

SUGGESTION BOT FOR INTERVIEW SKILLS MANAGEMENT

A Report

Submitted in partial fulfilment of the

**BE IV SEMESTER DATABASE MANAGEMENT SYSTEMS LAB
INFORMATION TECHNOLOGY**

BY

J Gayatri <1602-18-737-068>

Under the guidance of

B Leelavathy



Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2020

BONAFIDE CERTIFICATE

This is to certify that this project entitles “**SUGGESTION BOT FOR INTERVIEW SKILLS MANAGEMENT**” is a bonafide mini project work of Ms. **J Gayatri** bearing the hall ticket number **1602-18-737-068** who carried out the project under my supervision in the year **2020** certified further my best knowledge.

Signature of the examiner

B. LEELAVATHY

Associate professor

Department of Information Technology

ABSTRACT

This project is called “Suggestion Bot for Interview Skills Management”. In today’s competitive world, there are various skills that one requires in order to be a part of the competing group. With the number of engineers increasing each passing day, companies often look for students with unique and diverse talents when hiring. Everyone has a dream to crack a job in one of the top companies yet are often unaware of the different requirements different companies have. Thus, this suggestion bot is here to save you and your dreams. This bot takes all your skills into consideration and lets you choose your dream company. It then provides suggestions related to the areas you need to improve and onto which level you need to improve.

INTRODUCTION

1. Requirements about project domain in general

Aim:

To do this project, insight into java and Structured Query Language are required. The project is creation of Java GUI based Suggestion Bot which takes values like user details, company details, skills of the user and suggest the user required skills for a particular company. These values are to be taken through Java GUI and updated into the database using JDBC connectivity.

2. Information about the project

The project aims at providing a platform made from Java GUI, to the user where he/she can enter their details, choose a company of their choice, and get the suggestions based on their skill levels. The main objective of the project is to understand the procedure of Java Database Connectivity.

3. Architecture and Technology used

Technology:

The software used is Java Eclipse and SQL * Plus- Oracle 11g Enterprise Edition.

Java AWT:

Java AWT (Abstract Window Toolkit) is an API to develop graphical user interface 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 which means that its components are using the resources of OS.

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 use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

Java-SQL Connectivity using 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. It provides methods to query and update data in a database and is oriented towards relational databases.

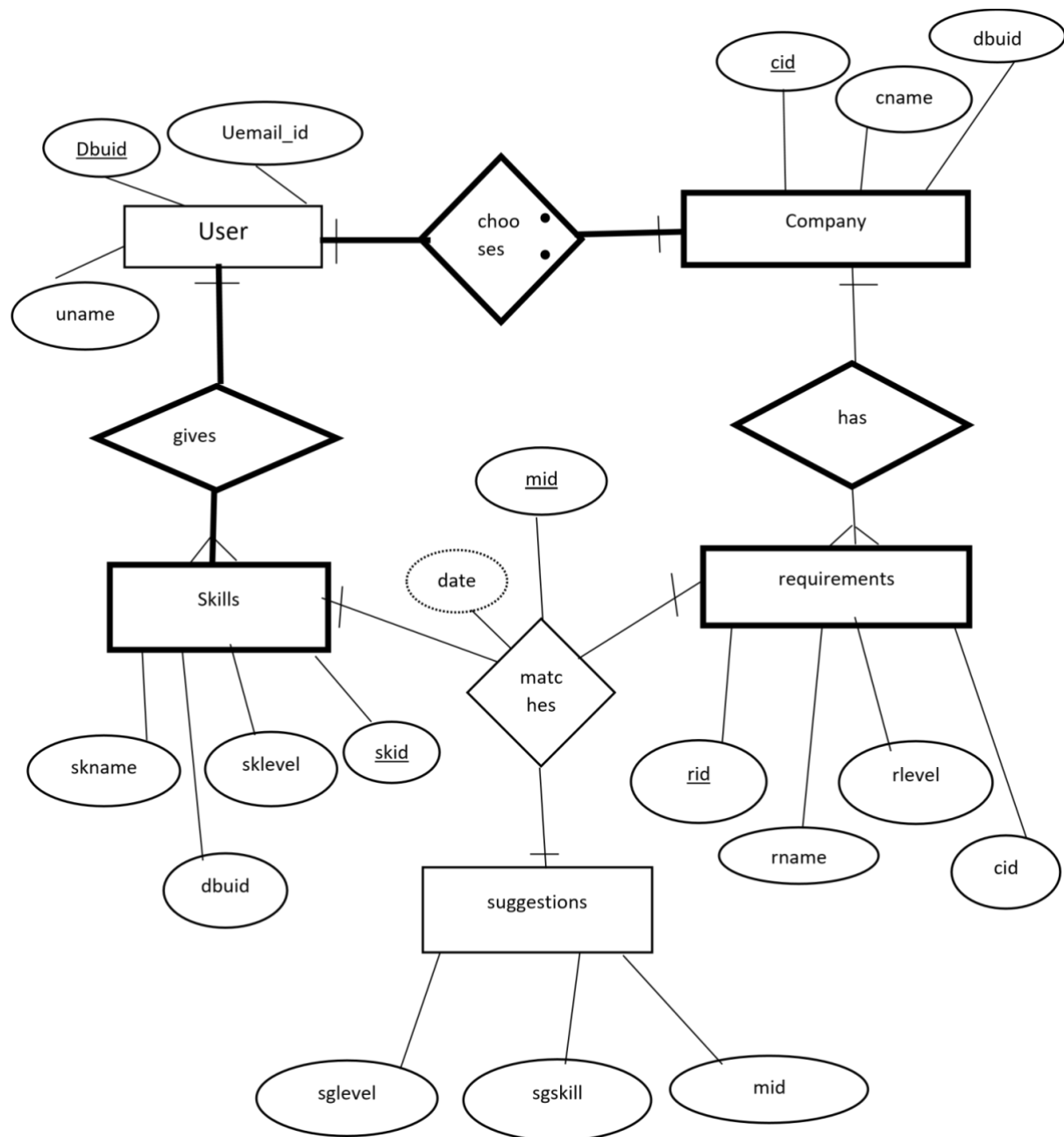
4. Design

Requirement Analysis:

<u>Table</u>	<u>Attributes</u>
EndUser	euid number(5) euname varchar2(20) euemail_id varchar(20)
Skill	skid number(5) skname varchar2(20) sklevel varchar2(20)
Companies	cid number(5) cname varchar2(20)
User_company	cid number(5) user_id number(5) preference varchar2(10)

Requirements	rid number(5) rname varchar2(20) rlevel varchar2(20)
User_skill	Euid number(5) Skid number(5) Since date
Company_reuirements	Cmp_id number(5) Req_id number(5) since date
Suggestions	sgskill varchar2(20) sglevel varchar2(20) mid number(5) euid number(5)
Requirement_suggestion	Sugg_id number(5) Req_id number(5) Skills_id number(5) Since date

Entity-Relation Diagram:



Mapping Cardinalities and Constraints:

One user can choose one company. Thus, company and user has one to one mapping cardinalities.

One user can give many skills. Thus, skills and user has one to many mapping cardinalities.

Each company/ one company can have any number (many) requirements. Thus, company and requirements has one to many mapping cardinalities.

Each skill has one respective requirement which will be given by a suggestion. Thus, skills, requirements and suggestion have a one to one mapping cardinalities.

Since, user is participating completely, user has total participation, which is indicated by the bold line.

Table Creation: {DDL Commands}

Create table enduser(euid number(5), euname varchar2(20),
euemail_id varchar2(20), primary key(euid));

Create table skill(skid number(5), skname varchar2(20), sklevel
varchar2(20), primary key(skid));

Create table companies(cid number(5), cname varchar2(20), primary
key(cid));

Create table requirement(rid number(5), rname varchar2(20), rlevel
varchar2(20), primary key(rid));

Create table suggestion(sgskill varchar2(20), sglevel varchar2(20),
mid number(5), u_id number(5), primary key(mid));

Create table user_skill(euid number(5), skid number(5), since
varchar2(20));

Create table cmp_req(cmp_id number(5), req_id number(5), since
varchar2(20));

Create table user_company(cid number(5), user_id number(5),
preference varchar2(20));

Create table req_sugg(sugg_id number(5), since varchar2(20),
skill_id number(5), reqs_id number(5));

DBMS Mini Project

TOPIC: Suggestion Bot for Interview Skills Management

```
Run SQL Command Line
SQL> desc enduser;
Name                                     Null?   Type
-----
EUID                                     NOT NULL NUMBER(5)
EUNAME                                  VARCHAR2(20)
EUEMAIL_ID                             VARCHAR2(20)

SQL> desc skill;
Name                                     Null?   Type
-----
SKID                                     NOT NULL NUMBER(5)
SKNAME                                  VARCHAR2(20)
SKLEVEL                                VARCHAR2(20)

SQL> desc companies;
Name                                     Null?   Type
-----
CID                                     NOT NULL NUMBER(5)
CNAME                                  VARCHAR2(20)

SQL> desc requirement;
Name                                     Null?   Type
-----
RID                                     NOT NULL NUMBER(5)
RNAME                                  VARCHAR2(20)
```

```
Run SQL Command Line
SQL> desc requirement;
Name                                     Null?   Type
-----
RID                                     NOT NULL NUMBER(5)
RNAME                                  VARCHAR2(20)
RLEVEL                                VARCHAR2(20)

SQL> desc suggestion;
Name                                     Null?   Type
-----
SGSKILL                                VARCHAR2(20)
SGLEVEL                                VARCHAR2(20)
MID                                     NOT NULL NUMBER(5)
U_ID                                    NUMBER(5)

SQL> desc user_skill;
Name                                     Null?   Type
-----
EUID                                     NOT NULL NUMBER(5)
SKID                                     NOT NULL NUMBER(5)
SINCE                                  VARCHAR2(20)

SQL> desc cmp_req;
Name                                     Null?   Type
-----
```

```

Run SQL Command Line
6 rows selected.

SQL> desc req_sugg;
Name                                Null?     Type
-----
SUGG_ID                             NUMBER(5)
SINCE                                DATE
SKILL_ID                             NUMBER(5)
REQS_ID                              NUMBER(5)

SQL> desc cmp_req;
Name                                Null?     Type
-----
CMP_ID                               NUMBER(5)
REQ_ID                               NUMBER(5)
SINCE                                NUMBER(5)

SQL> desc user_company;
Name                                Null?     Type
-----
CID                                  NUMBER(5)
USER_ID                             NUMBER(5)
PREFERENCE                           VARCHAR2(20)

SQL>

```

Adding foreign keys to the tables:

Table 1: Suggestion

Alter table suggestion add constraint u_id foreign key(u_id) references enduser(euid) on delete cascade;

Table 2: User_company

Alter table user_company add constraint cid foreign key(cid) references companies(cid) on delete cascade;

Alter table user_company add constraint user_id foreign key(user_id) references enduser(user_id) on delete cascade;

Table 3: User_skill

Alter table user_skill add constraint (euid) foreign key(euid) references enduser(euid) on delete cascade;

Alter table user_skill add constraint (skid) foreign key(skid) references skill on delete cascade;

Table 4: company_requirement

Alter table cmp_req add constraint(cmp_id) foreign key(cmp_id) references companies(cid) on delete cascade;

Alter table cmp_req add constraint (req_id) foreign key(req_id) references requirement(rid) on delete cascade;

Table 5: requirements_suggestions

Alter table req_sugg add constraint(sugg_id) foreign key(sugg_id) references suggestion(mid) on delete cascade;

Alter table req_sugg add constraint(req_id) foreign key(req_id) references requirement(rid) on delete cascade;

Alter table req_sugg add constraint(skill_id) foreign key(skill_id) references skill(skid) on delete cascade;

5. Implementation

Front end programs and connectivity:

Program to Insert Users into the “User” table of the database:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class InsertUser extends Frame
{
    Button UserB;
    TextField euidTf, eunameTf, emailidTf;
    TextArea errorText;
    Connection conn;
    Statement st;
    public InsertUser()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Cannot find and load driver");
            System.exit(1);
        }
        connectToDB();
    }

    public void buildGUI()
    {
        UserB = new Button("Insert User");
        UserB.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {

```

```

        try
        {
            String query= "INSERT INTO enduser VALUES(" +
euidTf.getText() + ", " + "" + eunameTf.getText() + "," + "" + emailidTf.getText() + "" + ")";
            int i = st.executeUpdate(query);
            errorText.append("\nInserted " + i + " rows successfully");
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});

```

```

euidTf = new TextField(15);
eunameTf = new TextField(15);
emailidTf = 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("User ID:"));
first.add(euidTf);
first.add(new Label("Name:"));
first.add(eunameTf);
first.add(new Label("Email ID:"));
first.add(emailidTf);
first.setBounds(125,90,200,100);

```

```

Panel second = new Panel(new GridLayout(4, 1));
second.add(UserB);
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("To Insert New Users");
setSize(500, 600);
setVisible(true);

```

```

}

```

```

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

public void connectToDB()
{
    try
    {
        conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","gayatri","manager");
        st = conn.createStatement();

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

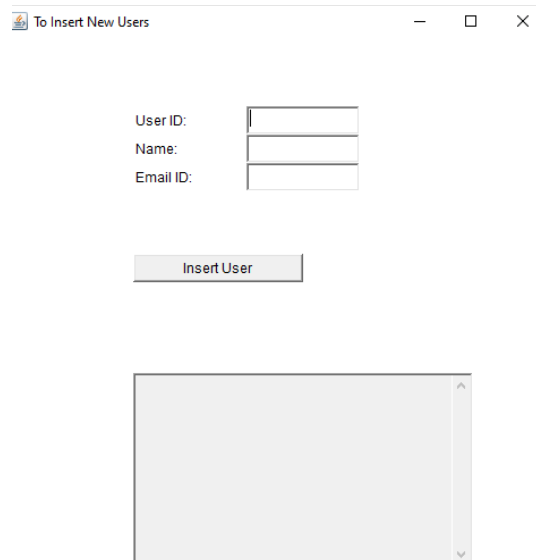
public static void main(String[] args)
{
    InsertUser user = new InsertUser();

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

    user.buildGUI();
}
}

```

OUTPUT:



Program to Update Users into the “User” table of the database:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateUser extends Frame
{
    Button UserB;
    List userList;
    TextField euidTf, eunameTf, emailidTf;
    TextArea errorText;
    Connection conn;
    Statement st;
    ResultSet rs;

    public UpdateUser()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Cannot find and load driver");
            System.exit(1);
        }
        connectToDB();
    }

    private void loadUser()
    {
        try
        {
```

```

        rs = st.executeQuery("SELECT EUID FROM enduser");
        while (rs.next())
        {
            userList.add(rs.getString("EUID"));
        }
    }
    catch (SQLException e)
    {
        displaySQLErrors(e);
    }
}

public void buildGUI()
{
    userList = new List(10);
    loadUser();
    add(userList);

    userList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = st.executeQuery("SELECT * FROM enduser where
EUID =" +userList.getSelectedItem());
                rs.next();
                euidTf.setText(rs.getString("EUID"));
                eunameTf.setText(rs.getString("EUNAME"));
                emailidTf.setText(rs.getString("EUEMAIL_ID"));

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

    UserB= new Button("Update User");
    UserB.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                Statement statement = conn.createStatement();
                int i = statement.executeUpdate("UPDATE enduser "
                + "SET euname=" + eunameTf.getText() + ", "

```

```

        + "euemail_id=" + emailidTf.getText() + "WHERE
euid = "
        + userList.getSelectedItemId());
        errorText.append("\nUpdated " + i + " rows
successfully");

        userList.removeAll();
        loadUser();
    }
    catch (SQLException insertException)
    {
        displaySQLErrors(insertException);
    }
}

euidTf = new TextField(15);
euidTf.setEditable(false);
eunameTf = new TextField(15);
emailidTf = 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("User ID:"));
first.add(euidTf);
first.add(new Label("Name:"));
first.add(eunameTf);
first.add(new Label("Email ID:"));
first.add(emailidTf);

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

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

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

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

}
public void connectToDB()
{

```



```

        try
        {
            conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","gayatri","manager");
            st = conn.createStatement();

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

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

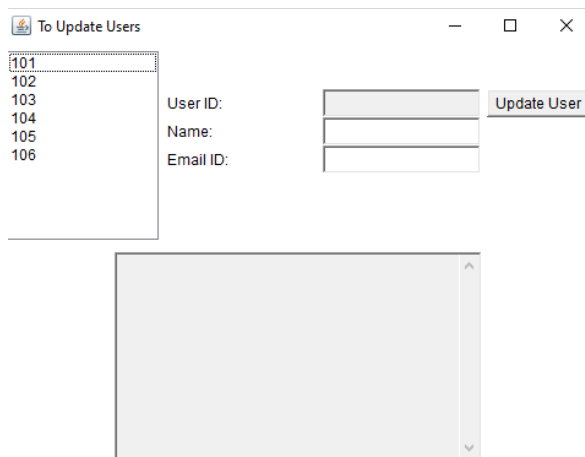
    public static void main(String[] args)
    {
        UpdateUser upUs = new UpdateUser();

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

        upUs.buildGUI();
    }
}

```

OUTPUT:



Program to Delete Users from the “User” table of the database:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteUser extends Frame
{
    Button UserB;
    List userList;
    TextField euidTf, eunameTf, emailidTf;
    TextArea errorText;
    Connection conn;
    Statement st;
    ResultSet rs;

    public DeleteUser()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Cannot find and load driver");
            System.exit(1);
        }
        connectToDB();
    }

    private void loadUsers()
    {
        try
        {
            rs = st.executeQuery("SELECT * FROM enduser");
            while (rs.next())
```

```

        {
            userList.add(rs.getString("EUID"));
        }
    }
    catch (SQLException e)
    {
        displaySQLErrors(e);
    }
}

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

    userList.addItemListener(new ItemListener()
    {
        public void itemStateChanged(ItemEvent e)
        {
            try
            {
                rs = st.executeQuery("SELECT * FROM enduser");
                while (rs.next())
                {
                    if
(rs.getString("EUID").equals(userList.getSelectedItem()))
                        break;
                }
                if (!rs.isAfterLast())
                {
                    euidTf.setText(rs.getString("EUID"));
                    eunameTf.setText(rs.getString("EUNAME"));

                    emailidTf.setText(rs.getString("EUEMAIL_ID"));
                }
            }
            catch (SQLException selectException)
            {
                displaySQLErrors(selectException);
            }
        }
    });

    UserB = new Button("Delete User");
    UserB.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try

```

```

        {
            Statement statement = conn.createStatement();
            int i = statement.executeUpdate("DELETE FROM
enduser WHERE EUID = "
                                + userList.getSelectedItemAt());
            errorText.append("\nDeleted " + i + " rows
successfully");

            euidTf.setText(null);
            eunameTf.setText(null);
            emailidTf.setText(null);
            userList.removeAll();
            loadUsers();
        }
        catch (SQLException insertException)
        {
            displaySQLErrors(insertException);
        }
    }
});

euidTf = new TextField(15);
eunameTf = new TextField(15);
emailidTf = 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("User ID:"));
first.add(euidTf);
first.add(new Label("Name:"));
first.add(eunameTf);
first.add(new Label("Email ID:"));
first.add(emailidTf);

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

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

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

setTitle("To Delete Users");
setSize(450, 600);
setLayout(new FlowLayout());
setVisible(true);

```

```

    }

    public void connectToDB()
    {
        try
        {
            conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","gayatri","manager");
            st = conn.createStatement();

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

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

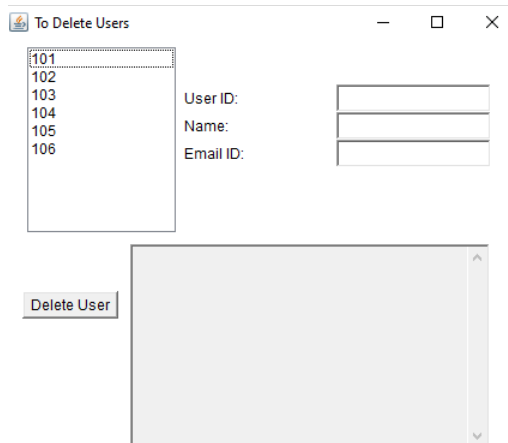
    public static void main(String[] args)
    {
        DeleteUser dels = new DeleteUser();

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

        dels.buildGUI();
    }
}

```

OUTPUT:



GitHub Link

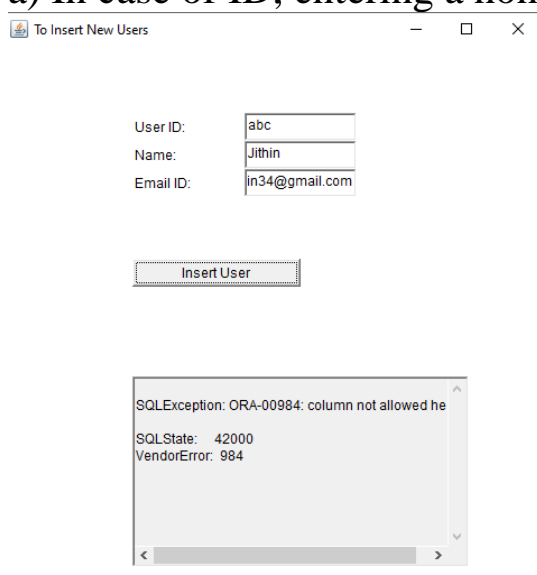
<https://github.com/jgavatri068/gayatri>

6. Testing

This section of the report deals with the testing of the connection between java GUI and the database established previously.

1. Testing for incorrect format/data type of details entered when inserting values into the database of user table using the GUI designed.

a) In case of ID, entering a non-number value.



```
SQL> insert into enduser values(&eid, '&euname','&euemail_id');
Enter value for eid: abc
Enter value for euname: Jithin
Enter value for euemail_id: jithin34@gmail.com
old 1: insert into enduser values(&eid, '&euname','&euemail_id')
new 1: insert into enduser values(abc, 'Jithin','jithin34@gmail.com')
insert into enduser values(abc, 'Jithin','jithin34@gmail.com')
*
ERROR at line 1:
ORA-00984: column not allowed here
```

b) Entering date in incorrect format

The screenshot shows two windows. The 'Run SQL Command Line' window displays the following SQL commands and errors:

```
SQL> insert into req_sugg values(&sugg_id, '&since',&skill_id,&req_id);
Enter value for sugg_id: 7
Enter value for since: 00:00:00
Enter value for skill_id: 7
Enter value for reqs_id: 1007
old 1: insert into req_sugg values(&sugg_id, '&since',&skill_id,&req_id);
new 1: insert into req_sugg values(7, '00:00:00',7,1007)
insert into req_sugg values(7, '00:00:00',7,1007)
*
ERROR at line 1:
ORA-00925: missing INTO keyword

SQL> insert into req_sugg values(&sugg_id, '&since',&skill_id,&req_id);
Enter value for sugg_id: 7
Enter value for since: 00:00:00
Enter value for skill_id: 7
Enter value for reqs_id: 1007
old 1: insert into req_sugg values(&sugg_id, '&since',&skill_id,&req_id);
new 1: insert into req_sugg values(7, '00:00:00',7,1007)
insert into req_sugg values(7, '00:00:00',7,1007)
*
ERROR at line 1:
ORA-01847: day of month must be between 1 and last day of month
```

The 'Insert Required Suggestions' form shows the following input fields:

- Suggestion ID: 7
- Day: 00:00:00
- Skill ID: 7
- Req ID: 1007

The form has an 'Insert Required Suggestions' button. Below the form, an error message is displayed:

```
SQLException: ORA-01847: day of month must be between 1 and last day of month
SQLState: 22008
VendorError: 1847
```

2. Testing for updating a non-existent value in the database.

This feature has been omitted while coding. That is, the primary key that uniquely identifies a row has been set to un-editable while coding to ensure updating primary key is not possible and to avoid further confusion.

Updating other details such as name and email id (attributes of user entity) in case of user's table is possible.

The screenshot shows two windows. The 'Select Run SQL Command Line' window displays the following SQL commands and results:

```
SQL> select * from enduser;

EUID EUNAME EUEMAIL_ID
-----
101 Riya riya1@gmail.com
102 Sameera sameera2@gmail.com
103 Mihir mihir1345@gmail.com
104 Lake lakeadams6@gmail.com
105 Molly molly789@gmail.com
106 Andrew Mcknight andrew12@gmail.com

6 rows selected.
```

The 'To Update Users' form shows the following input fields:

- User ID: 106
- Name: Andrew Mcknight
- Email ID: andrew12@gmail.com

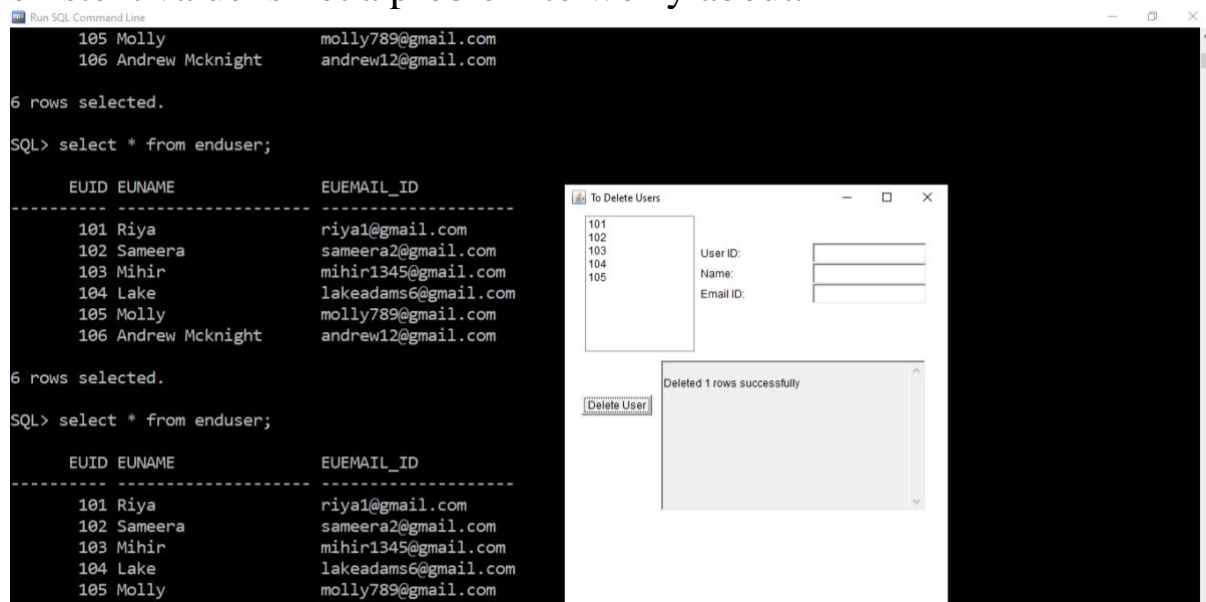
The form has an 'Update User' button. Below the form, a message is displayed:

```
Updated 1 rows successfully
```

The user id text field in the GUI coloured in grey indicates that it is un-editable.

3. Testing for a deleting a non-existent value from the database

The delete GUI is designed in such a way that it provides the user with a view of all the values inserted into the table and lets the user choose from one of them to delete the values. Thus, deleting a non-existent value is not a problem to worry about.



4. Testing for inserting values into child table those of which are not present in the parent table.

DBMS Mini Project

TOPIC: Suggestion Bot for Interview Skills Management

Run SQL Command Line

SQL> select * from companies;

CID	CNAME
1	Oracle
2	Microsoft
3	Google

SQL> select * from requirement;

RID	RNAME	RLEVEL
1001	Programming	Expert
1002	Aptitude	Expert
1003	Programming	Expert
1004	Communication	Expert
1005	Programming	Expert
1006	Aptitude	Expert
1007	Writing	Expert

7 rows selected.

Insert Company's Requirements

Company ID: 4

Req ID: 1007

Since: 2010

Insert Company's Requirements

SQLException: ORA-02291: integrity constraint (GA'

SQLState: 23000

VendorError: 2291

Insert Company's Requirements

Company ID: 4

Req ID: 1007

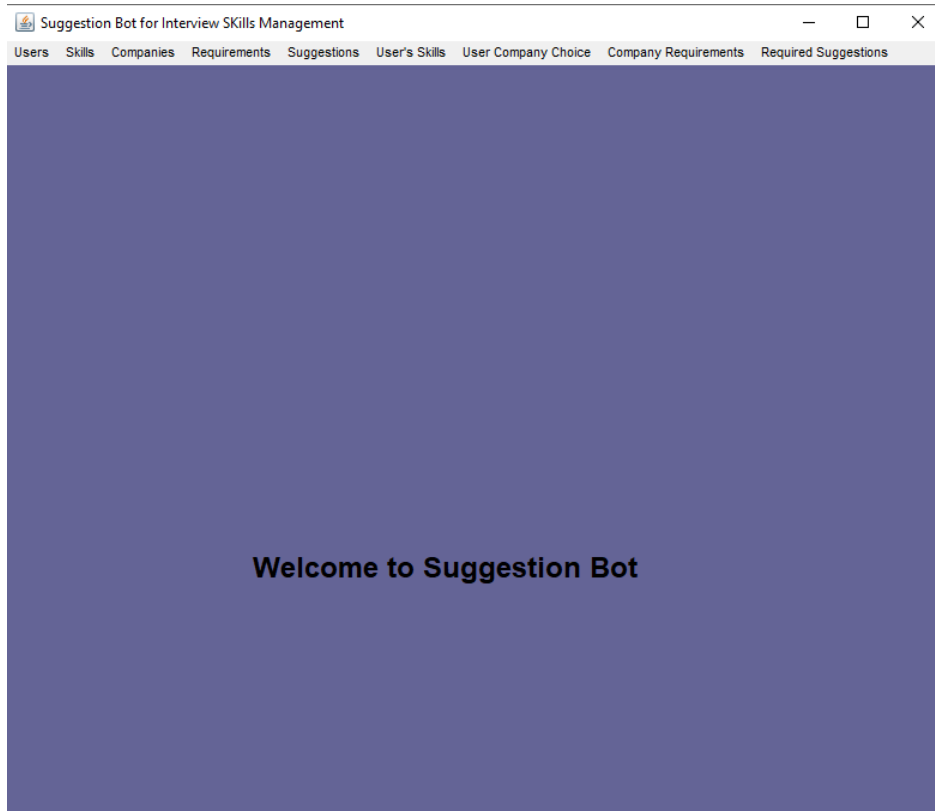
Since: 2010

Insert Company's Requirements

t(GAYATRI.CMP_ID) violated - parent key not found

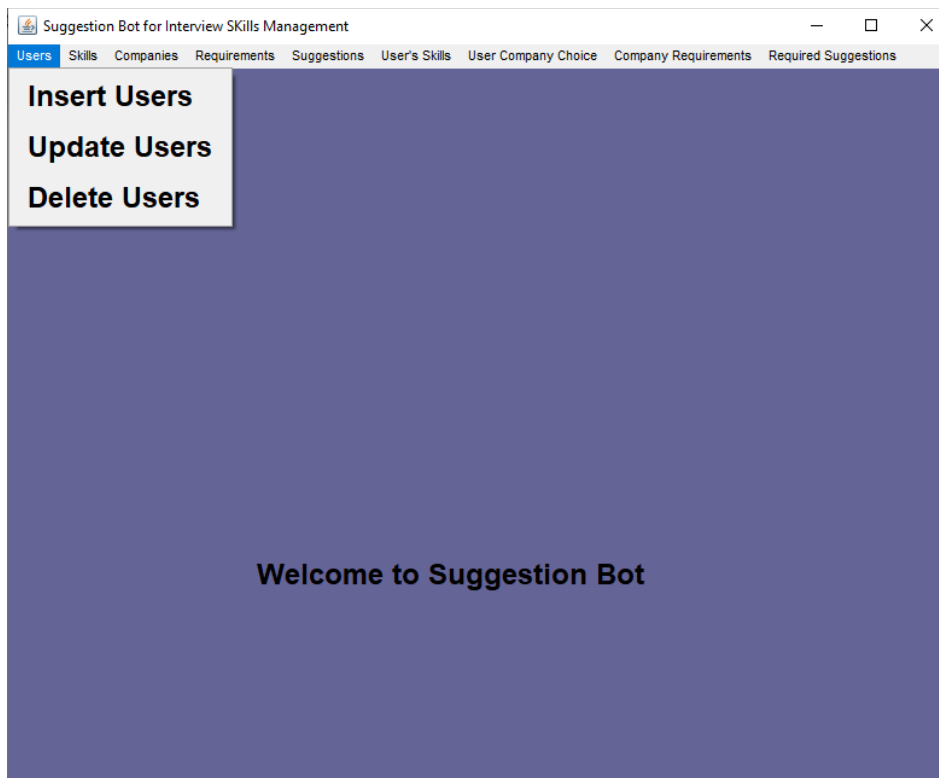
RESULTS

Main GUI:



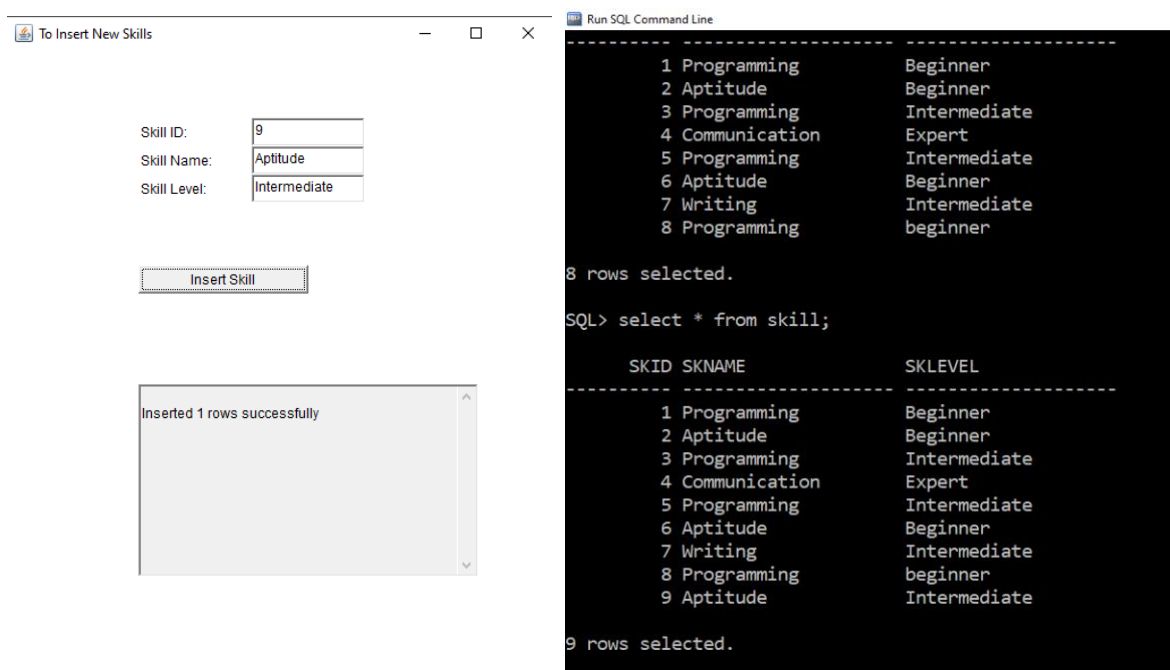
DBMS Mini Project

TOPIC: Suggestion Bot for Interview Skills Management

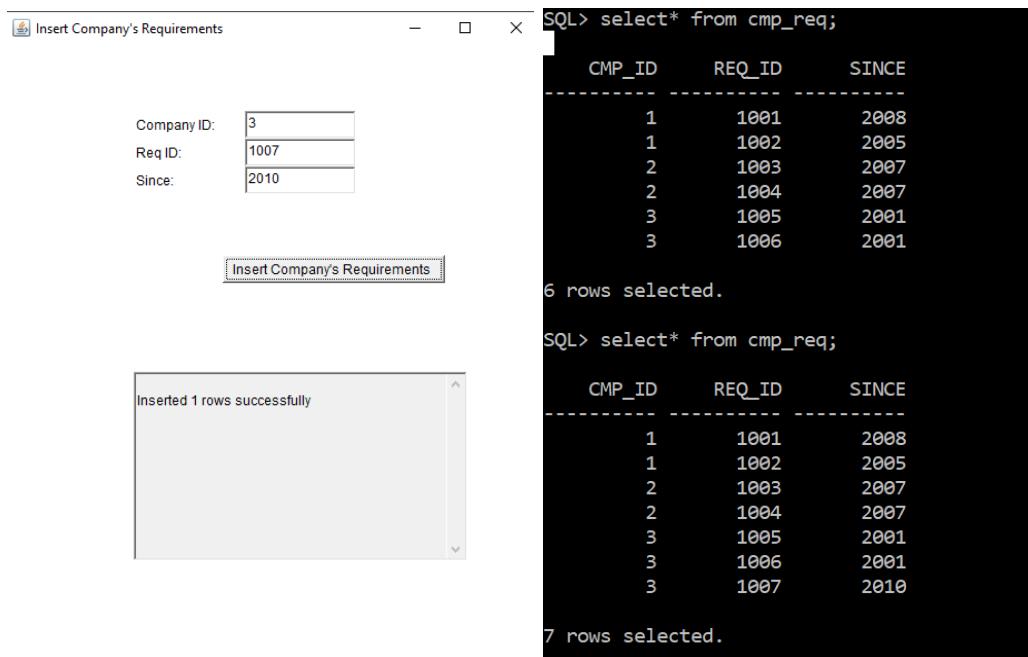


Inserting values into tables:

1) Into Skill table



2) Into Company's Requirements Table



The application window 'Insert Company's Requirements' has the following fields:

Company ID:	3
Req ID:	1007
Since:	2010

Below the fields is a button labeled 'Insert Company's Requirements'. A message box below the button states: 'Inserted 1 rows successfully'.

The terminal window shows the following SQL queries and results:

```
SQL> select* from cmp_req;
```

CMP_ID	REQ_ID	SINCE
1	1001	2008
1	1002	2005
2	1003	2007
2	1004	2007
3	1005	2001
3	1006	2001

6 rows selected.

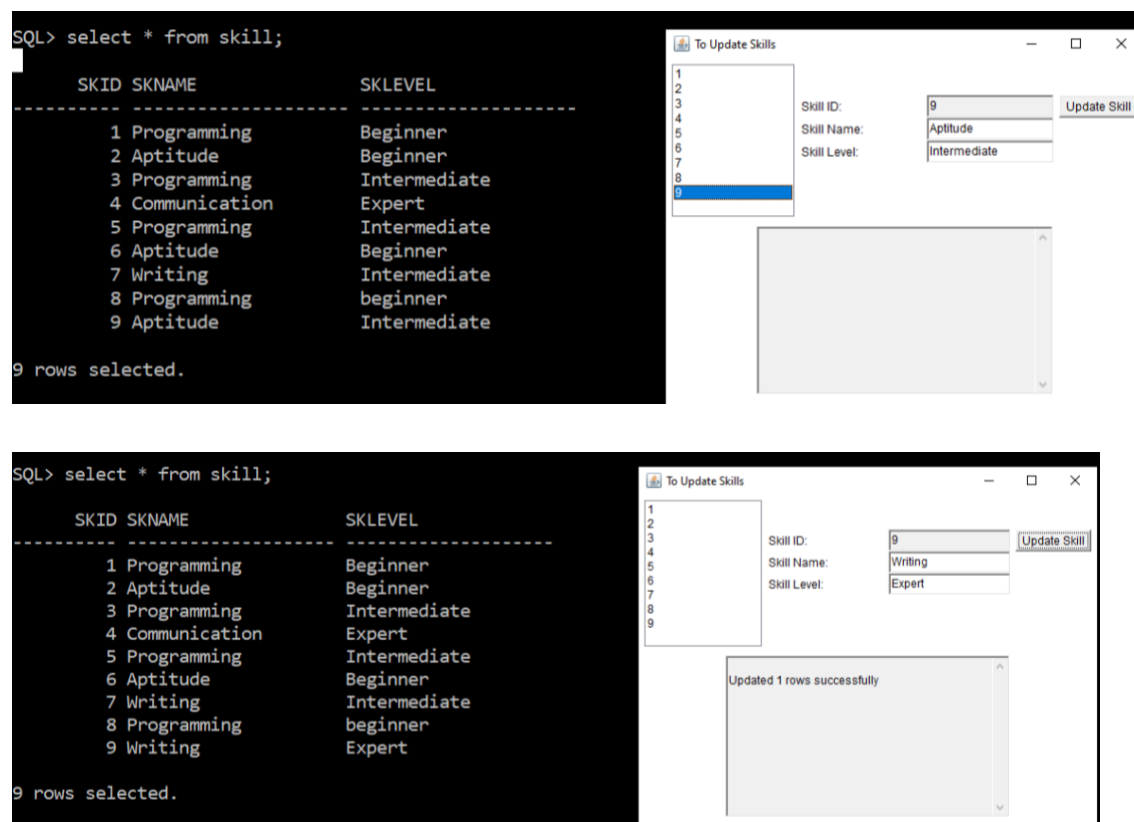
```
SQL> select* from cmp_req;
```

CMP_ID	REQ_ID	SINCE
1	1001	2008
1	1002	2005
2	1003	2007
2	1004	2007
3	1005	2001
3	1006	2001
3	1007	2010

7 rows selected.

Updating values into tables:

1) Updating into Skills Table



The application window 'To Update Skills' has the following fields:

Skill ID:	9
Skill Name:	Aptitude
Skill Level:	Intermediate

Below the fields is a button labeled 'Update Skill'. A message box below the button states: 'Updated 1 rows successfully'.

The terminal window shows the following SQL query and results:

```
SQL> select * from skill;
```

SKID	SKNAME	SKLEVEL
1	Programming	Beginner
2	Aptitude	Beginner
3	Programming	Intermediate
4	Communication	Expert
5	Programming	Intermediate
6	Aptitude	Beginner
7	Writing	Intermediate
8	Programming	beginner
9	Aptitude	Intermediate

9 rows selected.

2) Updating into Company's Requirements table

```

8 Programming      beginner
9 Writing          Expert

9 rows selected.

SQL> select * from cmp_req;

  CMP_ID  REQ_ID  SINCE
-----
      1    1001   2008
      1    1002   2005
      2    1003   2007
      2    1004   2007
      3    1005   2001
      3    1006   2001
      3    1007   2010

7 rows selected.

SQL>

      3    1006   2001
      3    1007   2010

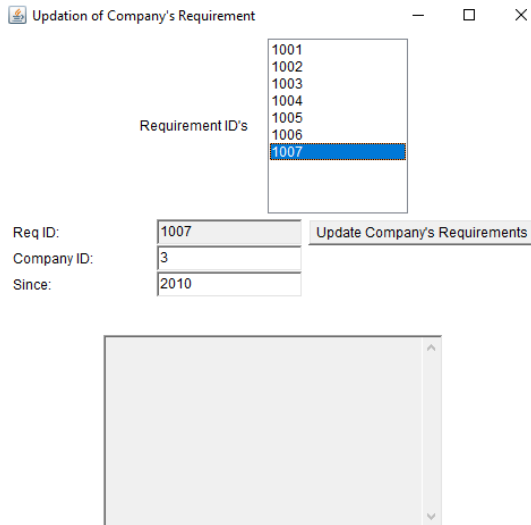
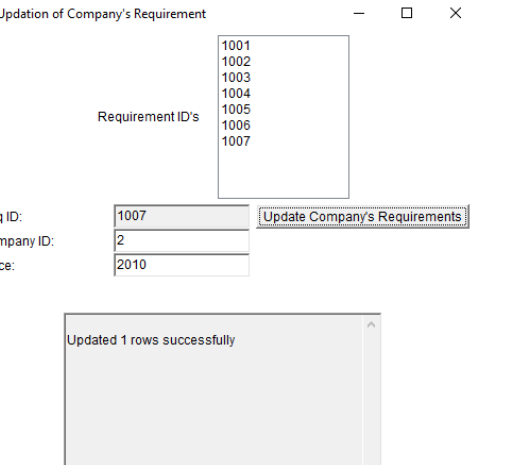
7 rows selected.

SQL> select * from cmp_req;

  CMP_ID  REQ_ID  SINCE
-----
      1    1001   2008
      1    1002   2005
      2    1003   2007
      2    1004   2007
      3    1005   2001
      3    1006   2001
      2    1007   2010

7 rows selected.

```

Deleting values from tables:

1) Deleting from skills table:

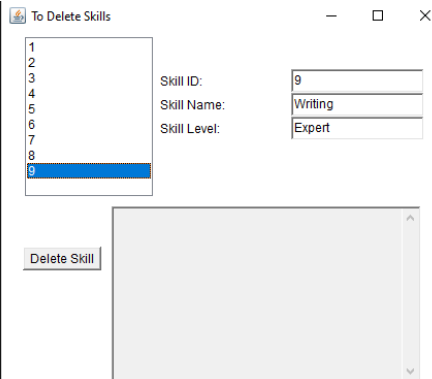
```

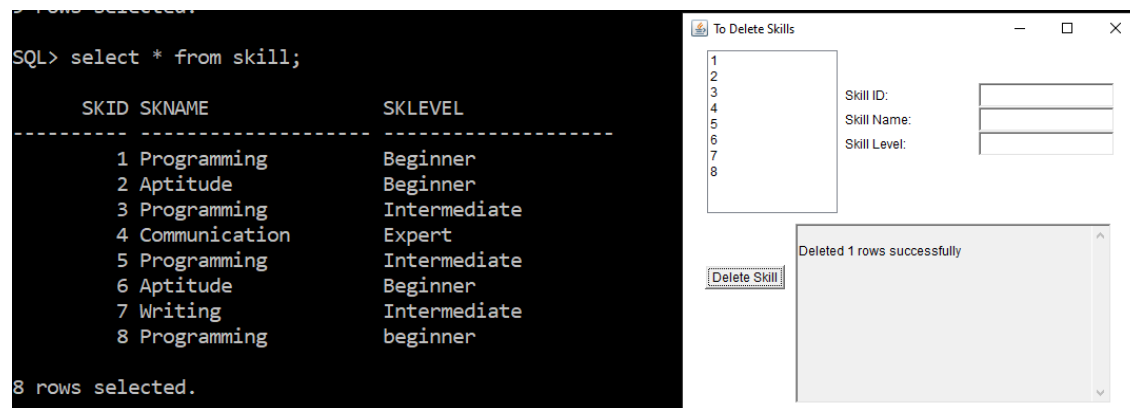
SQL> select * from skill;

  SKID  SKNAME          SKLEVEL
-----
      1 Programming      Beginner
      2 Aptitude        Beginner
      3 Programming      Intermediate
      4 Communication    Expert
      5 Programming      Intermediate
      6 Aptitude          Beginner
      7 Writing           Intermediate
      8 Programming      beginner
      9 Writing           Expert

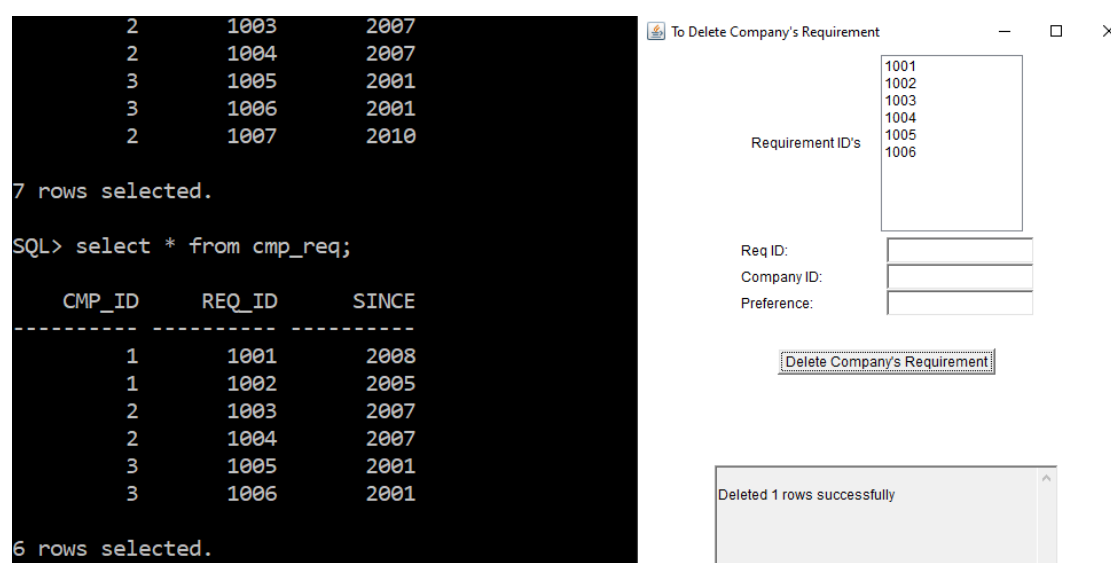
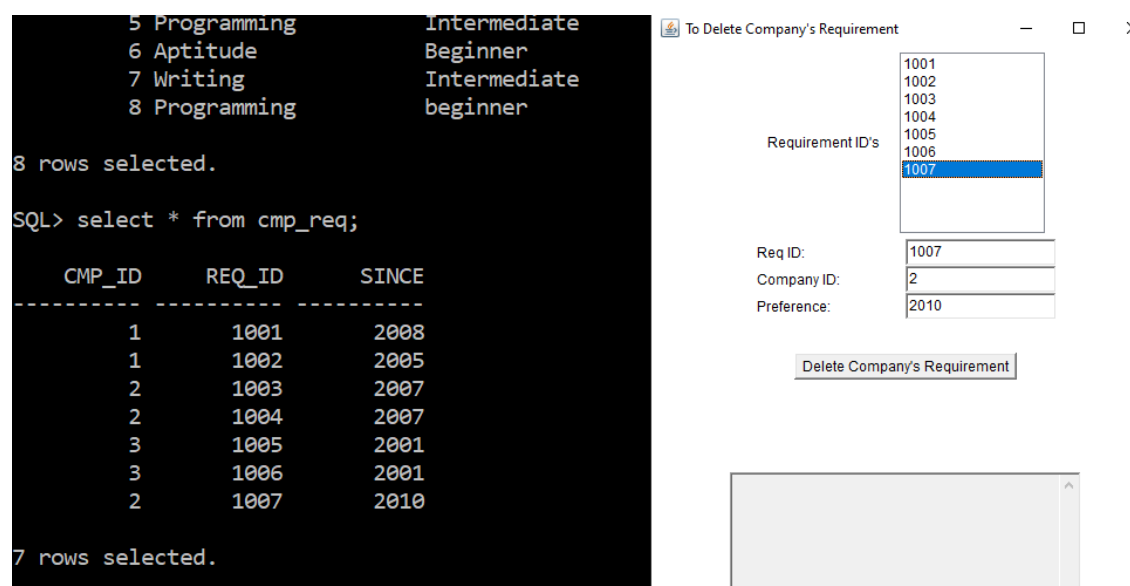
9 rows selected.

```





2) Deleting from Company's Requirements table



DISCUSSION AND FUTURE WORK

This project can be further developed into a software or an app depending on the clients need. In the cut-throat environment of today's day and age, a suggestion bot that gives you tips on area and /or skills that an individual may need to improve always comes in handy.

We can also further develop the project by introducing other modules such as platforms that teach the required skills, providing video lectures of certain concepts and also a module that provides assistance in campus recruitment training and other such areas.

REFERENCES

- Abraham Silberschatz, Henry F. Korth and S. Sudarshan, *Database System Concepts*, McGraw-Hill Education (Asia), Fifth Edition, 2006.
- Raghu Ramakrishnan *Database Management System*, Third Edition.