("");
            emailTextField.setText("");
        }

        if(e.getSource()==goToLogin)
        {
            String cmd = e.getActionCommand();

            if(cmd.equals("Open"))
            {
             frame.setVisible(false);
             dispose();
             new LoginPage();
            }
        }
    }
}
```

LoginPage.java

```
public class LoginPage extends JFrame implements ActionListener
{
       public void goToLogin()
       {
              frame.setVisible(true);
       }

    JFrame frame;
    JLabel usernameLabel=new JLabel("USERNAME");
    JLabel passwordLabel=new JLabel("PASSWORD");
    static JTextField usernameTextField=new JTextField();
    static JPasswordField passwordField=new JPasswordField();
    JButton loginButton=new JButton("LOGIN");
    JButton resetButton=new JButton("RESET");
    JLabel hyperlink = new JLabel("Forgot password");
    Font f=new Font("ARIAL", Font.PLAIN,13);
    Color myTextColor;

    LoginPage()
    {
       createWindow();
       addComponentsToFrame();
       setLocationAndSize();
        actionEvent();
    }

    public static JTextField getUsernameTextField()
    {

       return usernameTextField;
    }

    public static void clearScreen()
    {
        usernameTextField.setText("");
        passwordField.setText("");
```

```
}

public void createWindow()
{
    frame=new JFrame();
    frame.setTitle("Login Page");
    frame.setFont(new Font("Dialog", Font.PLAIN, 96));
  frame.setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Project\\src\\
  Socket\\secureft.png"));
}

public void setLocationAndSize()
{
  usernameLabel.setBounds(40,40,80,70);
  usernameLabel.setForeground(Color.white);
  passwordLabel.setBounds(40,90,80,70);
  passwordLabel.setForeground(Color.white);
  usernameTextField.setBounds(180,63,165,23);
   passwordField.setBounds(180,113,165,23);
   hyperlink.setBounds(135,218,100,70);
   hyperlink.setFont(f);
   hyperlink.setForeground(Color.white.brighter());
   hyperlink.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
   loginButton.setBounds(60,200,100,35);
   resetButton.setBounds(210,200,100,35);
}

public void addComponentsToFrame()
{
    frame.add(usernameLabel);
    frame.add(passwordLabel);
    frame.add(usernameTextField);
    frame.add(passwordField);
    frame.add(loginButton);
    frame.add(resetButton);
    frame.add(hyperlink);
     ImageIcon background=new
  ImageIcon("C:\\Project\\src\\Socket\\background.jpg");
     Image img=background.getImage();
     Image temp=img.getScaledInstance(380,300,Image.SCALE_SMOOTH);
     background=new ImageIcon(temp);
     JLabel back=new JLabel(background);
     frame.add(back);
     frame.pack();
    frame.setLocationRelativeTo(null);
    frame.getContentPane().setLayout(null);
    frame.setSize(380,300);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setResizable(false);
}

public void actionEvent()
{
    loginButton.addActionListener(this);
    resetButton.addActionListener(this);
}
```

```
@Override
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==loginButton)
    {
         String username = usernameTextField.getText();
         String password = passwordField.getText();
         String passwordt = null;
         String salt = null;
         Boolean LoggedIn = true;
         try
         {
          Connection
          connection=DriverManager.getConnection("jdbc:mysql://*redacted*:3308
          /safesend?useTimezone=true&serverTimezone=UTC","root","*redacted*");
          PreparedStatement st = (PreparedStatement)
          connection.prepareStatement("Select SALT, PASSWORD from users where
          USERNAME=?");
          PreparedStatement updateLoggedIn = (PreparedStatement)
          connection.prepareStatement("Update users SET LoggedIn=true where
          USERNAME=?");
          PreparedStatement generateSessionID = (PreparedStatement)
          connection.prepareStatement("Update users SET SessionID=? where
          USERNAME=?");
          st.setString(1, username);
          updateLoggedIn.setString(1, username);
          generateSessionID.setString(2, username);
          ResultSet rs = st.executeQuery();
            if (rs.next())
            {
                salt = rs.getString("SALT");
                passwordt = rs.getString("PASSWORD");

                       String hashedPassword =
                       passwordFunction.generateSecurePassword(password,
                       salt);
                       if (hashedPassword.equals(passwordt))
                       {
                              String suuid =
                              passwordFunction.generateSessionID();
                              generateSessionID.setString(1, suuid);
                              generateSessionID.executeUpdate();
                              updateLoggedIn.executeUpdate();
                              JOptionPane.showMessageDialog(null, "You have
                              successfully logged in");

                              frame.dispose();
                              UserHome uh = new UserHome();
                              uh.setTitle("Welcome");
                       }
                       else
                       {
                              JOptionPane.showMessageDialog(null, "Wrong
                              Username & Password");
                       }
                } else
                       {
```

```
                                JOptionPane.showMessageDialog(null, "Wrong
                                Username & Password");
                        }
                } catch (Exception sqlException)
                        {
                                sqlException.printStackTrace();
                        }
                }

        if(e.getSource()==resetButton)
        {
            usernameTextField.setText("");
            passwordField.setText("");
        }
    }
}
```

UserHome.java

```
public class UserHome extends JFrame implements ActionListener
{
        File selectedFile = null;
        File[] SendingArray = {};

        JFrame frame;
        JButton sendFileButton=new JButton("SEND");
        JButton logoutButton=new JButton("LOGOUT");
        JLabel enterRecipientLabel = new JLabel("Enter recipient");
        static JTextField recipientTextField=new JTextField();
        JFileChooser openFileChooser;
        Font f=new Font("ARIAL", Font.PLAIN,13);
        Color myTextColor;


        public static void main(String[] args)

        {
                EventQueue.invokeLater(new Runnable()
                {
                        public void run()
                        {
                                try
                                {
                                        UserHome window = new UserHome();
                                        window.frame.setVisible(true);
                                } catch (Exception e)
                                {
                                        e.printStackTrace();
                                }
                        }
                });
        }


        UserHome()
        {
        createWindow();
```

```
    addComponentsToFrame();
    setLocationAndSize();
     actionEvent();
    }

    public void FileChooser()
    {
          openFileChooser = new JFileChooser();
    }

    public static void setIP (String user)
    {
        if (user.equals("t1"))
        {
          SocketServer.setSender("192.168.1.16");
        }
        else if (user.equals("t2"))
        {
          SocketServer.setSender("192.168.1.13");
        }
        else
        {
          System.out.println("User does not exist.");
        }
}

    public void createWindow()
{
     frame=new JFrame();
     frame.setTitle("Send a file");
     JButton chooseFileButton = new JButton("Choose file");
      chooseFileButton.setFont(new Font("Arial", Font.PLAIN, 16));
      chooseFileButton.setBounds(62, 60, 125, 30);
      frame.getContentPane().add(chooseFileButton);
      new Thread(new Runnable()


    {
          @Override

               public void run()
               {
                      boolean check = true;
                         while(check == true)
                         {
                               try
                               {
                                      new SocketServer();
                                      SocketClient.receive();
                               } catch (IOException e)
                               {

                                      System.out.println("Trying to connect to
                                      server...");
                               }
                      }
               }
          }).start();
```

```
File one = new File("C:\\Documents");
File two =  new File("C:\\Documents");
File three =  new File("C:\\Documents");
File[] ReceivingArray = {one,two,three};
File dir = new File("C:\\Documents\\Receiving");

dir.mkdir();
new Thread(new Runnable()
{
	WatchEvent.Kind<?> kind = null;
	int checkOnce = 0;
	WatchService watcher;

	@Override
	public void run()
	{
		boolean always = true;
		while(always == true)
	{
			try
			{
				watcher =
				FileSystems.getDefault().newWatchService()
				;
			} catch (IOException e2)
			{
				e2.printStackTrace();
			}
		Path Pdir = dir.toPath();
		WatchKey key = null;
		try
		{
		    key = Pdir.register(watcher,
						ENTRY_CREATE,
						ENTRY_DELETE,
						ENTRY_MODIFY);
		}
		catch (IOException x)
		{
		    System.err.println(x);
		}
		try
		{
			key = watcher.take();
		} catch (InterruptedException e1)
		{
			e1.printStackTrace();
		}
		for (WatchEvent<?> event: key.pollEvents())
		{
			kind = event.kind();
		}
		if (kind == ENTRY_MODIFY)
		{
			checkOnce = 1;
		}
		if(checkOnce == 1)
		{
```

```
                File[] listOfFiles = dir.listFiles();
                                        ReceivingArray[2] =
                                        listOfFiles[1];
                                        ReceivingArray[1] =
                                        listOfFiles[2];
                                        ReceivingArray[0] =
                                        listOfFiles[0];
                        try
                        {
                            RSAEncryptionWithAES.decrypt(ReceivingArra
                    y[0], ReceivingArray[1], ReceivingArray[2]);
                            MessageDigest shaDigest =
                            MessageDigest.getInstance("SHA-256");
                            byte[] buffer =
                            Files.readAllBytes(Paths.get("C:\\Document
                            s\\Receiving\\encryptedFile"));
                            shaDigest.update(buffer);
                            shaDigest.toString();
                            byte[] digest = shaDigest.digest();
                            String digestInStringForm =
                            Base64.toBase64String(digest);
                            System.out.println("Hash of file: " +
                            digestInStringForm);
                            checkOnce = 0;
                            key.reset();
                            kind = null;
                            }
                            catch (Exception e)
                            {
                                    e.printStackTrace();
                            }
                }
        }

      }
  }).start();

chooseFileButton.addActionListener(new ActionListener()
{

 @Override
 public void actionPerformed(ActionEvent arg0)
 {
  JFileChooser chooseFileButton = new JFileChooser();
  JFileChooser openFileChooser = new
  JFileChooser(FileSystemView.getFileSystemView().getHomeDirectory());
  openFileChooser.setDialogTitle("Select a file");
  int returnValue = openFileChooser.showOpenDialog(frame);

  if (returnValue == JFileChooser.APPROVE_OPTION)
  {
        try
        {
                selectedFile = openFileChooser.getSelectedFile();
                System.out.println("Selected file: " +
                selectedFile.getAbsolutePath());

        } catch (Exception e)
```

```
                {
                        e.printStackTrace();
                }
                }
        }
    }
    );
    frame.setFont(new Font("Dialog", Font.PLAIN, 96));
   frame.setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Project\\src\\
   Socket\\secureft.png"));
}

public void setLocationAndSize()
{
   sendFileButton.setBounds(60,250,285,45);
   sendFileButton.setFont(new Font("Arial", Font.PLAIN, 18));
   logoutButton.setBounds(135,315,120,40);
   logoutButton.setFont(new Font("Arial", Font.PLAIN, 14));
   enterRecipientLabel.setFont(new Font("Arial", Font.PLAIN, 16));
   enterRecipientLabel.setBounds(62, 150, 140, 29);
   enterRecipientLabel.setForeground(Color.white);
   recipientTextField.setBounds(190, 150, 158, 29);
}

public void addComponentsToFrame()
{
    frame.add(sendFileButton);
    frame.add(logoutButton);
    frame.add(enterRecipientLabel);
    frame.add(recipientTextField);
     ImageIcon background=new
   ImageIcon("C:\\Project\\src\\Socket\\background.jpg");
     Image img=background.getImage();
     Image temp=img.getScaledInstance(400,400,Image.SCALE_SMOOTH);
     background=new ImageIcon(temp);
     JLabel back=new JLabel(background);
     frame.add(back);
     frame.pack();
    frame.setLocationRelativeTo(null);
    frame.getContentPane().setLayout(null);
    frame.setSize(400,400);
    frame.setVisible(true);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setResizable(false);
}

public void actionEvent()
{
    sendFileButton.addActionListener(this);
    logoutButton.addActionListener(this);
}

@Override
public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==logoutButton)
    {
         String username = LoginPage.getUsernameTextField().getText();
```

```
            String sessionID = LoginPage.getUsernameTextField().getText();

            try
           {
            Connection
connection=DriverManager.getConnection("jdbc:mysql://*redacted*:3308/safesend?useT
imezone=true&serverTimezone=UTC","root","*redacted*");
            PreparedStatement getSessionID = (PreparedStatement)
connection.prepareStatement("Select SessionID from users where USERNAME=?");
            PreparedStatement updateLoggedIn = (PreparedStatement)
connection.prepareStatement("Update users SET LoggedIn=false where USERNAME=?");
            PreparedStatement updateSessionID = (PreparedStatement)
connection.prepareStatement("Update users SET SessionID=null where USERNAME=?");
            getSessionID.setString(1, username);
            updateLoggedIn.setString(1, username);
            updateSessionID.setString(1, username);
            ResultSet rs = getSessionID.executeQuery();
            updateLoggedIn.executeUpdate();
            updateSessionID.executeUpdate();

            int a = JOptionPane.showConfirmDialog(null, "Are you sure?");

            if (a == JOptionPane.YES_OPTION)
            {
                   frame.dispose();
                   LoginPage frame = new LoginPage();
                   LoginPage.clearScreen();
                  JOptionPane.showMessageDialog(null, "You are now logged out.
                  Please log in to continue!");
            }
            else
            {
            }
            } catch (SQLException sqlException)
            {
              sqlException.printStackTrace();
            }
      }

        if(e.getSource()==sendFileButton)
        {
            try
            {
                   SendingArray =
            RSAEncryptionWithAES.encrypt(selectedFile.getAbsolutePath());
                 setIP(recipientTextField.getText());
                 SocketServer.send(SendingArray[0]);
                 SocketServer.send(SendingArray[1]);
                 SocketServer.send(SendingArray[2]);
                 MessageDigest shaDigest = MessageDigest.getInstance("SHA-256");
                   byte[] buffer =
            Files.readAllBytes(Paths.get("C:\\Documents\\encryptedFile"));
                 shaDigest.update(buffer);
                 shaDigest.toString();
                 byte[] digest = shaDigest.digest();
                 String digestInStringForm = Base64.toBase64String(digest);
                 System.out.println("Hash of file: " + digestInStringForm);
            } catch (Exception e1)
```

```
            {
                    System.out.println("Please select a file");
            }
        }
       }
    }
```

RSAEncryptionWithAES.java

```
public class RSAEncryptionWithAES
{
      public static File[] encrypt(String Path) throws Exception
    {
      FileInputStream fis = null;
        String str = "";
        try
        {
            fis = new FileInputStream(Path);
            int content;
            while ((content = fis.read()) != -1)
            {
                str += (char) content;
            }
        } catch (IOException e)
        {
            e.printStackTrace();
        }
        finally
        {
            try
            {
                if (fis != null)
                    fis.close();
            } catch (IOException ex)
            {
                ex.printStackTrace();
            }
        }

        PrivateKey privateKey = getPrivateRSAKey();
        String secretAESKeyString = getSecretAESKeyAsString();
        IvParameterSpec ivSpec = generateIv();
        File ivSpecInFile = FileIv(ivSpec);
        File encryptedText = encryptTextUsingAES(str, secretAESKeyString, ivSpec);
       File encryptedAESKeyString = encryptAESKey(secretAESKeyString,
      privateKey);
        File[] fileArray = {encryptedText, encryptedAESKeyString, ivSpecInFile};
             return fileArray;
    }

    public static File FileIv(IvParameterSpec ivSpec) throws Exception
    {
             byte[] byteCipherText = ivSpec.getIV();
          FileOutputStream fos = new FileOutputStream("C:\\Documents\\IvFile");
          InputStream fis = new ByteArrayInputStream(byteCipherText);
           int length;
           while((length = fis.read(byteCipherText)) >= 0)
```

```
        {
           fos.write(byteCipherText, 0, length);
        }
        fis.close();
        fos.close();
        File c = new File("C:\\Documents\\IvFile");
           return c;
     }

   public static IvParameterSpec DeFileIv(File ivSpec) throws Exception
   {
     byte[] iv =
     Files.readAllBytes(Paths.get("C:\\Documents\\Receiving\\IvFile"));
     IvParameterSpec IVVal = new IvParameterSpec(iv, 0, 16);
           return IVVal;
     }

     public static String getSecretAESKeyAsString()
   {
       KeyGenerator generator;
           try {
           generator = KeyGenerator.getInstance("AES");
           generator.init(128);
           SecretKey secKey = generator.generateKey();
           String encodedKey = Base64.toBase64String(secKey.getEncoded());
           return encodedKey;
           } catch (NoSuchAlgorithmException e)
           {
                 e.printStackTrace();
                 String encodedKeyForReturn = null;
                 return encodedKeyForReturn;
           }
   }

   public static File encryptTextUsingAES(String str, String aesKeyString,
IvParameterSpec ivSpec) throws Exception
   {
       byte[] decodedKey = Base64.decode(aesKeyString);
       Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
       SecretKey originalKey = new SecretKeySpec(decodedKey, 0,
       decodedKey.length, "AES");
       aesCipher.init(Cipher.ENCRYPT_MODE, originalKey, ivSpec);
       byte[] CipherText = aesCipher.doFinal(str.getBytes("UTF-8"));
       byte[] byteCipherText = Base64.encode(CipherText);
       FileOutputStream fos = new FileOutputStream("C:Documents\\encryptedFile");
       InputStream fis = new ByteArrayInputStream(byteCipherText);
        int length;
        while((length = fis.read(byteCipherText)) >= 0)
        {
           fos.write(byteCipherText, 0, length);
        }
        fis.close();
        fos.close();
        File a = new File("C:\\Documents\\encryptedFile");
        return a;
   }

   public static KeyPair genRSAKeys() throws Exception
```

```
    {
        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
        KeyPair keyPair = keyPairGenerator.generateKeyPair();
        return keyPair;
    }

    private static File encryptAESKey(String plainAESKey, PrivateKey privateKey)
throws Exception
    {
        Cipher cipher = Cipher.getInstance("RSA");
        cipher.init(Cipher.ENCRYPT_MODE, privateKey);
       String temp =
      Base64.toBase64String(cipher.doFinal(Base64.decode(plainAESKey)));
        byte[] byteCipherText = temp.getBytes();
        FileOutputStream fos = new FileOutputStream("C:\\Documents\\keyFile");
        InputStream fis = new ByteArrayInputStream(byteCipherText);
          int length;
          while((length = fis.read(byteCipherText)) >= 0)
          {
             fos.write(byteCipherText, 0, length);
          }
          fis.close();
          fos.close();
          File b = new File("C:\\Documents\\keyFile");
          return b;
    }

      public static void decrypt(File encryptedFile, File encryptedAESKey, File
FileIv) throws Exception
    {
        PublicKey publicKey = getPublicRSAKey();
        String decryptedAESKeyString = decryptAESKey(encryptedAESKey, publicKey);
             decryptTextUsingAES(encryptedFile, decryptedAESKeyString, FileIv);
    }

      private static String decryptAESKey(File encryptedAESKey, PublicKey
publicKey) throws Exception
    {

        byte[] data =
        Files.readAllBytes(Paths.get("C:\\Documents\\Receiving\\keyFile"));
       Cipher cipher = Cipher.getInstance("RSA");
       cipher.init(Cipher.DECRYPT_MODE, publicKey);
       String temp = Base64.toBase64String(cipher.doFinal(Base64.decode(data)));
       return temp;
    }

      public static String decryptTextUsingAES(File encryptedFile, String aesKey,
File FileIv) throws Exception
      {
             FileInputStream fis = null;
        String str = "";
        try
        {
            fis = new FileInputStream(encryptedFile);
            int content;
            while ((content = fis.read()) != -1)
            {
```

```
            str += (char) content;
        }
    } catch (IOException e)
    {
        e.printStackTrace();
    } finally
    {
        try
        {
            if (fis != null)
            {
                fis.close();
            }
        } catch (IOException ex)
        {
            ex.printStackTrace();
        }
    }
         byte[] decodedKey = Base64.decode(aesKey);
         SecretKey originalKey = new SecretKeySpec(decodedKey, 0,
         decodedKey.length, "AES");
         Cipher aesCipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
         IvParameterSpec IVVal = DeFileIv(FileIv);
         aesCipher.init(Cipher.DECRYPT_MODE, originalKey, IVVal);
         byte[] decodedText = Base64.decode(str.getBytes("UTF-8"));
         byte[] bytePlainText = aesCipher.doFinal(decodedText);
         FileOutputStream fos = new
         FileOutputStream("C:\\Documents\\decryptedFile");
    InputStream fis2 = new ByteArrayInputStream(bytePlainText);
       int length;
       while((length = fis2.read(bytePlainText)) >= 0)
       {
          fos.write(bytePlainText, 0, length);
       }
       fis2.close();
       fos.close();
       return new String(bytePlainText);
}

   public static PrivateKey getPrivateRSAKey()
      {
                 try
                 {
                        String username =
                        LoginPage.getUsernameTextField().getText();
                        Connection connection =
                        DriverManager.getConnection("jdbc:mysql://*redacted*:33
                        08/safesend?useTimezone=true&serverTimezone=UTC","root"
                        ,"*redacted*");
                        PreparedStatement query =
                        connection.prepareStatement("SELECT Private FROM users
                        where USERNAME=?");
                        query.setString(1, username);
                        ResultSet rs = query.executeQuery();
                        if (rs.next());
                        {
                        byte[] decodedKey = "*redacted*".getBytes();
```

```
                        SecretKey originalKey = new SecretKeySpec(decodedKey,
                        0, decodedKey.length, "AES");
                        String retrivedPrivateKey = rs.getString("Private");
                        byte[] decodedPrivateKey =
                        Base64.decode(retrivedPrivateKey);
                        Cipher aesCipher = Cipher.getInstance("AES");
                         aesCipher.init(Cipher.DECRYPT_MODE, originalKey);
                    byte[] privateBytes = aesCipher.doFinal(decodedPrivateKey);
                    byte[] privateBytes2 = Base64.decode(privateBytes);
                        PKCS8EncodedKeySpec  privateKeySpec  = new
                        PKCS8EncodedKeySpec(privateBytes2);
                        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
                        PrivateKey privKey =
                        keyFactory.generatePrivate(privateKeySpec);
                        return privKey;
                    }
                }
                    catch (SQLException | InvalidKeySpecException |
NoSuchAlgorithmException | IllegalBlockSizeException | BadPaddingException e)
                {
                        e.printStackTrace();
                        return null;
                    } catch (InvalidKeyException e)
                    {
                        e.printStackTrace();
                        return null;
                    } catch (NoSuchPaddingException e)
                    {
                        e.printStackTrace();
                        return null;
                    }
            }

        public static PublicKey getPublicRSAKey()
            {
                try
                {
                        String username = SocketClient.getSender();
                    Connection connection =
                    DriverManager.getConnection("jdbc:mysql://*redacted*:3308/
                    safesend?useTimezone=true&serverTimezone=UTC","root","*red
                    acted*");
                    PreparedStatement query =
                    connection.prepareStatement("SELECT Public FROM users
                    WHERE USERNAME=?");
                        query.setString(1,  username);
                    ResultSet rs = query.executeQuery();
                    if (rs.next());
                    {
                        byte[] decodedKey = "*redacted*".getBytes();
                        SecretKey originalKey = new SecretKeySpec(decodedKey,
                        0, decodedKey.length, "AES");
                        String retrivedPublicKey = rs.getString("Public");
                        byte[] DecodedpublicBytes =
                        Base64.decode(retrivedPublicKey);
                        Cipher aesCipher = Cipher.getInstance("AES");
                    aesCipher.init(Cipher.DECRYPT_MODE, originalKey);
                    byte[] publicBytes = aesCipher.doFinal(DecodedpublicBytes);
```

```
                    byte[] publicBytes2 = Base64.decode(publicBytes);
                    X509EncodedKeySpec keySpec = new
                    X509EncodedKeySpec(publicBytes2);
                            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
                            PublicKey pubKey;
                            pubKey = keyFactory.generatePublic(keySpec);
                            return pubKey;
                    }
                }
                    catch (SQLException | InvalidKeySpecException |
NoSuchAlgorithmException | IllegalBlockSizeException | BadPaddingException e)
                {
                            e.printStackTrace();
                            return null;
                    } catch (InvalidKeyException e)
                {
                            e.printStackTrace();
                            return null;
                    } catch (NoSuchPaddingException e)
                {
                            e.printStackTrace();
                            return null;
                    }
            }

        public static IvParameterSpec generateIv()
        {
            SecureRandom secureRandom = new SecureRandom();
            byte[] IV = new byte[16];
            secureRandom.nextBytes(IV);
            IvParameterSpec ivSpec = new IvParameterSpec(IV);
            return ivSpec;
        }
}
```

SocketClient.java

```
public class SocketClient
{
  public static int SOCKET_PORT = 15010;
  public static String FILE_TO_RECEIVE = "C:\\Documents\\copy.txt";
  public final static int FILE_SIZE = 6022386;
  static Socket sock = null;

  public static String getSender()
  {
        String currentSock = sock.toString();
        if(currentSock.startsWith("Socket[addr=/192.168.1.16"))
        {
              return "t1";
        }
        else if(currentSock.startsWith("Socket[addr=/192.168.1.13"))
        {
              return "t2";
        }
        else return null;
  }
```

```
  public static void receive() throws IOException
  {
          int bytesRead;
          int current = 0;
          FileOutputStream fos = null;
          BufferedOutputStream bos = null;
          ServerSocket servsock = new ServerSocket(SOCKET_PORT);
          System.out.println("Waiting...");
          InputStream is = null;
          DataInputStream din = null;
          try
          {
            while (true)
            {
            sock = servsock.accept();
            System.out.println("Accepted connection : " + sock);

            byte [] mybytearray  = new byte [FILE_SIZE];
            is = sock.getInputStream();
            din = new DataInputStream(is);
            String fileName = din.readUTF();
            String deleteStart = "C:\\Documents";
            fileName = fileName.substring(deleteStart.length()-2);
            fos = new FileOutputStream("C:\\Documents\\Receiving\\" + fileName);
            bos = new BufferedOutputStream(fos);
            bytesRead = is.read(mybytearray,0,mybytearray.length);
            current = bytesRead;

            do
            {
               bytesRead = is.read(mybytearray, current, (mybytearray.length-
current));
               if(bytesRead >= 0) current += bytesRead;
            }
            while(bytesRead > -1);

            bos.write(mybytearray, 0 , current);
            bos.flush();
            System.out.println("File " + FILE_TO_RECEIVE + " downloaded (" +
current + " bytes read)");
          }
          }
          finally
          {
            if (fos != null) fos.close();
            if (bos != null) bos.close();
            if (sock != null) sock.close();
            if (servsock != null) servsock.close();
            if (din != null) din.close();
            if (is != null) is.close();
          }
  }
}
```

SocketServer.java

```
public class SocketServer
{
  public final static int SOCKET_PORT = 15010;
  public static String SERVER = "192.168.1.16";

  public static void setSender(String input)
  {
        SERVER = input;
  }

  public static void send (File fileforsending) throws IOException
  {
          String FILE_TO_SEND = fileforsending.getAbsolutePath();
          FileInputStream fis = null;
          BufferedInputStream bis = null;
          OutputStream os = null;
          Socket sock = null;
          DataOutputStream dos = null;
          try
          {
              System.out.println("Waiting...");
              try
              {
                sock = new Socket(SERVER, SOCKET_PORT);
                System.out.println("Accepted connection : " + sock);
                os = sock.getOutputStream();
                dos = new DataOutputStream(os);
                dos.writeUTF(FILE_TO_SEND);

                File myFile = new File (FILE_TO_SEND);
                byte [] mybytearray  = new byte [(int)myFile.length()];
                fis = new FileInputStream(myFile);
                bis = new BufferedInputStream(fis);
                bis.read(mybytearray,0,mybytearray.length);
                os = sock.getOutputStream();
               System.out.println("Sending " + FILE_TO_SEND + "(" +
            mybytearray.length + " bytes)");
                os.write(mybytearray,0,mybytearray.length);
                os.flush();
                System.out.println("Done.");
              }
              finally
              {
                if (bis != null)
                   {
                     bis.close();
                   }
               else
                   {
                         System.out.println("Error occured with closing the
                         buffer input stream.");
                   }
                if (os != null)
                   {
                     os.close();
                   }
```

```
                else
                   {
                         System.out.println("Error occured with closing the
                         output stream.");
                   }
                if (sock!=null)
                   {
                     sock.close();
                   }
                else
                   {
                         System.out.println("Error occured with closing the
                         socket.");
                   }
                if (fis!=null)
                   {
                     fis.close();
                   }
                else
                   {
                     System.out.println("Error occured with closing the fis.");
                   }
                if (dos!=null)
                   {
                     dos.close();
                   }
                else
                   {
                     System.out.println("Error occured with closing the dos.");
                   }
             }
          }
            finally
            {
                   System.out.println("Success.");
            }
      }
}
```

passwordFunction.java

```
public class passwordFunction
{

    private static final Random RANDOM = new SecureRandom();
    private static final String ALPHABET =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
    private static final int ITERATIONS = 10000;
    private static final int KEY_LENGTH = 256;

     public static String getSalt(int length)
     {
        StringBuilder returnValue = new StringBuilder(length);
        for (int i = 0; i < length; i++)
        {
```

```
            returnValue.append(ALPHABET.charAt(RANDOM.nextInt(ALPHABET.length())
            ));
        }
        return new String(returnValue);
    }
    public static byte[] hash(char[] password, byte[] salt)
    {
        PBEKeySpec spec = new PBEKeySpec(password, salt, ITERATIONS, KEY_LENGTH);
        Arrays.fill(password, Character.MIN_VALUE);
        try {
            SecretKeyFactory skf =
            SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
           return skf.generateSecret(spec).getEncoded();
        } catch (NoSuchAlgorithmException | InvalidKeySpecException e)
        {
            throw new AssertionError("Error while hashing password: " +
            e.getMessage(), e);
        } finally
        {
           spec.clearPassword();
        }
    }
    public static String generateSecurePassword(String password, String salt)
    {
        String returnValue = null;
        byte[] securePassword = hash(password.toCharArray(), salt.getBytes());

        returnValue = Base64.getEncoder().encodeToString(securePassword);

        return returnValue;
    }

    public static boolean verifyUserPassword(String providedPassword,
            String securedPassword, String salt)
    {
        boolean returnValue = false;

        String newSecurePassword = generateSecurePassword(providedPassword, salt);

        returnValue = newSecurePassword.equalsIgnoreCase(securedPassword);

        return returnValue;
    }

    public static String generateSessionID()
    {
        String suuid = UUID.randomUUID().toString();
        return suuid;
    }
}
```