JAVA AWT BASED -ONLINE STREAMING OF VIDEO DATABASE SYSTEM -SQL CONNECTIVITY USING JDBC

A report submitted in partial fulfillment of the Requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By- M.Sannihitha (1602-18-737-104)

Under the guidance

Of

B.Leelavathy

Department of Information Technology

VASAVI COLLEGE OF ENGINEERING

AUTONOMOUS (AFFILIATED TO O.U)

IBRAHIMBAGH, HYDERABAD-500031

2019-20

BONAFIDE CERTIFICATE

This is to certify that this project report titled

"ONLINE STREAMING OF VIDEO DATABASE SYSTEM" is the bonafide mini project work of **Ms. Mula Sannihitha** Bearing hall ticket number **1602-18-737-104** under the guidance of **B. Leelavathy** during 4th semester B.E for the academic year **2019-2020.**

External Examiner

Internal Examiner
B.LEELAVATHY
Assistant professor
Department of Information Technology

AIM AND PRIORITY OF THE PROJECT:

To create a GUI based form for the project of **ONLINE STREAMING OF VIDEO DATABASE SYSTEM** where in a user watches a lot of vehicles and the genuineness of the view and keeps the count of views of all the videos.

The values entered (insertion, updating and deletion) by the user for Respective table in **GUI** should be updated in the database using **JDBC.**

ABSTRACT:

This project is application software developed for monitoring the online video streaming which mainly focuses on basic operations like uploading a video, updating new information, searching videos and identifying the members who are genuinely watching the video. It is implemented with the help of 5 tables namely USERS, VIEWS, VIDEOS, UPLOADS, ADMIN. The USERS and ADMIN tables consist of information regarding members and the admin respectively. Admin uploads a video which is described in another table namely UPLOADS while the relationship between USERS and VIDEOS tables is established by VIEWS. Uid, Vid, Aid are the primary keys of the tables USERS, VIDEOS, ADMIN respectively and are foreign keys in the tables VIEW and UPLOADS. A one to one mapping for the ADMIN table and a total participation of USERS tables is established.

This article aims to provide a structured approach that admin can use to identify the members who watch the video genuinely and generate reports accordingly.

A. REQUIREMENTS:

Tables Required: (5) Users, Views, Videos, Uploads, Admin.

TABLE	DSECRPTION	ATTRIBUTE	DATATYPE
USERS	User ID	usid	NUMBER(20)
	User Name	name	VARCHAR2(20)
	User Password	password	VARCHAR2(20)
	User Age	age	NUMBER(3)
VIDEOS	Video ID	vid	NUMBER(20)
	Video Duration	duration	NUMBER(20)
	Video Topic	topic	VARCHAR2(20)
	Number of views of Video	numviews	NUMBER(20)
VIEWS	User ID	usid	NUMBER(20)
	Video ID	vid	NUMBER(20)
	Genuineness of the View	genuineness	VARCHAR2(20)
ADMIN	Admin ID	aid	NUMBER(20)
	Admin Name	name	VARCHAR2(20)

	Admin Age	age	NUMBER(3)
UPLOADS	Video ID	vid	NUMBER(20)
	Admin ID	aid	NUMBER(20)
	Time of Upload	since	VARCHAR2(10)

C.ARCHITECTURE AND TECHNOLOGY USED:

Java Eclipse, Oracle 11g Database, java SE version 8, SQL *plus, java AWT

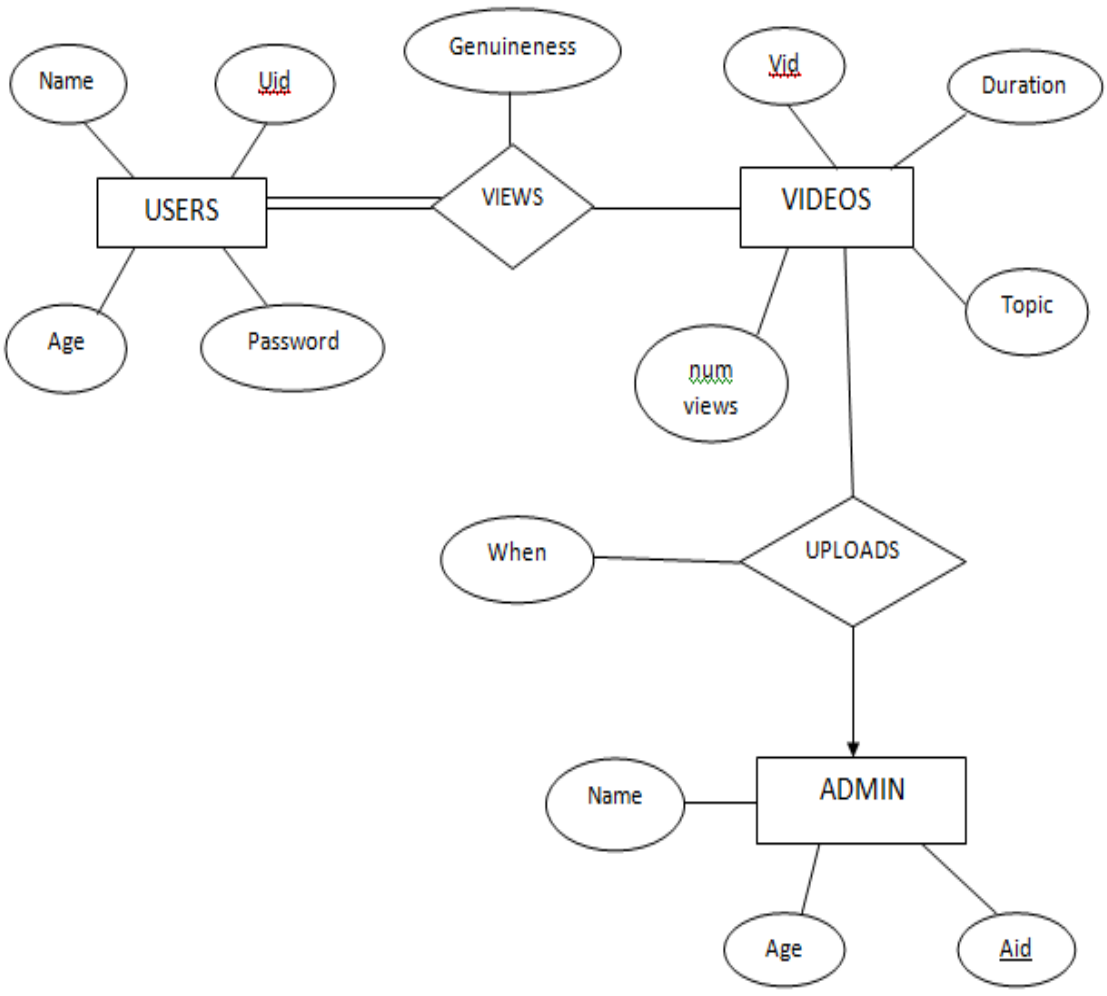
Eclipse: It is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug in system for customizing the environment. The Eclipse software development kit (SDK), which includes java development tools is meant for java developers.

SQL *plus: SQL *plus is a command line tool proprietary to oracle. You can send SQL Queries to the server using the tool. It can also help you format the result of a query. SQL is the query language that is used to communicate with the oracle server to access and modify data.

JAVA AWT: Abstract window toolkit is an API to develop GUI or Window based applications in java. Java AWT components are platform dependent i.e., components are displayed according to the view of the operating system. AWT is a heavy weight that is components are using the resources of O.S.

JDBC: Java Database Connectivity is an application programming interface (API) for the programming language java**,** which defines how a client may access a database. It is a java based data access technology used for java database connectivity. It is part of the java Standard Edition Platform, from Oracle Corporation.

D.ER DIAGRAM OF THIS PROJECT:



OUTPUTS

DDL COMMANDS

1. **Users**
2. **Videos**
3. **Views**
4. **Admin**
5. **Uploads**

```
SQL> create table videos(
  2  vid number(20) primary key,
  3  duration number(20),
  4  topic varchar2(20),
  5  numviews number(20));

Table created.

SQL> create table users(
  2  usid number(20) primary key,
  3  name varchar2(20),
  4  password varchar2(20),
  5  age number(3));

Table created.

SQL> create table views(
  2  usid number(20),
  3  views number(20),
  4  genuineness varchar2(20),
  5  foreign key(usid) references users on delete cascade);

Table created.

SQL> create table admin(
  2  aid number(20) primary key,
  3  name varchar2(20),
  4  age number(3));

Table created.

SQL> create table uploads(
  2  vid number(20),
  3  aid number(20),
  4  since varchar2(10),
  5  foreign key(vid) references videos,
  6  foreign key(aid) references admin on delete cascade);

Table created.
```

DESCRIPTION OF TABLES

```
SQL> desc videos
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VID                                       NOT NULL NUMBER(20)
 DURATION                                           NUMBER(20)
 TOPIC                                              VARCHAR2(20)
 NUMVIEWS                                           NUMBER(20)

SQL> desc users
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USID                                      NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(20)
 PASSWORD                                           VARCHAR2(20)
 AGE                                                NUMBER(3)

SQL> desc views
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USID                                               NUMBER(20)
 VIEWS                                              NUMBER(20)
 GENUINENESS                                        VARCHAR2(20)

SQL> desc uploads
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VID                                                NUMBER(20)
 AID                                                NUMBER(20)
 SINCE                                              VARCHAR2(10)

SQL> desc admin
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AID                                       NOT NULL NUMBER(20)
 NAME                                               VARCHAR2(20)
 AGE                                                NUMBER(3)
```

E. JAVA-SQL CONNECTIVITY USING JDBC:

I) FRONT END PROGRAMS AND CONNECTIVITY

The connection to the database can be performed using java programming **(JDBC API)** as**:**

```
public void connectToDB() {
try {
        connection=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:ORCL","msr","vasavi");
                statement = connection.createStatement();
}
catch (SQLException connectException) {
    System.out.println(connectException.getMessage());
    System.out.println(connectException.getSQLState());
    System.out.println(connectException.getErrorCode());
    System.exit(1);
}
catch(Exception e){
   System.out.println("Unable to find and load driver");
   System.exit(1);} }
```

AS THIS PROJECT CONTAINS 5 TABLES

i.**e. USERS,VIDEOS,VIEWS,ADMIN & UPLOADS.**

BELOW IS THE CODE FOR ALL **DML OPERATIONS** ON THE TABLE **USER**

INSERT USER:

```
import java.awt.*;
import java.awt.GridLayout;
import java.awt.event.*;
import java.sql.*;

public class InsertUser extends Frame
{
Button insuserbtn;
TextField usidtxt, nametxt, pwdtxt, agetxt;
TextArea errtxt;
Connection connection;
Statement statement;
public InsertUser()
{
try
```

```
{
Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (Exception e)
{
System.err.println("Unable to find and load driver");
System.exit(1);
}
connectToDB();
}

public void connectToDB()
    {
try
{
 connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"msr","vasavi");
 statement = connection.createStatement();

}
catch (SQLException connectException)
{
 System.out.println(connectException.getMessage());
 System.out.println(connectException.getSQLState());
```

```
 System.out.println(connectException.getErrorCode());
 System.exit(1);
}
   }

public void buildGUI()
{

insuserbtn = new Button("Enter");
insuserbtn.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{

 String query= "INSERT INTO users VALUES(" + usidtxt.getText()
+ ", " + "'" + nametxt.getText() + "'"+"," +"'"+ pwdtxt.getText() +"'"+
"," + agetxt.getText() + ")";
 int i = statement.executeUpdate(query);
//  System.out.print("jgig\nbj");
 errtxt.append("\nInserted " + i + " rows successfully");
//  System.out.print("jgig\nbj");
}
catch (SQLException insertException)
```

```
{
 displaySQLErrors(insertException);
}
}
});


usidtxt = new TextField(15);
nametxt = new TextField(15);
pwdtxt = new TextField(15);
agetxt = new TextField(15);


errtxt = new TextArea(10, 40);
errtxt.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("User ID:"));
first.add(usidtxt);
first.add(new Label("Name:"));
first.add(nametxt);
first.add(new Label("Password"));
first.add(pwdtxt);
first.add(new Label("Age:"));
```

```
first.add(agetxt);
first.setBounds(125,90,200,100);

Panel second = new Panel(new GridLayout(4, 1));
second.add(insuserbtn);
        second.setBounds(125,220,150,100);

Panel third = new Panel();
third.add(errtxt);
third.setBounds(125,320,300,200);

setLayout(null);

add(first);
add(second);
add(third);

setTitle("INSERT USER");
setSize(500, 600);
setVisible(true);

}

private void displaySQLErrors(SQLException e)
{
```

```
errtxt.append("\nSQLException: " + e.getMessage() + "\n");
errtxt.append("SQLState:     " + e.getSQLState() + "\n");
errtxt.append("VendorError:  " + e.getErrorCode() + "\n");
}

public static void main(String[] args)
{
InsertUser user = new InsertUser();

user.addWindowListener(new WindowAdapter(){
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});
user.buildGUI();

}
```

UPDATE USER:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateUser extends Frame
{
```

```
Button upduserbtn;
List USIDList;
TextField usidtxt, nametxt, pwdtxt, agetxt;
TextArea errtxt;
Connection connection;
Statement statement;
ResultSet rs;

public UpdateUser()
{
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (Exception e)
{
System.err.println("Unable to find and load driver");
System.exit(1);
}
connectToDB();
}

public void connectToDB()
   {
try
```

```
{
 connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"msr","vasavi");
 statement = connection.createStatement();


}
catch (SQLException connectException)
{
 System.out.println(connectException.getMessage());
 System.out.println(connectException.getSQLState());
 System.out.println(connectException.getErrorCode());
 System.exit(1);
}
    }
private void loadUsers()
{
try
{
 rs = statement.executeQuery("SELECT USID FROM users");
 while (rs.next())
 {
USIDList.add(rs.getString("USID"));
 }
}
```

```
catch (SQLException e)
{
 displaySQLErrors(e);
}
}
public void buildGUI()
{
   USIDList = new List(10);
loadUsers();
add(USIDList);

//When a list item is selected populate the text fields
USIDList.addItemListener(new ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
try
{
rs = statement.executeQuery("SELECT * FROM users where USID
="+USIDList.getSelectedItem());
rs.next();
usidtxt.setText(rs.getString("USID"));
nametxt.setText(rs.getString("NAME"));
pwdtxt.setText(rs.getString("PASSWORD"));
agetxt.setText(rs.getString("AGE"));
```

```
}
catch (SQLException selectException)
{
displaySQLErrors(selectException);
}
}
});


//Handle Update User Button
upduserbtn = new Button("Update");
upduserbtn.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{
Statement statement = connection.createStatement();
int i = statement.executeUpdate("UPDATE users "
+ "SET name='" + nametxt.getText() + "', "
+ "password='" + pwdtxt.getText() + "' , "
+ "age ="+ agetxt.getText() + " WHERE usid = "
+ USIDList.getSelectedItem());
errtxt.append("\nUpdated " + i + " rows successfully");
```

```
USIDList.removeAll();
loadUsers();
}
catch (SQLException insertException)
{
displaySQLErrors(insertException);
}
}
});

usidtxt = new TextField(15);
usidtxt.setEditable(false);
nametxt = new TextField(15);
pwdtxt = new TextField(15);
agetxt = new TextField(15);
errtxt = new TextArea(10, 40);
errtxt.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));
first.add(new Label("Sailor ID:"));
first.add(usidtxt);
first.add(new Label("Name:"));
first.add(nametxt);
```

```
first.add(new Label("Password"));
first.add(pwdtxt);
first.add(new Label("Age:"));
first.add(agetxt);

Panel second = new Panel(new GridLayout(4, 1));
second.add(upduserbtn);

Panel third = new Panel();
third.add(errtxt);

add(first);
add(second);
add(third);

setTitle(“UPDATE USER”);
setSize(500, 600);
setLayout(new FlowLayout());
setVisible(true);

}



private void displaySQLErrors(SQLException e)
```

```
{
errtxt.append("\nSQLException: " + e.getMessage() + "\n");
errtxt.append("SQLState:      " + e.getSQLState() + "\n");
errtxt.append("VendorError:  " + e.getErrorCode() + "\n");
}

public static void main(String[] args)
{
UpdateUser upu = new UpdateUser();

upu.addWindowListener(new WindowAdapter(){
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});

upu.buildGUI();
}
}
```

DELETE USER:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteUser extends Frame
{
Button dltuserbtn;
List USIDList;
TextField usidtxt, nametxt, pwdtxt, agetxt;
TextArea errtxt;
Connection connection;
Statement statement;
ResultSet rs;

public DeleteUser()
{
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch (Exception e)
{
System.err.println("Unable to find and load driver");
System.exit(1);
}
connectToDB();
```

```
}

public void connectToDB()
    {
try
{
 connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"msr","vasavi");
 statement = connection.createStatement();


}
catch (SQLException connectException)
{
 System.out.println(connectException.getMessage());
 System.out.println(connectException.getSQLState());
 System.out.println(connectException.getErrorCode());
 System.exit(1);
}
    }

private void loadUsers()
{
try
{
```

```
 rs = statement.executeQuery("SELECT * FROM users");
 while (rs.next())
 {
USIDList.add(rs.getString("USID"));
 }
}
catch (SQLException e)
{
 displaySQLErrors(e);
}
}


public void buildGUI()
{
   USIDList = new List(10);
loadUsers();
add(USIDList);

//When a list item is selected populate the text fields
USIDList.addItemListener(new ItemListener()
{
public void itemStateChanged(ItemEvent e)
{
try
```

```
{
rs = statement.executeQuery("SELECT * FROM users");
while (rs.next())
{
if (rs.getString("USID").equals(USIDList.getSelectedItem()))
break;
}
if (!rs.isAfterLast())
{
usidtxt.setText(rs.getString("USID"));
nametxt.setText(rs.getString("NAME"));
pwdtxt.setText(rs.getString("PASSWORD"));
agetxt.setText(rs.getString("AGE"));
}
}
catch (SQLException selectException)
{
displaySQLErrors(selectException);
}
}
});


//Handle Delete User Button
dltuserbtn = new Button("Delete");
```

```
dltuserbtn.addActionListener(new ActionListener()
{
public void actionPerformed(ActionEvent e)
{
try
{
Statement statement = connection.createStatement();
int i = statement.executeUpdate("DELETE FROM users WHERE
USID = "
+ USIDList.getSelectedItem());
errtxt.append("\nDeleted " + i + " rows successfully");
usidtxt.setText(null);
nametxt.setText(null);
pwdtxt.setText(null);
agetxt.setText(null);
USIDList.removeAll();
loadUsers();
}
catch (SQLException insertException)
{
displaySQLErrors(insertException);
}
}
});
```

```
usidtxt = new TextField(15);
nametxt = new TextField(15);
pwdtxt = new TextField(15);
agetxt = new TextField(15);

errtxt = new TextArea(10, 40);
errtxt.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("User ID:"));
first.add(usidtxt);
first.add(new Label("Name:"));
first.add(nametxt);
first.add(new Label("Password:"));
first.add(pwdtxt);
first.add(new Label("Age:"));
first.add(agetxt);

Panel second = new Panel(new GridLayout(4, 1));
second.add(dltuserbtn);

Panel third = new Panel();
third.add(errtxt);
```

```
add(first);
add(second);
add(third);

setTitle("DELETE USER");
setSize(450, 600);
setLayout(new FlowLayout());
setVisible(true);

}

private void displaySQLErrors(SQLException e)
{
errtxt.append("\nSQLException: " + e.getMessage() + "\n");
errtxt.append("SQLState:     " + e.getSQLState() + "\n");
errtxt.append("VendorError:  " + e.getErrorCode() + "\n");
}



public static void main(String[] args)
{
DeleteUser delu = new DeleteUser();

delu.addWindowListener(new WindowAdapter(){
```

```
 public void windowClosing(WindowEvent e)
 {
System.exit(0);
 }
});

delu.buildGUI();
}
}
```

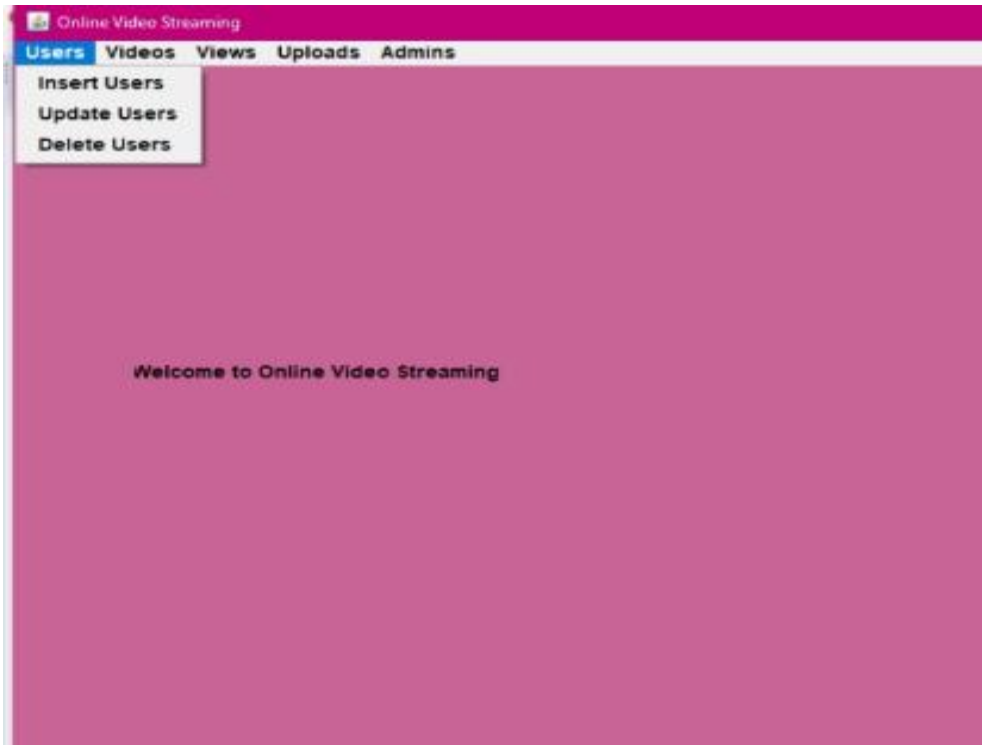
GITHUB LINK AND FOLDER STRUCTURE:

https://github.com/SannihithaReddy/DBMS-Assignment

The zip folder contains all the source codes that are part of this mini project such as other Insert, Update and Delete files of remaining tables.

OUTPUTS

THE FIRST FRAME THAT WILL BE VISIBLE WHEN WE EXECUTE THE PROGRAM WILL BE



OUTPUT OF INSERT USER:


```
SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
        99 sai                  1234                         18

SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
        99 sai                  1234                         18
       104 sannihitha           456                          50

SQL>
```

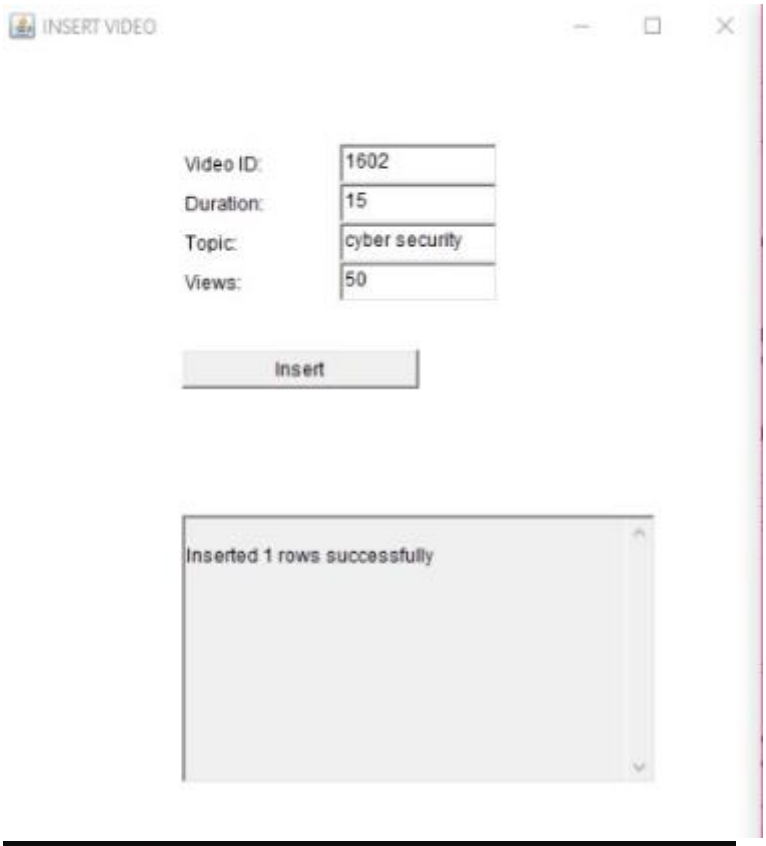
OUTPUT OF UPDATE USER:


```
SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
        99 sai                  1234                         18
       104 sannihitha           456                          50

SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
        99 sai                  1234                         18
       104 sannihitha           456                          47

SQL>
```

OUTPUT OF DELETE USER:

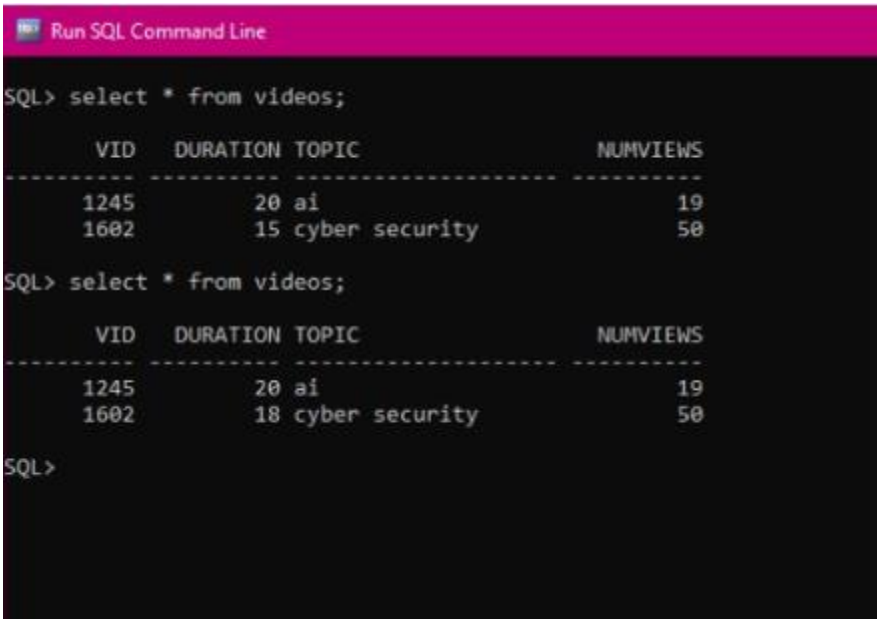

```
SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
        99 sai                  1234                         18
       104 sannihitha           456                          47

SQL> select * from users;

      USID NAME                 PASSWORD                 AGE
---------- -------------------- -------------------- ----------
        97 mani                 9876                         18
       101 andy                 opo                          66
       100 mahesh               pop                          50
       104 sannihitha           456                          47

SQL>
```

OUTPUT OF INSERT VIDEO:




```
SQL> select * from videos;

       VID   DURATION TOPIC                NUMVIEWS
---------- ---------- -------------------- ----------
      1245         20 ai                           19
       777         35 daa                         233

SQL> select * from videos;

       VID   DURATION TOPIC                NUMVIEWS
---------- ---------- -------------------- ----------
      1245         20 ai                           19
       777         35 daa                         233
      1602         15 cyber security               50

SQL>
```

OUTPUT OF UPDATE VIDEO:



OUTPUT OF DELETE VIDEO:


```
SQL> select * from videos;

       VID   DURATION TOPIC                    NUMVIEWS
---------- ---------- -------------------- ----------
      1245         20 ai                           19
       777         35 daa                         233
      1602         15 cyber security               50

SQL> select * from videos;

       VID   DURATION TOPIC                    NUMVIEWS
---------- ---------- -------------------- ----------
      1245         20 ai                           19
      1602         15 cyber security               50

SQL>
```

TESTING:

If a user enters an invalid value then an error message is displayed and exception is raised.

RESULT:

The process of entering information into the frame created by java code so that the data is reflected in the database using **JDBC connectivity** is done successfully.

DISCUSSION AND FUTURE WORK!

The application till now done is a basic interface in which a user Can enter the details and watch videos. So **in future the project will be edited in such a manner that will be able to identify the genuine viewers.**

REFERENCES:

https://docs.oracle.com/javase/8/docs/api/

https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/

https://www.javatpoint.com/java-awt