MESSAGE ENCODING DECODING SYSTEM

A MINI-PROJECT REPORT

Submitted by

JOHN PRATHAP SINGH S	**220701112**
KESAVAN M	**220701129**

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

2023-2024

BONAFIDE CERTIFICATE

Certified that this project **“MESSAGE ENCODING DECODING SYSTEM”** is the bonafide work of **“JOHN PRATHAP SINGH S (220701112)” & KESAVAN M (220701129)”** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.R.SABITHA	**Ms.V.JANANEE**
Professor and Academic Head	**Assistant Professor (SG)**
Computer Science and Engineering,	Computer Science and Engineering,
Rajalakshmi Engineering College (Autonomous)	Rajalakshmi Engineering College (Autonomous)
Thandalam, Chennai – 602 105	Thandalam, Chennai – 602 105

Submitted for the Practical Examination held on ____________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The "MessageEncodingDecoding" Java program exemplifies secure data encryption and storage using the Advanced Encryption Standard (AES) algorithm, while emphasizing robust key management practices. The program allows users to input a message, encrypt it, and then decrypt it, demonstrating data protection with encryption keys stored in a KeyStore.

Key Features:

1) User Interaction: The program interacts with the user, allowing them to input a message to be encrypted and later decrypted.

2) AES Encryption: It employs the AES encryption algorithm to secure the user's data. AES is a widely accepted encryption algorithm that provides strong data protection.

3) Key Management: The program utilizes a KeyStore to securely manage the encryption keys. It generates a random AES key, stores it in the KeyStore, and subsequently retrieves the key for data encryption and decryption.

4) Dynamic Key Generation: If the KeyStore does not exist, the program creates one and generates a new AES encryption key. It ensures that a valid key is always available for secure operations.

5) Data Integrity: The program maintains data integrity by verifying that the stored key is of the correct type (SecretKey) before use.

6) File Persistence: The generated KeyStore and keys are persisted in a file ("keystore.jceks") to ensure data and key retention across program runs.

7) Base64 Encoding: Before display, the encrypted data is converted to Base64 format, enabling it to be safely printed and decoded.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S.MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. P.Kumar M.E Ph.D.,** for being ever supporting force during our project work.

We also extend our sincere and hearty thanks to our internal guide **Mrs.V.Jananee** , for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. **JOHN PRATHAP SINGH S**
2. **KESAVAN M**

TABLE OF CONTENTS

CHAPTER - 1 : INTRODUCTION

1.1 INTRODUCTION :

This project improves the secrecy of information which is communication between two or more person. There are many data encryption and decryption algorithm this project's algorithm is derived from one of them.

1.2 MESSAGE ENCODING DECODING SYSTEM :

This Message Encoding Decoding algorithm is developed using Advanced Encryption Standard(AES) with key storage which stores key from every data transmitted which is used for encrypting data and send the data to the receiver end and use that same key for decrypting the encrypted data.

1.3 IMPLEMENTATION OF THE PROJECT :

The implementation of this project comprises of two distinct version:

i) A Console Based Application written in Java Language.

ii) A Graphical User Interface (GUI) which uses Java Swing.

CONSOLE BASED USING JAVA

This Java program performs the following operations:

1. User Input: The program prompts the user to enter a message through the console.
2. AES Encryption and Decryption: It generates or retrieves an AES secret key securely using a keystore. It then encrypts the user entered message using the AES key and subsequently decrypts the encrypted data.
3. Base64 Conversion: The encrypted data is converted to a Base64-encoded string for easy storage and transmission. The decrypted data is converted back to a regular string.

4. Database Interaction:The program connects to a MySQL database and inserts the encrypted and decrypted data into a table named 'message.'
5. Key Management:The AES key is stored securely in a keystore file named 'keystore.jceks.' If the keystore file doesn't exist, it is created. The key is either retrieved from the keystore or generated if it doesn't exist.
6. JDBC Connection: The program uses JDBC (Java Database Connectivity) to connect to the MySQL database. It includes the necessary driver for MySQL and specifies the connection details such as the URL, username, and password.

JAVA SWING

This Java program using Swing performs the following operations:

1. GUI Setup: The program creates a Swing-based graphical user interface (GUI) with a JFrame, JLabels, a JTextField and JButtons for encryption, decryption, and inserting data into a database.

2. Key Generation: It provides a method `getKey` that generates or retrieves an AES key using the `generateOrRetrieveAESKey` method from the `MessageEncodingDecodingUsingSwing` class.

3. Database Connection: It establishes a connection to a MySQL database named 'sakila' on localhost. It also creates a table named 'message' if it does not already exist.

4. Encryption: When the user clicks the "Encrypted Data" button, the program reads the user input, gets an AES key,encrypts the data, and converts the encrypted data to a Base64-encoded string.

5. Decryption: When the user clicks the "Decrypted Data" button, the program decrypts the previously encrypted data using the same AES key.

6. Database Insertion: When the user clicks the "Insert data into Database" button, the program inserts the encrypted and decrypted data into the 'message' table in the database.

CHAPTER – 2 : SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	JAVA
Back - End	:	MYSQL
Language	:	JAVA , MYSQL

CHAPTER - 3 : CODING

MESSAGE ENCODING DECODING USING JAVA

SOURCE CODE

CONSOLE-BASED :

```
import java.io.*;

import java.sql.*;

import java.security.*;

import java.util.Base64;

import java.util.Scanner;

import javax.crypto.*;

public class MessageEncodingDecoding {

    public static void main(String[] args) throws Exception {

        Scanner s = new Scanner(System.in);

        System.out.println("Enter a message:");

        String originalData = s.nextLine();

        SecretKey key = generateOrRetrieveAESKey(); // Generate or retrieve the
key securely

        byte[] encryptedData = null;
        byte[] decryptedData = null;

        try {

            // Encrypt the data

            Cipher cip = Cipher.getInstance("AES");

            cip.init(Cipher.ENCRYPT_MODE, key);

            encryptedData = cip.doFinal(originalData.getBytes());

            // Decrypt the data

            cip.init(Cipher.DECRYPT_MODE, key);

            decryptedData = cip.doFinal(encryptedData);

        } catch (Exception e) {

            System.out.println("Error in encrypting or decrypting the data: " +
e.getMessage());

        }

        // Convert the encrypted and decrypted data to Base64 for printing

        String encryptedDataStr =
Base64.getEncoder().encodeToString(encryptedData);

        String decryptedDataStr = new String(decryptedData);
        // JDBC Connection Information
        try {
            // Register JDBC driver
            Class.forName("com.mysql.jdbc.Driver");
            // Open a connection
            Connection conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/mini?allowPublicKeyRetrieval=true&characterEn
coding=utf8&useSSL=false&useUnicode=true","root", "StJo2912#_");
            // Insert the encrypted data into the database
            String insertSql = "INSERT INTO message
(encrypted_data,decrypted_data) VALUES (?, ?)";
            try (PreparedStatement preparedStatement =
conn.prepareStatement(insertSql)) {
            preparedStatement.setString(1, encryptedDataStr);
            preparedStatement.setString(2, decryptedDataStr);
            try {
                preparedStatement.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace(); // or use a logging framework to log the
exception

                System.out.println("Error executing SQL query: " +
e.getMessage());

            }

            conn.close();

        }

    } catch (ClassNotFoundException | SQLException e) {

        System.out.println("Error in database operations: " + e.getMessage());

    }

}

private static SecretKey generateOrRetrieveAESKey() throws Exception {

    KeyStore keyStore = KeyStore.getInstance("JCEKS");

    char[] keystorePassword = "keystore_password".toCharArray();

    FileInputStream fis = null;

    try {

        File keystoreFile = new File("keystore.jceks"); // Create a File object

        if (keystoreFile.exists()) {
```

```
        fis = new FileInputStream(keystoreFile);

        keyStore.load(fis, keystorePassword);

    } else {

        keyStore.load(null, keystorePassword);

        try (FileOutputStream fos = new FileOutputStream(keystoreFile)) {

            keyStore.store(fos, keystorePassword);

        }

    }

    String keyAlias = "aes_key";

    SecretKey key;

    if (keyStore.containsAlias(keyAlias)) {

        Key keyFromKeystore = keyStore.getKey(keyAlias,
keystorePassword);

        if (keyFromKeystore instanceof SecretKey)

            key = (SecretKey) keyFromKeystore;

        else

            throw new RuntimeException("Key in the keystore is not a
SecretKey");
```

```
    }
```

```
            } else {
                KeyGenerator kg = KeyGenerator.getInstance("AES");
                kg.init(256, new SecureRandom());
                key = kg.generateKey();
                keyStore.setKeyEntry(keyAlias, key, keystorePassword, null);
                try (FileOutputStream fos = new FileOutputStream(keystoreFile)) {
                    keyStore.store(fos, keystorePassword);
                }
            }
            return key;
        } finally {
            if (fis != null) {
                fis.close();
            }
        }
    }
}
```

JAVA SWING :

```
import java.io.*;

import java.sql.*;

import java.security.*;

import java.util.Base64;

import javax.crypto.*;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;



public class MessageEncodingDecodingUsingSwing {

    public static void main(String[] args) throws Exception, SQLException {

        SwingUtilities.invokeLater(() -> {

            InputOutput obj = new InputOutput();

            obj.setVisible(true);

        });

    }
```

```
public static SecretKey getKey() {
    try {
        return generateOrRetrieveAESKey();
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}


private static SecretKey generateOrRetrieveAESKey() throws Exception {
    KeyStore keyStore = KeyStore.getInstance("JCEKS");
    char[] keystorePassword = "keystore_password".toCharArray();
    FileInputStream fis = null;
    try {
        File keystoreFile = new File("keystore.jceks");
        if (keystoreFile.exists()) {
            fis = new FileInputStream(keystoreFile);
            keyStore.load(fis, keystorePassword);

        } else {

            keyStore.load(null, keystorePassword);

            try (FileOutputStream fos = new FileOutputStream(keystoreFile)) {

                keyStore.store(fos, keystorePassword);

            }

        }

        String keyAlias = "aes_key";

        SecretKey key;

        if (keyStore.containsAlias(keyAlias)) {

            Key keyFromKeystore = keyStore.getKey(keyAlias,
keystorePassword);

            if (keyFromKeystore instanceof SecretKey)

                key = (SecretKey) keyFromKeystore;

            else

                throw new RuntimeException("Key in the keystore is not a
SecretKey");

        } else {
                KeyGenerator kg = KeyGenerator.getInstance("AES");
                kg.init(256, new SecureRandom());
                key = kg.generateKey();
                keyStore.setKeyEntry(keyAlias, key, keystorePassword, null);
                try (FileOutputStream fos = new FileOutputStream(keystoreFile)) {
                    keyStore.store(fos, keystorePassword);
                }
            }
            return key;
        } finally {
            if (fis != null) {
                fis.close();
            }
        }
    }
}
```

```
class InputOutput extends JFrame {

    JLabel l1, l2;

    JTextField t1;

    JButton b1, b2, b3;

    Cipher cip;

    SecretKey key;

    Connection conn;

    byte[] encryptedData;

    String encryptedDataStr;

    String decryptedDataStr;

    public InputOutput() {

        setDefaultCloseOperation(EXIT_ON_CLOSE);

        setLayout(null);

        l1 = new JLabel("Enter your message");

        l1.setBounds(40, 10, 300, 30);

        add(l1);

        l2 = new JLabel("");
```

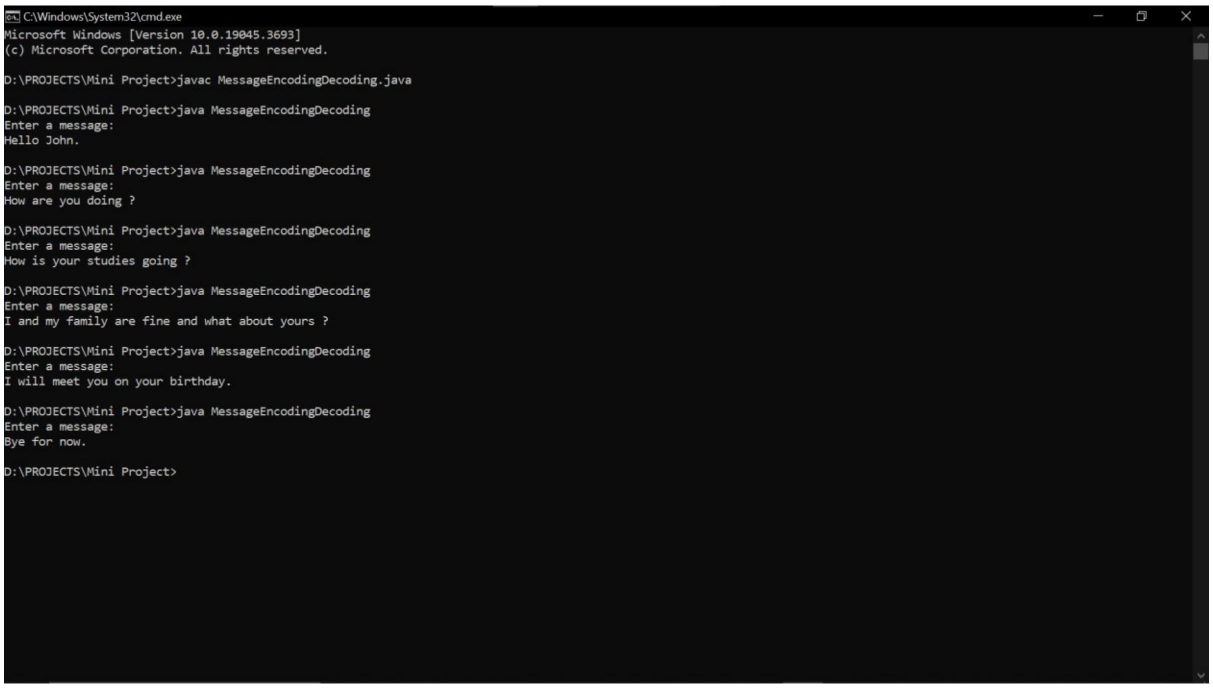
```
l2.setBounds(60, 260, 300, 40);

add(l2);

t1 = new JTextField(200);

b1 = new JButton("Encrypted Data");

b2 = new JButton("Decrypted Data");

b3 = new JButton("Insert data into Database");

t1.setBounds(40, 60, 500, 40);

b1.setBounds(60, 140, 150, 30);

b2.setBounds(60, 180, 150, 30);

b3.setBounds(60, 220, 200, 30);

add(t1);

add(b1);

add(b2);

add(b3);



try {

    Class.forName("com.mysql.jdbc.Driver");
```

```
            try {

                conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/sakila?allowPublicKeyRetrieval=true&characterEn
coding=utf8&useSSL=false&useUnicode=true",

                        "root", "StJo2912#_");

                Statement stat = conn.createStatement();

                String createTable = "CREATE TABLE IF NOT EXISTS message(No
INT AUTO_INCREMENT PRIMARY KEY,encrypted_data
TEXT,decrypted_data TEXT)";

                stat.executeUpdate(createTable);

                stat.close();

            } catch (SQLException ae) {

                l2.setText("Error in database operations: " + ae.getMessage());

            }

        } catch (ClassNotFoundException ae) {

            l2.setText("Error in database operations: " + ae.getMessage());

        }
        b1.addActionListener(e -> {

            try {

                String originalData = t1.getText();

                key = MessageEncodingDecodingUsingSwing.getKey();

                cip = Cipher.getInstance("AES");

                cip.init(Cipher.ENCRYPT_MODE, key);

                encryptedData = cip.doFinal(originalData.getBytes());

                encryptedDataStr =
Base64.getEncoder().encodeToString(encryptedData);

            } catch (Exception ae) {

                l2.setText("Error in encrypting data");

            }

        });



        b2.addActionListener(e -> {

            try {

                cip.init(Cipher.DECRYPT_MODE, key);

                byte[] decryptedData = cip.doFinal(encryptedData);
            decryptedDataStr = new String(decryptedData);

         } catch (Exception ae) {

            l2.setText("Error in decrypting data");

         }

      });



      b3.addActionListener(e -> {

         String insertSql = "INSERT INTO message
(encrypted_data,decrypted_data) VALUES (?, ?)";

         try (PreparedStatement preparedStatement =
conn.prepareStatement(insertSql)) {

            preparedStatement.setString(1, encryptedDataStr);

            preparedStatement.setString(2, decryptedDataStr);

            try {

               preparedStatement.executeUpdate();

            } catch (SQLException ae) {

               ae.printStackTrace();

               l2.setText("Error executing SQL query: " + ae.getMessage());
```

```
                }
            } catch (SQLException ae) {
                l2.setText("Error in database operations: " + ae.getMessage());
            }
        });
    }
}
```

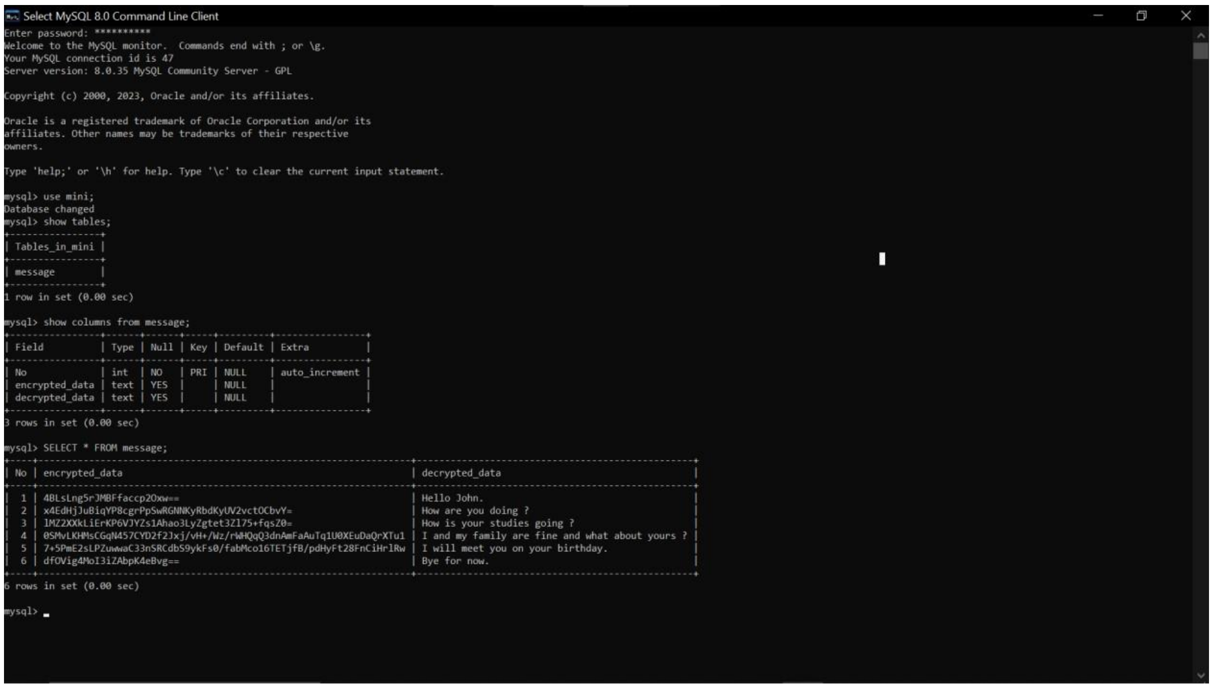
CHAPTER – 4 : SCREEN SHOTS

OUTPUT FOR CONSOLE –BASED



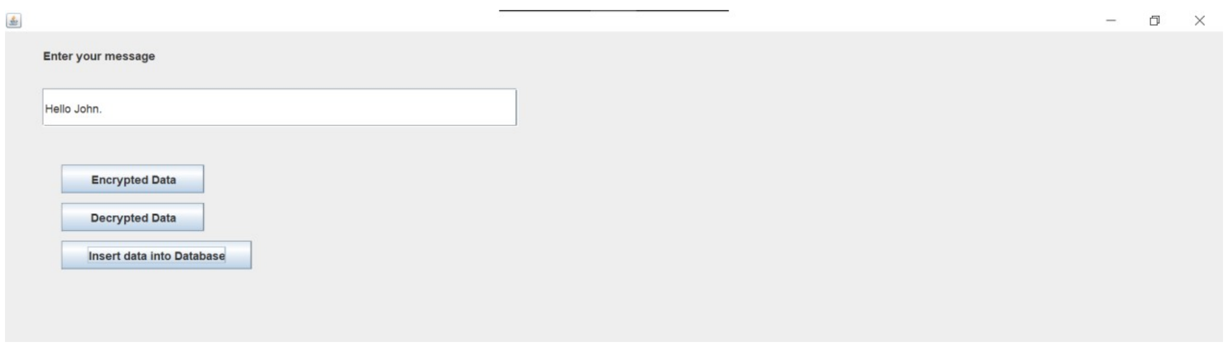
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3693]
(c) Microsoft Corporation. All rights reserved.

D:\PROJECTS\Mini Project>javac MessageEncodingDecoding.java

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
Hello John.

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
How are you doing ?

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
How is your studies going ?

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
I and my family are fine and what about yours ?

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
I will meet you on your birthday.

D:\PROJECTS\Mini Project>java MessageEncodingDecoding
Enter a message:
Bye for now.

D:\PROJECTS\Mini Project>
```

DATABASE CONNNECTION :

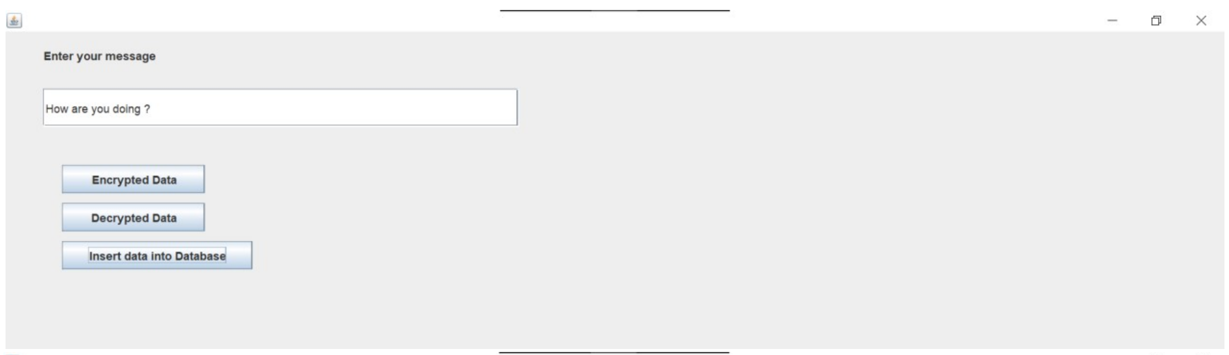


```
Select MySQL 8.0 Command Line Client
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 47
Server version: 8.0.35 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use mini;
Database changed
mysql> show tables;
+----------------+
| Tables_in_mini |
+----------------+
| message        |
+----------------+
1 row in set (0.00 sec)

mysql> show columns from message;
+----------------+------+------+-----+---------+----------------+
| Field          | Type | Null | Key | Default | Extra          |
+----------------+------+------+-----+---------+----------------+
| No             | int  | NO   | PRI | NULL    | auto_increment |
| encrypted_data | text | YES  |     | NULL    |                |
| decrypted_data | text | YES  |     | NULL    |                |
+----------------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM message;
+----+------------------------------------------------------------------+-------------------------------------------------+
| No | encrypted_data                                                   | decrypted_data                                  |
+----+------------------------------------------------------------------+-------------------------------------------------+
|  1 | 4BLsLng5rJMBFfaccp2Oxw==                                         | Hello John.                                     |
|  2 | x4EdHjJuBiqYP8cgrPpSwRGNNKyRbdKyUV2vctOCbvY=                     | How are you doing ?                             |
|  3 | 1MZ2XXkLiErKP6VJYZs1Ahao3LyZgtet3Z175+fqsZ0=                     | How is your studies going ?                     |
|  4 | 0SMvLKHMsCGqN457CYD2f2Jxj/vH+/Wz/rWHQqQ3dnAmFaAuTq1U0XEuDaQrXTu1 | I and my family are fine and what about yours ? |
|  5 | 7+5PmE2sLPZuwwaC33nSRCdbS9ykFs0/fabMco16TETjfB/pdHyFt28FnCiHrlRw | I will meet you on your birthday.               |
|  6 | dfOVig4MoI3iZAbpK4eBvg==                                         | Bye for now.                                    |
+----+------------------------------------------------------------------+-------------------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

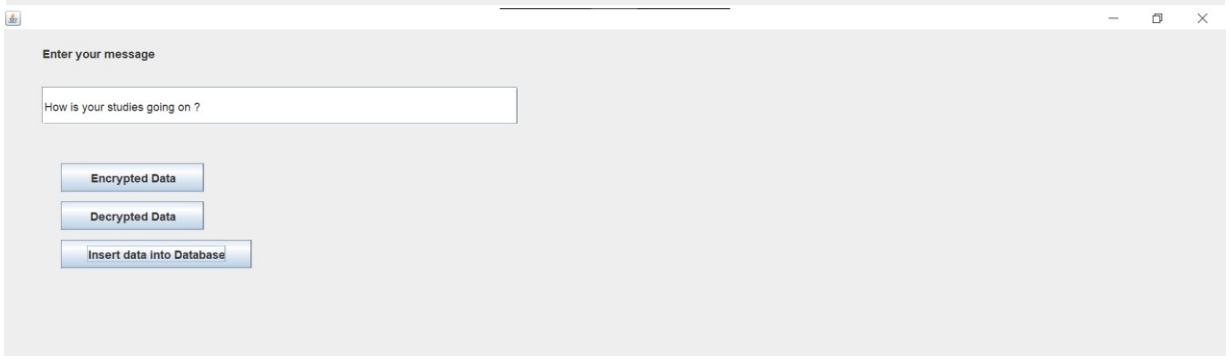
OUTPUT FOR JAVA SWING


Enter your message

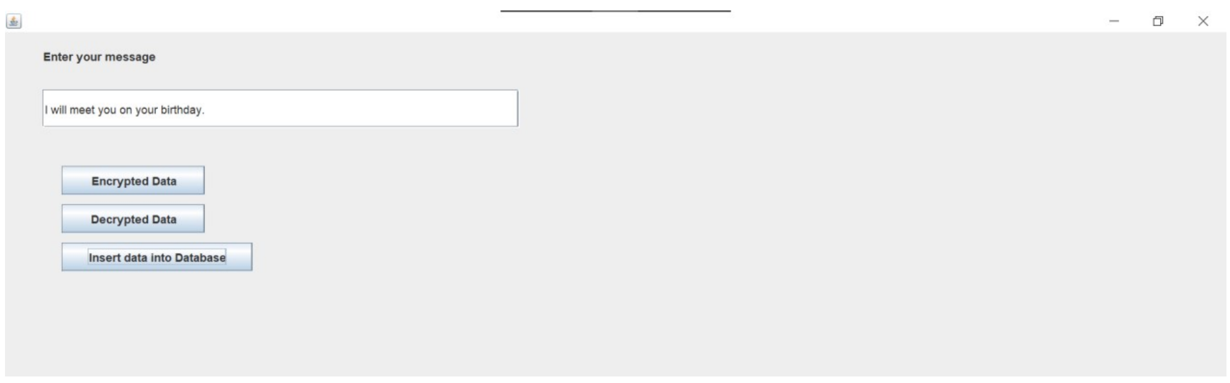
Hello John.


Enter your message

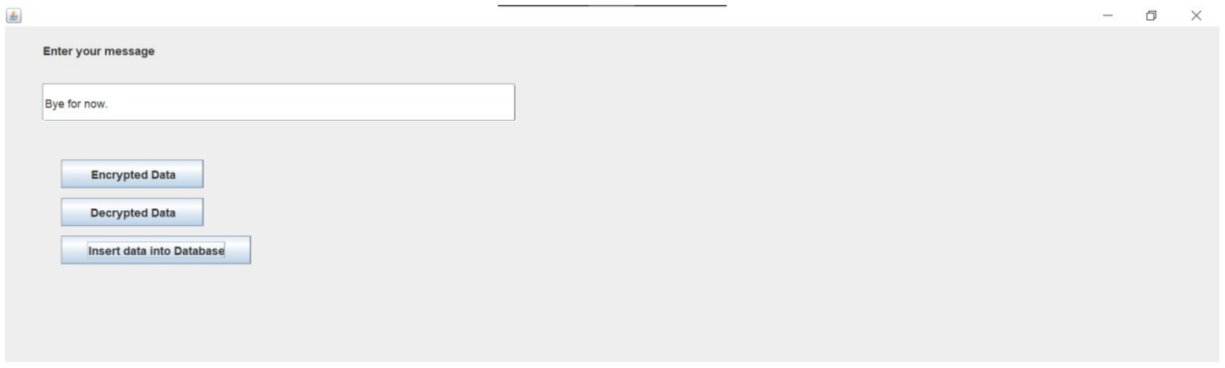
How are you doing ?


Enter your message

How is your studies going on ?


Enter your message

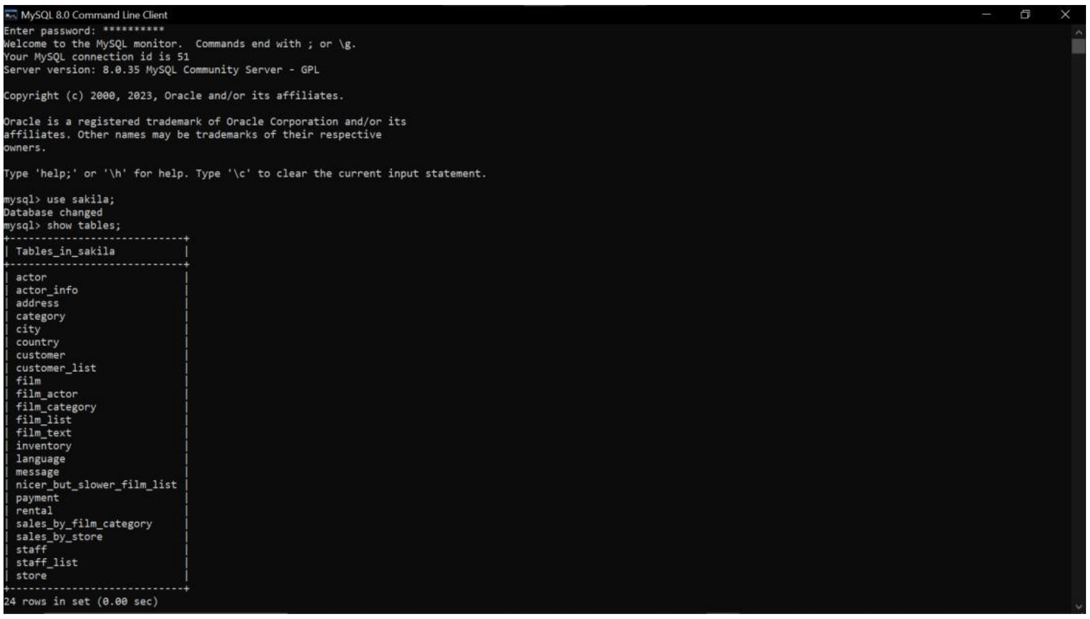
I will meet you on your birthday.



Enter your message

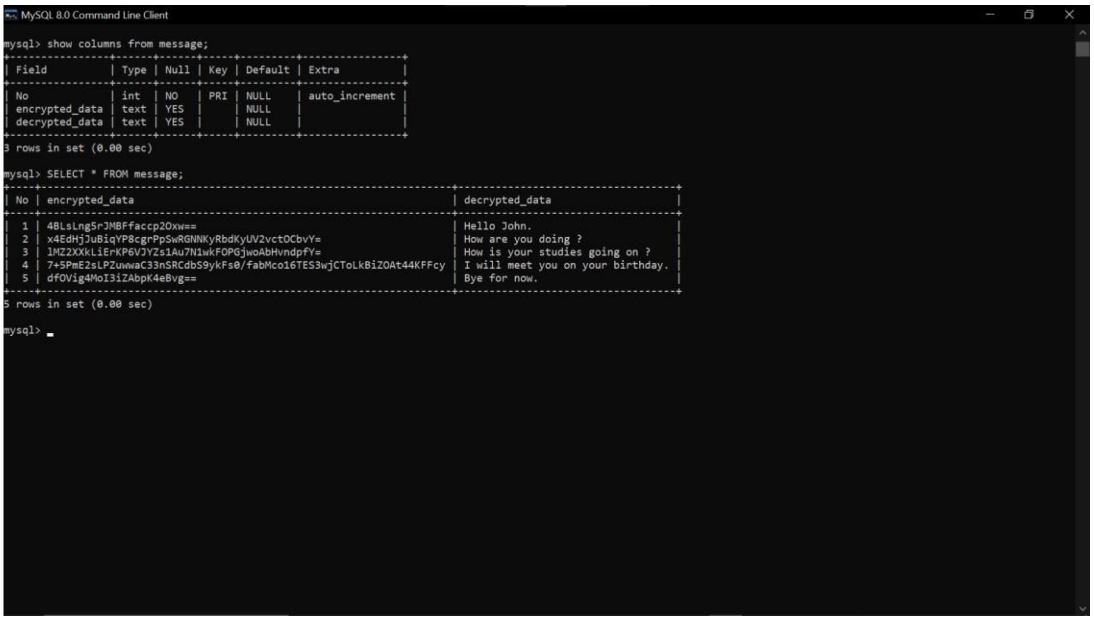
Bye for now.

DATABASE CONNNECTION :



MySQL 8.0 Command Line Client

```
Enter password: **********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 51
Server version: 8.0.35 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sakila;
Database changed
mysql> show tables;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
| city                       |
| country                    |
| customer                   |
| customer_list              |
| film                       |
| film_actor                 |
| film_category              |
| film_list                  |
| film_text                  |
| inventory                  |
| language                   |
| message                    |
| nicer_but_slower_film_list |
| payment                    |
| rental                     |
| sales_by_film_category     |
| sales_by_store             |
| staff                      |
| staff_list                 |
| store                      |
+----------------------------+
24 rows in set (0.00 sec)
```



MySQL 8.0 Command Line Client

```
mysql> show columns from message;
+----------------+------+------+-----+---------+----------------+
| Field          | Type | Null | Key | Default | Extra          |
+----------------+------+------+-----+---------+----------------+
| No             | int  | NO   | PRI | NULL    | auto_increment |
| encrypted_data | text | YES  |     | NULL    |                |
| decrypted_data | text | YES  |     | NULL    |                |
+----------------+------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM message;
+----+--------------------------------------------------------------+------------------------------------+
| No | encrypted_data                                               | decrypted_data                     |
+----+--------------------------------------------------------------+------------------------------------+
|  1 | 4BLsLng5rJMBFfaccp2Oxw==                                     | Hello John.                        |
|  2 | x4EdHjJuBiqYP8cgrPpSwRGNNKyRbdKyUV2vctOCbvY=                 | How are you doing ?                |
|  3 | lMZ2XXkLiErKP6VJYZs1Au7N1wkFOPGjwoAbHvndpfY=                 | How is your studies going on ?     |
|  4 | 7+5PmE2sLPZuwwaC33nSRCdbS9ykFs0/fabMco16TES3wjCToLkBiZOAt44KFFcy | I will meet you on your birthday. |
|  5 | dfOVig4MoI3iZAbpK4eBvg==                                     | Bye for now.                       |
+----+--------------------------------------------------------------+------------------------------------+
5 rows in set (0.00 sec)

mysql> _
```

CHAPTER 5

CONCLUSION

Thus, the program has been created successfully to MESSAGE ENCODING DECODING SYSTEM . The console-based version id designed for users comfortable with command-lines interfaces. On the other hand, the GUI version provides user-friendly and interactive experience for MESSAGE ENCODING DECODING SYSTEM.

This code can serve as a foundation for secure data storage and retrieval in applications that require data confidentiality. It showcases essential practices in key management and data encryption, enabling users to apply these techniques to enhance the security of their applications.

CHAPTER 6

REFERENCES

The below websites and books are useful in gaining knowledge and for creating this project :

WEBSITES :

1. https://www.javatpoint.com/aes-256-encryption-in-java
2. https://www.javatpoint.com/example-to-connect-to-the-mysql-database
3. https://www.javatpoint.com/java-swing

BOOKS :

1. JAVA - THE COMPLETE REFERENCE
2. RSA : Data Encryption and Data Decryption.