**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:**

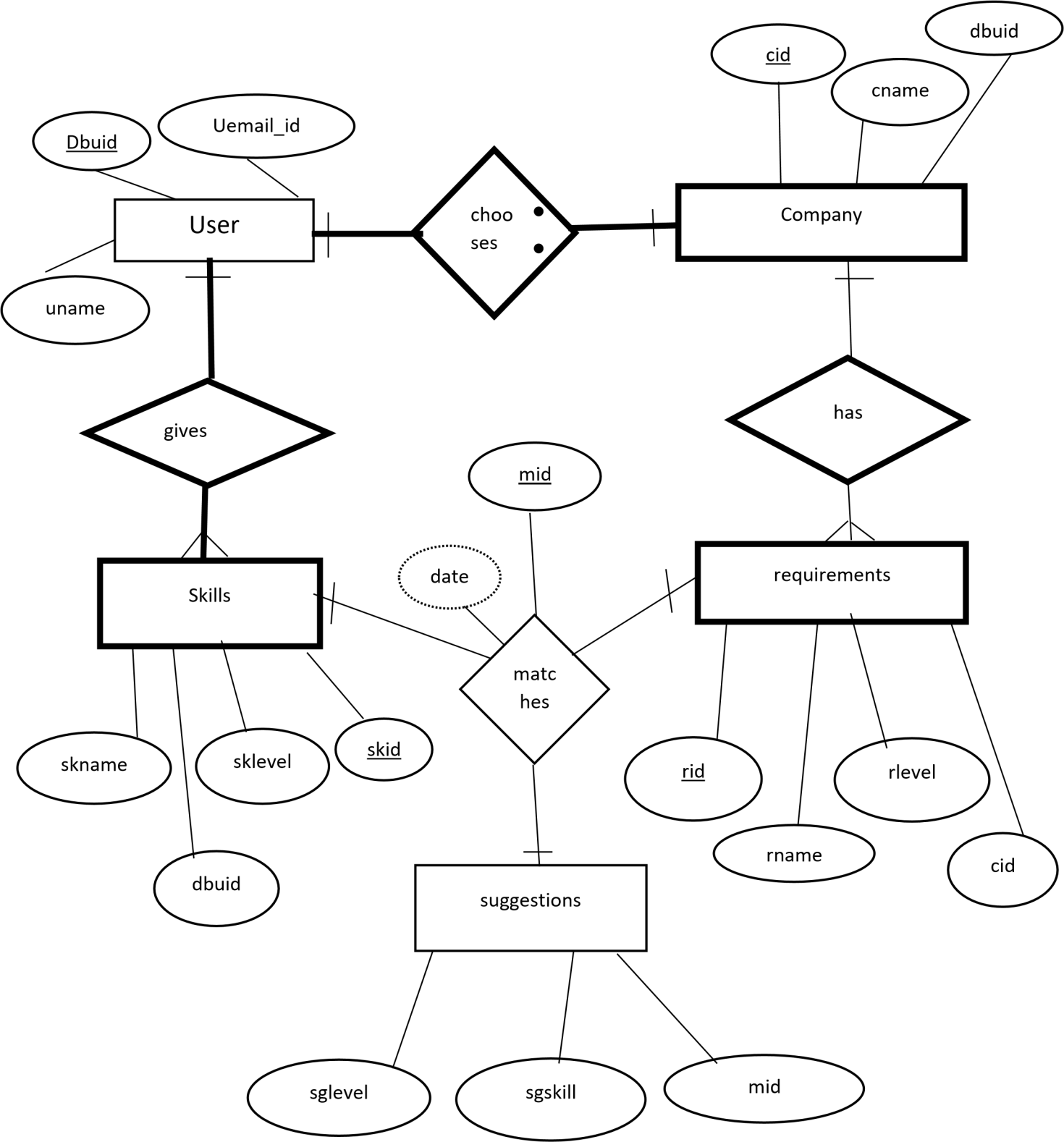
Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

**4. Design**

**Requirement Analysis:**

|  |  |
| --- | --- |
| **Table** | **Attributes** |
| 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));

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**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** displaySQLErrors(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:

A screenshot of a computer

Description automatically generated

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.getSelectedItem());

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:

A screenshot of a computer

Description automatically generated

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.getSelectedItem());

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** 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)

{

DeleteUser dels = **new** DeleteUser();

dels.addWindowListener(**new** WindowAdapter(){

**public** **void** windowClosing(WindowEvent e)

{

System.*exit*(0);

}

});

dels.buildGUI();

}

}

OUTPUT:

A screenshot of a computer

Description automatically generated

**GitHub Link**

<https://github.com/jgayatri068/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.

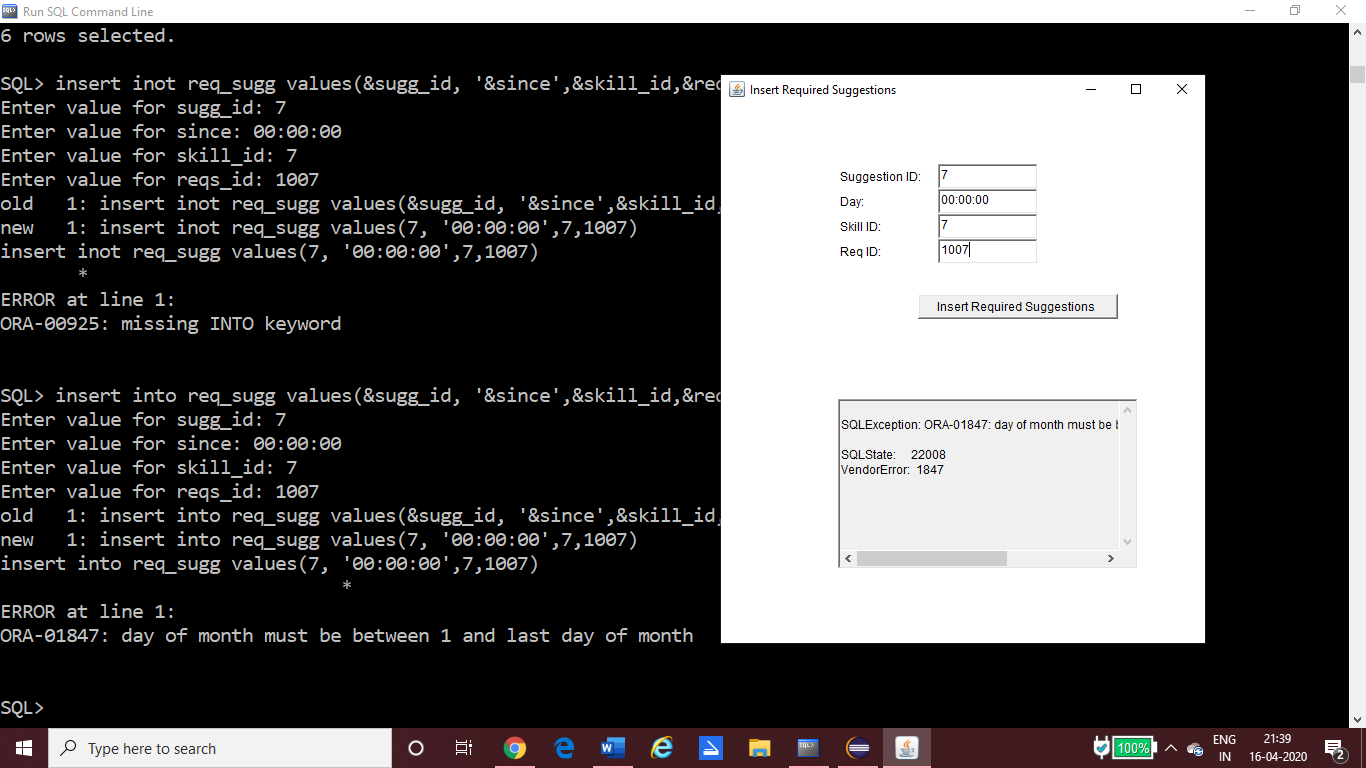
A screenshot of a social media post

Description automatically generated

A screenshot of a computer screen

Description automatically generated

b) Entering date in incorrect format



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.

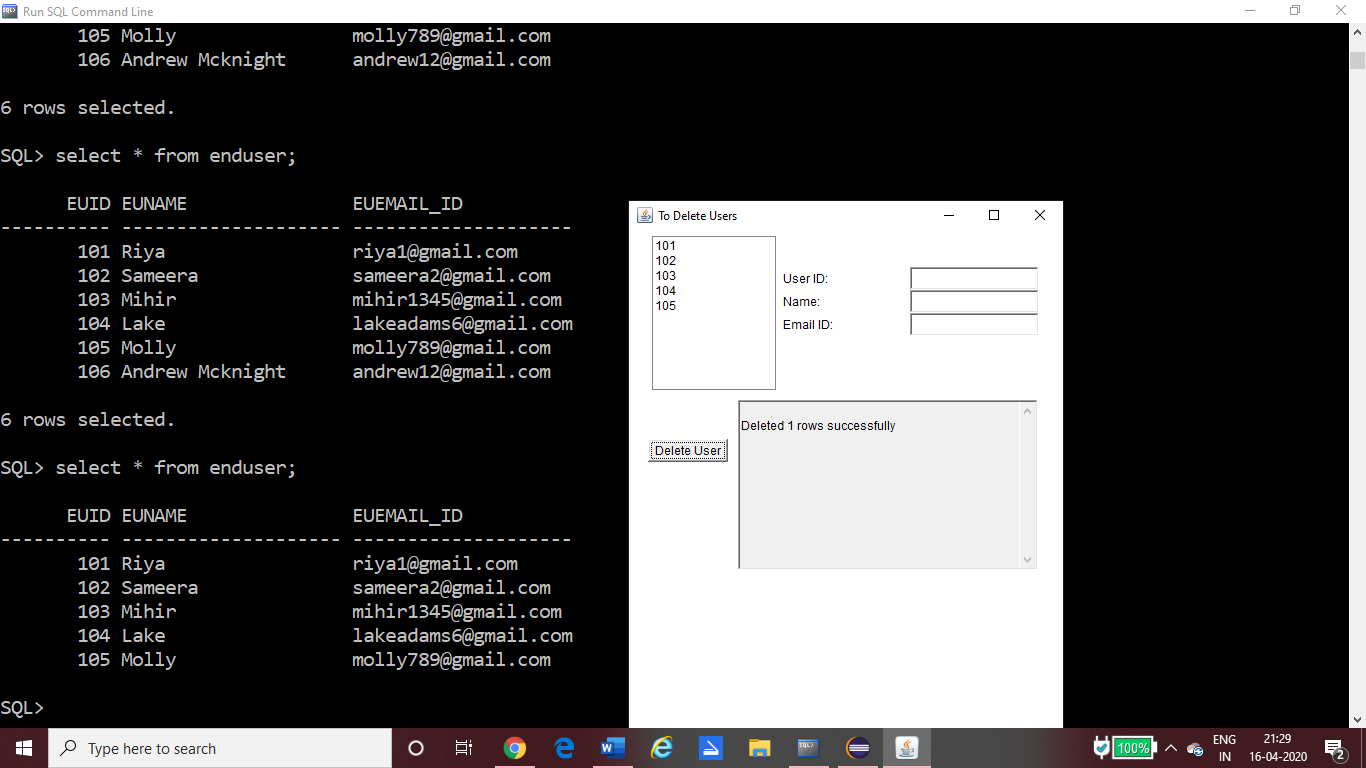
A screenshot of a computer screen

Description automatically generated

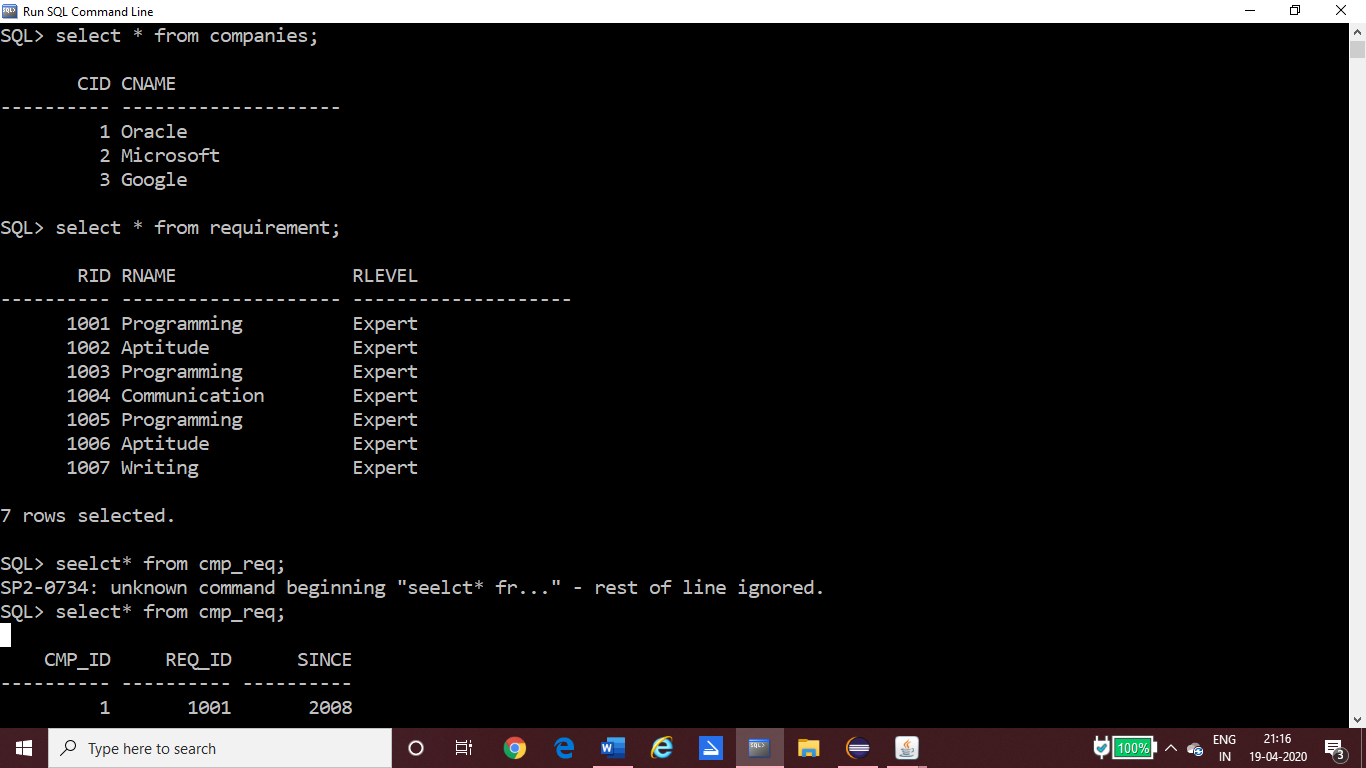
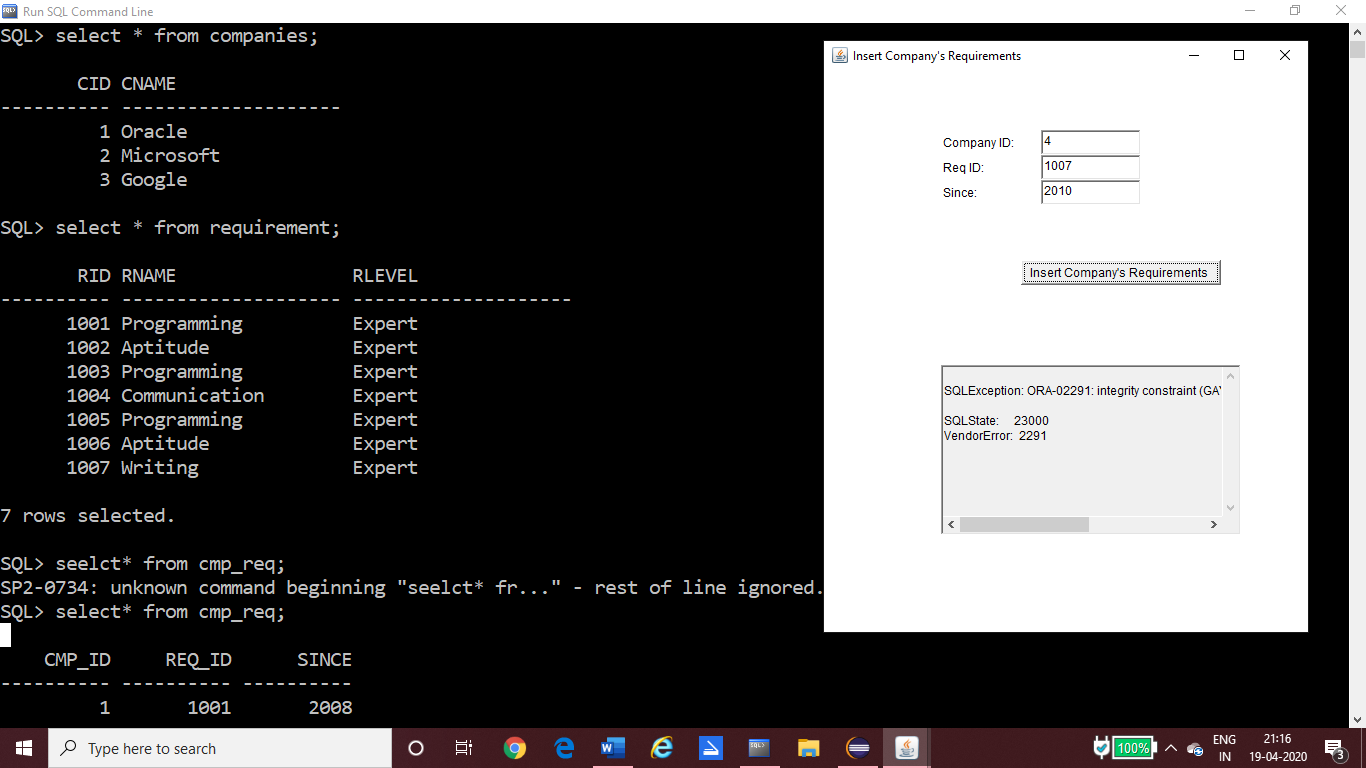
The user id text field in the GUI coloured in grey indicates that it is un-editable.

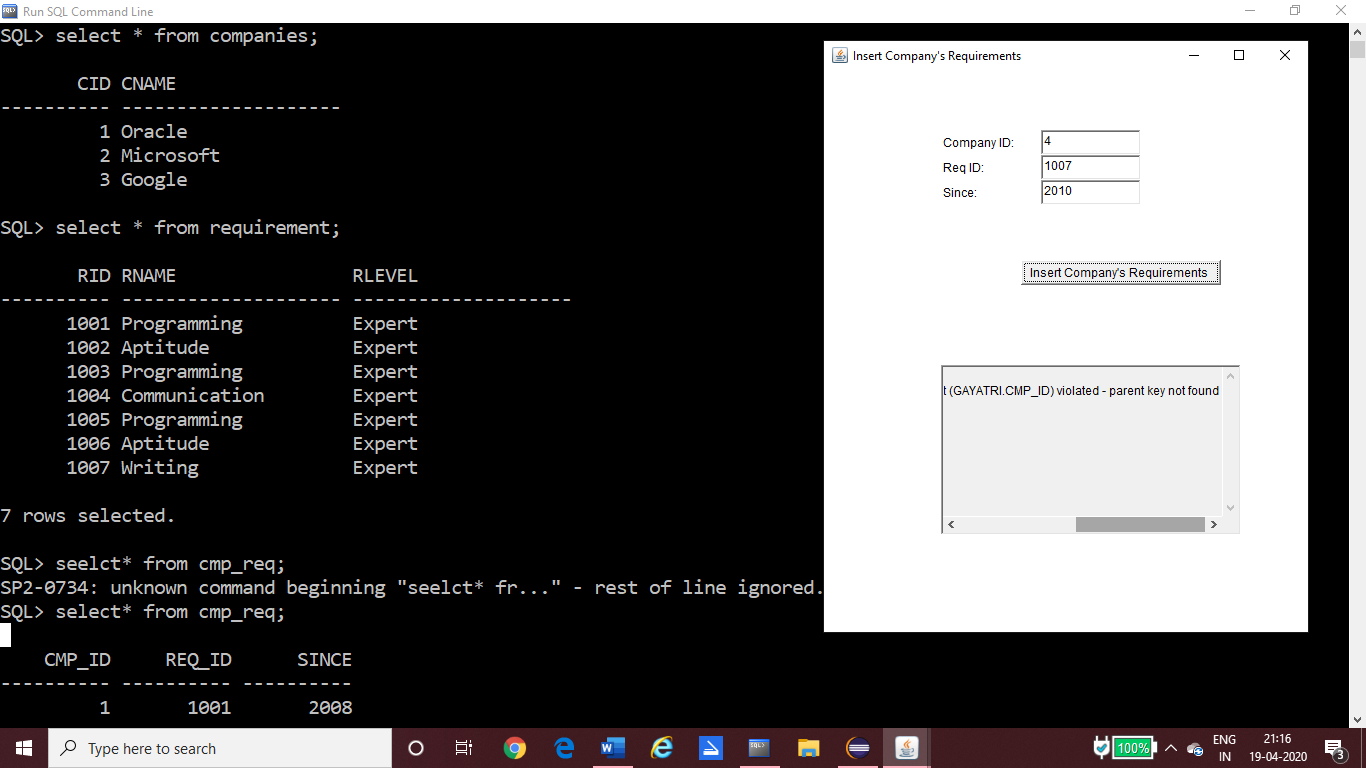
3. Testing for a deleting a non-existent value from the database

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.



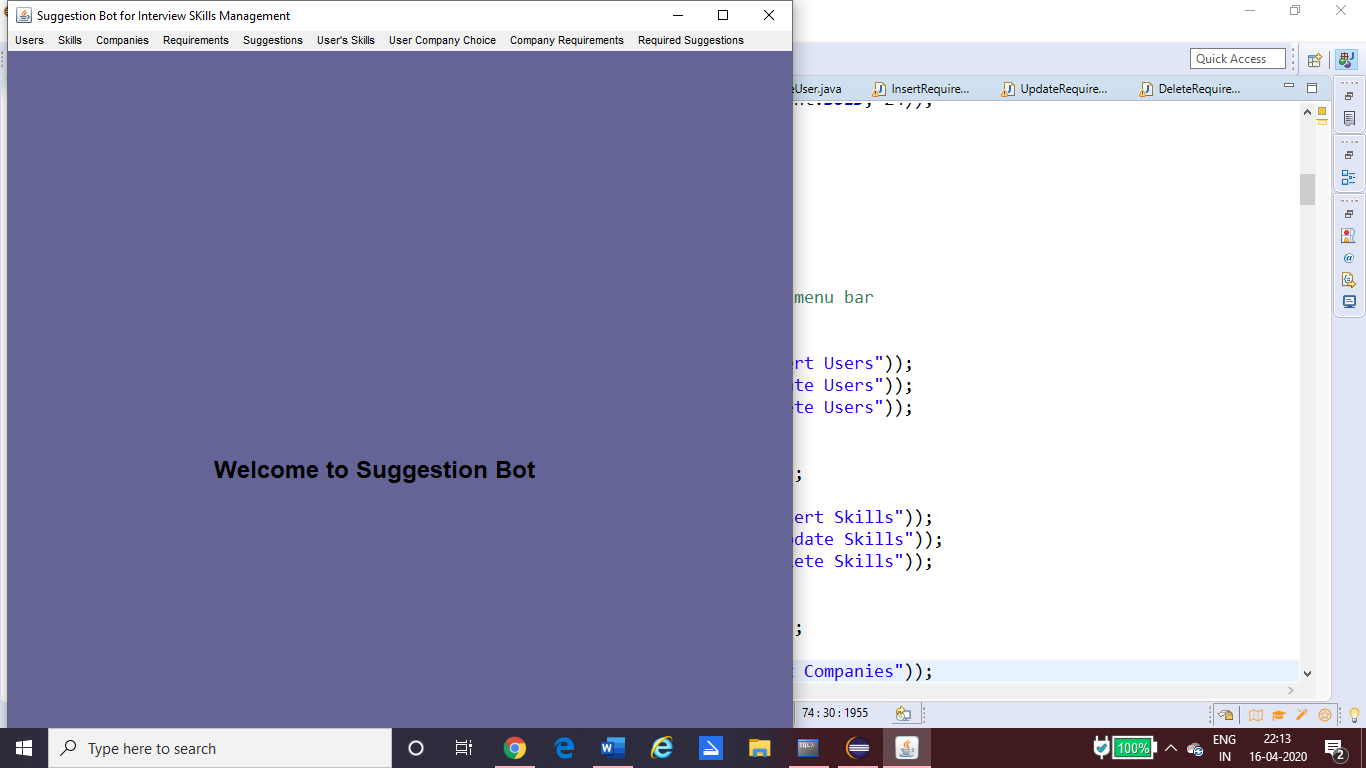
4. Testing for inserting values into child table those of which are not present in the parent table.

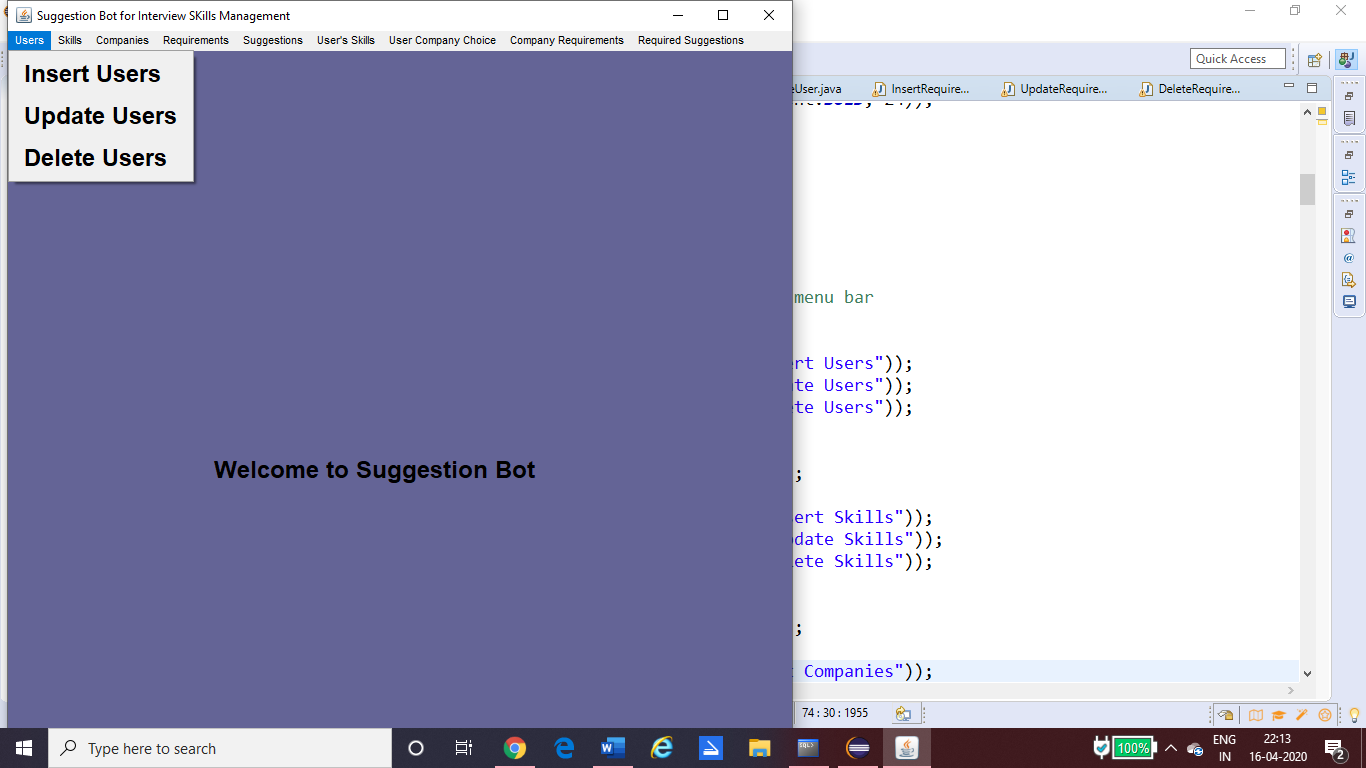
 



**RESULTS**

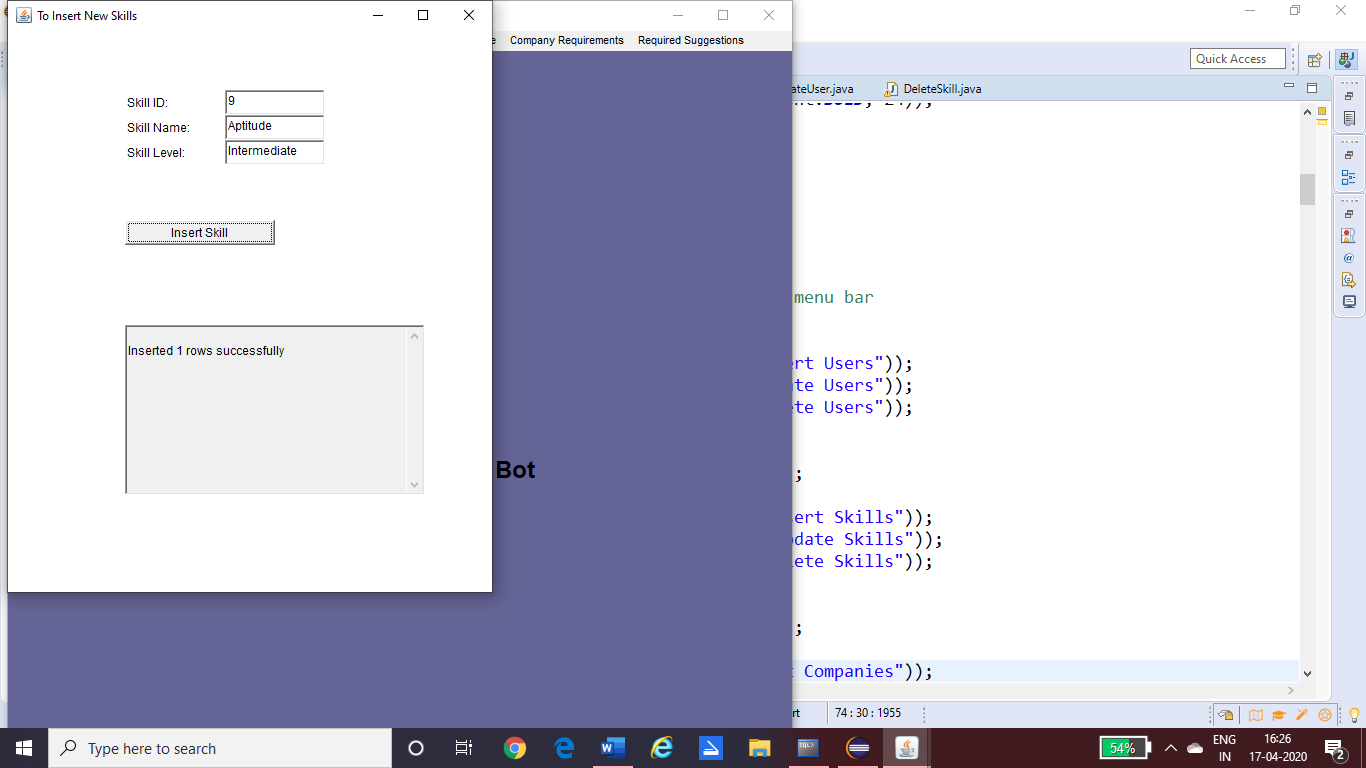
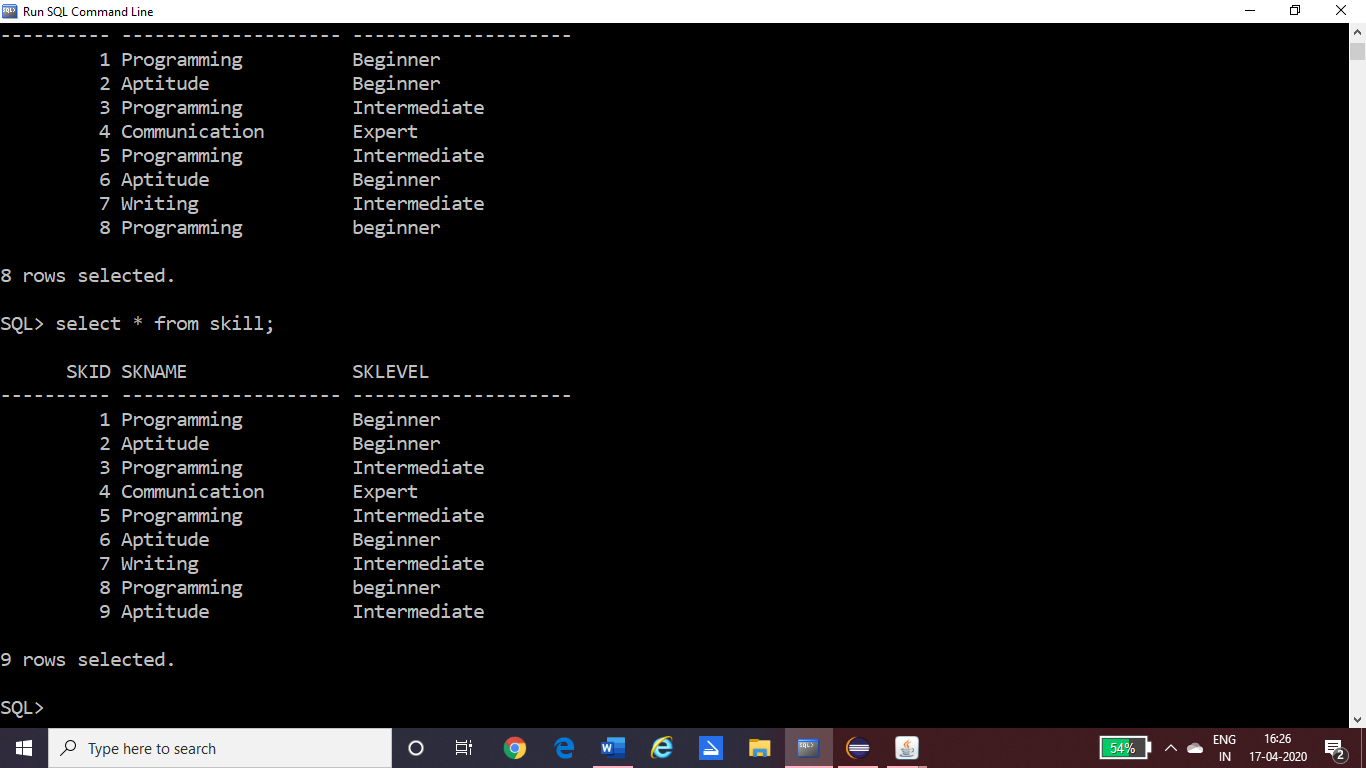
Main GUI:



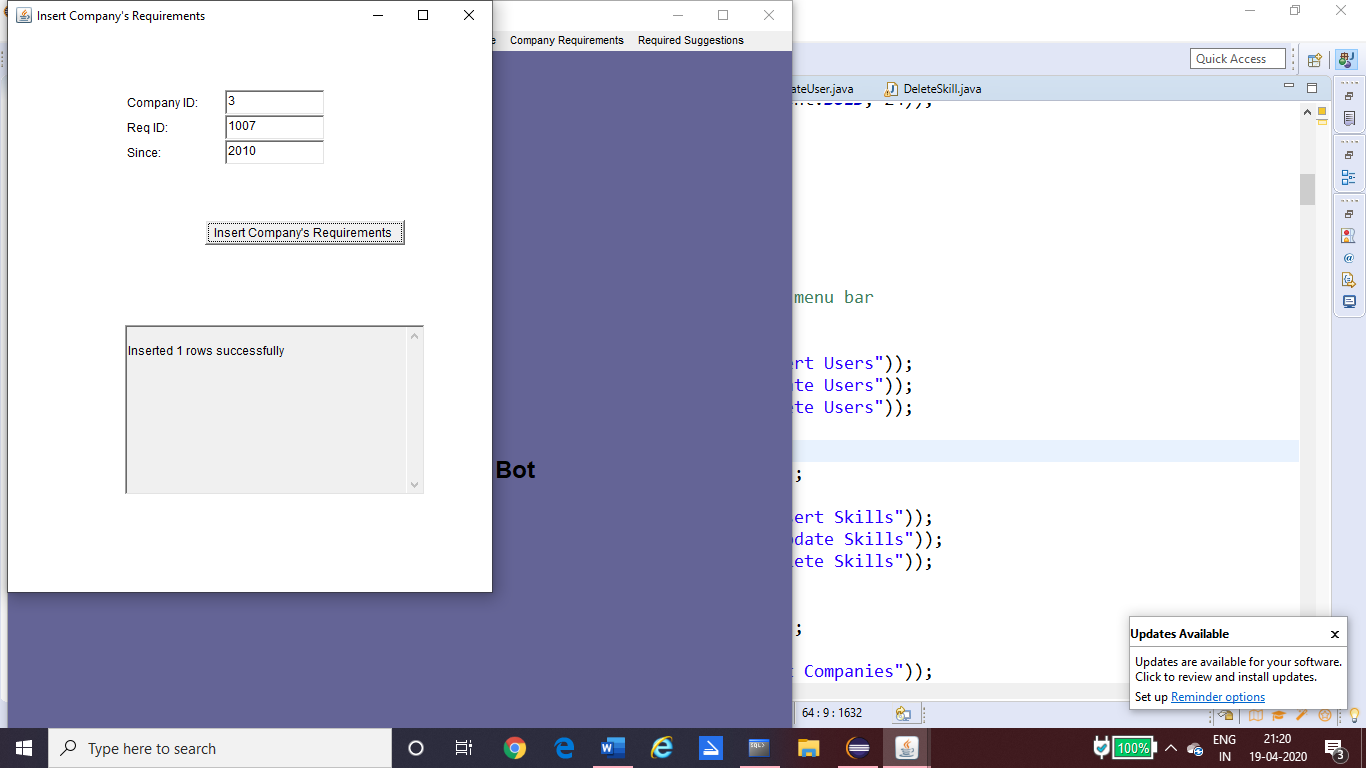
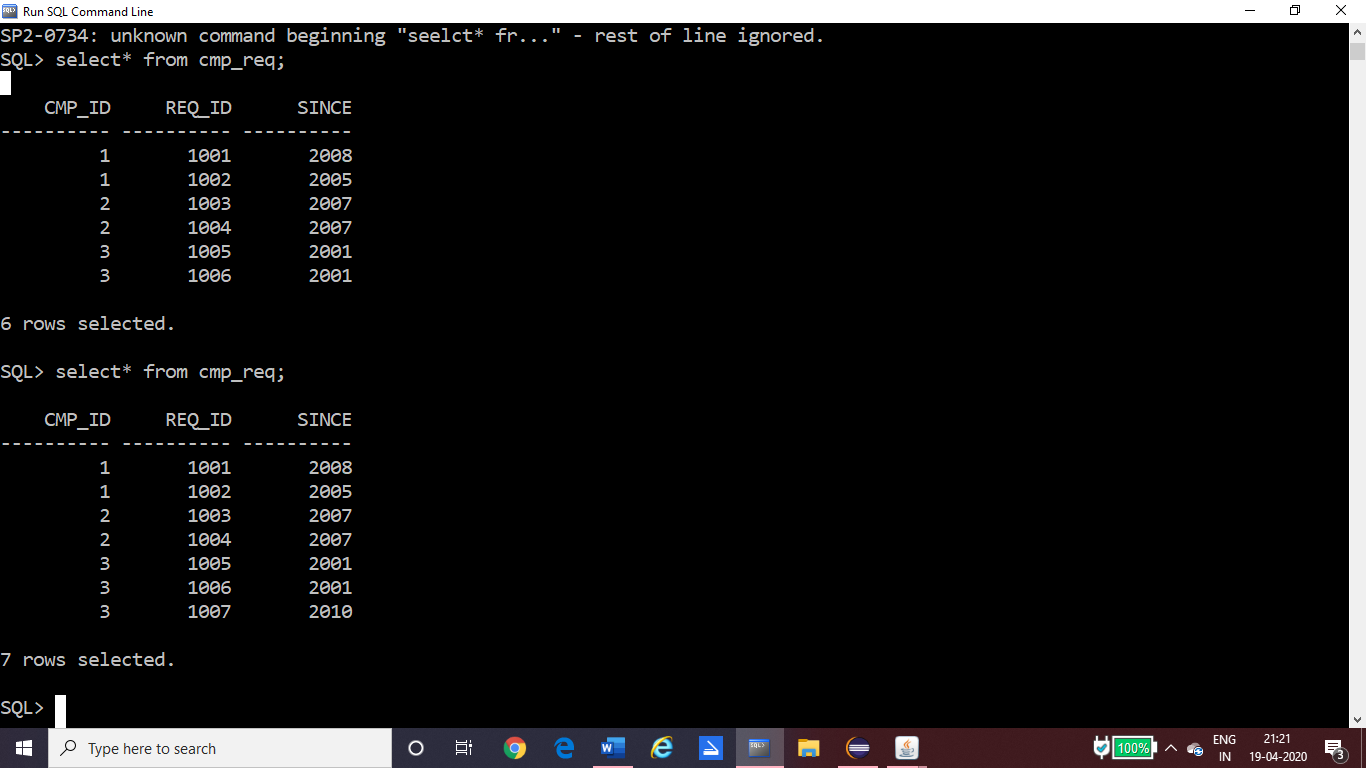


Inserting values into tables:

1) Into Skill table

 ****

2) Into Company’s Requirements Table

Updating values into tables:

1) Updating into Skills Table

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

2) Updating into Company’s Requirements table

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Deleting values from tables:

1) Deleting from skills table:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

2) Deleting from Company’s Requirements table

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**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.