***VCE NETWORK CONNECTION***

***MANAGEMENT SYSTEM***

***DONE BY:-***

***A.RUCHITA***

***IT-B***

***1602-18-737-064***

**JAVA AWT BASED- NETWORK CONNECTION MANAGEMENT SYSTEM - SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

**A.RUCHITA <1602-18-737-064>**

Under the guidance of B.Leelavathy

****

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2019**

**ABSTRACT**:-

The Network Connection Management System is a Web-based system that provides a single-stop for registration and maintenance of the network connection database of the campus network. The database contains both data (wired and wireless) and voice network connections in the Main Campus and the Student Residences. Network management system (NMS) is important both in ensuring the correct operation of network devices and in maintaining the services that run on them. This project has total of 11 tables .It describes how the network is being connected in our college across the various blocks. When you enter the data it is stored in the data base and is displayed as of when it is needed.

**AIM:**

To create a **Java GUI based NETWORK CONNECTION MANAGEMENT SYSTEM** which takes the values like: computer ID, computer name , manufacturer , type , count, routers speed , username , website , block name , hod , server ipaddress , operating system name , version etc from the user. These values are to be updated in the database using **JDBC connectivity.**

**INTRODUCTION**

**Requirements:**

**List of tables:**

* Internet
* Computers
* Routers
* Block
* Server
* operating\_system
* contains
* has
* are\_having
* provides\_network\_to
* connected\_to

**List of attributes with their domain types:-**

*Internet*:

Mac address : mac\_address-varchar2(20)

HTML: html -varchar2(30)

Service provider: serv\_provider- varchar2(20));

*Computers:*

Id of the computer: cid- varchar2(20)

Type of the computer: type - varchar2(20)

Count of computers: count -number

Manufacturer name: manufacturer - varchar2(20)

Type of model: model-varchar2(20)

*Routers*:

Website name :website -varchar2(50)

Speed of the router:speed- varchar2(10)

Model :model- varchar2 (20)

Username: username -varchar2(20)

Company :company- varchar2 (20)

*Block*:

Name of the block:bname-varchar2(20)

Name of the HOD:hod -varchar2(20)

Branch of the block:branch varchar2(20)

*Server*:

Ipaddress : ipaddress varchar2(20)

*Operating\_system*:

Name of the operating system : osname- varchar2(20)

Version: version -varchar2(20)

Vendor: vendor-varchar2(20)

**ARCHITECTURE AND TECHNOLOGY :**

**Software used:**

Java Eclipse, Oracle 11g Database, Java SE version 7, 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**

**ER-Diagram**

**Internet\_server**

**Server**

**Internet**

**Server\_block**

**Block\_computers**

**Block\_ routers**

**Routers**

Computer

**Operating system**

**Computers\_operatingsystem**

**Block**

**DDL COMMANDS**

create table **internet**(

mac\_address varchar2(20) primary key,html varchar2(30),serv\_provider varchar2(20));

create table **computers**(

cid varchar2(20) primary key, type varchar2(20), count number, manufacturer varchar2(20),model varchar2(20)) ;

create table **routers** (

website varchar2(50) primary key ,speed varchar2(10),model varchar2 (20), username varchar2(20),company varchar2 (20)) ;

create table **Block**(

bname varchar2(20)primary key ,hod varchar2(20),branch varchar2(20));

create table **server** ( ipaddress varchar2(20)primary key);

create table **operating\_system**(

osname varchar2(20) , version varchar2(20), vendor varchar2(20), primary key(osname));

create table **Block\_computer**(

cid varchar2(20), bname varchar2(20),foreign key(cid) references computers(cid), foreign key (bname) references block( bname),primary key(cid,bname)) ;

create table **Computers\_operatingsystem**(

cid varchar2(20), osname varchar2(20),foreign key (cid) references computers, foreign key (osname) reference operating\_system) ;

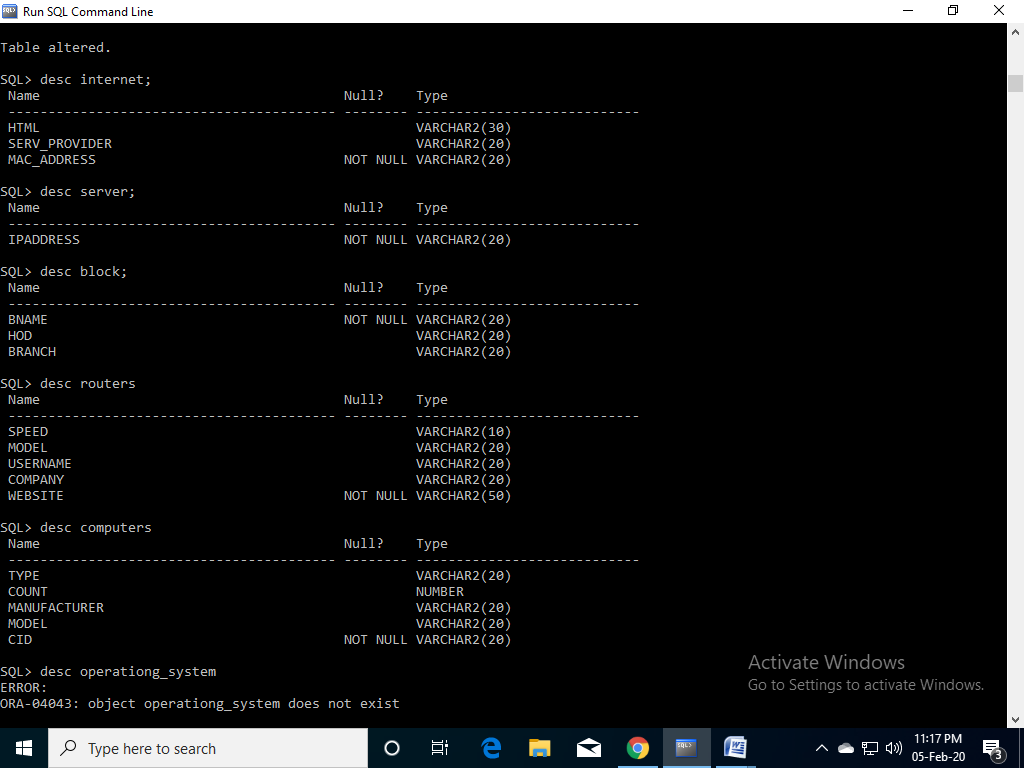
create table **Block\_routers**(

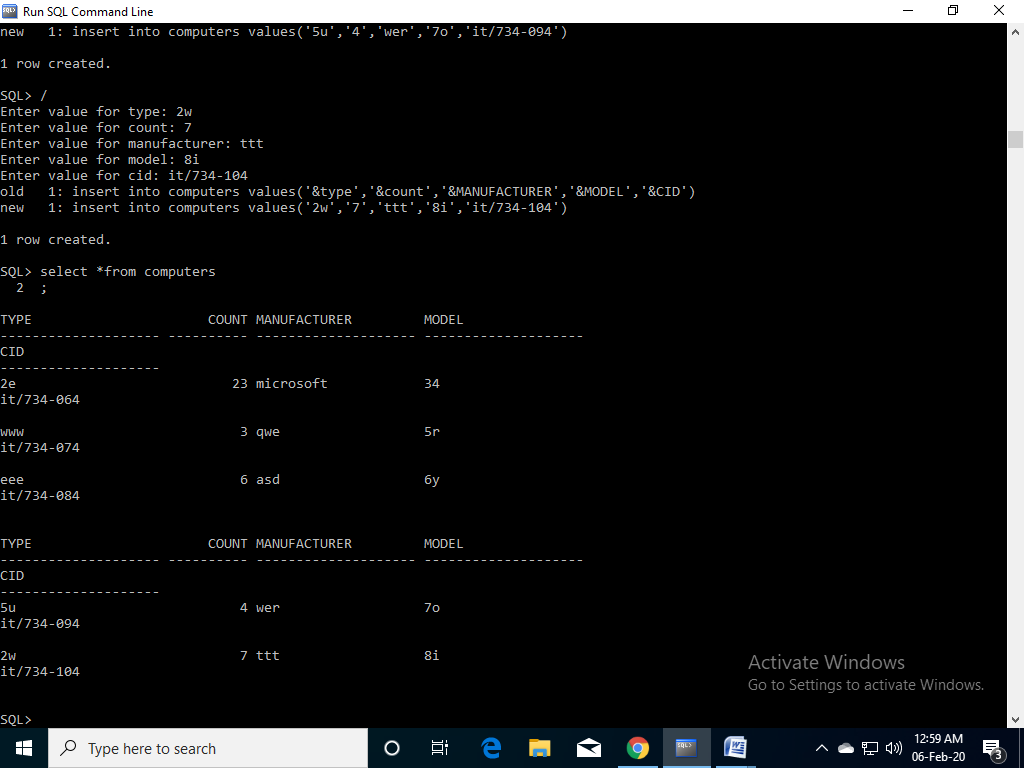
website varchar2(50), bname varchar2(20), foreign key(website) references routers, foreign key(bname) references block);

create table **Server\_block**(

bname varchar2(20),ipaddress varchar(20),foreign key(ipaddress) references server, foreign key(bname) references block);

create table **Internet\_server**( mac varchar2(20),ipadd varchar2(20),foreign key(mac) references internet, foreign key(ipadd) references server);

****

****

**Java-SQL Connectivity using JDBC:**

**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:

**private** **void** connToDb(){

**try** {

Class.*forName*("oracle.jdbc.driver.OracleDriver");

connection = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1522:xe","rachana","vasavi");

statement = connection.createStatement();

} **catch** (SQLException connectException) {

System.*out*.println(connectException.getMessage());

System.*out*.println(connectException.getSQLState());

System.*out*.println(connectException.getErrorCode());

System.*exit*(1);

}

**catch** (Exception e)

{

System.*err*.println("Unable to find and load driver");

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.

**Table Created in SQL for above mentioned purpose is as:**

create table **computers**

(

cid varchar2(20) primary key ,

type varchar2(20) ,

count number ,

manufacturer varchar2(20) ,

model varchar2(20)

) ;

**Program to insert computers:**

package java\_ass2;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class InsertComputers extends Frame

{

Button insertComputersButton;

TextField cidText, typeText, countText, manufacturerText;

TextArea errorText;

Connection connection;

Statement statement;

public InsertComputers()

{

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","ruchi","04032001");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

public void buildGUI()

{

//Handle Insert Account Button

insertComputersButton = new Button("Insert Computers");

insertComputersButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO Computers VALUES('" + cidText.getText() + "', " + "'" + typeText.getText() + "'," + countText.getText() + ",'" + manufacturerText.getText() + "')";

int i = statement.executeUpdate(query);

errorText.append("\nInserted " + i + " rows successfully");

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

cidText = new TextField(15);

typeText = new TextField(15);

countText = new TextField(15);

manufacturerText = 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("Computer ID:"));

first.add(cidText);

first.add(new Label("Type:"));

first.add(typeText);

first.add(new Label("Count:"));

first.add(countText);

first.add(new Label("Manufacturer:"));

first.add(manufacturerText);

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

Panel second = new Panel(new GridLayout(4, 1));

second.add(insertComputersButton);

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 Computers 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)

{

InsertComputers cc = new InsertComputers();

cc.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

cc.buildGUI();

}

}

**Program to update computers:**

package java\_ass2;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class ViewComputers extends Frame

{

Button updateComputersButton;

List ComputersList;

TextField cidText, typeText, countText, manufacturerText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public ViewComputers()

{

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","ruchi","04032001");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

private void loadComputers()

{

try

{

rs = statement.executeQuery("SELECT \* FROM Computers");

while (rs.next())

{

ComputersList.add(rs.getString("CID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

ComputersList = new List(6);

loadComputers();

add(ComputersList);

//When a list item is selected populate the text fields

ComputersList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM Computers");

while (rs.next())

{

if (rs.getString("CID").equals(ComputersList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

cidText.setText(rs.getString("CID"));

typeText.setText(rs.getString("TYPE"));

countText.setText(rs.getString("COUNT"));

manufacturerText.setText(rs.getString("MANUFACTURER"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Computers Button

updateComputersButton = new Button("Update Computers");

updateComputersButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE Computers "

+ "SET COUNT=" + countText.getText()

+ " WHERE cid = '" + ComputersList.getSelectedItem() + "'");

errorText.append("\nUpdated " + i + " rows successfully");

ComputersList.removeAll();

loadComputers();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

cidText = new TextField(15);

cidText.setEditable(false);

typeText = new TextField(15);

typeText.setEditable(false);

countText = new TextField(15);

manufacturerText = new TextField(15);

manufacturerText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

first.setLayout(new GridLayout(4, 2));

first.add(new Label("Computer ID:"));

first.add(cidText);

first.add(new Label("Type:"));

first.add(typeText);

first.add(new Label("Count:"));

first.add(countText);

first.add(new Label("Manufacturer:"));

first.add(manufacturerText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(updateComputersButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setTitle("Update Computers");

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)

{

ViewComputers upb = new ViewComputers();

upb.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

upb.buildGUI();

}

}

**Program to delete computers:**

package java\_ass2;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

public class DeleteComputers extends Frame

{

Button DeleteComputersButton;

List ComputersIDList;

TextField cidText, typeText, countText, manufacturerText;

TextArea errorText;

Connection connection;

Statement statement;

ResultSet rs;

public DeleteComputers()

{

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","ruchi","04032001");

statement = connection.createStatement();

}

catch (SQLException connectException)

{

System.out.println(connectException.getMessage());

System.out.println(connectException.getSQLState());

System.out.println(connectException.getErrorCode());

System.exit(1);

}

}

private void loadComputers()

{

try

{

rs = statement.executeQuery("SELECT \* FROM computers");

while (rs.next())

{

ComputersIDList.add(rs.getString("CID"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void buildGUI()

{

ComputersIDList = new List(10);

loadComputers();

add(ComputersIDList);

//When a list item is selected populate the text fields

ComputersIDList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

rs = statement.executeQuery("SELECT \* FROM computers");

while (rs.next())

{

if (rs.getString("CID").equals(ComputersIDList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

cidText.setText(rs.getString("CID"));

typeText.setText(rs.getString("TYPE"));

countText.setText(rs.getString("COUNT"));

manufacturerText.setText(rs.getString("MANUFACTURER"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Delete Computers Button

DeleteComputersButton = new Button("Delete Computers");

DeleteComputersButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM computers WHERE CID = '" + ComputersIDList.getSelectedItem()+"'");

errorText.append("\nDeleted " + i + " rows successfully");

cidText.setText(null);

typeText.setText(null);

countText.setText(null);

manufacturerText.setText(null);

ComputersIDList.removeAll();

loadComputers();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

cidText = new TextField(15);

typeText = new TextField(15);

countText = new TextField(15);

manufacturerText = 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("Computers ID:"));

first.add(cidText);

first.add(new Label("Type:"));

first.add(typeText);

first.add(new Label("Count:"));

first.add(countText);

first.add(new Label("Manufacturer:"));

first.add(manufacturerText);

Panel second = new Panel(new GridLayout(4, 1));

second.add(DeleteComputersButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setTitle("Remove computers");

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)

{

DeleteComputers dels = new DeleteComputers();

dels.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

});

dels.buildGUI();

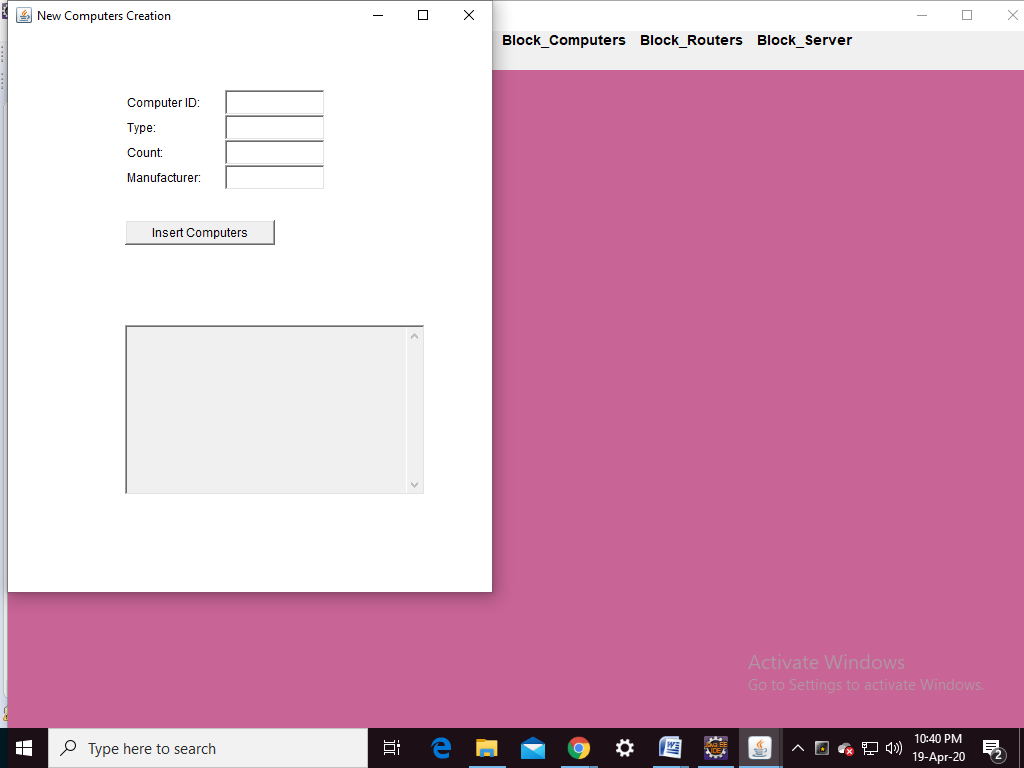
}

}

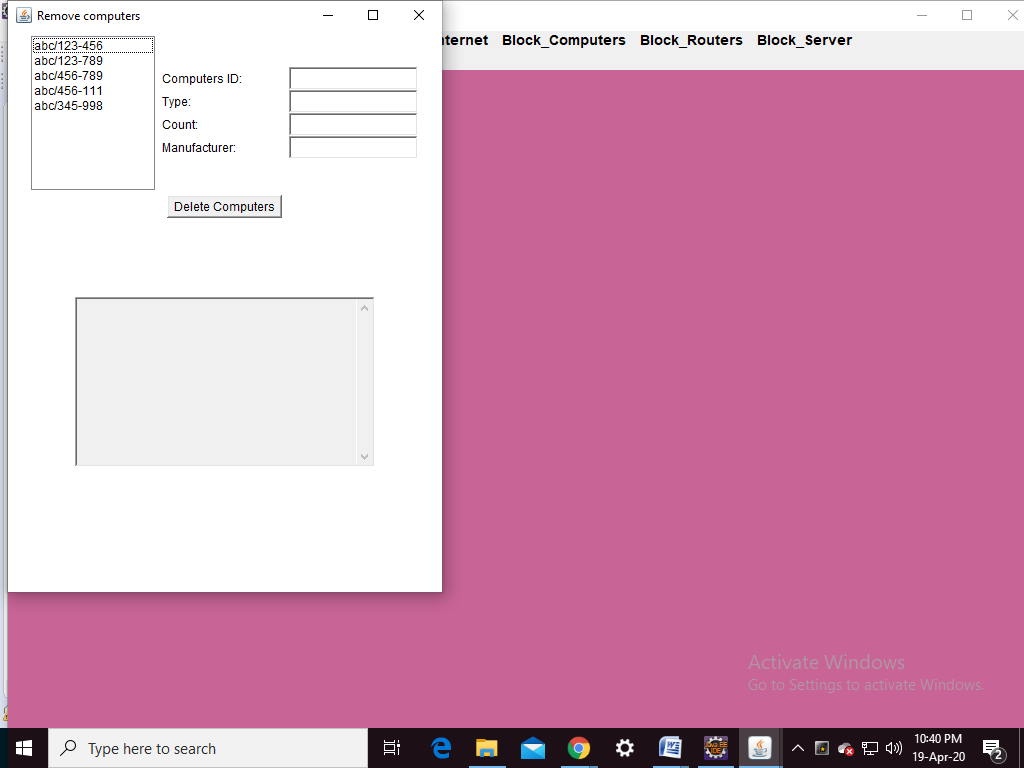
**OUTPUT SCREENSHOTS:**

**Java GUI Screenshot:**

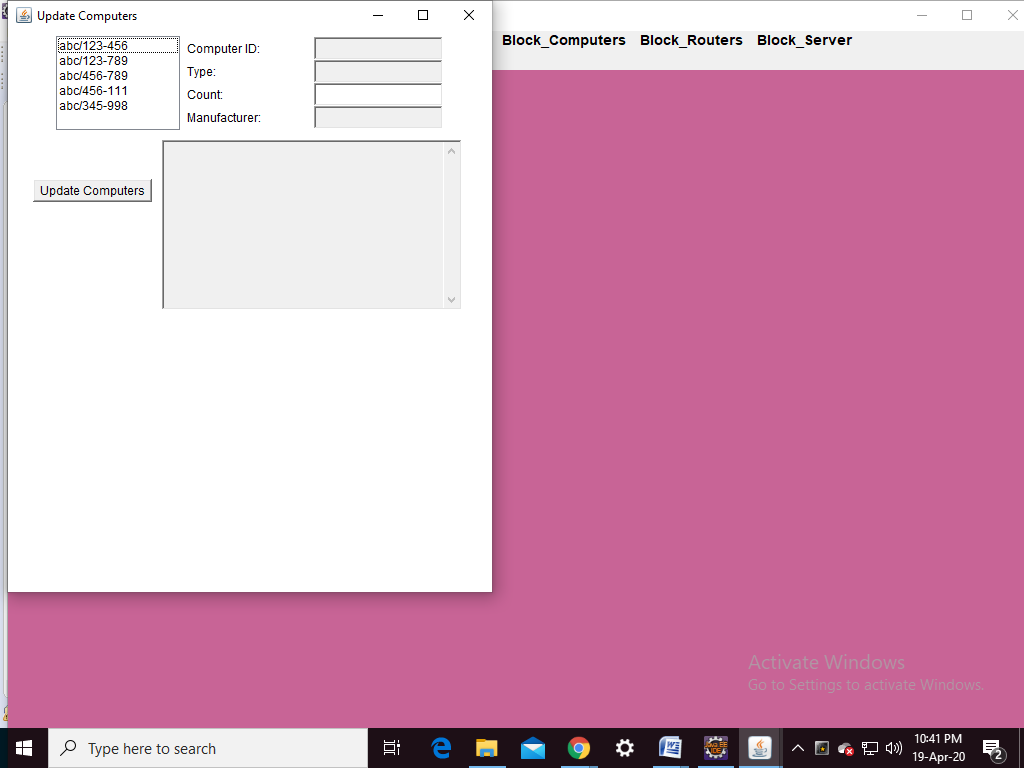
1) For inserting computers



2) for deleting computers



3) for updating computers



**GITHUB Link:**

<https://github.com/ruchita0403-dot/VCE-Network-Connection-Management-System>

Github link

# VCE Network Connection Management System.pdf

DbmsReport

Assignment-2

Assignment-2

METADATA

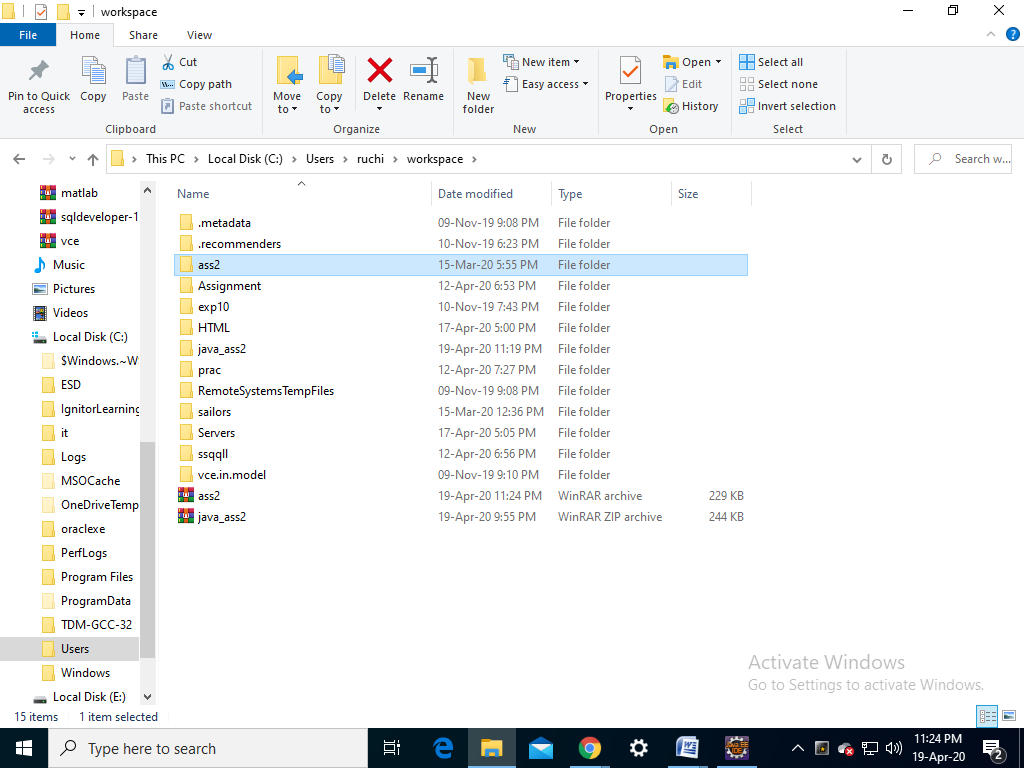
ASS-2 settings

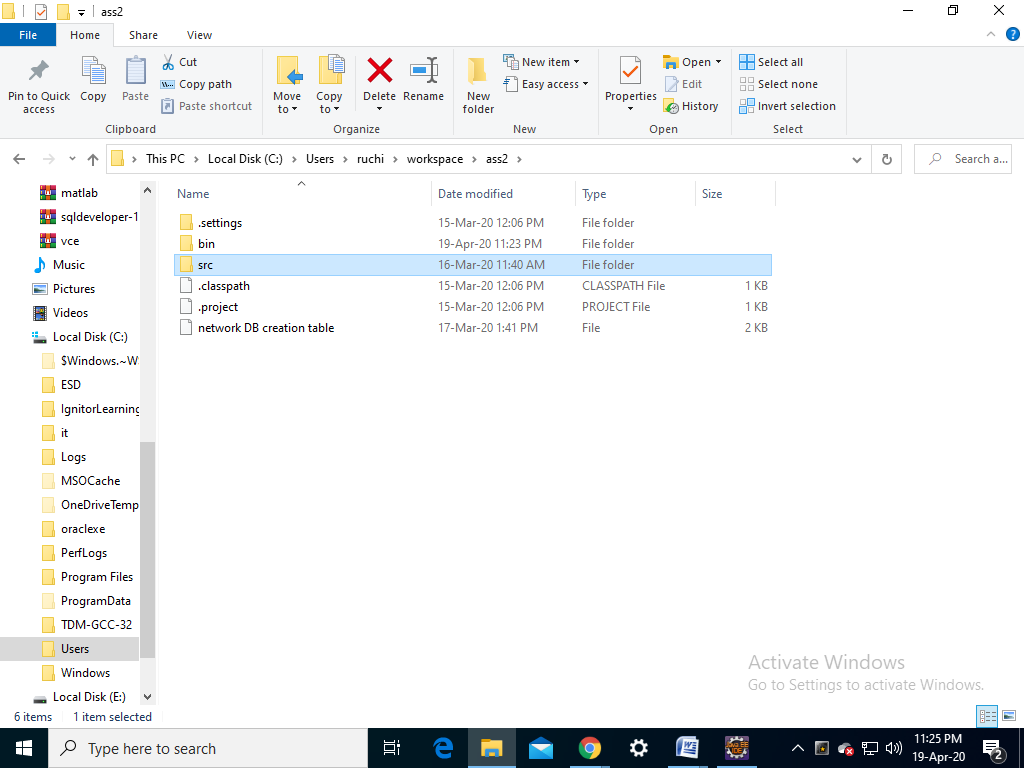
bin

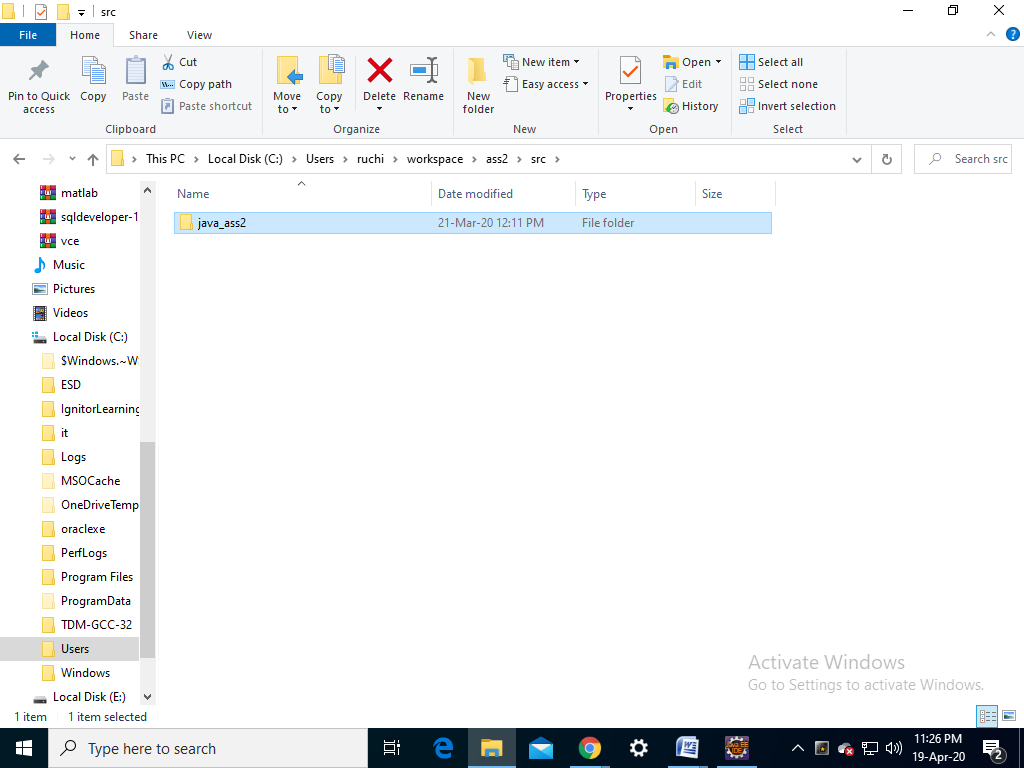
src programs of vce network connection managemnt

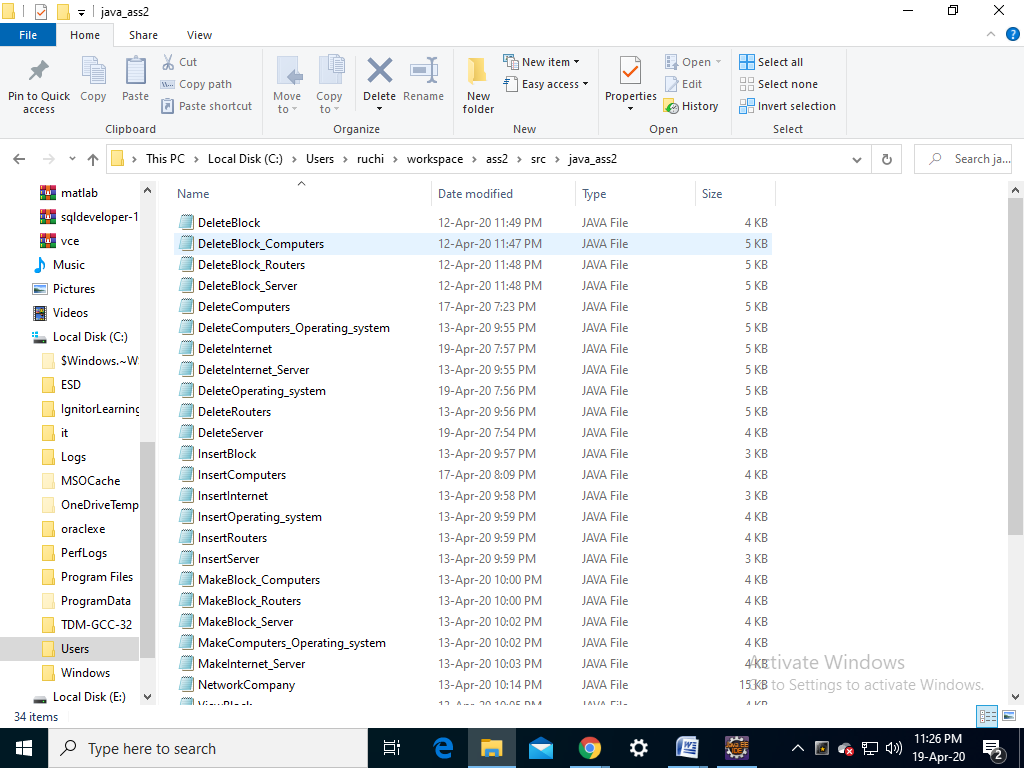
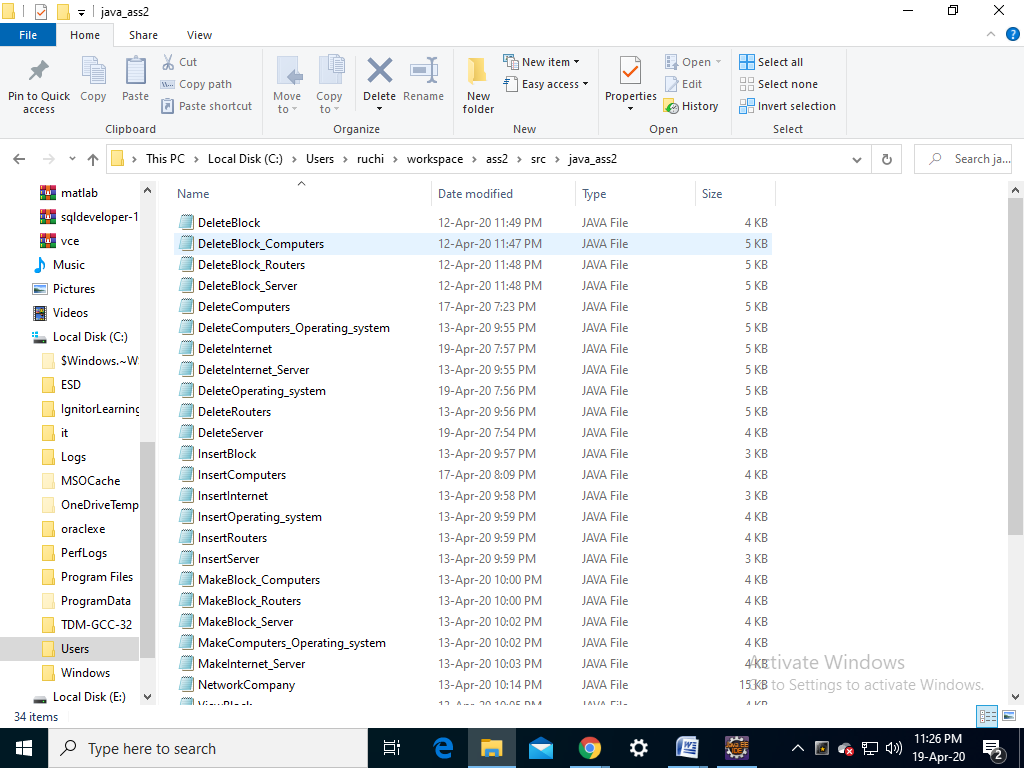
classpath

project

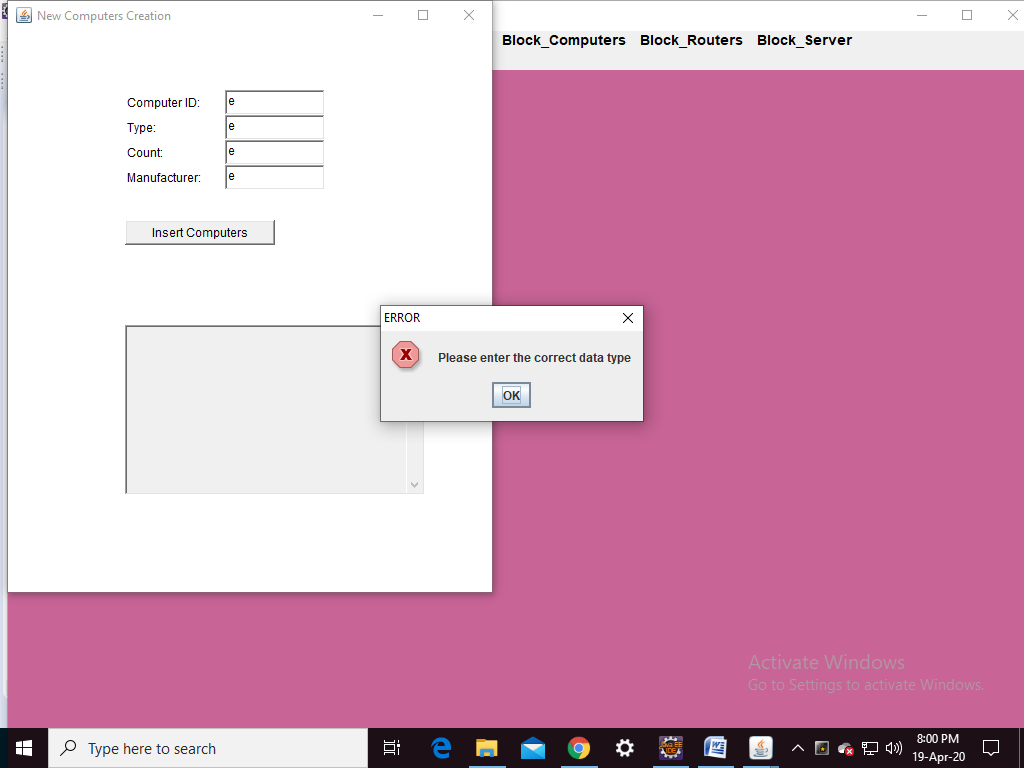






**TESTING**



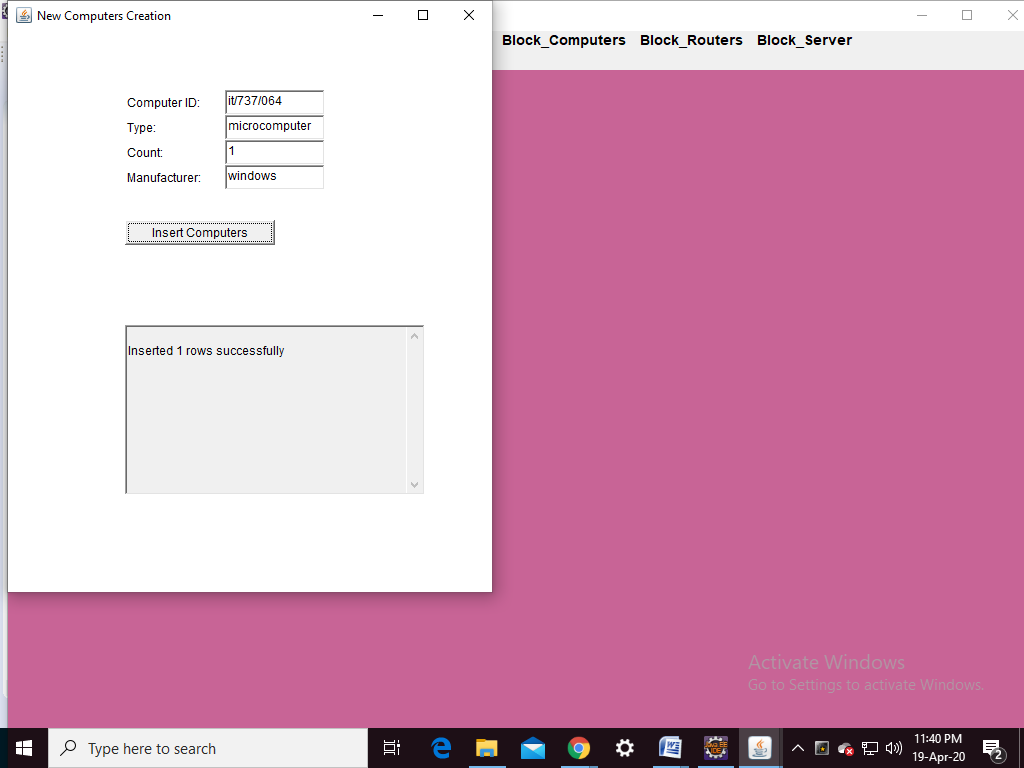
**DML COMMANDS**

1. Insert into internet values(‘&html’,’&serv\_provider’,’&mac\_address’);
2. Insert into computers values ( ‘&type ’ , ’ & count ’ , ’ & manufacturer ’ , ’ & mode ’ ,’ &cid’);
3. Insert into server values(‘&ipaddress’);
4. Insert into block values(‘&bname’,’&hod’,’&branch’);
5. Insert into server values(‘&ipaddress’);
6. Insert into routers values(‘&website’,’&speed’,’&model’,’& username’,’&company’);
7. Insert into operating\_system values(‘&osname’,’& version’,’& vendor’);
8. Insert into connected\_to values(‘&mac’,’& ipadd’);
9. Insert into provides\_network\_to(‘&bname’,’& ipaddress’);
10. Insert into contains values (‘&cid’,’&bname’);
11. Insert into has values (‘&cid’,’& osname’);
12. Insert into are\_having values (‘&website’,’& bname’);

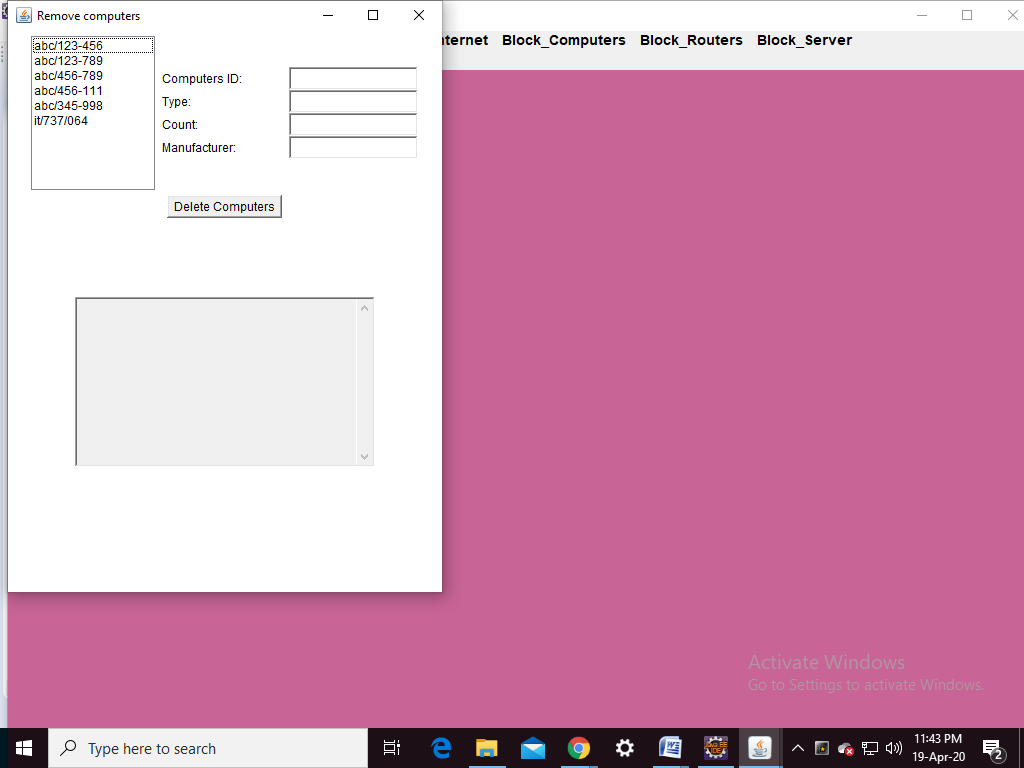
**OUTPUT SCREENSHOTS:**

**Java GUI Screenshot:**

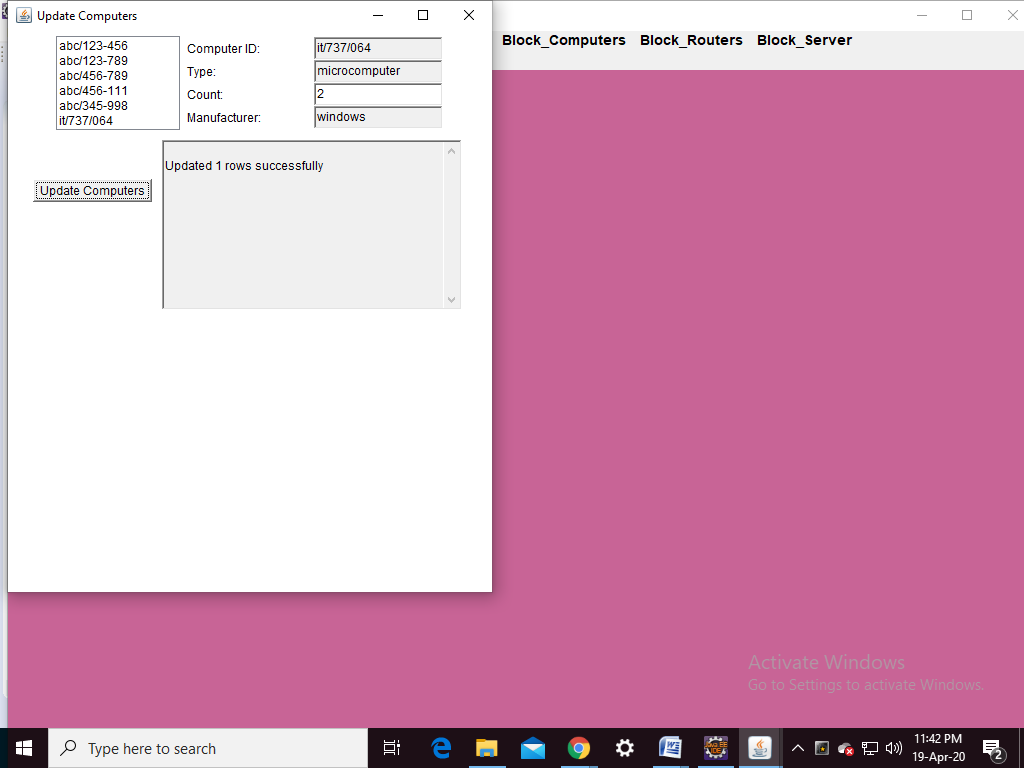
1) For inserting computers



2) for deleting computers

****

3) for updating computers

****

**DISCUSSION & FUTURE WORK :**

The application done till now is to store all the information related to the network connection of our college . Furthermore, other programming languages can also be used along with database by connecting SQL with it. This application can be extended further more to store network connections of other colleges, organizations etc

**CONCLUSION:**

Thus, a Java AWT based network connection management system is created which is connected to the Oracle 11g database. Therefore, all the entries in the form are directly updated on the network table created in the database.

**REFERENCES :**

<https://www.oracle.com/technetwork/java/javase/documentation/index.html>

<https://nptel.ac.in/courses/106105175/>

<https://google.github.io/styleguide/javaguide.html>

<https://nptel.ac.in/courses/106105191/>