CRYPTOGRAPHIC ALGORITHMS AND ARCHITECTURE USING JAVA - 19CSH21

MINI PROJECT REPORT

IMPLEMENTATION OF FOUR PILLARS OF CYBER SECURITY IN FILE SHARING SYSTEM

DHARSHINI R (21BAD008)

SANDHIYA DEVI B (21BAD022)

SUJI S (21BAD026)

B.Tech Artificial Intelligence and Data Science,

Mepco Schlenk Engineering College, Sivakasi.

Problem Statement

File Sharing has grown in popularity and frequency as people work remotely and enterprises move to the cloud. However, any time employees use technology to share files between devices, there are security risks involved. File sharing can introduce risks of malware infection, hacking, and loss or exposure of sensitive information. Without proper security measures in place, the benefits of file sharing can be significantly outweighed by the potential for exposing your company’s sensitive data to new security threats.

Objective

To ensure the safety of the file and preventing it from security risks by implementing the four pillars of cyber security namely Authorization, Authentication, Confidentiality and Integrity.

Implementation

Frontend – using Java

//Client Side

```
import javax.swing.*;
import java.awt.*;
import javax.swing.border.EmptyBorder;
import java.awt.event.*;
import java.sql.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.file.Files;
import java.security.*;
import javax.crypto.*;
import javax.crypto.spec.SecretKeySpec;
import java.io.FileOutputStream;
public class Client extends JFrame implements ActionListener{
   JLabel l1,l2,l3;
   JPanel p;
```

```
JFrame frame;
JTextField t1;
JTextField t2;
JButton b;
JCheckBox showPassword;
Statement st;
Connection conn;
ResultSet rs;
public Client()
{
   //LOGIN FRAME CREATION
   //FOR IMPLEMENTATION OF AUTHENTICATION
   showPassword=new JCheckBox("Show Password");
   l1=new JLabel("USERNAME");
   l2=new JLabel("PASSWORD");
   t1=new JTextField(15);
   t2=new JTextField(15);
   b=new JButton("LOGIN");
   b.addActionListener(this);
   p=new JPanel(new GridLayout(4, 2,2,5));
   p.add(l1);
   p.add(t1);
   p.add(l2);
   p.add(t2);
   p.add(showPassword);
   p.setVisible(true);
   frame=new JFrame();
   frame.add(p, BorderLayout.CENTER);
   frame.add(b, BorderLayout.SOUTH);
   frame.setSize(400,200);
   frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```
    frame.setVisible(true);
    frame.setTitle("Client Login Page");
    frame.setResizable(false);

}
//IMPLEMENTATION OF CONFIDENTIALITY USING THE AES ALGORITHM
//UTILITY FUNCTION FOR FILE ENCRYTION
private static byte[] encrypt(byte[] input, String password) throws Exception {
    SecretKeySpec keySpec = getKeySpec(password);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.ENCRYPT_MODE, keySpec);
    return cipher.doFinal(input);
}
//UTILITY FUNCTION FOR FILE DECRYPTION
private static byte[] decrypt(byte[] input, String password) throws Exception {
    SecretKeySpec keySpec = getKeySpec(password);
    Cipher cipher = Cipher.getInstance("AES");
    cipher.init(Cipher.DECRYPT_MODE, keySpec);
    return cipher.doFinal(input);
}
 //UTILITY FUNCTION FOR GENERATING SECRET KEY
   private static SecretKeySpec getKeySpec(String password) throws Exception {
    byte[] keyBytes = password.getBytes();
    MessageDigest sha = MessageDigest.getInstance("SHA-256");
    keyBytes = sha.digest(keyBytes);
    keyBytes = copyOf(keyBytes, 16);
    return new SecretKeySpec(keyBytes, "AES");
}

private static byte[] copyOf(byte[] original, int newLength) {
    byte[] copy = new byte[newLength];
    System.arraycopy(original, 0, copy, 0, Math.min(original.length, newLength));
    return copy;
}

private static void saveToFile(byte[] content, String filePath) throws Exception {
    FileOutputStream outputStream = new FileOutputStream(filePath);
    outputStream.write(content);
    outputStream.close();
}
//FRAME FOR THE CLIENT SIDE USAGE
public void nextFrame()
{
    final File[] fileToSend=new File[1];
    JFrame jFrame=new JFrame("Client Side");
    jFrame.setSize(900,900);
    jFrame.setLayout(new BoxLayout(jFrame.getContentPane(),BoxLayout.Y_AXIS));
    jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JLabel jTitle= new JLabel("File Sender");
    jTitle.setFont(new Font("Arial",Font.BOLD,25));
    jTitle.setBorder(new EmptyBorder(20,0,10,0));
    jTitle.setAlignmentX(CENTER_ALIGNMENT);
    JLabel jFileName=new JLabel("Choose a file to send");
    jFileName.setFont(new Font("Arial",Font.BOLD,20));
    jFileName.setBorder(new EmptyBorder(50,0,0,0));
    jFileName.setAlignmentX(CENTER_ALIGNMENT);
    JPanel jpbutton=new JPanel();
    JPanel jpbutton1=new JPanel();
    jpbutton1.setBorder(new EmptyBorder(150,0,10,0));
    jpbutton.setBorder(new EmptyBorder(75,0,10,0));
    JButton encrytpButton=new JButton("Encrypt File");
    encrytpButton.setPreferredSize(new Dimension(150,75));
encrytpButton.setFont(new Font("Arial",Font.BOLD,20));

JButton decrypButton=new JButton("Decrypt File");
decrypButton.setPreferredSize(new Dimension(150,75));
decrypButton.setFont(new Font("Arial",Font.BOLD,20));
jpbutton1.add(encrytpButton);
jpbutton1.add(decrypButton);
JButton jbSendFile=new JButton("Send File");
jbSendFile.setPreferredSize(new Dimension(150,75));
jbSendFile.setFont(new Font("Arial",Font.BOLD,20));
JButton jbChooseFile=new JButton("Choose File");
jbChooseFile.setPreferredSize(new Dimension(150,75));
jbChooseFile.setFont(new Font("Arial",Font.BOLD,20));
jpbutton.add(jbSendFile);
jpbutton.add(jbChooseFile);
JButton open_e=new JButton("Open Encrypt File");
open_e.setPreferredSize(new Dimension(200,75));
open_e.setFont(new Font("Arial",Font.BOLD,20));
jpbutton1.add(open_e);
JButton check=new JButton("Check Integrity");
check.setPreferredSize(new Dimension(200,75));
check.setFont(new Font("Arial",Font.BOLD,20));
jpbutton1.add(check);
JButton open_d=new JButton("Open Decrypt File");
open_d.setPreferredSize(new Dimension(200,75));
open_d.setFont(new Font("Arial",Font.BOLD,20));
jpbutton1.add(open_d);

//FUNCTION FOR CHOOSING THE FILE FROM THE DESKTOP

jbChooseFile.addActionListener(new ActionListener(){
```

```
            public void actionPerformed(ActionEvent e)
            {
                JFileChooser jFileChooser=new JFileChooser();
                jFileChooser.setDialogTitle("Choose file to send");
                if(jFileChooser.showOpenDialog(null)==JFileChooser.APPROVE_OPTION)
                {
                    fileToSend[0]=jFileChooser.getSelectedFile();
                    jFileName.setText("The file you want to send is "+fileToSend[0].getName());

                }
                JOptionPane.showMessageDialog((Component)e.getSource(),"File            uploaded
","Message",JOptionPane.PLAIN_MESSAGE);
            }
        });

        jbSendFile.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                if(fileToSend[0]==null)
                {
                    jFileName.setText("Please Choose a file first");

                }
                else{

                    JOptionPane.showMessageDialog((Component)e.getSource(),"File            sent
!!","Message",JOptionPane.PLAIN_MESSAGE);

                }
            }
        });

        //ACTION LISTENER TO ENCRYPT THE SELECTED FILE
        encrytpButton.addActionListener(new ActionListener(){
           public void actionPerformed(ActionEvent e)
           {
              try {
                 String inputFilePath = fileToSend[0].getAbsolutePath();
                 String outputFilePath = "C:/Users/maha9/OneDrive/Desktop/encrypted.txt";
                 String password = "mySecretPassword";

                 byte[] input = Files.readAllBytes(new File(inputFilePath).toPath());

                 byte[] encrypted = encrypt(input, password);

                 saveToFile(encrypted, outputFilePath);
                 JOptionPane.showMessageDialog((Component)e.getSource(),"File
  Encrypted","Message",JOptionPane.PLAIN_MESSAGE);



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

           //ACTION LISTENER TO OPEN THE ENCRYTED FILE
           open_e.addActionListener(new ActionListener(){
              public void actionPerformed(ActionEvent e)
              {
                 try {
                    File file = new File("C:/Users/maha9/OneDrive/Desktop/encrypted.txt");
                    Desktop desktop = Desktop.getDesktop();
                    desktop.open(file);
                } catch (IOException e10) {
                    e10.printStackTrace();
                }
            }});

        //ACTION LISTENER TO DECRYPT THE SELECTED FILE
        decrypButton.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e)
            {
                try {
                    String inputFilePath = fileToSend[0].getAbsolutePath(); // replace with the path of
your input file
                    String outputFilePath = "C:/Users/maha9/OneDrive/Desktop/encrypted.txt"; //
replace with the path where you want to save the encrypted file
                    String decryptedFilePath = "C:/Users/maha9/OneDrive/Desktop/decrypted.txt";
// replace with the path where you want to save the decrypted file
                    String password = "mySecretPassword"; // replace with your own secret password

                    byte[] input = Files.readAllBytes(new File(inputFilePath).toPath());

                    byte[] encrypted = encrypt(input, password);

                    saveToFile(encrypted, outputFilePath);
                    byte[] decrypted = decrypt(encrypted, password);

                    saveToFile(decrypted, decryptedFilePath);
                    JOptionPane.showMessageDialog((Component)e.getSource(),"File
Decrypted","Message",JOptionPane.PLAIN_MESSAGE);
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }});
    //ACTION LISTENER TO OPEN THE DECRYPTED FILE
    open_d.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e)
        {
            try {
                File file = new File("C:/Users/maha9/OneDrive/Desktop/decrypted.txt");
                Desktop desktop = Desktop.getDesktop();
                desktop.open(file);
            } catch (IOException e10) {
                e10.printStackTrace();
            }
        }});
        //CREATING THE HASH VALUE TO CHECK INTEGRITY
        check.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent e)
            {
                String filePath = fileToSend[0].getAbsolutePath().toString();
try {
    MessageDigest md = MessageDigest.getInstance("MD5");
    FileInputStream fis = new FileInputStream(filePath);
    byte[] buffer = new byte[1024];
    int nread;
    while ((nread = fis.read(buffer)) !=-1) {
        md.update(buffer, 0, nread);
    }
    byte[] hash = md.digest();
    // Convert the byte array to a hex string
```

```
            StringBuilder sb = new StringBuilder();
            for (byte b : hash) {
              sb.append(String.format("%02x", b));
            }
            String md5Hash = sb.toString();
            System.out.println("MD5 hash value: " + md5Hash);
            if(md5Hash.equals("04afc2aa639e376d135dd0df5d2256f6"))
            {
              //"04afc2aa639e376d135dd0df5d2256f6"---> already generated hash value for the
particular file(ex1_ans.txt)
              JOptionPane.showMessageDialog((Component)e.getSource(),"No
modifications","Hash value",JOptionPane.PLAIN_MESSAGE);
            }
            else{
              JOptionPane.showMessageDialog((Component)e.getSource(),"modifications
happened erorr!!!","Hash value",JOptionPane.PLAIN_MESSAGE);
            }
            fis.close();
          } catch (NoSuchAlgorithmException | IOException e1) {
            e1.printStackTrace();
          }
                    }});
        jFrame.add(jTitle);
        jFrame.add(jFileName);
        jFrame.add(jpbutton);
        jFrame.add(jpbutton1);
        jFrame.setVisible(true);
```

```
  }
  //AUTHENTICATION AND AUTHORIZATION CHECKING FRAME USING THE JDBC CONCEPT(password)
  public void actionPerformed(ActionEvent evt)
  {
    try{
      if(evt.getSource()==b){
        PreparedStatement pst;
        String s=t1.getText();
        pst=conn.prepareStatement("select password,role from Users where name=?");
        pst.setString(1, s);
        rs=pst.executeQuery();
        int flag=0;
        while(rs.next()){
          String i=rs.getString("password");
          String i1=rs.getString("role");
          if(i.equals(t2.getText())&&i1.equals("c"))//password checking and role(client ('c')or server('s')--implementation of authorization)
          {
            flag=1;
            JOptionPane.showMessageDialog((Component)evt.getSource(),"Login Successfull","Message",JOptionPane.PLAIN_MESSAGE);
            nextFrame();
          }
        }

        if(flag==0){
          JOptionPane.showMessageDialog((Component)evt.getSource(),"Invalid password check your role correspondingly","Message",JOptionPane.PLAIN_MESSAGE);
        }
  }
}
```

```
catch(Exception e)
{
}
}
//CREATING THE CONNECTION BETWEEN THE DATABASE AND THE JAVA CODE
public void getcon(){
   try{
      System.out.println("Hello, World!");
      Class.forName("com.mysql.cj.jdbc.Driver");
      String url1 = "jdbc:mysql://localhost:3306/cyber_proj_db?useSSL=false";
      String user = "root";
      String password1 = "dharshini1109";
      conn = DriverManager.getConnection(url1, user, password1);
      System.out.println("Loaded success");
   }
   catch(Exception e){
      System.out.println("Failed");
   }
 }
 //MAIN
 public static void main(String[] args) throws Exception {
   Client aobj=new Client();
   aobj.getcon();
   }
}
 //SERVER
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.border.EmptyBorder;
```

```
public class Server extends JFrame implements ActionListener{
   JLabel l1,l2,l3;
   JPanel p;
   JFrame frame;
   JTextField t1;
   JTextField t2;
   JButton b;
   JCheckBox showPassword;
   Statement st;
   Connection conn;
   ResultSet rs;
   //LOGIN FRAME CREATION FOR SERVER
   public Server()
   {
      showPassword=new JCheckBox("Show Password");
      l1=new JLabel("USERNAME");
      l2=new JLabel("PASSWORD");
      t1=new JTextField(15);
      t2=new JTextField(15);
      b=new JButton("LOGIN");
      b.addActionListener(this);
      p=new JPanel(new GridLayout(4, 2,2,5));
      p.add(l1);
      p.add(t1);
      p.add(l2);
      p.add(t2);
      p.add(showPassword);
      p.setVisible(true);
      frame=new JFrame();
      frame.add(p, BorderLayout.CENTER);
      frame.add(b, BorderLayout.SOUTH);
```

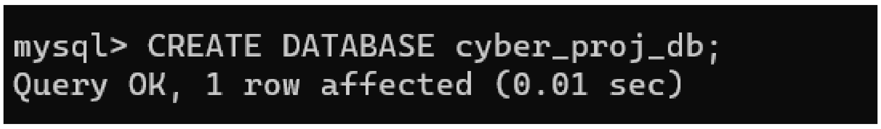
```
        frame.setSize(400,200);
        frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
        frame.setVisible(true);
        frame.setTitle("Login Page");
        frame.setResizable(false);


    }
    public void nextFrame()
    {
        JFrame jFrame=new JFrame("Server");
        jFrame.setSize(900,900);
        jFrame.setLayout(new BoxLayout(jFrame.getContentPane(),BoxLayout.Y_AXIS));
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel jTitle= new JLabel("File Receiver");
        jTitle.setFont(new Font("Arial",Font.BOLD,25));
        jTitle.setBorder(new EmptyBorder(20,0,10,0));
        jTitle.setAlignmentX(CENTER_ALIGNMENT);
        jFrame.add(jTitle);
        jFrame.setVisible(true);
    }
```

//AUTHENTICATION AND AUTHORIZATION CHECKING FRAME USING THE JDBC CONCEPT(password)

```
    public void actionPerformed(ActionEvent evt)
    {
        try{
            if(evt.getSource()==b){
                PreparedStatement pst;
                String s=t1.getText();
                pst=conn.prepareStatement("select password,role from Users where name=?");
                pst.setString(1, s);
                rs=pst.executeQuery();
                int flag=0;
                while(rs.next()){
                   String i=rs.getString("password");
                   String i1=rs.getString("role");
                   if(i.equals(t2.getText())&&i1.equals("s"))//password checking and role(client ('c')or server('s')--implementation of authorization)
                   {
                      flag=1;
                      JOptionPane.showMessageDialog((Component)evt.getSource(),"Login Successfull","Message",JOptionPane.PLAIN_MESSAGE);
                      nextFrame();
                   }
                }
                if(flag==0){
                   JOptionPane.showMessageDialog((Component)evt.getSource(),"Invalid    password and check your role correspondingly","Message",JOptionPane.PLAIN_MESSAGE);
                }
    }
 }
 catch(Exception e)
 {
 }
 }
 //CREATING THE CONNECTION BETWEEN THE DATABASE AND THE JAVA CODE
 public void getcon(){
    try{
        System.out.println("Hello, World!");
        Class.forName("com.mysql.cj.jdbc.Driver");
        String url1 = "jdbc:mysql://localhost:3306/cyber_proj_db?useSSL=false";
        String user = "root";
        String password = "dharshini1109";
        conn = DriverManager.getConnection(url1, user, password);
```

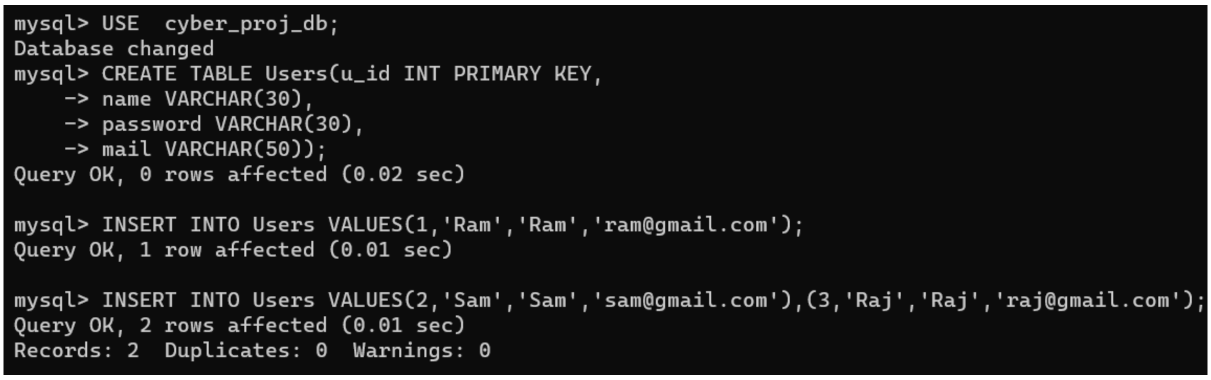
```
        System.out.println("Loaded success");
    }
    catch(Exception e){
        System.out.println("Failed");
    }
 }
//MAIN
 public static void main(String[] args) throws Exception {
    Server aobj=new Server();
    aobj.getcon();
    }
}
```

Backend – Database creation using mySQL



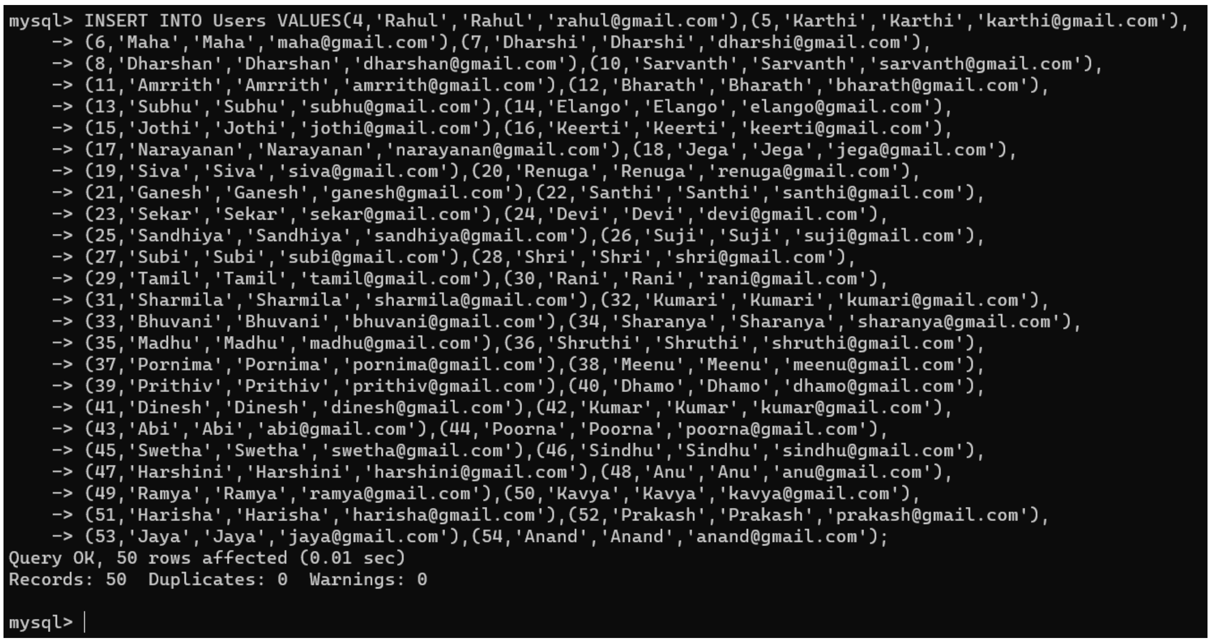
```
mysql> CREATE DATABASE cyber_proj_db;
Query OK, 1 row affected (0.01 sec)
```

Figure 1: Database Creation



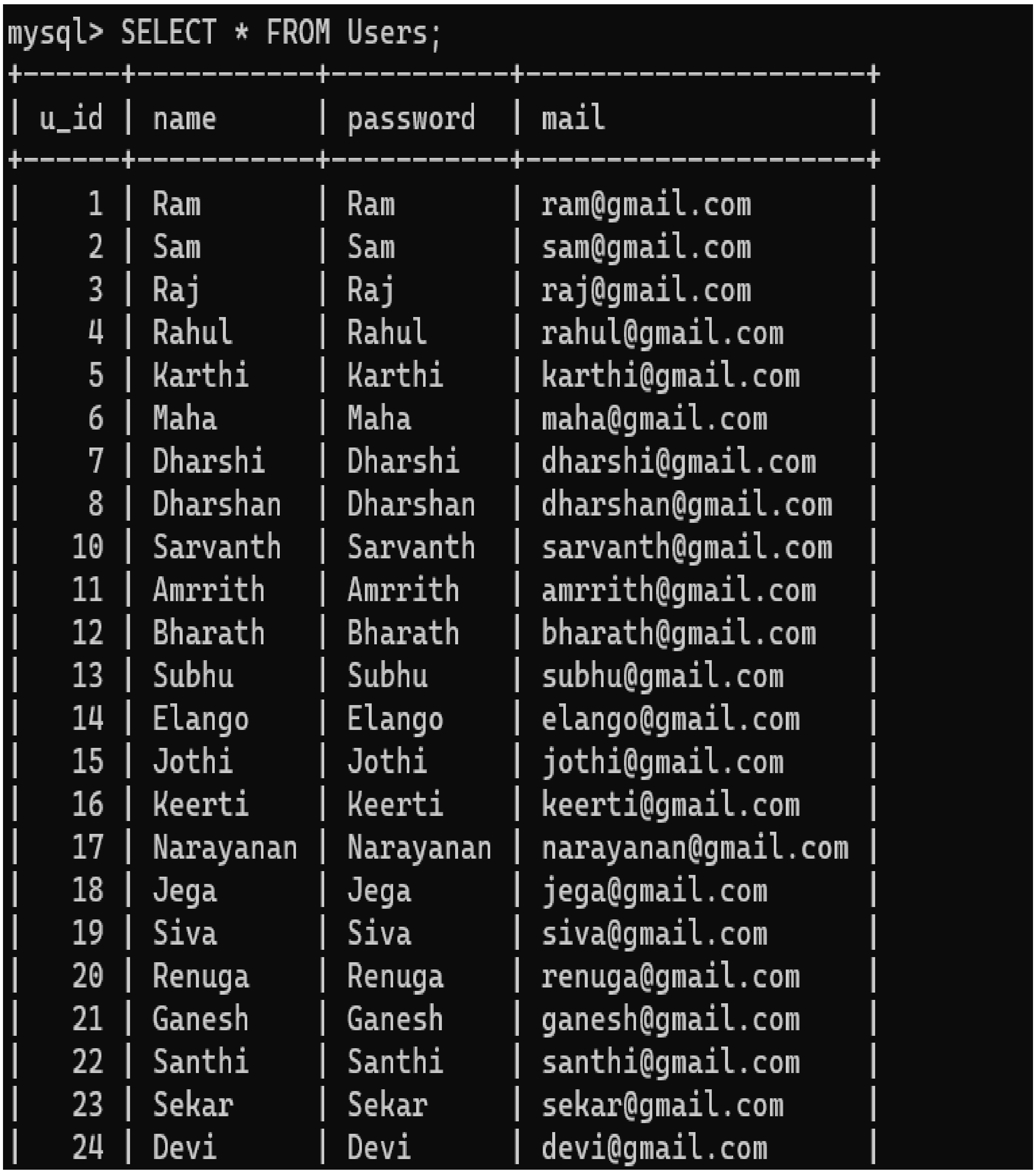
```
mysql> USE  cyber_proj_db;
Database changed
mysql> CREATE TABLE Users(u_id INT PRIMARY KEY,
    -> name VARCHAR(30),
    -> password VARCHAR(30),
    -> mail VARCHAR(50));
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Users VALUES(1,'Ram','Ram','ram@gmail.com');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO Users VALUES(2,'Sam','Sam','sam@gmail.com'),(3,'Raj','Raj','raj@gmail.com');
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

Figure 2: Creating Table USERS to store user details for AUTHORIZATION



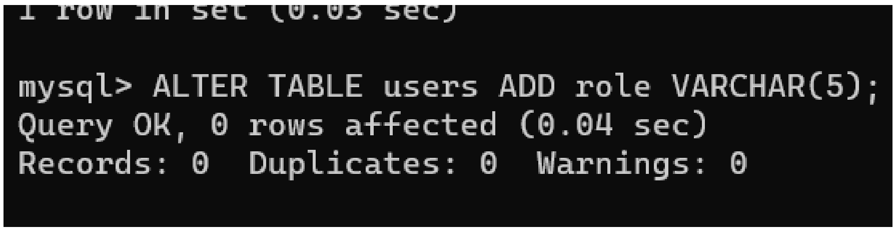
```
mysql> INSERT INTO Users VALUES(4,'Rahul','Rahul','rahul@gmail.com'),(5,'Karthi','Karthi','karthi@gmail.com'),
    -> (6,'Maha','Maha','maha@gmail.com'),(7,'Dharshi','Dharshi','dharshi@gmail.com'),
    -> (8,'Dharshan','Dharshan','dharshan@gmail.com'),(10,'Sarvanth','Sarvanth','sarvanth@gmail.com'),
    -> (11,'Amrrith','Amrrith','amrrith@gmail.com'),(12,'Bharath','Bharath','bharath@gmail.com'),
    -> (13,'Subhu','Subhu','subhu@gmail.com'),(14,'Elango','Elango','elango@gmail.com'),
    -> (15,'Jothi','Jothi','jothi@gmail.com'),(16,'Keerti','Keerti','keerti@gmail.com'),
    -> (17,'Narayanan','Narayanan','narayanan@gmail.com'),(18,'Jega','Jega','jega@gmail.com'),
    -> (19,'Siva','Siva','siva@gmail.com'),(20,'Renuga','Renuga','renuga@gmail.com'),
    -> (21,'Ganesh','Ganesh','ganesh@gmail.com'),(22,'Santhi','Santhi','santhi@gmail.com'),
    -> (23,'Sekar','Sekar','sekar@gmail.com'),(24,'Devi','Devi','devi@gmail.com'),
    -> (25,'Sandhiya','Sandhiya','sandhiya@gmail.com'),(26,'Suji','Suji','suji@gmail.com'),
    -> (27,'Subi','Subi','subi@gmail.com'),(28,'Shri','Shri','shri@gmail.com'),
    -> (29,'Tamil','Tamil','tamil@gmail.com'),(30,'Rani','Rani','rani@gmail.com'),
    -> (31,'Sharmila','Sharmila','sharmila@gmail.com'),(32,'Kumari','Kumari','kumari@gmail.com'),
    -> (33,'Bhuvani','Bhuvani','bhuvani@gmail.com'),(34,'Sharanya','Sharanya','sharanya@gmail.com'),
    -> (35,'Madhu','Madhu','madhu@gmail.com'),(36,'Shruthi','Shruthi','shruthi@gmail.com'),
    -> (37,'Pornima','Pornima','pornima@gmail.com'),(38,'Meenu','Meenu','meenu@gmail.com'),
    -> (39,'Prithiv','Prithiv','prithiv@gmail.com'),(40,'Dhamo','Dhamo','dhamo@gmail.com'),
    -> (41,'Dinesh','Dinesh','dinesh@gmail.com'),(42,'Kumar','Kumar','kumar@gmail.com'),
    -> (43,'Abi','Abi','abi@gmail.com'),(44,'Poorna','Poorna','poorna@gmail.com'),
    -> (45,'Swetha','Swetha','swetha@gmail.com'),(46,'Sindhu','Sindhu','sindhu@gmail.com'),
    -> (47,'Harshini','Harshini','harshini@gmail.com'),(48,'Anu','Anu','anu@gmail.com'),
    -> (49,'Ramya','Ramya','ramya@gmail.com'),(50,'Kavya','Kavya','kavya@gmail.com'),
    -> (51,'Harisha','Harisha','harisha@gmail.com'),(52,'Prakash','Prakash','prakash@gmail.com'),
    -> (53,'Jaya','Jaya','jaya@gmail.com'),(54,'Anand','Anand','anand@gmail.com');
Query OK, 50 rows affected (0.01 sec)
Records: 50  Duplicates: 0  Warnings: 0

mysql> |
```

Figure 3: Inserting record into table -Users



```
mysql> SELECT * FROM Users;
+------+-----------+-----------+---------------------+
| u_id | name      | password  | mail                |
+------+-----------+-----------+---------------------+
|    1 | Ram       | Ram       | ram@gmail.com       |
|    2 | Sam       | Sam       | sam@gmail.com       |
|    3 | Raj       | Raj       | raj@gmail.com       |
|    4 | Rahul     | Rahul     | rahul@gmail.com     |
|    5 | Karthi    | Karthi    | karthi@gmail.com    |
|    6 | Maha      | Maha      | maha@gmail.com      |
|    7 | Dharshi   | Dharshi   | dharshi@gmail.com   |
|    8 | Dharshan  | Dharshan  | dharshan@gmail.com  |
|   10 | Sarvanth  | Sarvanth  | sarvanth@gmail.com  |
|   11 | Amrrith   | Amrrith   | amrrith@gmail.com   |
|   12 | Bharath   | Bharath   | bharath@gmail.com   |
|   13 | Subhu     | Subhu     | subhu@gmail.com     |
|   14 | Elango    | Elango    | elango@gmail.com    |
|   15 | Jothi     | Jothi     | jothi@gmail.com     |
|   16 | Keerti    | Keerti    | keerti@gmail.com    |
|   17 | Narayanan | Narayanan | narayanan@gmail.com |
|   18 | Jega      | Jega      | jega@gmail.com      |
|   19 | Siva      | Siva      | siva@gmail.com      |
|   20 | Renuga    | Renuga    | renuga@gmail.com    |
|   21 | Ganesh    | Ganesh    | ganesh@gmail.com    |
|   22 | Santhi    | Santhi    | santhi@gmail.com    |
|   23 | Sekar     | Sekar     | sekar@gmail.com     |
|   24 | Devi      | Devi      | devi@gmail.com      |
```

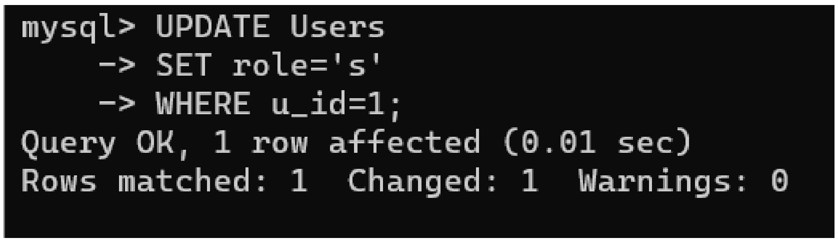
Figure 4: Records in Users



```
1 row in set (0.03 sec)

mysql> ALTER TABLE users ADD role VARCHAR(5);
Query OK, 0 rows affected (0.04 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

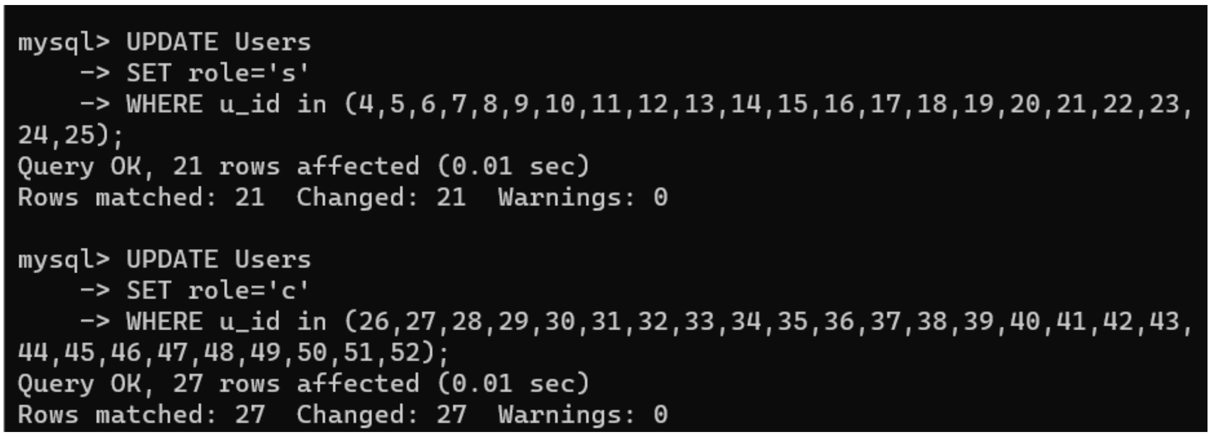
Figure 5: Updating Table with the column role for AUTHENTICATION

Server i.e., receivers can log into the Server module only whereas Clients i.e., File senders can log into the Client Module.



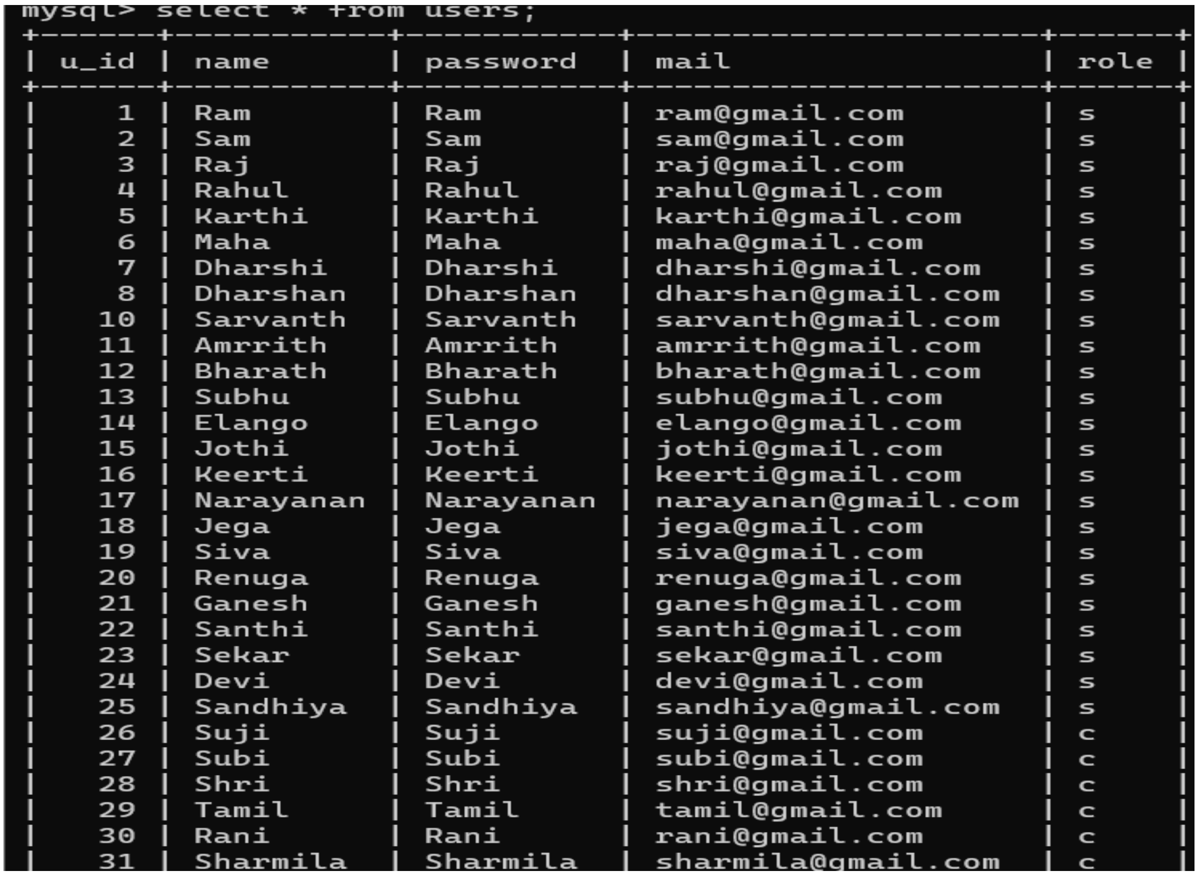
```
mysql> UPDATE Users
    -> SET role='s'
    -> WHERE u_id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Figure 6: Updating record for the role column (server – s, client -c)



```
mysql> UPDATE Users
    -> SET role='s'
    -> WHERE u_id in (4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
24,25);
Query OK, 21 rows affected (0.01 sec)
Rows matched: 21  Changed: 21  Warnings: 0

mysql> UPDATE Users
    -> SET role='c'
    -> WHERE u_id in (26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,
44,45,46,47,48,49,50,51,52);
Query OK, 27 rows affected (0.01 sec)
Rows matched: 27  Changed: 27  Warnings: 0
```

Figure 7: Updating the other records



```
mysql> select * from users;
+------+-----------+-----------+---------------------+------+
| u_id | name      | password  | mail                | role |
+------+-----------+-----------+---------------------+------+
|    1 | Ram       | Ram       | ram@gmail.com       | s    |
|    2 | Sam       | Sam       | sam@gmail.com       | s    |
|    3 | Raj       | Raj       | raj@gmail.com       | s    |
|    4 | Rahul     | Rahul     | rahul@gmail.com     | s    |
|    5 | Karthi    | Karthi    | karthi@gmail.com    | s    |
|    6 | Maha      | Maha      | maha@gmail.com      | s    |
|    7 | Dharshi   | Dharshi   | dharshi@gmail.com   | s    |
|    8 | Dharshan  | Dharshan  | dharshan@gmail.com  | s    |
|   10 | Sarvanth  | Sarvanth  | sarvanth@gmail.com  | s    |
|   11 | Amrrith   | Amrrith   | amrrith@gmail.com   | s    |
|   12 | Bharath   | Bharath   | bharath@gmail.com   | s    |
|   13 | Subhu     | Subhu     | subhu@gmail.com     | s    |
|   14 | Elango    | Elango    | elango@gmail.com    | s    |
|   15 | Jothi     | Jothi     | jothi@gmail.com     | s    |
|   16 | Keerti    | Keerti    | keerti@gmail.com    | s    |
|   17 | Narayanan | Narayanan | narayanan@gmail.com | s    |
|   18 | Jega      | Jega      | jega@gmail.com      | s    |
|   19 | Siva      | Siva      | siva@gmail.com      | s    |
|   20 | Renuga    | Renuga    | renuga@gmail.com    | s    |
|   21 | Ganesh    | Ganesh    | ganesh@gmail.com    | s    |
|   22 | Santhi    | Santhi    | santhi@gmail.com    | s    |
|   23 | Sekar     | Sekar     | sekar@gmail.com     | s    |
|   24 | Devi      | Devi      | devi@gmail.com      | s    |
|   25 | Sandhiya  | Sandhiya  | sandhiya@gmail.com  | s    |
|   26 | Suji      | Suji      | suji@gmail.com      | c    |
|   27 | Subi      | Subi      | subi@gmail.com      | c    |
|   28 | Shri      | Shri      | shri@gmail.com      | c    |
|   29 | Tamil     | Tamil     | tamil@gmail.com     | c    |
|   30 | Rani      | Rani      | rani@gmail.com      | c    |
|   31 | Sharmila  | Sharmila  | sharmila@gmail.com  | c    |
```

Figure 8: Updated records in the database

Algorithms used to achieve the four pillars

- AES to achieve confidentiality.
- MD5 to achieve integrity.
- Authorization is achieved by checking the access permission of a user as server or client that is defined in database.
- Authentication is achieved by checking the credentials that are predefined in the database.

AES Algorithm

The Advanced Encryption Standard (AES) is a symmetric block cipher chosen by the U.S. government to protect classified information.

AES is implemented in software and hardware throughout the world to encrypt sensitive data. It is essential for government computer security, cybersecurity and electronic data protection.

Working Concept of AES algorithm

AES includes three block ciphers:

- AES-128 uses a 128-bit key length to encrypt and decrypt a block of messages.
- AES-192 uses a 192-bit key length to encrypt and decrypt a block of messages.
- AES-256 uses a 256-bit key length to encrypt and decrypt a block of messages.
- Each cipher encrypts and decrypts data in blocks of 128 bits using cryptographic keys of 128, 192 and 256 bits, respectively.

Symmetric, also known as secret key, ciphers use the same key for encrypting and decrypting. The sender and the receiver must both know -- and use -- the same secret key.

Advantages

Security → Competing algorithms were to be judged on their ability to resist attack as compared to other submitted ciphers. Security strength was to be considered the most important factor in the competition.

Cost → Intended to be released on a global, nonexclusive and royalty-free basis, the candidate algorithms were to be evaluated on computational and memory efficiency.

Implementation → Factors to be considered included the algorithm's flexibility, suitability for hardware or software implementation, and overall simplicity.

Message Digest Algorithm

Message Digest is used to ensure the integrity of a message transmitted over an insecure channel (where the content of the message can be changed). The message is passed through a Cryptographic hash function. This function creates a compressed image of the message called Digest.

This message and digest pair is equivalent to a physical document and fingerprint of a person on that document. Unlike the physical document and the fingerprint, the message and the digest can be sent separately.

Most importantly, the digest should be unchanged during the transmission. The cryptographic hash function is a one way function, that is, a function which is practically infeasible to invert. This cryptographic hash function takes a message of variable length as input and creates a digest / hash / fingerprint of fixed length, which is used to verify the integrity of the message.

Message digest ensures the integrity of the document. To provide authenticity of the message, digest is encrypted with sender's private key. Now this digest is called digital signature, which can be only decrypted by the receiver who has sender's public key. Now the receiver can authenticate the sender and also verify the integrity of the sent message.

Output

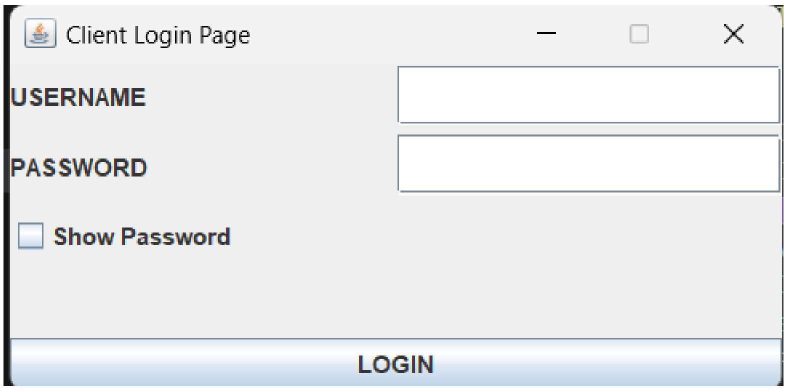
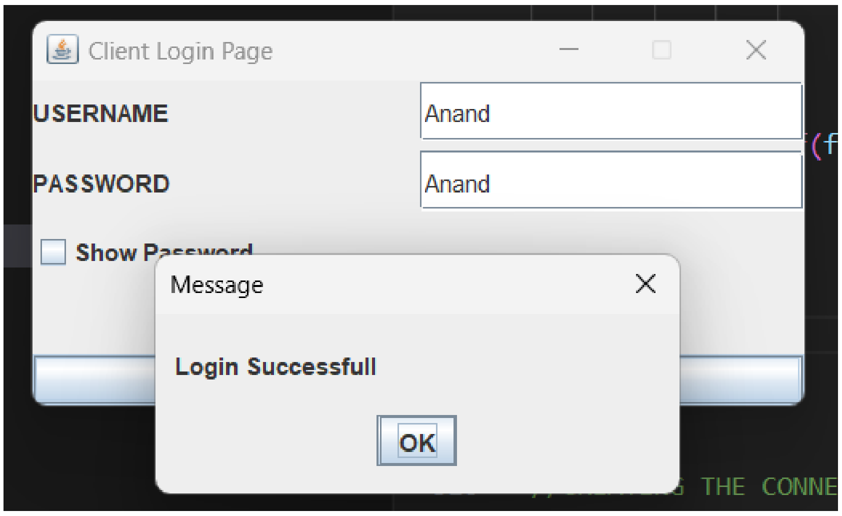


Figure 9: Client Login Page

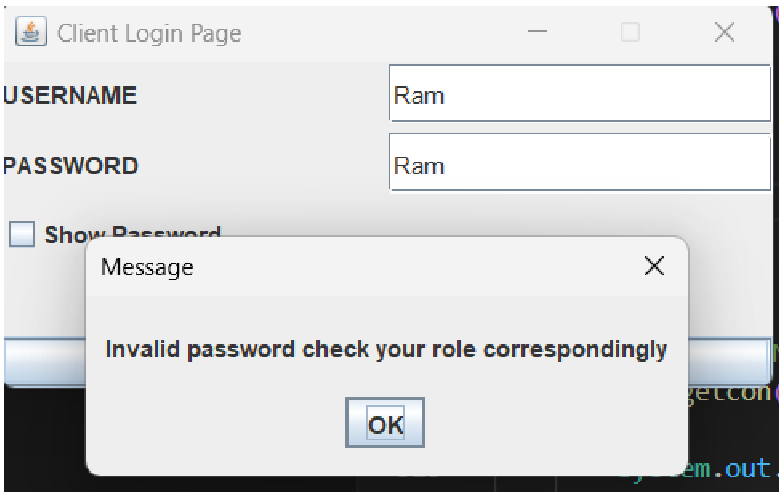


USERNAME Anand

PASSWORD Anand

Login Successfull

Figure 10: Anand is an authorized Client in database. Hence Login is Successful.

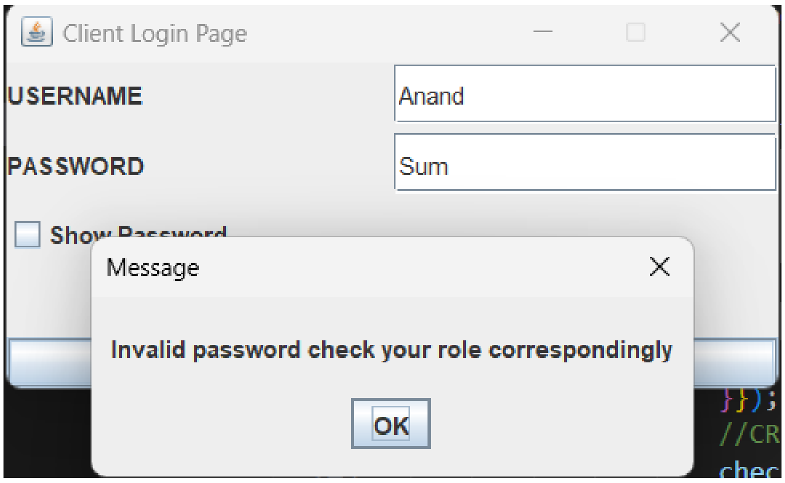


USERNAME Ram

PASSWORD Ram

Invalid password check your role correspondingly

Figure 11: Ram is an authorized server in database. Hence Client Login Failed.


Client Login Page

USERNAME Anand

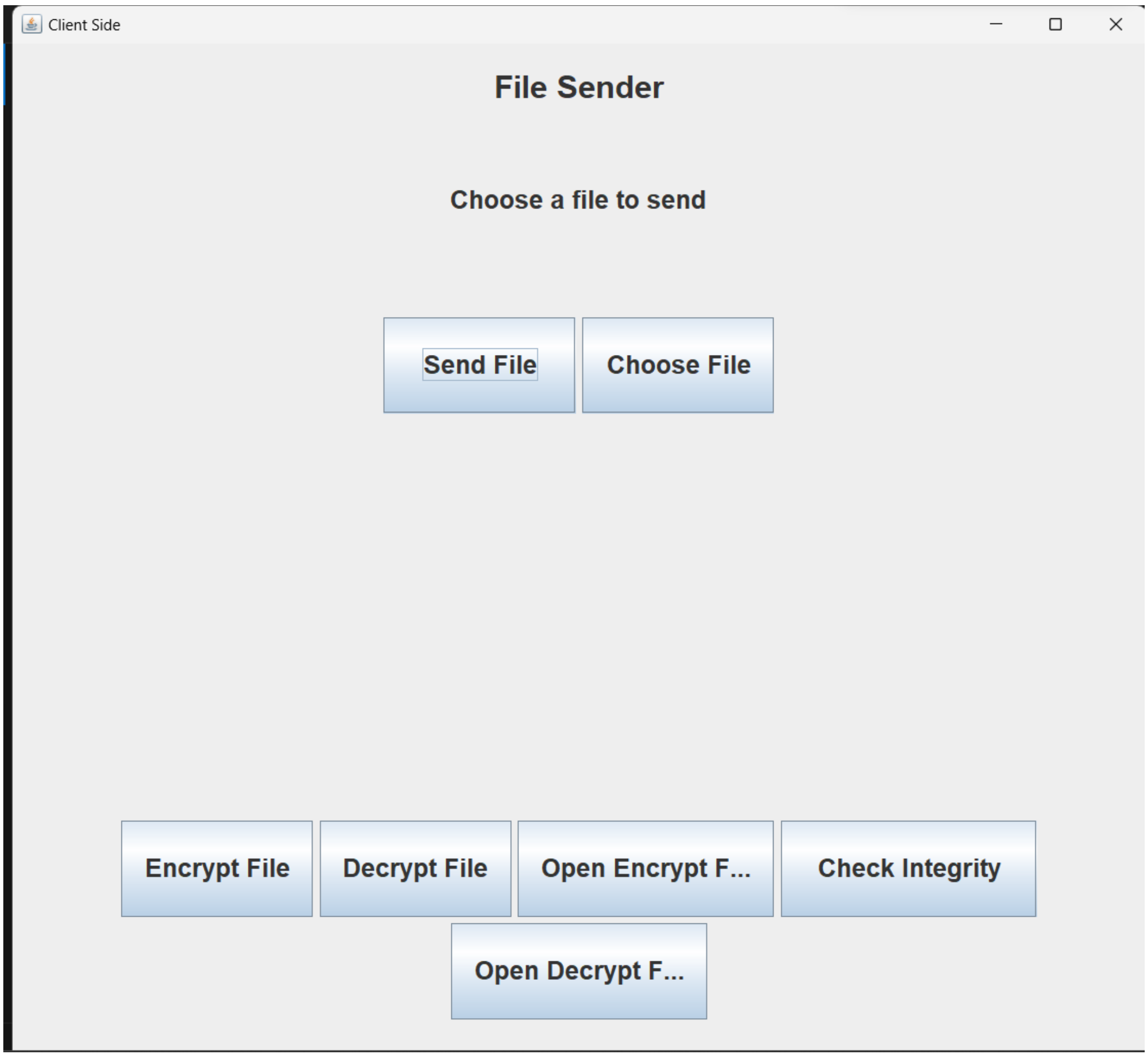
PASSWORD Sum

Show Password

Message

Invalid password check your role correspondingly

Figure 12: Here Login Failed as the entered password is incorrect.


Client Side

File Sender

Choose a file to send

Figure 13: This is the client side module

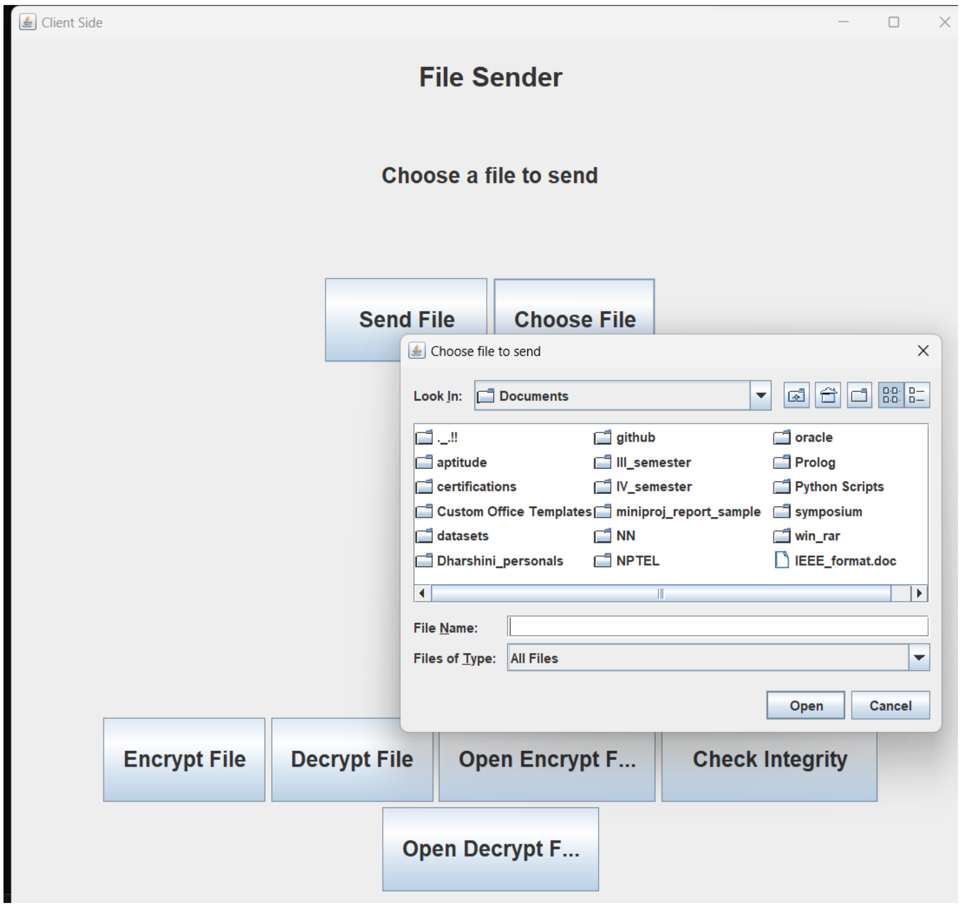
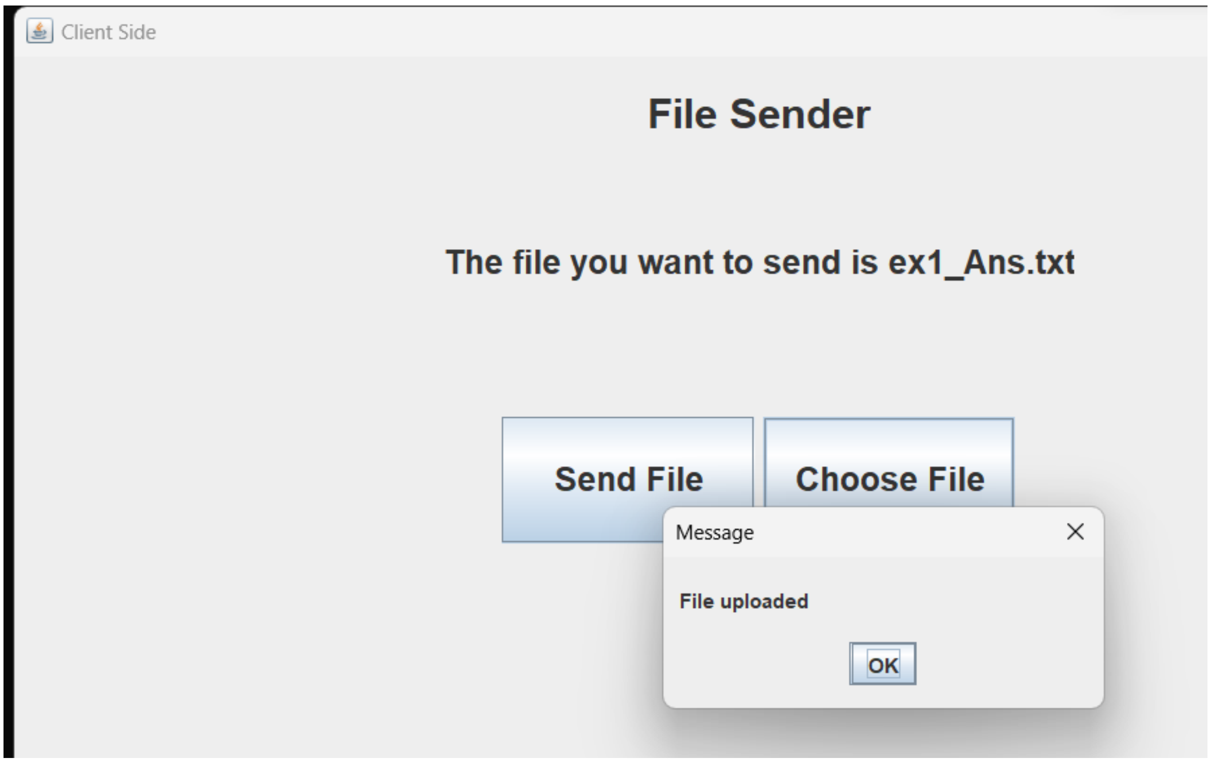


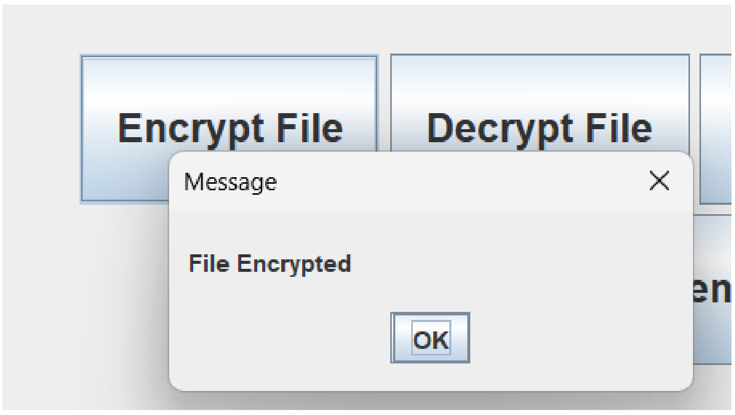
Figure 14: Choose File button is used to upload a file to be sent or perform other utilities



The file you want to send is ex1_Ans.txt

File uploaded

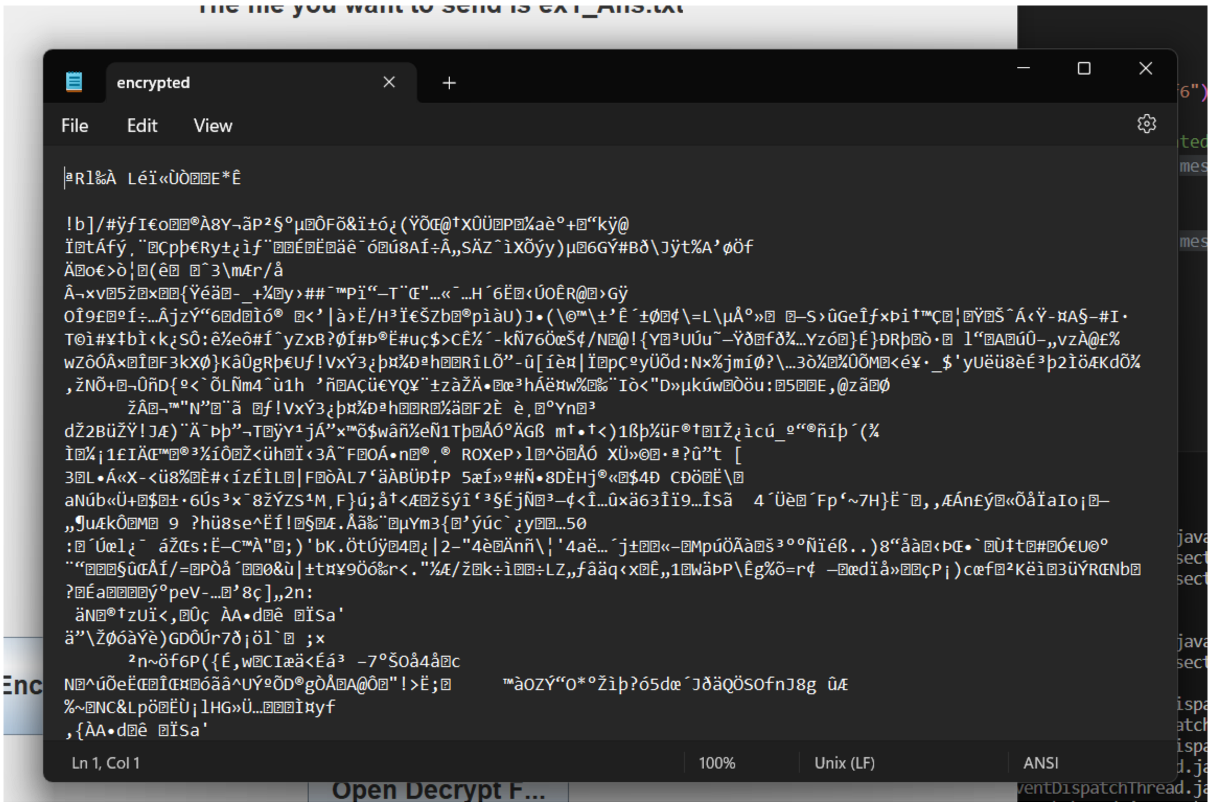
Figure 15: The file ex1_Ans.txt is selected and uploaded



File Encrypted

Figure 16: Encrypt File Button is used to encrypt the file

After encryption the pop-up message File Encrypted is displayed.



```
ªRl‰À Léï«ÙÒ□□E*Ê

!b]/#ÿƒI€o□□®À8Y¬ãP²§°µ□ÔF õ&ï±ó¿(ŸŒ@†XÛÜ□P□‰aè°+□“kÿ@
Ï□tÁfý¸¨□Çpþ€Ry±¿iƒ¨□□É□Ë□ãë¯õ□ú8AÍ÷Ã„SÄZˆìXÕýy)µ□6GÝ#Bð\Jÿt%A’øÖf
Ä□o€>ò¦□(ê□ □ˆ3\mÆr/å
Â¬xv□5ž□x□□{Ýéä□-_+¼□y›##¯™Pï“—T¨Œ"…«¨…H´6Ë□‹ÚOÊR@□›Gÿ
OÎ9£□ºÍ÷…ÃjzÝ“6□d□Ìó® □<’|à›É/H³Ï€ŠZb□®pìàU)J•(\©™\±’Ê´±Ø□¢\=L\µÅ°»□ □—S›ûGeÎƒ×Þi†™Ç□¦□ŸŠˆÁ‹Ÿ-¤A§—#I·
T©ì#¥‡bÌ‹k¿SÒ:ê½eô#Íˆ yZxB?ØÍ#Þ®Ë#uç$>CÊ½´-kÑ76ÖœŠ¢/N□@!{Y□³UÛu˜—Ÿð□fð¾…Yzó□}É}ÐRþ□ò·□ 1“□A□úÛ–„vzÀ@£%
wZôÓÂx□Î□F3kXØ}KâÛgRþ€Uƒ!VxÝ3¿þ¤%Ðªh□□RîLÕ”-û[íè¤|Ï□pÇºyÜÕd:N×%jmíØ?\…3ò¼□¼ÛÕM□<é¥·_$'yUëü8èÉ³þ2ÌöÆKdÕ%
,žNÕ+□¬ÛñD{º<`ÕLÑm4ˆù1h ’ñ□AÇü€YQ¥¨±àŽÄ•□œ³hÁë¤w%□‰¨Ìò<"D»µkúw□Òöu:□5□□E,@zã□Ø
      žÂ□¬™"N”□¨ã □ƒ!VxÝ3¿þ¤%Ðªh□□R□¼ä□F2È è¸□°Yn□³
dŽ2BüŽŸ!JÆ)¨Ä`Þþ”¬T□ÿY¹jÁ”x™õ$wâñ½eÑ1Tþ□ÅÓ°ÄGß m†•†<)1ßþ%üF®†□IŽ¿ìcú_º“®ñíþ´(¾
Ï□¼¡1£IÄŒ™□®³¼íÔ□Ž<üh□Ï‹3Ãˆ F□OÁ•n□®¸® ROXeP›1□^ö□ÅÓ XÜ»©□·ª?ü”t [
3□L•Á«X-<ü8%□È#‹ízÉÌL□|F□ÒAL7‘äÀBÜÐ‡P 5æÍ»º#Ñ•8DÈHj®«□$4Ð CÐöÐË\□
aNúb«Ü+□$□±·6Ús³x¨8žÝZS¹M¸F}ú;å†<Æ□žšýì‘³§ÉjÑ□³—¢<Î…û×ä63Îï9…ÎSã  4´Üè□´Fp‘~7H}Ë¨□,‚ÆÁn£ý□«ÕåÏaIo¡□—
„¶uÆkÔ□M□ 9 ?hü8se^ËÍ!□§□Æ.Åã‰¨□µYm3{□’ýúc`¿y□□…50
:□´Úœl¿¯ áŽŒs:Ë—C™À"□;)'bK.ÖtÚÿ□4□¿|2–"4è□Ãnñ\¦'4aë…´j±□□«—□MpúÖÃà□š³°°Ñïéß..)8“âà□‹ÞŒ•`□Ù‡t□#□Ó€U©°
¨“□□□§ûŒÅÍ/=□PÒå´□□0&ù|±t¤¥9ÖóÆr<."¼Æ/ž□k÷ì□□÷LZ„ƒãäq‹x□Ê„1□wäÞP\Êg%õ=r¢ —□œdìå»□□çP¡)cœf□²Këì□3üŸRŒNb□
?□Éa□□□□ý°peV-…□’8ç],,2n:
 äN□®†zUï<,□Ûç ÀA•d□ê □ÏSa'
ä”\ŽØóàÝè)GDÔÛr7ð¡öl`□ ;x
      ²n~öf6P({É,w□CIæä<Éá³ -7°ŠOå4â□c
N□^úÕeËŒ□ÎŒ¤□óãã^UÝºÕD®gÒÅ□A@Ô□"!>Ë;□     ™àOZÝ“O*°Žìþ?ó5dœ´JðäQÖSOfnJ8g ûÆ
%~□NC&LpöËÙ¡1HG»Ü…□□□Ì¤yf
,{ÀA•d□ê □ÏSa'
```

Ln 1, Col 1   100%   Unix (LF)   ANSI

Figure 17: Open Encrypt File Button is used to display the encrypted file.

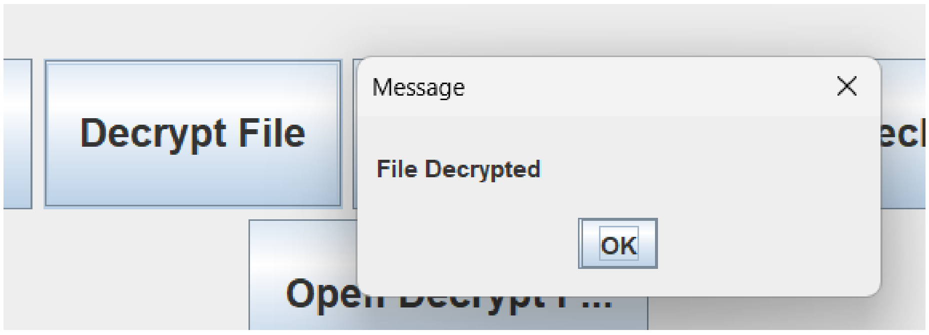
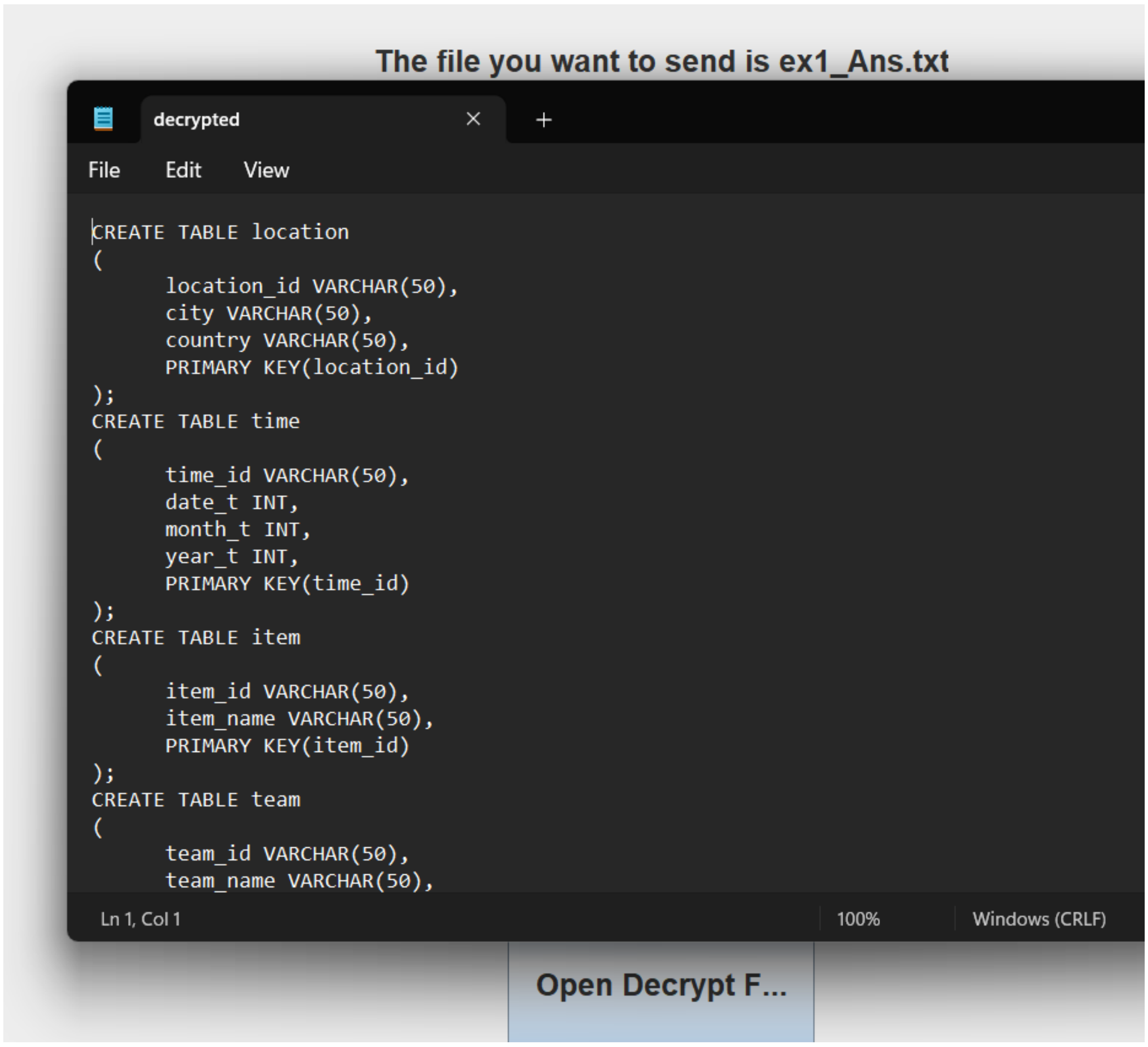


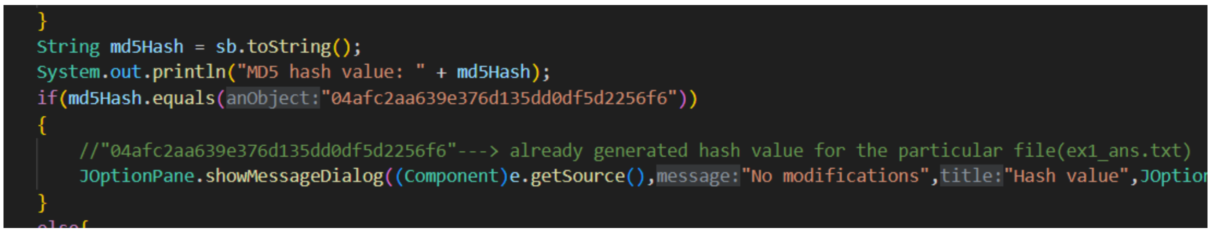
Figure 17: Decrypt File Button is used to decrypt the file

After decryption the pop-up message File Decrypted is displayed.



```
CREATE TABLE location
(
	location_id VARCHAR(50),
	city VARCHAR(50),
	country VARCHAR(50),
	PRIMARY KEY(location_id)
);
CREATE TABLE time
(
	time_id VARCHAR(50),
	date_t INT,
	month_t INT,
	year_t INT,
	PRIMARY KEY(time_id)
);
CREATE TABLE item
(
	item_id VARCHAR(50),
	item_name VARCHAR(50),
	PRIMARY KEY(item_id)
);
CREATE TABLE team
(
	team_id VARCHAR(50),
	team_name VARCHAR(50),
```

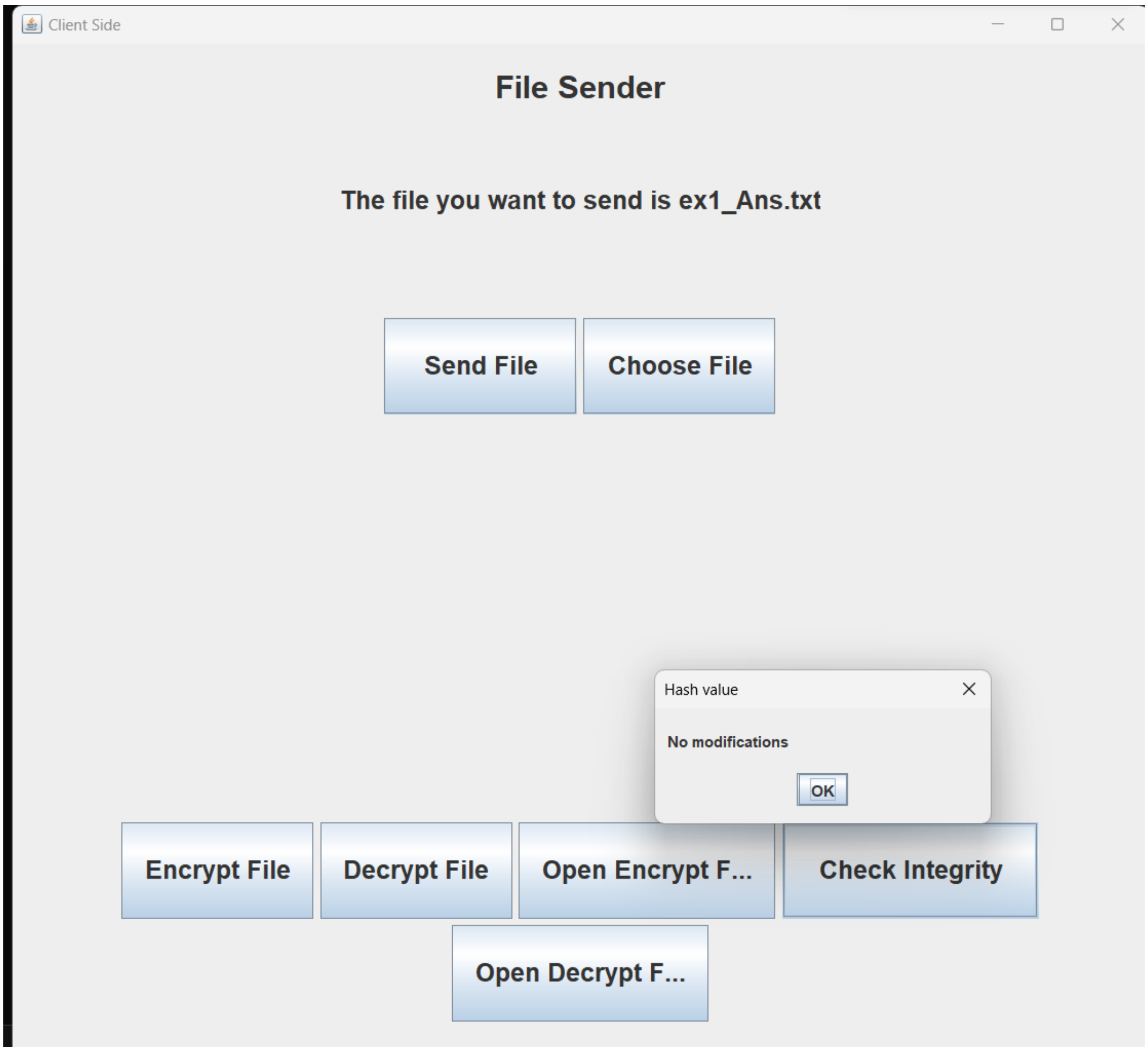
Figure 18: Open Decrypt File Button is used to display the decrypted file.



```
}
String md5Hash = sb.toString();
System.out.println("MD5 hash value: " + md5Hash);
if(md5Hash.equals(anObject:"04afc2aa639e376d135dd0df5d2256f6"))
{
    //"04afc2aa639e376d135dd0df5d2256f6"---> already generated hash value for the particular file(ex1_ans.txt)
    JOptionPane.showMessageDialog((Component)e.getSource(),message:"No modifications",title:"Hash value",JOptio
}
```

Figure 19: Here the hash value "04afc2aa639e376d135dd0df5d2256f6" is generated for the file ex1_Ans.txt

Then the integrity is checked for the particular file if modifications happened or not.



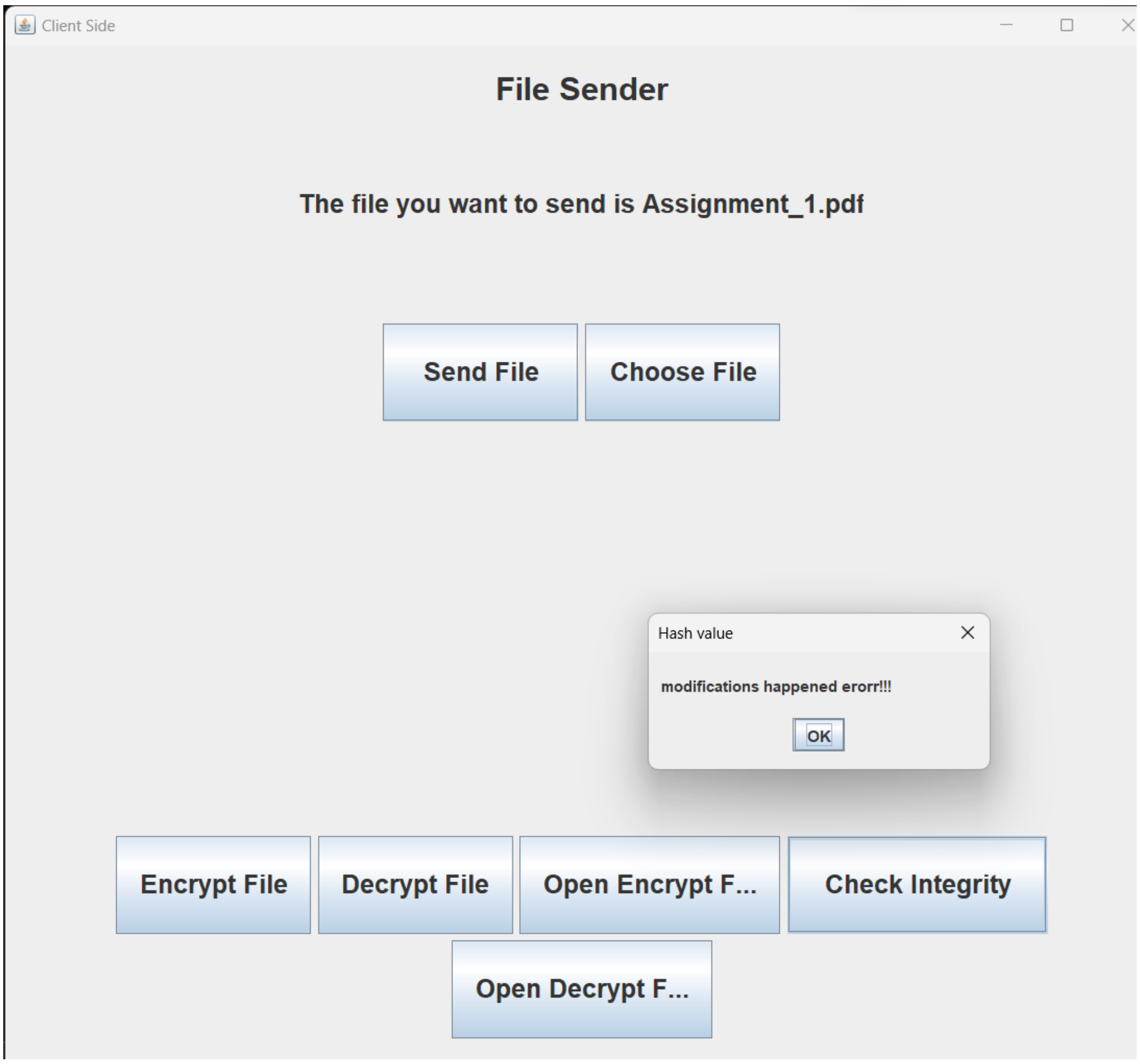
File Sender

The file you want to send is ex1_Ans.txt

Hash value

No modifications

Figure 20: The popup displays no modification for the file ex1_Ans.txt



modifications happened erorr!!!

Figure 21: Here the message "modifications happened error !!!" is displayed for other file Assignment_1.pdf

Because in the code I have specified to check the hash value with the hash value generated for ex1_Ans.txt

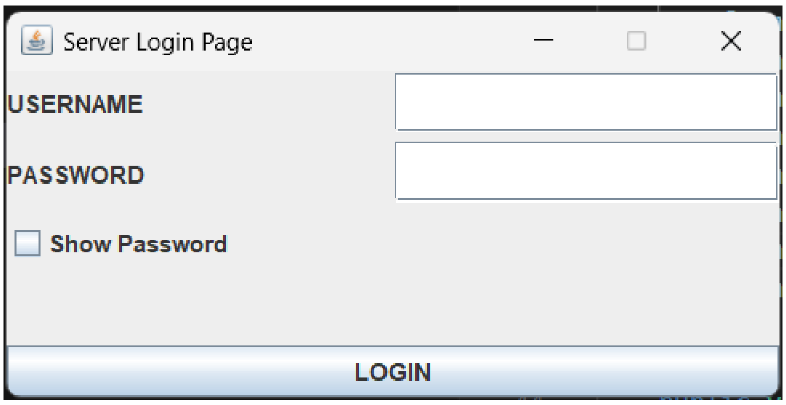
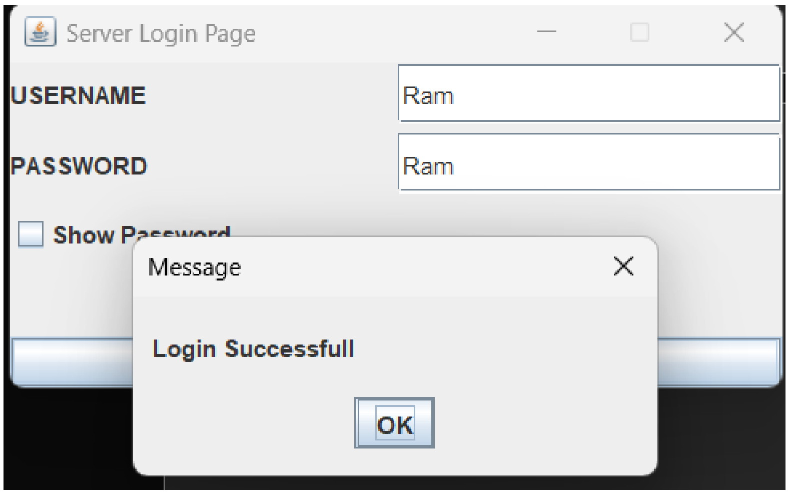


Figure 22: Server Login Page


Server Login Page

USERNAME Ram

PASSWORD Ram

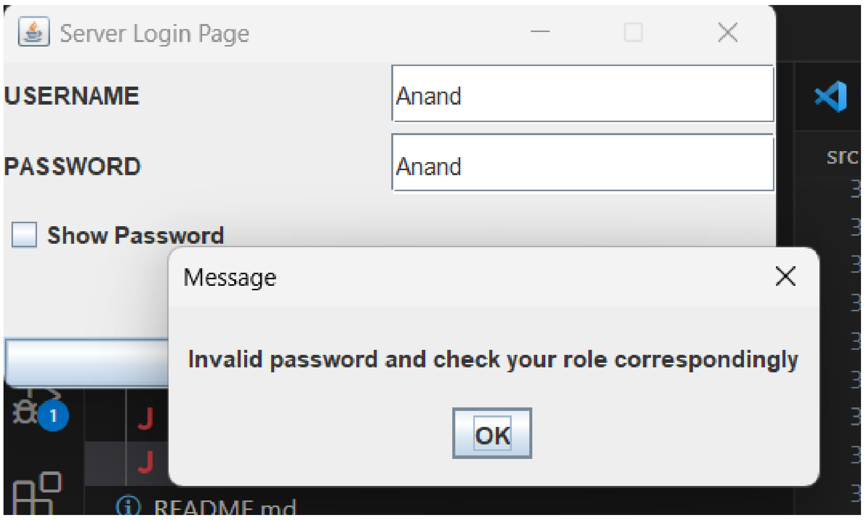
Show Password

Message

Login Successfull

OK

Figure 23: Ram is an authorized Server in database. Hence Login is Successful.


Server Login Page

USERNAME Anand

PASSWORD Anand

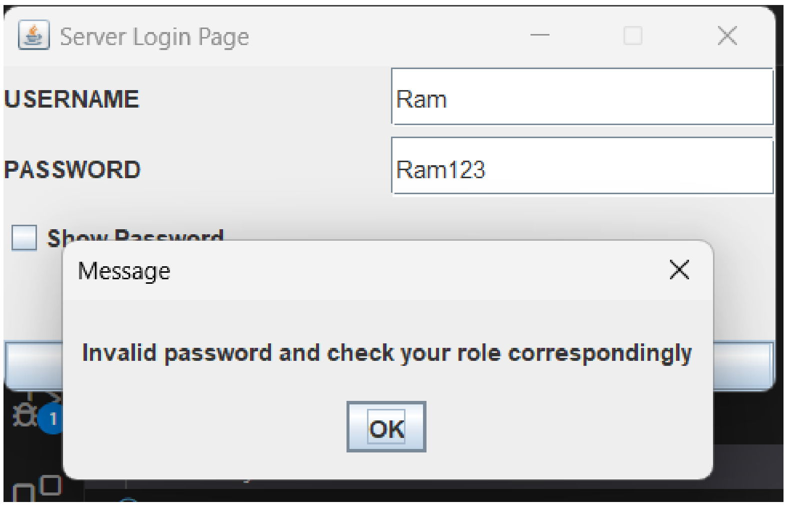
Show Password

Message

Invalid password and check your role correspondingly

OK

Figure 24: Anand is an authorized client in database. Hence Server Login Failed.


Server Login Page

USERNAME Ram

PASSWORD Ram123

Show Password

Message

Invalid password and check your role correspondingly

OK

Figure 25: Here Login Failed as the entered password is incorrect.

Server

File Receiver

Figure 26 : File Receiver frame when logined successfully