JAVA AWT BASED- INFORMATION RETRIEVAL OF GOOGLE QUERIES OF POSITIVE AND NEGATIVE FEEDBACK- 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

Eesha Sarang Gandhi <1602-18-737-067>

Under the guidance of Ms B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

BONAFIDE CERTIFICATE

This is to certify that this project report titled '**Information Retrieval of Google Queries of Positive and Negative Feedback**' is a project work of Ms. Eesha Gandhi bearing roll no. 1602-18-737-067 who carried out the project under my supervision in the IV semester for the academic year 2019- 2020.

Signature
Internal Examiner

Signature
External Examiner

ABSTRACT

Whenever we are in search of an instant solution to any problem, Google is the answer to most of us in such a case. Infact it is the only thing that comes to our mind when we have a question in our minds. Google has 90.46% of the search engine market share worldwide. An average person conducts atleast 3-4 searches every single day.

Numerous queries are asked and answered every single day. Inorder to keep a record of them we need to access the details of the users who post queries, the ones who provide solutions and the ones who rate the solutions or give a suggestion about them.The rating of the query depends on the exactness of the answer and queries with positive feedback comparatively have much higher rating than the queries with negative feedback.The rating and feedback of a particular query help the user to access the solution much quickly and effectively.

This makes things much simpler and easier. Any person searching an answer for a previously asked query can find the precise answer based on the feedback and the job gets done really quickly.

EESHA SARANG GANDHI
1602-18-737-067

INTRODUCTION

REQUIREMENTS

Table name	Attributes
Interrogators	i_id varchar2(10) i_name char(20)
Analyzers	a_id varchar2(10) a_name char(20)
Queries	q_id number(10) q_name varchar2(100)
Info_Retrieval	i_id varchar2(10) q_id number(10)
Rating	a_id varchar2(10) q_id number(10) feedback char(30)

AIM AND PRIORITY OF THE PROJECT

To create a Java GUI based information retrieval system of Google queries which takes the values like: user id, user name etc. from the person who posts the query/answers the query/rates the query, it also stores information about the query and its feedback. These values are to be updated in the database using JDBC connectivity .

ARCHITECTURE AND TECHNOLOGY

Software used:

Java Eclipse, Oracle 11g Database, Java SE version 13, SQL*Plus.

Java AWT:

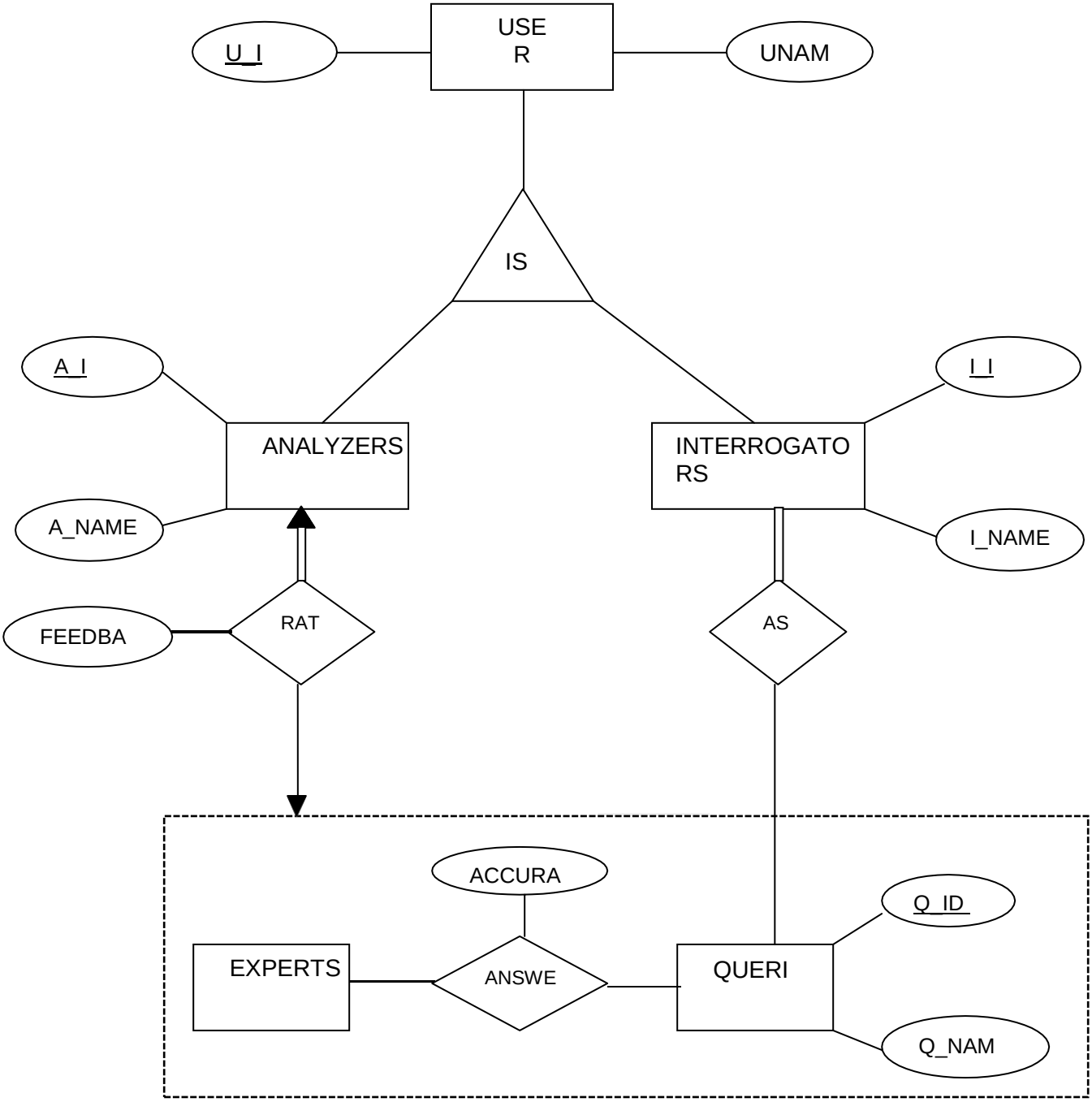
Java AWT (Abstract Window Toolkit) is an API to develop GUI or window-based applications in java.
Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.
The java.awt package provides classes for AWT API such as TextField, Label, TextArea,
RadioButton, CheckBox, Choice, List etc.

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

DESIGN

Entity relationship diagram



Database Design:

DDL Operations

Interrogators table:

```
SQL> create table interrogators(
  2  i_id varchar2(10) primary key,
  3  i_name char(20));

Table created.

SQL> desc interrogators;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 I_ID                                      NOT NULL VARCHAR2(10)
 I_NAME                                             CHAR(20)
```

Analyzers table:

```
SQL> desc analyzers;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 A_ID                                      NOT NULL VARCHAR2(10)
 A_NAME                                             CHAR(20)

SQL> create table queries(
  2  q_id number(10) primary key,
  3  q_name varchar2(100));
```

Queries table:

```
SQL> create table queries(
  2  q_id number(10) primary key,
  3  q_name varchar2(100));

Table created.

SQL> desc queries;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 Q_ID                                      NOT NULL NUMBER(10)
 Q_NAME                                             VARCHAR2(100)
```

Info Retrieval table:

```
SQL> create table info_retrieval(
  2  i_id varchar2(10),
  3  q_id number(10),
  4  foreign key(i_id) references interrogators(i_id),
  5  foreign key(q_id) references queries(q_id),
  6  primary key(i_id,q_id));

Table created.

SQL> desc info_retrieval;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 I_ID                                      NOT NULL VARCHAR2(10)
 Q_ID                                      NOT NULL NUMBER(10)
```

Rating table:

```
SQL> create table rating(
  2  a_id varchar2(10),
  3  feedback char(30),
  4  q_id number(10),
  5  foreign key(q_id) references queries(q_id),
  6  foreign key(a_id) references analyzers(a_id),
  7  primary key(a_id,q_id));

Table created.

SQL> desc rating;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 A_ID                                      NOT NULL VARCHAR2(10)
 FEEDBACK                                           CHAR(30)
 Q_ID                                      NOT NULL NUMBER(10)
```

DML Operations

Interrogators table:

```
SQL> insert into interrogators values(&i_id,'&i_name');
Enter value for i_id: 43
Enter value for i_name: piyush
old   1: insert into interrogators values(&i_id,'&i_name')
new   1: insert into interrogators values(43,'piyush')

1 row created.

SQL> /
Enter value for i_id: 31
Enter value for i_name: mohak
old   1: insert into interrogators values(&i_id,'&i_name')
new   1: insert into interrogators values(31,'mohak')

1 row created.

SQL> /
Enter value for i_id: 86
Enter value for i_name: khushi
old   1: insert into interrogators values(&i_id,'&i_name')
new   1: insert into interrogators values(86,'khushi')

1 row created.

SQL> /
Enter value for i_id: 60
Enter value for i_name: farhan
old   1: insert into interrogators values(&i_id,'&i_name')
new   1: insert into interrogators values(60,'farhan')

1 row created.

SQL> /
Enter value for i_id: 15
Enter value for i_name: devika
old   1: insert into interrogators values(&i_id,'&i_name')
new   1: insert into interrogators values(15,'devika')

1 row created.
```

```
SQL> select * from interrogators;

I_ID       I_NAME
---------- --------------------
43         piyush
31         mohak
86         khushi
60         farhan
15         devika
```

Analyzers table:

```
SQL> insert into analyzers values(10,'rohan');

1 row created.

SQL> insert into analyzers values(20,'eshan');

1 row created.

SQL> insert into analyzers values(30,'manvi');

1 row created.

SQL> insert into analyzers values(89,'kahani');

1 row created.

SQL> insert into analyzers values(45,'krish');

1 row created.

SQL> select * from analyzers;

A_ID       A_NAME
---------- --------------------
10         rohan
20         eshan
30         manvi
89         kahani
45         krish
```

Queries table:

```
SQL> select * from queries;

      Q_ID
----------
Q_NAME
--------------------------------------------------
       105
er_diagram

        47
is_a_hierachy

        35
integrity_constraints


      Q_ID
----------
Q_NAME
--------------------------------------------------
        29
relational algebra

        70
sql queries
```

Info Retrieval table:

```
SQL> insert into info_retrieval values(43,105);

1 row created.

SQL> insert into info_retrieval values(31,47);

1 row created.

SQL> insert into info_retrieval values(86,35);

1 row created.

SQL> insert into info_retrieval values(60,29);

1 row created.

SQL> insert into info_retrieval values(15,70);

1 row created.

SQL> select * from info_retrieval;

I_ID             Q_ID
---------- ----------
43                105
31                 47
86                 35
60                 29
15                 70
```

Rating table:

```
SQL> insert into rating values(10,'positive',105);

1 row created.

SQL> insert into rating values(20,'negative',47);

1 row created.

SQL> insert into rating values(30,'positive',35);

1 row created.

SQL> insert into rating values(89,'positive',29);

1 row created.

SQL> insert into rating values(45,'positive',70);

1 row created.

SQL> select * from rating;

A_ID       FEEDBACK                         Q_ID
---------- ------------------------------ ----------
10         positive                          105
20         negative                           47
30         positive                           35
89         positive                           29
45         positive                           70
```

IMPLEMENTATION

Front end programs and its connectivity

Java Database Connectivity (**JDBC**) 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. It provides methods to query and update data in a database and is oriented towards relational databases.

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:xe","eesha","eeshasoham");
                        statement = connection.createStatement();

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

Thus, the connection from Java to Oracle database is performed and therefore, can be used for updating tables in the database directly.

Insert queries:

```
package assignment2;

importjava.awt.*;
importjava.awt.event.*;
importjava.sql.*;
public class InsertQueries extends Frame
{
	Button insertQueriesButton;
	TextFieldq_idText, q_nameText;
	TextAreaerrorText;
	Connection connection;
	Statement statement;
	publicInsertQueries()
	{
		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","eesha","eeshasoham");
		statement = connection.createStatement();

		}
		catch (SQLExceptionconnectException)
		{
		System.out.println(connectException.getMessage());
		System.out.println(connectException.getSQLState());
		System.out.println(connectException.getErrorCode());
		System.exit(1);
		}
  }
	public void buildGUI()
	{
		//Handle Insert Account Button
		insertQueriesButton = new Button("Insert Query");
		insertQueriesButton.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent e)
			{
				try
				{
```

```
                                        String query= "INSERT INTO queries VALUES(" + q_idText.getText() + ", "
+ "'" + q_nameText.getText() + ")";
                                        int i = statement.executeUpdate(query);
                                        errorText.append("\nInserted " + i + " rows successfully");
                                        }
                                        catch (SQLExceptioninsertException)
                                        {
                                        displaySQLErrors(insertException);
                                        }
                                }
                        });


                        q_idText = new TextField(15);
                        q_nameText = new TextField(15);


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

                        Panel first = new Panel();
                        first.setLayout(new GridLayout(4, 2));
                        first.add(new Label("Query ID:"));
                        first.add(q_idText);
                        first.add(new Label("Query name:"));
                        first.add(q_nameText);
                        first.setBounds(125,90,200,100);

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

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

                        setLayout(null);

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

                        setTitle("New Query Creation");
                        setSize(500, 600);
                        setVisible(true);
        }



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

```
	public static void main(String[] args)
	{
		InsertQueries sail = new InsertQueries();

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

		sail.buildGUI();
	}
}
```

Update queries:

```
package assignment2;

importjava.awt.*;
importjava.awt.event.*;
importjava.sql.*;
public class UpdateQueries extends Frame
{
	Button updateQueriesButton;
	List queriesIDList;
	TextFieldq_idText, q_nameText;
	TextAreaerrorText;
	Connection connection;
	Statement statement;
	ResultSetrs;

	publicUpdateQueries()
	{
		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","eesha","eeshasoham");
		statement = connection.createStatement();
```

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

	private void loadQueries()
	{
		try
		{
		rs = statement.executeQuery("SELECT q_id FROM queries");
		while (rs.next())
		  {
			queriesIDList.add(rs.getString("q_id"));
		  }
		}
		catch (SQLException e)
		{
		displaySQLErrors(e);
		}
	}

	public void buildGUI()
	{
		queriesIDList = new List(10);
		loadQueries();
		add(queriesIDList);

		//When a list item is selected populate the text fields
		queriesIDList.addItemListener(new ItemListener()
		{
			public void itemStateChanged(ItemEvent e)
			{
				try
				{
					rs = statement.executeQuery("SELECT * FROM queries where q_id
="+queriesIDList.getSelectedItem());
					rs.next();
					q_idText.setText(rs.getString("q_id"));
					q_nameText.setText(rs.getString("q_name"));


				}
				catch (SQLExceptionselectException)
				{
					displaySQLErrors(selectException);
				}
			}
		});


		//Handle Update Sailor Button
```

```
		updateQueriesButton = new Button("Update Queries");
		updateQueriesButton.addActionListener(new ActionListener()
		{
			public void actionPerformed(ActionEvent e)
			{
				try
				{
					Statement statement = connection.createStatement();
					int i = statement.executeUpdate("UPDATE queries "
					+ "SET q_name='" + q_nameText.getText() + "', "
					+ " WHERE q_id= "
					+ queriesIDList.getSelectedItem());
					errorText.append("\nUpdated " + i + " rows successfully");
					queriesIDList.removeAll();
					loadQueries();
				}
				catch (SQLExceptioninsertException)
				{
					displaySQLErrors(insertException);
				}
			}
		});
		q_idText = new TextField(15);
		q_nameText = new TextField(15);



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

		Panel first = new Panel();
		first.setLayout(new GridLayout(4, 2));
		first.add(new Label("Query ID:"));
		first.add(q_idText);
		first.add(new Label("Query name:"));
		first.add(q_nameText);

		first.setBounds(125,90,200,100);


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

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

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

		setTitle("Update Queries");
		setSize(500, 600);
		setLayout(new FlowLayout());
		setVisible(true);

	}
```

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

	public static void main(String[] args)
	{
		UpdateQueries ups = new UpdateQueries();

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

		ups.buildGUI();
	}
}
```

Delete queries:

```
package assignment2;

importjava.awt.*;
importjava.awt.event.*;
importjava.sql.*;
public class DeleteQueries extends Frame
{
	Button deleteQueriesButton;
	List QueriesIDList;
	TextFieldq_idText, q_nameText;
	TextAreaerrorText;
	Connection connection;
	Statement statement;
	ResultSetrs;

	publicDeleteQueries ()
	{
		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","eesha","eeshasoham");
		statement = connection.createStatement();

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

	private void loadQueries ()
	{
		try
		{
		rs = statement.executeQuery("SELECT * FROM queries");
		while (rs.next())
		  {
			QueriesIDList.add(rs.getString("q_id"));
		  }
		}
		catch (SQLException e)
		{
		displaySQLErrors(e);
		}
	}

	public void buildGUI()
	{
		QueriesIDList = new List(10);
		loadQueries ();
		add(QueriesIDList);

		//When a list item is selected populate the text fields
		QueriesIDList.addItemListener(new ItemListener()
		{
			public void itemStateChanged(ItemEvent e)
			{
				try
				{
					rs = statement.executeQuery("SELECT * FROM Queries");
					while (rs.next())
					{

	if(rs.getString("e_id").equals(QueriesIDList.getSelectedItem()))
						break;
					}
					if (!rs.isAfterLast())
					{
						q_idText.setText(rs.getString("q_id"));
						q_nameText.setText(rs.getString("q_name"));
```

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


            //Handle Delete Sailor Button
            deleteQueriesButton = new Button("Delete Query");
            deleteQueriesButton.addActionListener(new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                        Statement statement = connection.createStatement();
                        int i = statement.executeUpdate("DELETE FROM Queries WHERE
q_id = "
                                +QueriesIDList.getSelectedItem());
                        errorText.append("\nDeleted " + i + " rows successfully");
                        q_idText.setText(null);
                        q_nameText.setText(null);

                        QueriesIDList.removeAll();
                        loadQueries();
                    }
                    catch (SQLExceptioninsertException)
                    {
                        displaySQLErrors(insertException);
                    }
                }
            });
            q_idText = new TextField(15);
            q_nameText = new TextField(15);



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

            Panel first = new Panel();
            first.setLayout(new GridLayout(4, 2));
            first.add(new Label("Query ID:"));
            first.add(q_idText);
            first.add(new Label("Query name:"));
            first.add(q_nameText);

            first.setBounds(125,90,200,100);

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

```
second.setBounds(125,220,150,100);


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

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

                setTitle("Remove Query");
                setSize(450, 600);
                setLayout(new FlowLayout());
                setVisible(true);

        }



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



        public static void main(String[] args)
        {
                DeleteQueriesdels = new DeleteQueries();

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

                dels.buildGUI();
        }
}
```

Main Program:

```
package assignment2;

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

class GoogleQueries extends Frame implements ActionListener
{
	String msg = "";
	Label ll;
	InsertInterrogators iinterrogators;
	UpdateInterrogators uinterrogators;
	DeleteInterrogators dinterrogators;
	InsertQueries iqueries;
	DeleteQueries dqueries;
	UpdateQueries uqueries;
	InsertAnalyzers ianalyzers;
	DeleteAnalyzers danalyzers;
	UpdateAnalyzers uanalyzers;
	InsertRating irating;
	DeleteRating drating;
	UpdateRating urating;
	InsertInfoRetrieval iinfoRetrieval;
	DeleteInfoRetrieval dinfoRetrieval;
	UpdateInfoRetrieval uinfoRetrieval;

	GoogleQueries()
	{
			ll = new Label();
			ll.setAlignment(Label.CENTER);
			ll.setBounds(100,250,250,200);
			ll.setText("WELCOME TO GOOGLE QUERIES");
			add(ll);

			// create menu bar and add it to frame
			MenuBar mbar = new MenuBar();
			setMenuBar(mbar);

			// create the menu items and add it to Menu
			Menu interrogators = new Menu("Interrogators");
			MenuItem item1, item2, item3;
			interrogators.add(item1 = new MenuItem("Insert Interrogators"));
			interrogators.add(item2 = new MenuItem("Update Interrogators"));
			interrogators.add(item3 = new MenuItem("Delete Interrogators"));
			mbar.add(interrogators);

			Menu queries = new Menu("Queries");
			MenuItem item4, item5, item6;
			queries.add(item4 = new MenuItem("Insert Queries"));
			queries.add(item5 = new MenuItem("Update Queries"));
			queries.add(item6 = new MenuItem("Delete Queries"));
			mbar.add(queries);

			Menu analyzers = new Menu("Analyzers");
```

```
                        MenuItem item7, item8, item9;
                        analyzers.add(item7 = new MenuItem("Insert Analyzers"));
                        analyzers.add(item8 = new MenuItem("Update Analyzers"));
                        analyzers.add(item9 = new MenuItem("Delete Analyzers"));
                        mbar.add(analyzers);

                        Menu rating= new Menu("Rating");
                        MenuItem item10,item11,item12;
                        rating.add(item10=new MenuItem("Insert Rating"));
                        rating.add(item11=new MenuItem("Update Rating"));
                        rating.add(item12=new MenuItem("Delete Rating"));
                        mbar.add(rating);

                        Menu infoRetrieval=new Menu("InfoRetrieval");
                        MenuItem item13,item14,item15;
                        infoRetrieval.add(item13=new MenuItem("Insert InfoRetrieval"));
                        infoRetrieval.add(item14=new MenuItem("Update InfoRetrieval"));
                        infoRetrieval.add(item15=new MenuItem("DeleteInfoRetrieval"));
                        mbar.add(infoRetrieval);

                        // register listeners
                        item1.addActionListener(this);
                        item2.addActionListener(this);
                        item3.addActionListener(this);
                        item4.addActionListener(this);
                        item5.addActionListener(this);
                        item6.addActionListener(this);
                        item7.addActionListener(this);
                        item8.addActionListener(this);
                        item9.addActionListener(this);
                        item10.addActionListener(this);
                        item11.addActionListener(this);
                        item12.addActionListener(this);
                        item13.addActionListener(this);
                        item14.addActionListener(this);
                        item15.addActionListener(this);

                         // Anonymous inner class which extends WindowAdaptor to handle the Window event:
windowClosing
                        addWindowListener(new WindowAdapter(){
                                public void windowClosing(WindowEvent we)
                                {
                                        System.exit(0);
                                }
                        });

                        //Frame properties
                        setTitle("Google Queries ");
                        Color clr = new Color(230, 190, 250);
                        setBackground(clr);
                        setFont(new Font("Cambria", Font.BOLD, 15));
                        setLayout(null);
                        setSize(500, 600);
                        setVisible(true);

        }
```

```
public void actionPerformed(ActionEvent ae)
{

	 String arg = ae.getActionCommand();
	 if(arg.equals("Insert Interrogators"))
	 {
		iinterrogators = new InsertInterrogators();
		iinterrogators.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			iinterrogators.dispose();
		}
		});
		iinterrogators.buildGUI();
	}

	 else if(arg.equals("Update Interrogators"))
	 {
		uinterrogators = new UpdateInterrogators();
		uinterrogators.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			uinterrogators.dispose();
		}
		});
		uinterrogators.buildGUI();
	 }

	 else if(arg.equals("Delete Interrogators"))
	 {
		dinterrogators = new DeleteInterrogators();
		dinterrogators.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			dinterrogators.dispose();
		}
		});
		dinterrogators.buildGUI();
	 }

	 else if(arg.equals("Insert Queries"))
	 {
		iqueries = new InsertQueries();
		iqueries.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			iqueries.dispose();
		}
		});
		iqueries.buildGUI();
	 }
	 else if(arg.equals("Update Queries"))
	 {
		uqueries = new UpdateQueries();
		uqueries.addWindowListener(new WindowAdapter(){
```

```
		public void windowClosing(WindowEvent e)
		{
			uqueries.dispose();
		}
		});
		uqueries.buildGUI();
	}
	else if(arg.equals("Delete Queries"))
	{
		dqueries = new DeleteQueries();
		dqueries.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			dqueries.dispose();
		}
		});
		dqueries.buildGUI();
	}
	else if(arg.equals("Insert Analyzers"))
	{
		ianalyzers = new InsertAnalyzers();
		ianalyzers.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			ianalyzers.dispose();
		}
		});
		ianalyzers.buildGUI();
	}
	else if(arg.equals("Update Analyzers"))
	{
		uanalyzers = new UpdateAnalyzers();
		uanalyzers.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			uanalyzers.dispose();
		}
		});
		uanalyzers.buildGUI();
	}
	else if(arg.equals("Delete Analyzers"))
	{
		danalyzers= new DeleteAnalyzers();
		danalyzers.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			danalyzers.dispose();
		}
		});
		danalyzers.buildGUI();
	}
	else if(arg.equals("Insert Rating"))
	{
		irating = new InsertRating();
		irating.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
```

```
		{
			irating.dispose();
		}
		});
		irating.buildGUI();
	}
	else if(arg.equals("Update Rating"))
	{
		urating = new UpdateRating();
		urating.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			urating.dispose();
		}
		});
		urating.buildGUI();
	}
	else if(arg.equals("Delete Rating"))
	{
		drating= new DeleteRating();
		drating.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			drating.dispose();
		}
		});
		drating.buildGUI();
	}
	else if(arg.equals("Insert InfoRetrieval"))
	{
		iinfoRetrieval = new InsertInfoRetrieval();
		iinfoRetrieval.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			iinfoRetrieval.dispose();
		}
		});
		iinfoRetrieval.buildGUI();
	}
	else if(arg.equals("Update InfoRetrieval"))
	{
		uinfoRetrieval = new UpdateInfoRetrieval();
		uinfoRetrieval.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
			uinfoRetrieval.dispose();
		}
		});
		uinfoRetrieval.buildGUI();
	}
	else if(arg.equals("Delete InfoRetrieval"))
	{
		dinfoRetrieval = new DeleteInfoRetrieval();
		dinfoRetrieval.addWindowListener(new WindowAdapter(){
		public void windowClosing(WindowEvent e)
		{
```

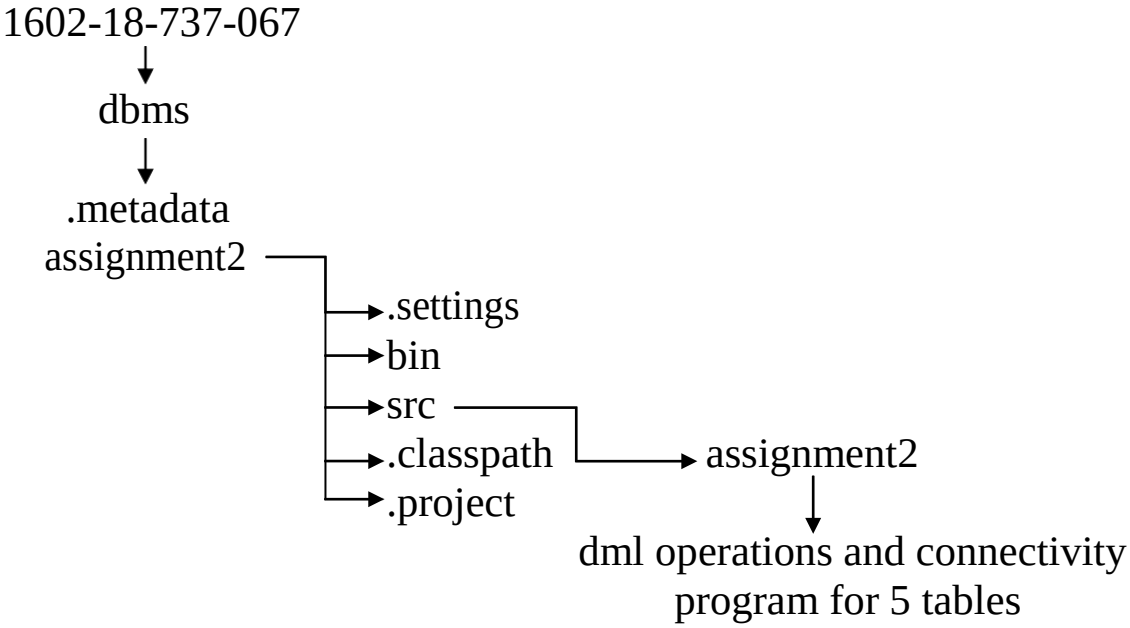
```
                                        dinfoRetrieval.dispose();
                                }
                                });
                                dinfoRetrieval.buildGUI();
                        }
                }
                public static void main(String ... args)
                {
                        new GoogleQueries();
                }
}
```

Github links and folder structure:

Link: https://github.com/eeshagandhi/google_queries

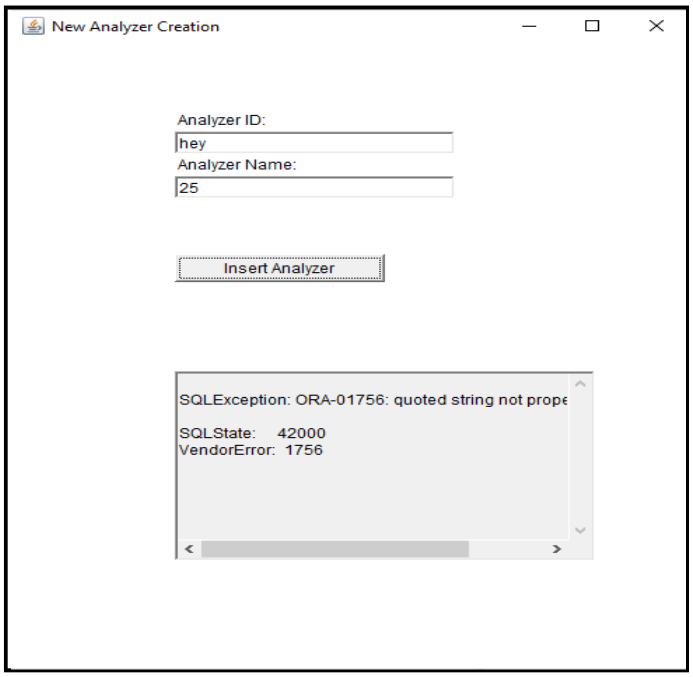
Folder structure:

1602-18-737-067.rar
google_queries.pdf (assignment 1)
report.pdf

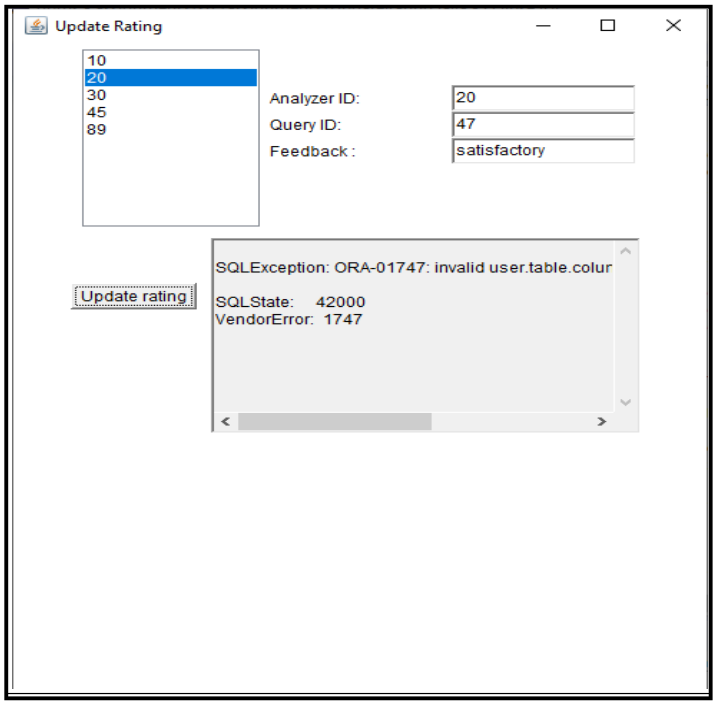


TESTING

The analyzer's id is a unique value and it can only have a numeric value and not anything else, similarly the name can only be a string. In case of any unexpected or incorrect input it throws an exception.



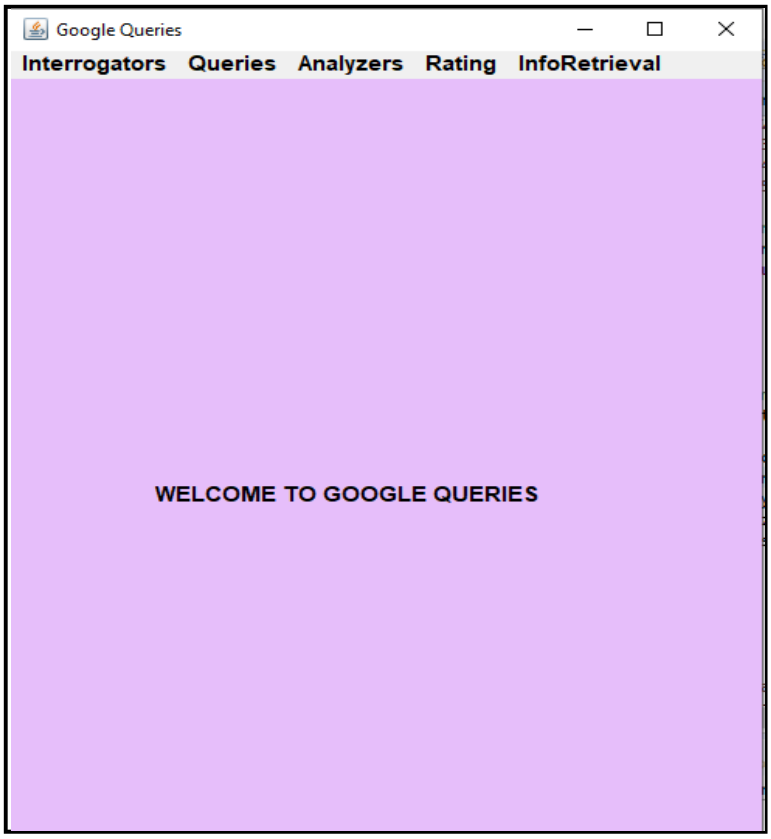
The feedback can be either positive or negative and nothing else since it's a quick poll and not a long review.



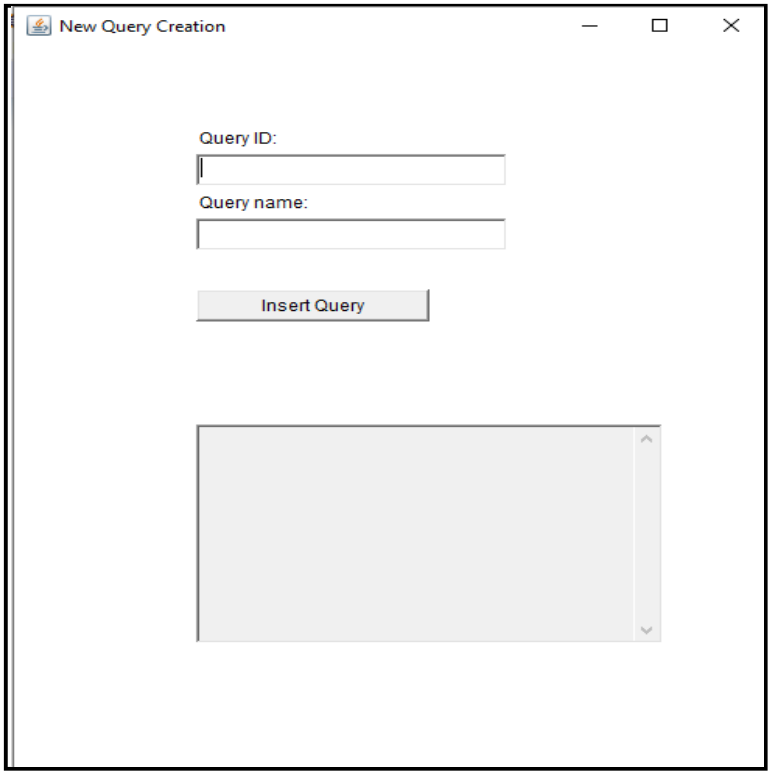
RESULTS

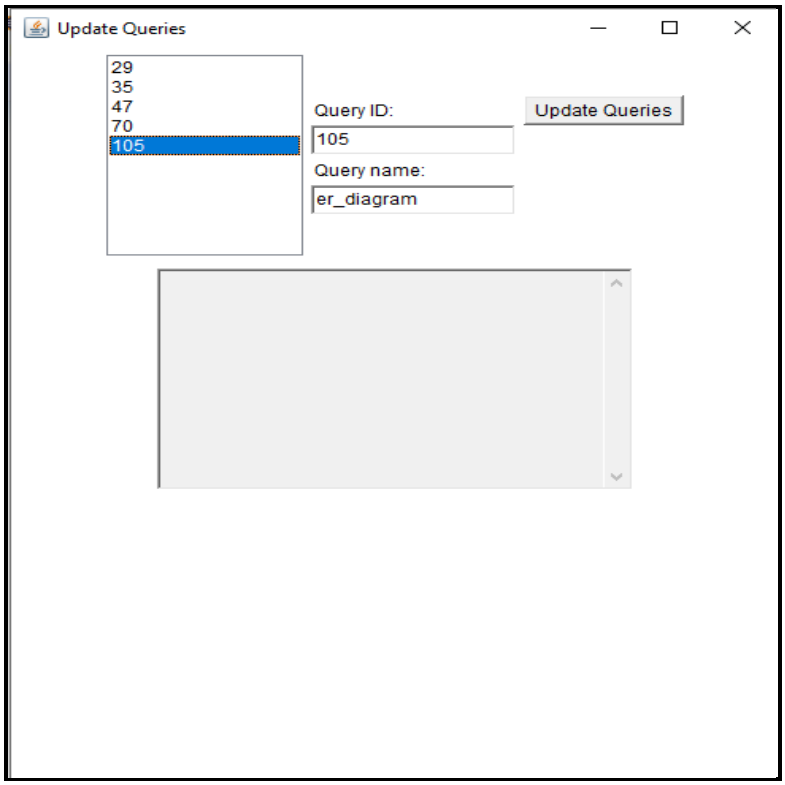
The main page:

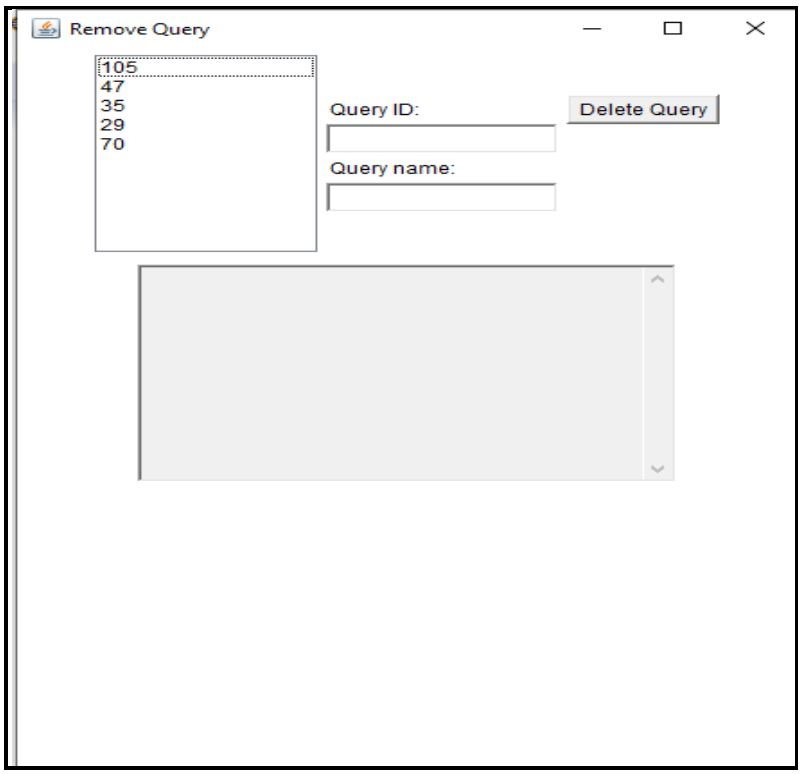
It displays the title of the project and it has menu items which are named after the five tables. Each menu item provides three options that can be performed on that particular table: Insert, Update and Delete.



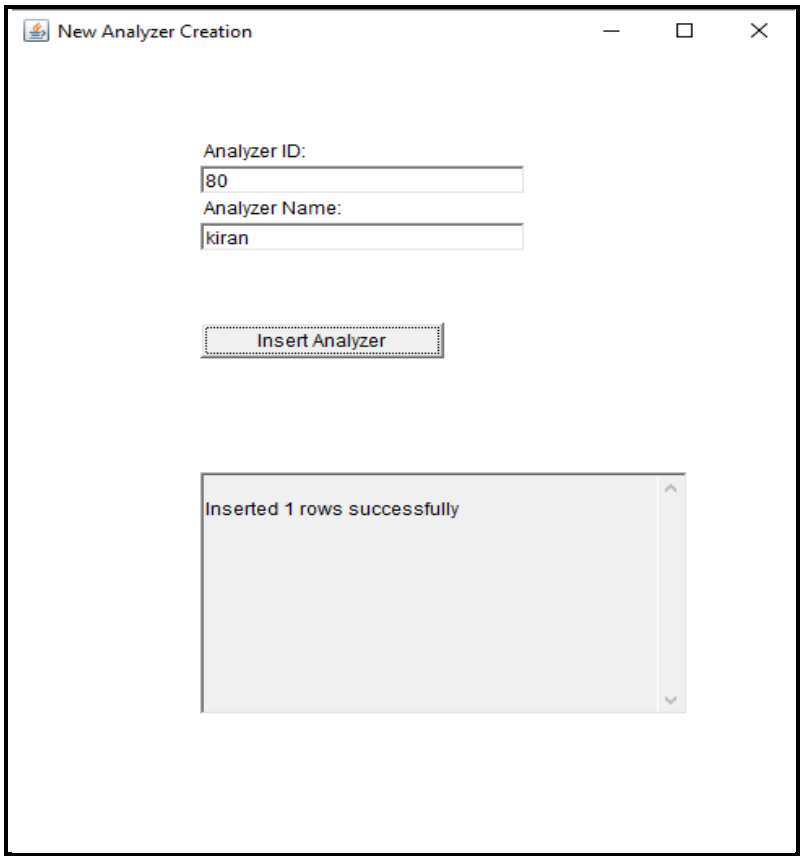
Queries:

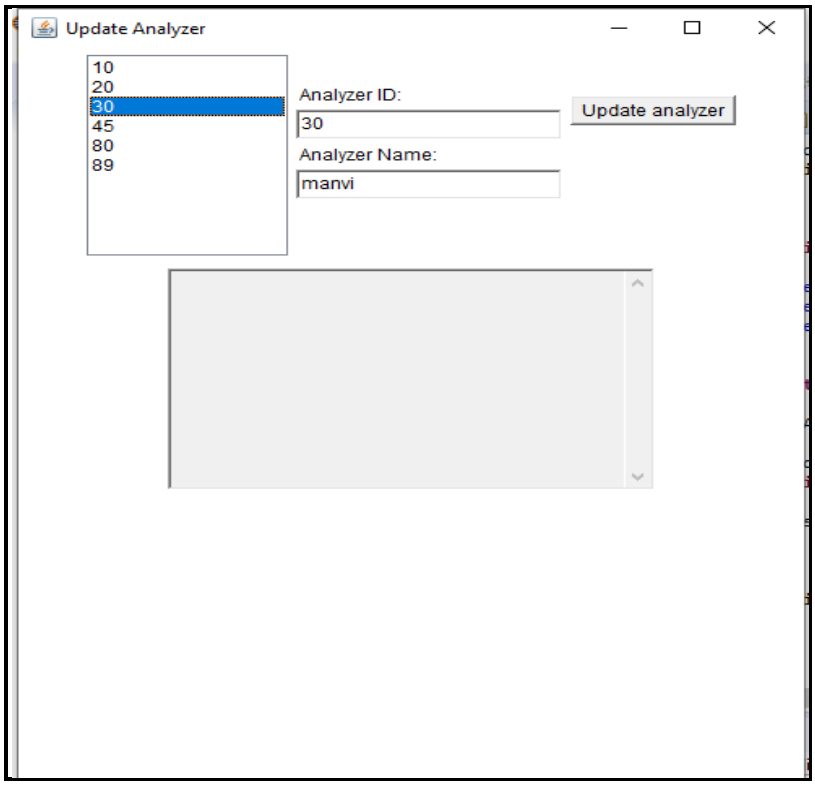

New Query Creation
Query ID:
Query name:
Insert Query

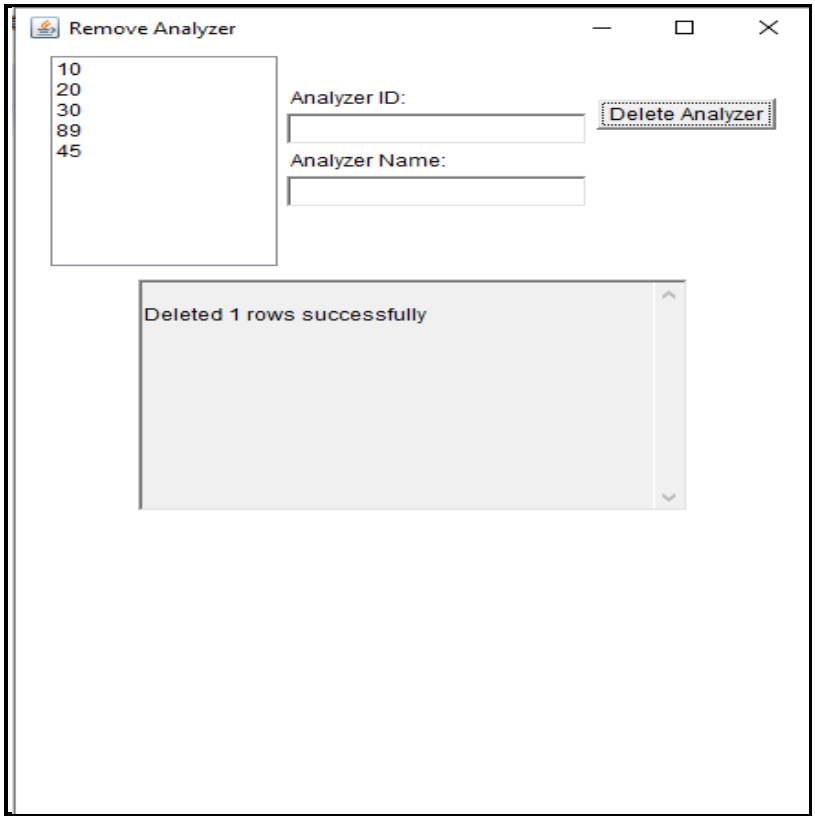

Update Queries
29
35
47
70
105
Query ID:
105
Query name:
er_diagram
Update Queries


Remove Query
105
47
35
29
70
Query ID:
Delete Query
Query name:

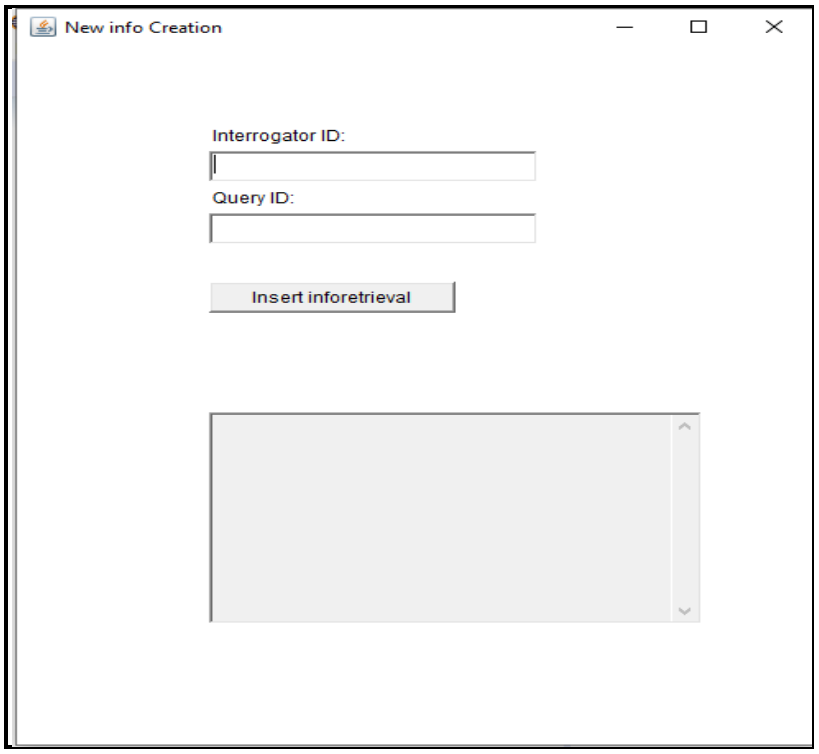
Analyzers:

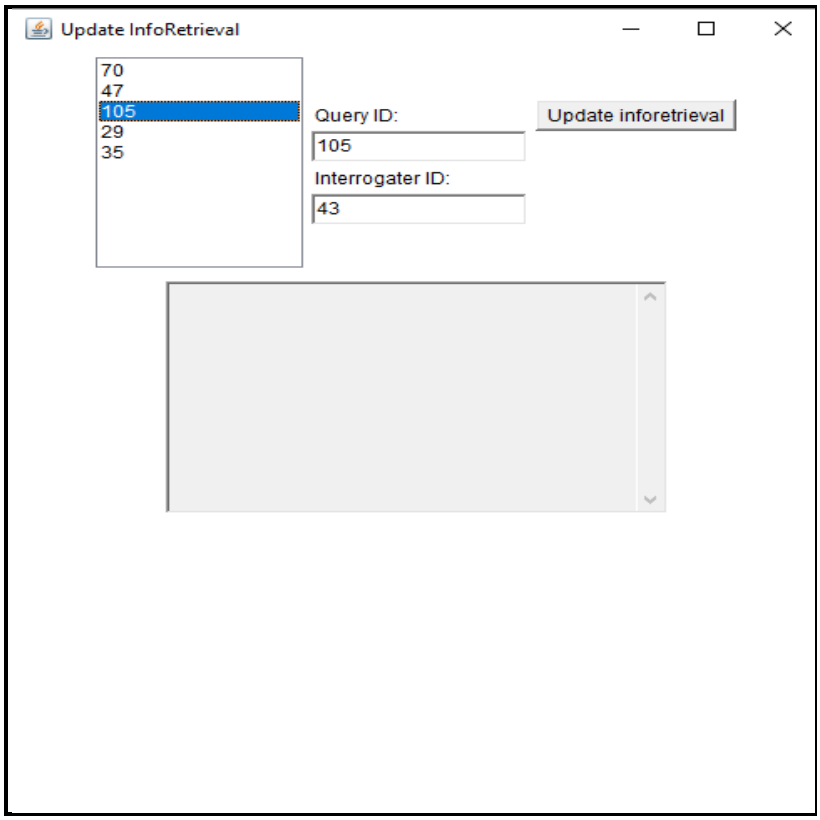

New Analyzer Creation
Analyzer ID:
80
Analyzer Name:
kiran
Insert Analyzer
Inserted 1 rows successfully


Update Analyzer
10
20
30
45
80
89
Analyzer ID:
30
Update analyzer
Analyzer Name:
manvi

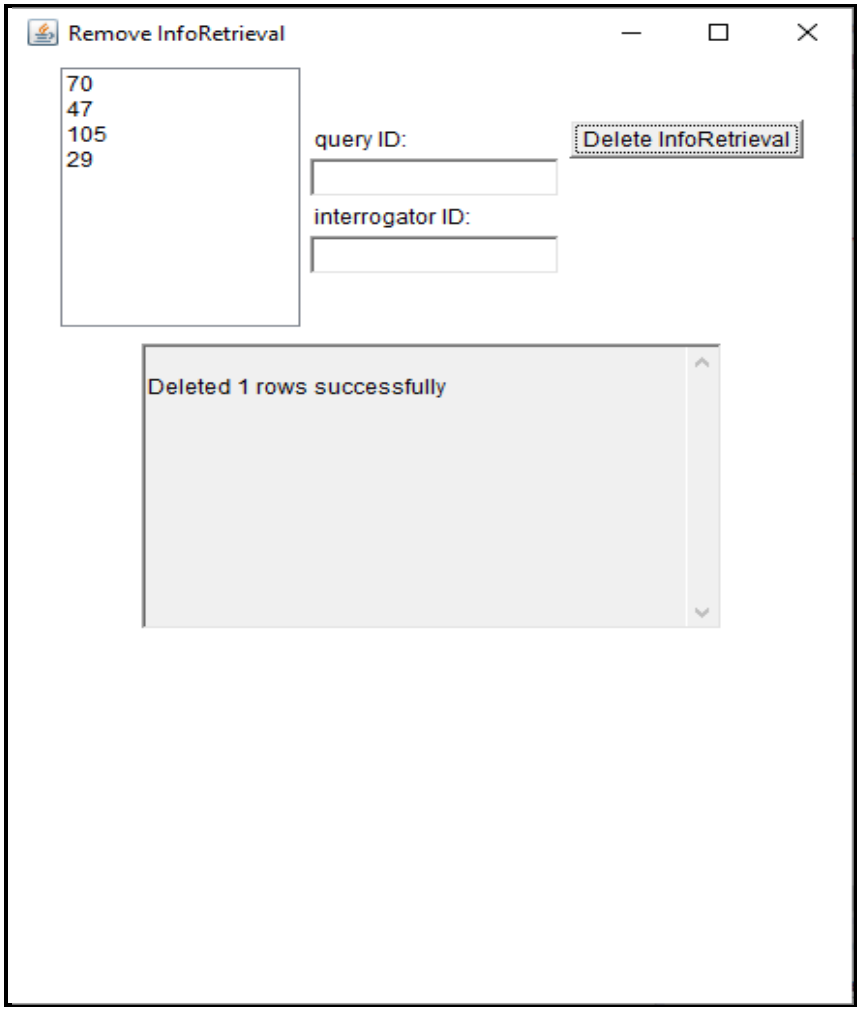

Remove Analyzer
10
20
30
89
45
Analyzer ID:
Delete Analyzer
Analyzer Name:
Deleted 1 rows successfully

Info Retrieval:





EESHA SARANG GANDHI
1602-18-737-067


Remove InfoRetrieval
70
47
105
29
query ID:
Delete InfoRetrieval
interrogator ID:
Deleted 1 rows successfully

DISCUSSION AND FUTURE WORK

So far this project has helped us retrieve the information about Google queries through the feedback and rating given by the users. The feedback eventually is the crucial aspect here as it is the parameter that rates the query and its solution. Qualitative feedbacks give a room for improvising things so that one can come up with a better solution.

In future the most probable aspects could be involving analyzers with a high intellect merely to improve the feedback quality. The query also could be analyzed based on how difficult or easy its solution could be. The feedback system could be made more comprehensive rather than just being stuck to either positive or negative feedbacks. Lastly, there could be a room for including advanced software and other technologies that could make the project more purposeful and better for future use.

REFERENCES

https://www.oracle.com/in/database/technologies/112010-win64soft.html

https://www.youtube.com/watch?v=fMp63HsIRbc&t=107s

https://mail.google.com/mail/u/0/#all/FMfcgxwHMGDXGBJRHPNwxkZSqQDLgPMR

https://mail.google.com/mail/u/0/#search/sam/FMfcgxwHMZGcqMdcjVXHfcLtWgdqPZdK

THANK YOU

EESHA SARANG GANDHI
1602-18-737-067