**JAVA SWING BASED – College Network Hardware Database- SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

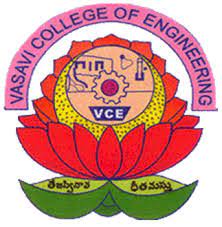
**IN**

**INFORMATION TECHNOLOGY**

**By**

**A. Venkat Sri Harsha <1602-21-737-063>**

**Under the guidance of Ms B. Leelavathy**



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2023**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled

‘***College Network Database***’

is a project work of **A.Venkat Sri Harsha**

bearing roll no. **1602-21-737-063**

who carried out

this project under my supervision

in the IV semester

of the academic year 2023

Signature Signature

External Examiner Internal Examiner

**ABSTRACT**

The database is designed to store information about network hardware connections. It maintains a comprehensive record of the various devices and their connections within the network infrastructure, including switches, routers, servers, and storage devices. The database also tracks configuration details, such as IP addresses, VLAN assignments, and port mappings, to ensure accurate and up-to-date documentation of the network topology. With this information, network administrators can quickly troubleshoot connectivity issues, plan for network expansion, and optimize performance.

**Requirement Analysis**

**List of Tables:**

1. Device

2. Connection

3. Users

4. Service

5. Warranty

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

DEVICE:

1. MAC address - CHAR(14)
2. IP address - CHAR(15)
3. Model - VARCHAR2(8)
4. Manufacturer - VARCHAR2(10)

CONNECTION:

1. From IP address - CHAR(15)
2. To IP address - CHAR(15)

USER:

1. User ID - VARCHAR2(10)
2. Department - VARCHAR(10)
3. IP address - CHAR(15)

SERVICE:

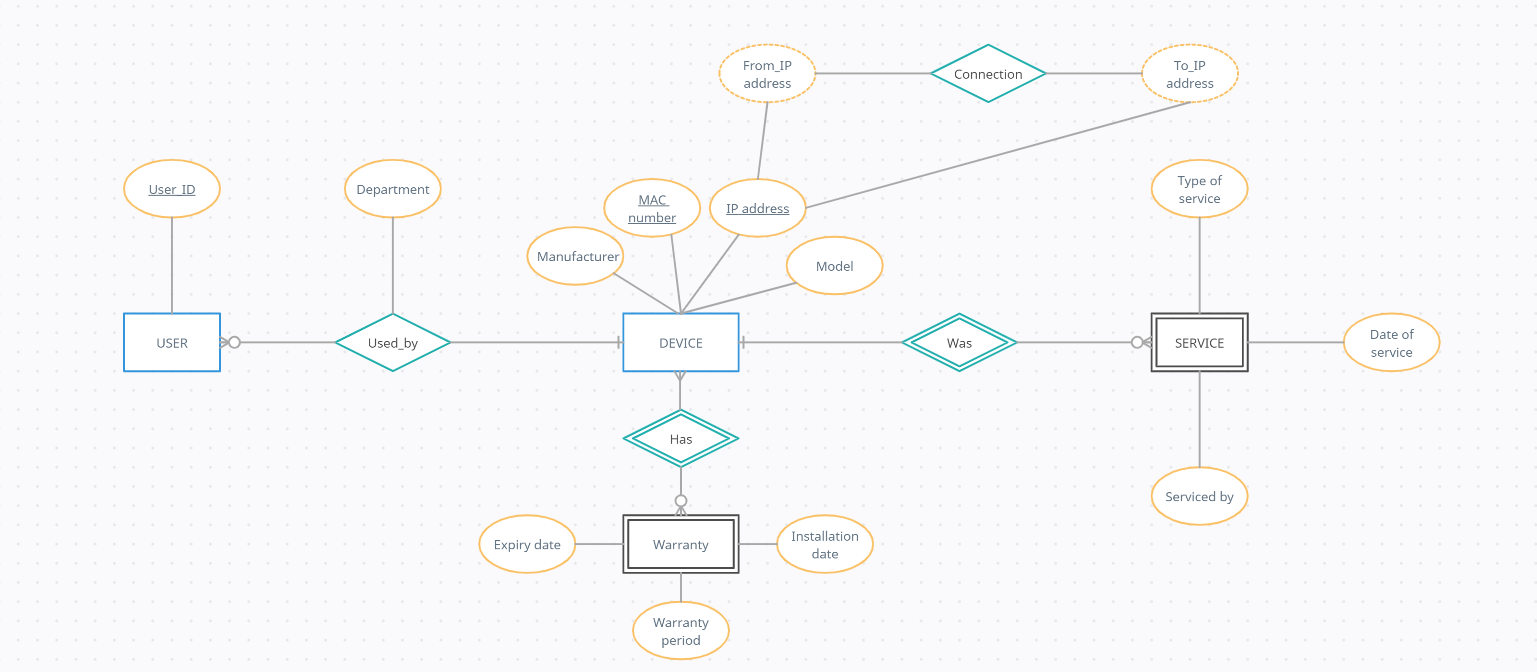
1. Type of service - VARCHAR(20)
2. Date of service - DATE
3. Serviced by - VARCHAR(20)
4. MAC address - CHAR(14)

WARRANTY:

1. MAC address - CHAR(14)
2. Expiry date - DATE
3. Installation date - DATE
4. Warranty Period - TINYINT(80)

**DESIGN**

**Entity Relationship Diagram**



**DDL Commands:**

CREATE TABLE DEVICE( mac\_address CHAR(14),

ip\_address CHAR(15),

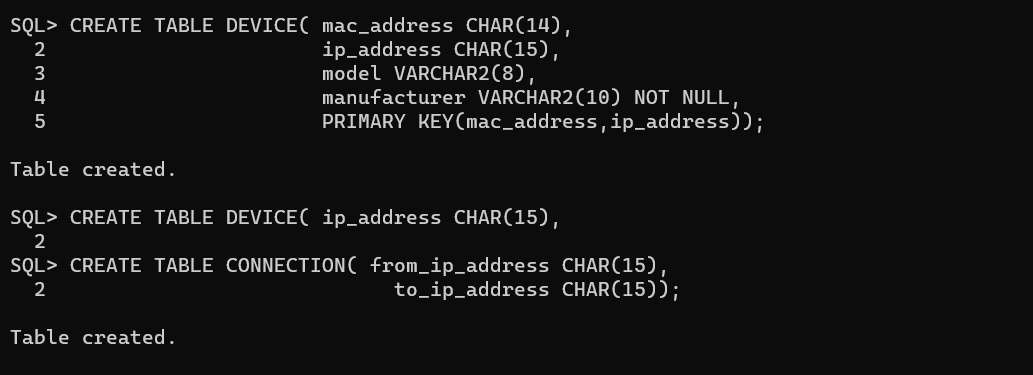
model VARCHAR2(8),

manufacturer VARCHAR2(10) NOT NULL,

PRIMARY KEY(mac\_address,ip\_address));

CREATE TABLE CONNECTION( from\_ip\_address CHAR(15),

to\_ip\_address CHAR(15));

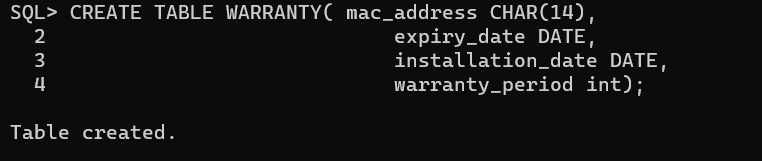


CREATE TABLE WARRANTY( mac\_address CHAR(14),

expiry\_date DATE,

installation\_date DATE,

warranty\_period int);



alter table device drop primary key;

alter table device add primary key(mac\_address);

alter table warranty ADD CONSTRAINT mac\_for FOREIGN KEY(mac\_address) REFERENCES DEVICE(mac\_address);

alter table warranty ADD CONSTRAINT warr\_check CHECK(warranty\_period<=80 AND expiry\_date>installation\_date);

alter table device add unique(ip\_address);

CREATE TABLE USERS( ip\_address CHAR(15),

user\_id VARCHAR(10) PRIMARY KEY,

department VARCHAR2(10),

FOREIGN KEY(ip\_address) REFERENCES DEVICE(ip\_address));

CREATE TABLE SERVICE(

mac\_address char(14),

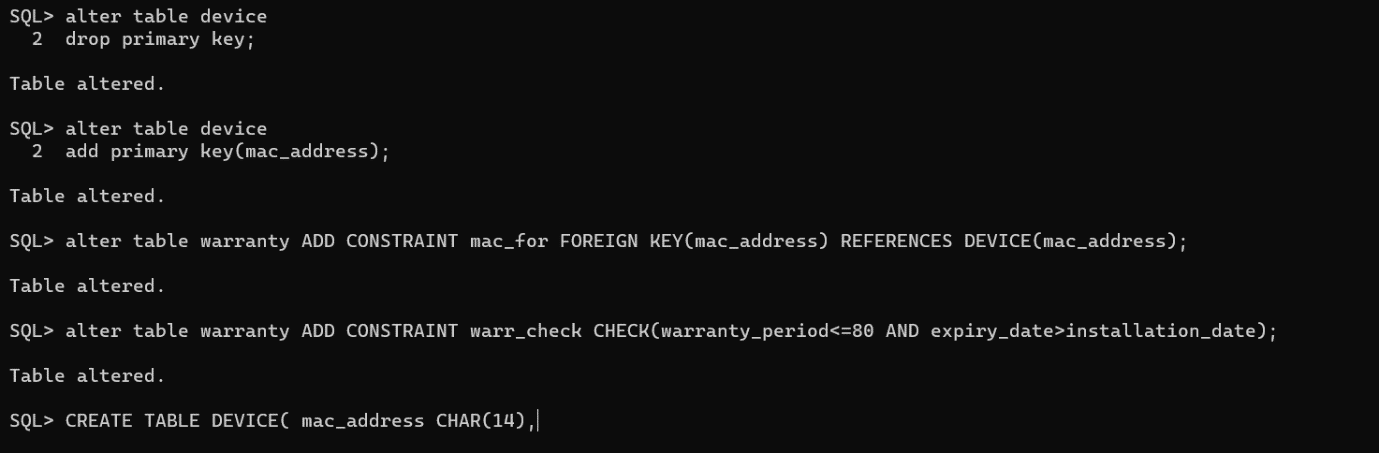
serviced\_by VARCHAR2(20),

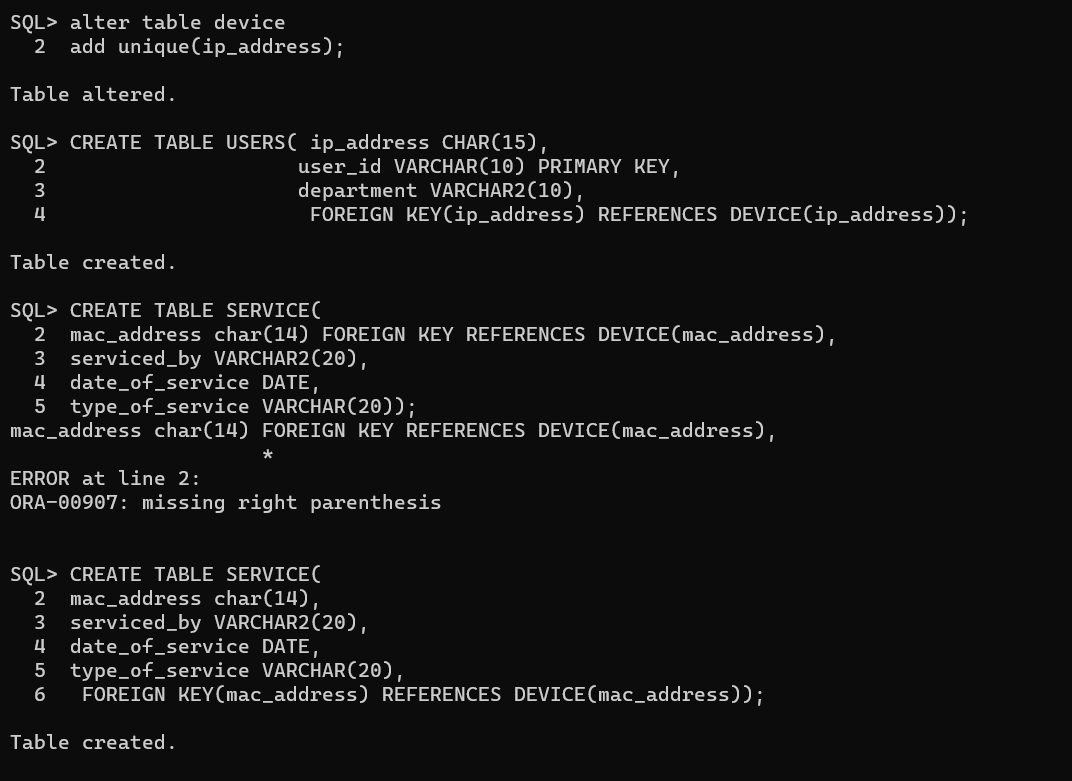
date\_of\_service DATE,

type\_of\_service VARCHAR(20),

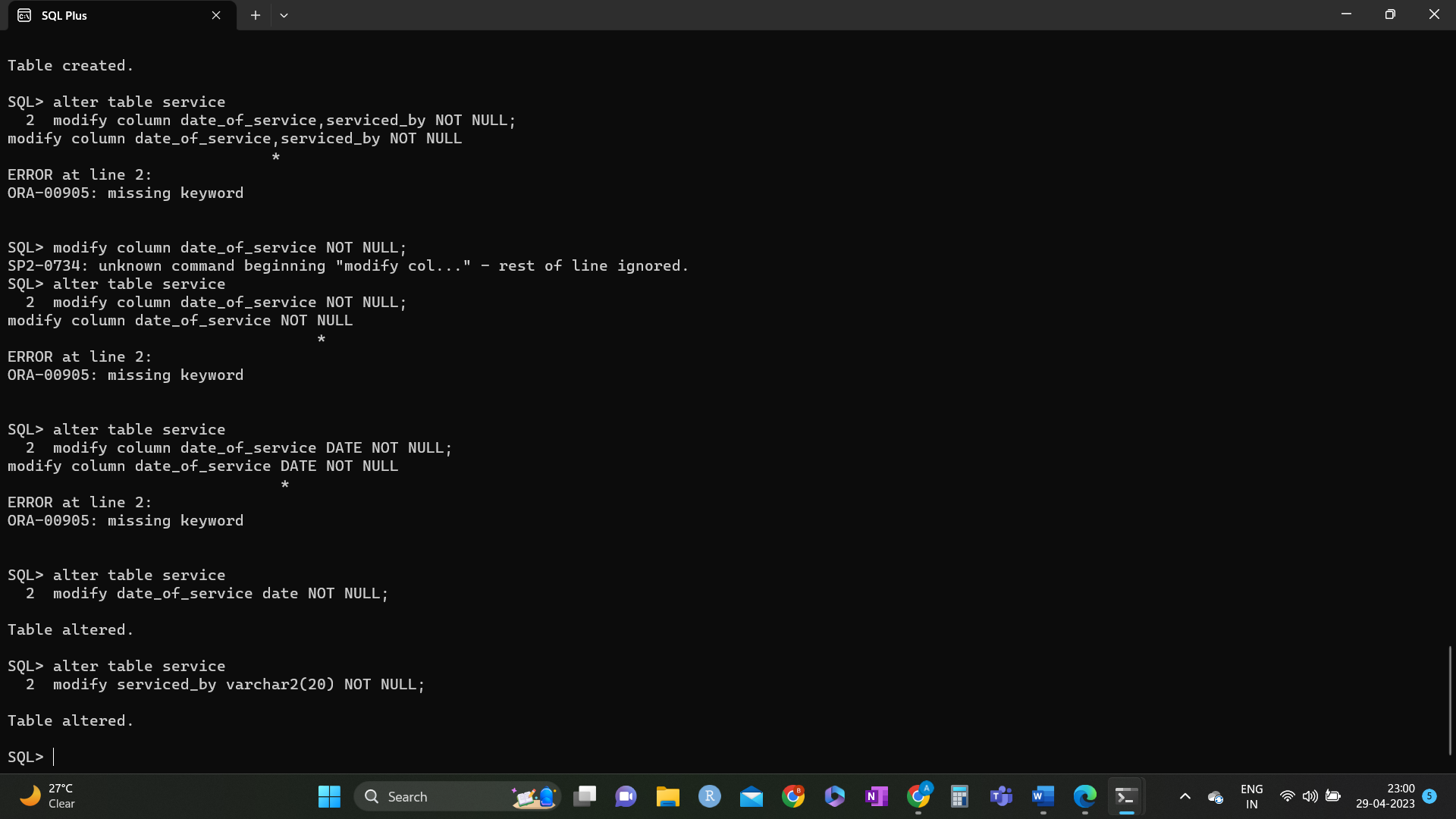
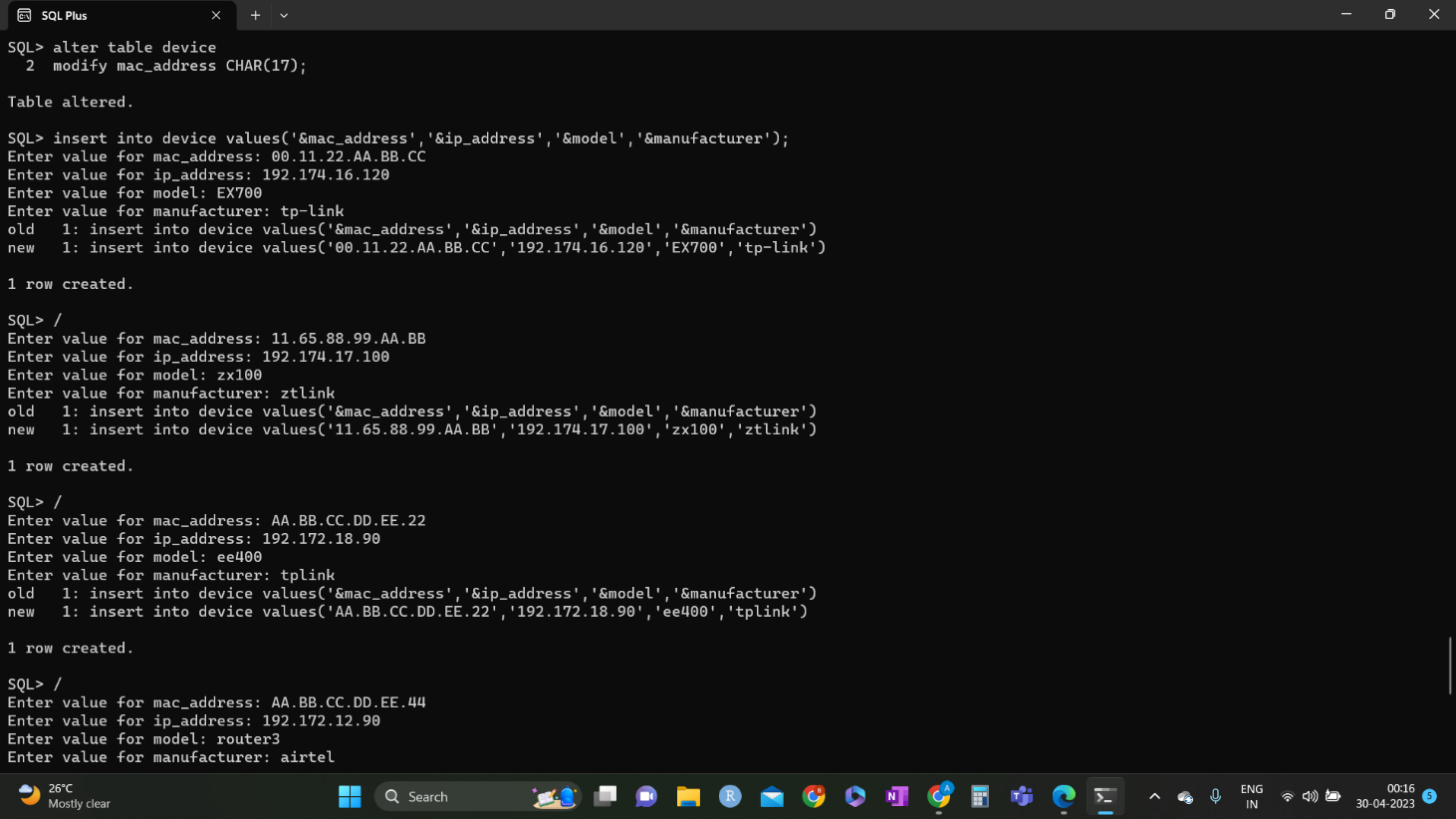
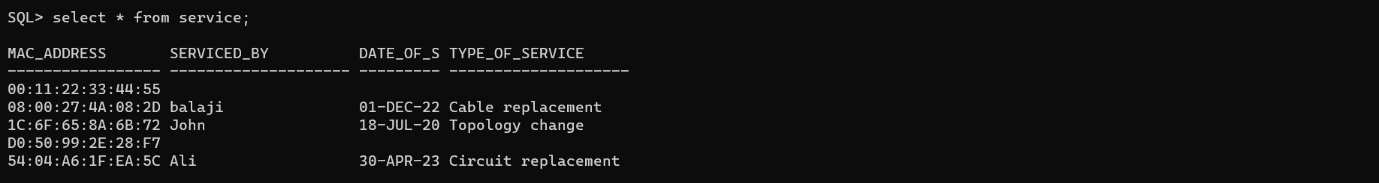
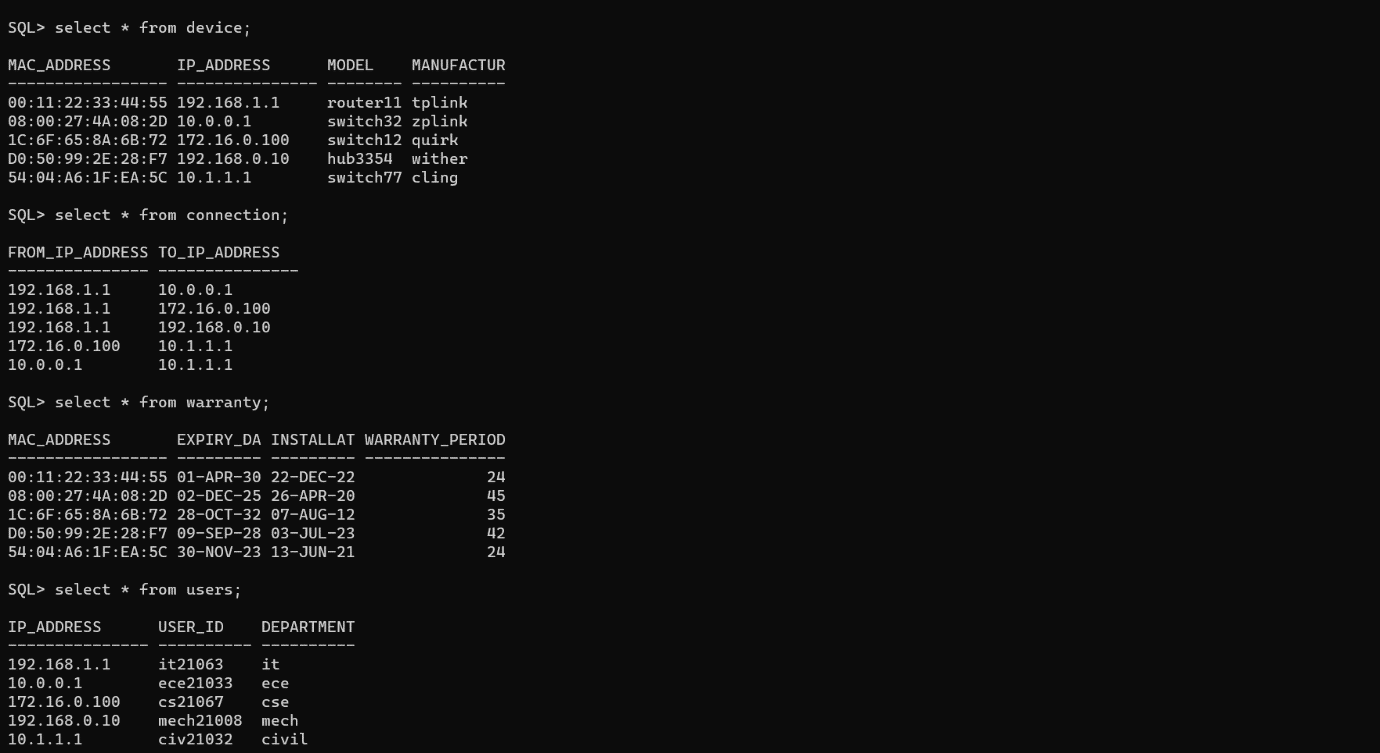
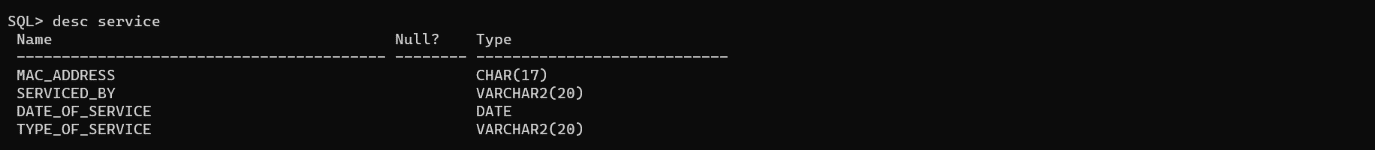
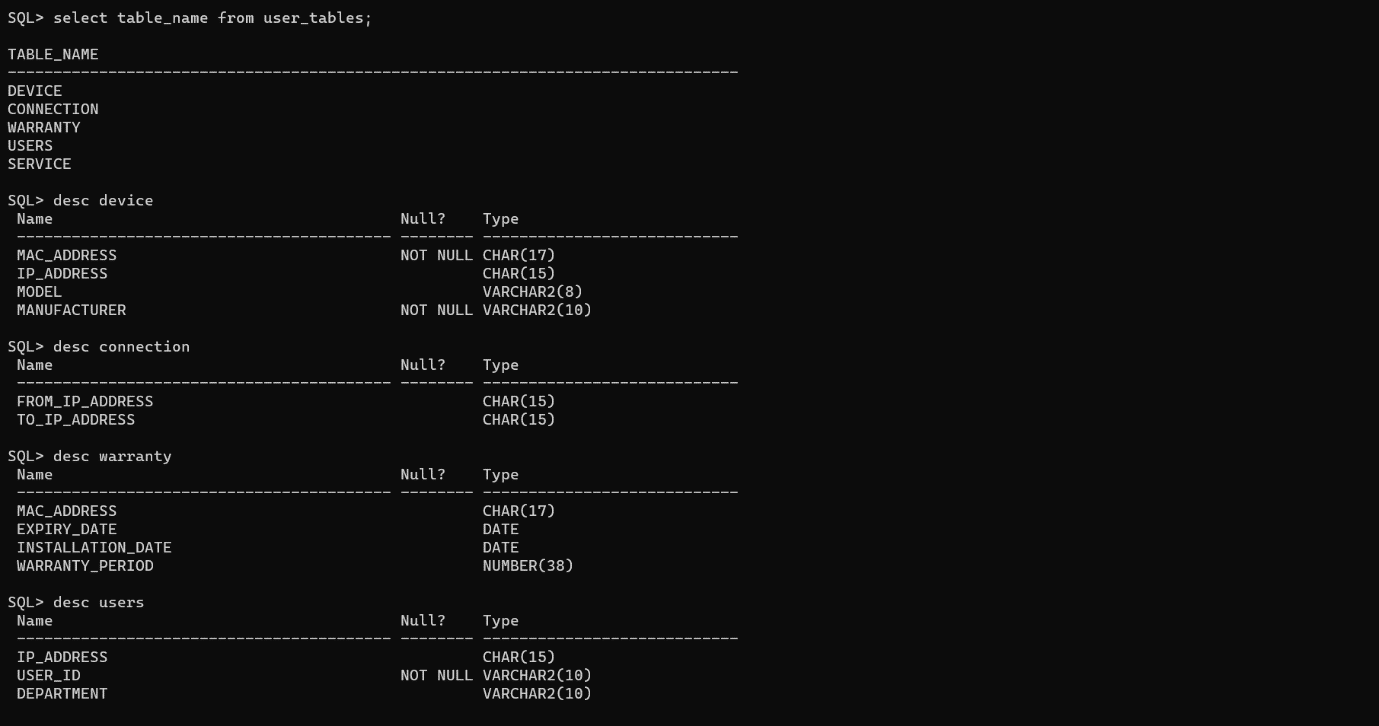
FOREIGN KEY(mac\_address) REFERENCES DEVICE(mac\_address));

alter table service modify date\_of\_service date NOT NULL; alter table service modify serviced\_by varchar2(20) NOT NULL;





**DML Commands:**

********

**IMPLEMENTATION**

**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:



**Front-end Programs (User Interfaces):**

import javax.swing.table.DefaultTableModel;

import java.text.SimpleDateFormat;

import java.util.Vector;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.event.\*;

import javax.swing.\*;

import java.sql.\*;

@SuppressWarnings("serial")

public class Network extends JFrame {

private JPanel contentPanel,masterPanel1,masterPanel2,masterPanel3,masterPanel4,masterPanel5;

private JPanel panel1, panel2, panel3, panel4, panel5,panelt1,panelt2,panelt3,panelt4,panelt5;

// private JMenu arr, menu2, menu3, menu4, menu5;

private JLabel device\_ip = new JLabel("Device ip-address:");

private JTextField device\_ip\_txt = new JTextField(15);

private JLabel user\_ip = new JLabel("Device ip-address:");

private JTextField user\_ip\_txt = new JTextField(15);

private JLabel user\_id = new JLabel("User id:");

private JTextField user\_id\_txt = new JTextField(15);

private JLabel device\_mac = new JLabel("Device mac-address:");

private JTextField device\_mac\_txt = new JTextField(17);

private JLabel device\_macs = new JLabel("Device mac-address:");

private JTextField device\_macs\_txt = new JTextField(17);

private JLabel device\_mac1 = new JLabel("Device mac-address:");

private JTextField device\_mac1\_txt = new JTextField(17);

private JLabel device\_model = new JLabel("Device model:");

private JTextField device\_model\_txt = new JTextField(8);

private JLabel device\_lastby=new JLabel("Serviced by:");

private JTextField device\_lastby\_txt=new JTextField(20);

private JLabel device\_lastser=new JLabel("Service done:");

private JLabel device\_last=new JLabel("Last serviced on:");

private JTextField device\_last\_txt=new JTextField(10);

private JTextField device\_lastser\_txt=new JTextField(20);

private JLabel dept = new JLabel("Department:");

private JTextField dept\_txt = new JTextField(15);

private JLabel device\_manf = new JLabel("Device Manufacturer:");

private JTextField device\_manf\_txt = new JTextField(10);

private JLabel device\_exp = new JLabel("Expiration date:");

private JTextField device\_exp\_txt = new JTextField(10);

private JLabel device\_inst = new JLabel("Installation date:");

private JTextField device\_inst\_txt = new JTextField(10);

private JLabel device\_warr = new JLabel("Warranty period:");

private JTextField device\_warr\_txt = new JTextField(4);

private JLabel device\_fromip = new JLabel("Device From ip-address:");

private JTextField device\_fromip\_txt = new JTextField(15);

private JLabel device\_toip = new JLabel("Device To ip-address:");

private JTextField device\_toip\_txt = new JTextField(15);

private JButton submit1=new JButton("Submit");

private JButton modify1=new JButton("Modify");

private JButton delete1=new JButton("Delete");

private JButton submit2=new JButton("Submit");

private JButton modify2=new JButton("Modify");

private JButton delete2=new JButton("Delete");

private JButton submit3=new JButton("Submit");

private JButton modify3=new JButton("Modify");

private JButton delete3=new JButton("Delete");

private JButton submit4=new JButton("Submit");

private JButton modify4=new JButton("Modify");

private JButton delete4=new JButton("Delete");

private JButton submit5=new JButton("Submit");

private JButton modify5=new JButton("Modify");

private JButton delete5=new JButton("Delete");

private JTable text1,text2,text3,text4,text5;

private JMenu menu1,menu2,menu3,menu4,menu5;

private JButton refresh1=new JButton("Refresh");

private JButton refresh2=new JButton("Refresh");

private JButton refresh3=new JButton("Refresh");

private JButton refresh4=new JButton("Refresh");

private JButton refresh5=new JButton("Refresh");

//private updateRecord(String s){}

public Network() {

// Set up the JFrame

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(870, 400);

setTitle("Network Management System");

// Create the content panel with CardLayout

masterPanel1=new JPanel(new GridLayout(1,2));

masterPanel2=new JPanel(new GridLayout(1,2));

masterPanel3=new JPanel(new GridLayout(1,2));

masterPanel4=new JPanel(new GridLayout(1,2));

masterPanel5=new JPanel(new GridLayout(1,2));

contentPanel = new JPanel(new CardLayout());

// Create the panels

panelt1=new JPanel();

panelt2=new JPanel();

panelt3=new JPanel();

panelt4=new JPanel();

panelt5=new JPanel();

panelt1.setLayout(new BorderLayout());

panelt2.setLayout(new BorderLayout());

panelt3.setLayout(new BorderLayout());

panelt4.setLayout(new BorderLayout());

panelt5.setLayout(new BorderLayout());

panel1 = new JPanel();

panel1.setLayout(new GridLayout(5,1));

panel2 = new JPanel();

panel2.setLayout(new GridLayout(5,1));

panel3 = new JPanel();

panel3.setLayout(new GridLayout(3,1));

panel4 = new JPanel();

panel4.setLayout(new GridLayout(4,1));

panel5 = new JPanel();

panel5.setLayout(new GridLayout(5,1));

// Add components to the panel1

JPanel pnl1 = new JPanel();

JPanel pnl2 = new JPanel();

JPanel pnl3 = new JPanel();

JPanel pnl4 = new JPanel();

JPanel c1 = new JPanel();

JPanel c2 = new JPanel();

JPanel c3 = new JPanel();

JPanel c4 = new JPanel();

JPanel c5 = new JPanel();

c1.add(submit1);

c1.add(delete1);

c1.add(modify1);

c2.add(submit2);

c2.add(delete2);

c2.add(modify2);

c3.add(submit3);

c3.add(delete3);

c3.add(modify3);

c4.add(submit4);

c4.add(delete4);

c4.add(modify4);

c5.add(submit5);

c5.add(delete5);

c5.add(modify5);

pnl2.add(device\_mac1);

pnl2.add(device\_mac1\_txt);

pnl1.add(device\_ip);

pnl1.add(device\_ip\_txt);

pnl3.add(device\_model);

pnl3.add(device\_model\_txt);

pnl4.add(device\_manf);

pnl4.add(device\_manf\_txt);

panel1.add(pnl1);

panel1.add(pnl2);

panel1.add(pnl3);

panel1.add(pnl4);

panel1.add(c1);

// Add components to the panel2

JPanel pnl5 = new JPanel();

JPanel pnl6 = new JPanel();

JPanel pnl7 = new JPanel();

JPanel pnl8 = new JPanel();

pnl5.add(device\_mac);

pnl5.add(device\_mac\_txt);

pnl6.add(device\_exp);

pnl6.add(device\_exp\_txt);

pnl7.add(device\_inst);

pnl7.add(device\_inst\_txt);

pnl8.add(device\_warr);

pnl8.add(device\_warr\_txt);

panel2.add(pnl5);

panel2.add(pnl6);

panel2.add(pnl7);

panel2.add(pnl8);

panel2.add(c2);

// Add components to the panel3

JPanel pnl9 = new JPanel();

JPanel pnl10 = new JPanel();

pnl9.add(device\_fromip);

pnl9.add(device\_fromip\_txt);

pnl10.add(device\_toip);

pnl10.add(device\_toip\_txt);

panel3.add(pnl9);

panel3.add(pnl10);

panel3.add(c3);

// Add components to the panel4

JPanel pnl11 = new JPanel();

JPanel pnl12 = new JPanel();

JPanel pnl13 = new JPanel();

pnl11.add(user\_ip);

pnl11.add(user\_ip\_txt);

pnl12.add(user\_id);

pnl12.add(user\_id\_txt);

pnl13.add(dept);

pnl13.add(dept\_txt);

panel4.add(pnl11);

panel4.add(pnl12);

panel4.add(pnl13);

panel4.add(c4);

// Add components to the panel5

JPanel pnl14 = new JPanel();

JPanel pnl15 = new JPanel();

JPanel pnl16 = new JPanel();

JPanel pnl17 = new JPanel();

pnl14.add(device\_macs);

pnl14.add(device\_macs\_txt);

pnl15.add(device\_lastby);

pnl15.add(device\_lastby\_txt);

pnl16.add(device\_last);

pnl16.add(device\_last\_txt);

pnl17.add(device\_lastser);

pnl17.add(device\_lastser\_txt);

panel5.add(pnl14);

panel5.add(pnl15);

panel5.add(pnl16);

panel5.add(pnl17);

panel5.add(c5);

//create tables and models for each database table

DefaultTableModel model1 = displayTableContents("device");

JTable table1 = new JTable(model1);

DefaultTableModel model2 = displayTableContents("warranty");

JTable table2 = new JTable(model2);

DefaultTableModel model3 = displayTableContents("connection");

JTable table3 = new JTable(model3);

DefaultTableModel model4 = displayTableContents("users");

JTable table4 = new JTable(model4);

DefaultTableModel model5 = displayTableContents("services");

JTable table5 = new JTable(model5);

JScrollPane scrollPane1 = new JScrollPane(table1);

JScrollPane scrollPane2 = new JScrollPane(table2);

JScrollPane scrollPane3 = new JScrollPane(table3);

JScrollPane scrollPane4 = new JScrollPane(table4);

JScrollPane scrollPane5 = new JScrollPane(table5);

//panelt1.add(table1);

panelt1.add(scrollPane1,BorderLayout.CENTER);

panelt1.add(refresh1,BorderLayout.SOUTH);

//panelt2.add(table2);

panelt2.add(scrollPane2,BorderLayout.CENTER);

panelt2.add(refresh2,BorderLayout.SOUTH);

//panelt3.add(table3);

panelt3.add(scrollPane3,BorderLayout.CENTER);

panelt3.add(refresh3,BorderLayout.SOUTH);

//panelt4.add(table4);

panelt4.add(scrollPane4,BorderLayout.CENTER);

panelt4.add(refresh4,BorderLayout.SOUTH);

//panelt5.add(table5);

panelt5.add(scrollPane5,BorderLayout.CENTER);

panelt5.add(refresh5,BorderLayout.SOUTH);

masterPanel1.add(panel1);masterPanel1.add(panelt1);

masterPanel2.add(panel2);masterPanel2.add(panelt2);

masterPanel3.add(panel3);masterPanel3.add(panelt3);

masterPanel4.add(panel4);masterPanel4.add(panelt4);

masterPanel5.add(panel5);masterPanel5.add(panelt5);

// Add the panels to the content panel

contentPanel.add(masterPanel1, "Panel-d");

contentPanel.add(masterPanel2, "Panel-w");

contentPanel.add(masterPanel3, "Panel-c");

contentPanel.add(masterPanel4, "Panel-u");

contentPanel.add(masterPanel5, "Panel-s");

// Create the menu items

JMenu menu1 = new JMenu("Device");

JMenu menu2 = new JMenu("Warranty");

JMenu menu3 = new JMenu("Connections");

JMenu menu4 = new JMenu("Users");

JMenu menu5 = new JMenu("Service");

JMenu menu6 = new JMenu("About");

// Create the menu bar

JMenuBar menuBar = new JMenuBar();

// Add the menus to the menu bar

menuBar.add(menu1);

menuBar.add(menu2);

menuBar.add(menu3);

menuBar.add(menu4);

menuBar.add(menu5);

menuBar.add(menu6);

// Set the menu bar to the JFrame

setJMenuBar(menuBar);

// Add MenuListener to the menu items

menu1.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

CardLayout cardLayout = (CardLayout) contentPanel.getLayout();

cardLayout.show(contentPanel, "Panel-d");

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

menu2.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

CardLayout cardLayout = (CardLayout) contentPanel.getLayout();

cardLayout.show(contentPanel, "Panel-w");

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

menu3.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

CardLayout cardLayout = (CardLayout) contentPanel.getLayout();

cardLayout.show(contentPanel, "Panel-c");

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

menu4.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

CardLayout cardLayout = (CardLayout) contentPanel.getLayout();

cardLayout.show(contentPanel, "Panel-u");

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

menu5.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

CardLayout cardLayout = (CardLayout) contentPanel.getLayout();

cardLayout.show(contentPanel, "Panel-s");

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

menu6.addMenuListener(new MenuListener() {

public void menuSelected(MenuEvent e) {

JOptionPane.showMessageDialog(Network.this,"The database is designed to store information about network hardware connections.\n It maintains a comprehensive record of the various devices and their connections within the network infrastructure, including switches, routers, servers, and storage devices.\n The database also tracks configuration details, such as IP addresses, VLAN assignments, and port mappings, to ensure accurate and up-to-date documentation of the network topology.\n With this information, network administrators can quickly troubleshoot connectivity issues, plan for network expansion, and optimize performance.","Project Information",JOptionPane.INFORMATION\_MESSAGE);

}

public void menuCanceled(MenuEvent e) {

}

public void menuDeselected(MenuEvent e) {

}

});

try{

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

}

//catch(SQLException e){System.out.println(e);}

catch(Exception ex){System.out.println(ex);}

submit1.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

String sql = "INSERT INTO device (mac\_address, ip\_address, model, manufacturer) " +

"VALUES ('" + device\_mac1\_txt.getText() + "', '" + device\_ip\_txt.getText() + "', '" + device\_model\_txt.getText() + "', '" + device\_manf\_txt.getText()+"')";

stmt.executeUpdate(sql);

JOptionPane.showMessageDialog(Network.this,"New record inserted into device table","Insertion status",JOptionPane.INFORMATION\_MESSAGE);

//conn.commit();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+sqle,"Insertion status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+ex,"Insertion status",JOptionPane.ERROR\_MESSAGE);}

}

});

modify1.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

if (table1.isEditing()) {

table1.getCellEditor().stopCellEditing(); // Stop cell editing and commit changes

}

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

int[] selectedRows = table1.getSelectedRows(); // Get selected rows

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(Network.this, "No rows selected", "Update Error", JOptionPane.WARNING\_MESSAGE);

return; // Return if no rows are selected

}

for (int row : selectedRows) {

String mac = table1.getValueAt(row, 0).toString();

String IP= table1.getValueAt(row, 1).toString();

String model= table1.getValueAt(row, 2).toString();

String manf= table1.getValueAt(row, 3).toString();

// Debug print statements

System.out.println("mac: " + mac);

System.out.println("IP: " + IP);

System.out.println("model: " + model);

System.out.println("manf: " + manf);

String sql = "UPDATE device SET " +

"mac\_address = '" + mac + "', " +

"ip\_address = '" + IP + "', "+

"model = '" + model + "', "+

"manufacturer = '" + manf + "' "+

"WHERE mac\_address = '" + mac + "' ";

int no = stmt.executeUpdate(sql);

//conn.commit();

if(no>0){

JOptionPane.showMessageDialog(Network.this,"Record Modified successfully","Updation status",JOptionPane.INFORMATION\_MESSAGE);}

else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the MAC address: " + device\_mac\_txt.getText(), "Updation status", JOptionPane.WARNING\_MESSAGE);

}

}

conn.close();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not update tuple:"+sqle,"Updation status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not update tuple:"+ex,"Updation status",JOptionPane.ERROR\_MESSAGE);}

}

});

delete1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

String macAddress = device\_mac1\_txt.getText();

String sql = "DELETE FROM device WHERE mac\_address='" + macAddress + "'";

int rowsAffected = stmt.executeUpdate(sql);

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(Network.this, "Record deleted successfully", "Deletion status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the MAC address: " + macAddress, "Deletion status", JOptionPane.WARNING\_MESSAGE);

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + sqle.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + ex.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

}

}

});

submit2.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

String sql = "INSERT INTO warranty (mac\_address, expiry\_date, installation\_date, warranty\_period) " +

"VALUES ('" + device\_mac\_txt.getText() + "', '" + device\_exp\_txt.getText() + "', '" + device\_inst\_txt.getText() + "', '" + device\_warr\_txt.getText()+"')";

stmt.executeUpdate(sql);

JOptionPane.showMessageDialog(Network.this,"New record inserted into warranty table","Insertion status",JOptionPane.INFORMATION\_MESSAGE);

//conn.commit();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+sqle,"Insertion status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+ex,"Insertion status",JOptionPane.ERROR\_MESSAGE);}

}

});

modify2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

if (table2.isEditing()) {

table2.getCellEditor().stopCellEditing(); // Stop cell editing and commit changes

}

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

int[] selectedRows = table2.getSelectedRows(); // Get selected rows

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(Network.this, "No rows selected", "Update Error", JOptionPane.WARNING\_MESSAGE);

return; // Return if no rows are selected

}

for (int row : selectedRows) {

String mac = table2.getValueAt(row, 0).toString();

String epd = table2.getValueAt(row, 1).toString();

String isd = table2.getValueAt(row, 2).toString();

int wp = Integer.parseInt(table2.getValueAt(row, 3).toString());

// Format date values as per the expected format in the SQL statement

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

String formattedEpd = dateFormat.format(dateFormat.parse(epd));

String formattedIsd = dateFormat.format(dateFormat.parse(isd));

String sql = "UPDATE warranty SET " +

"EXPIRY\_DATE = TO\_DATE('" + formattedEpd + "', 'YYYY-MM-DD'), " +

"INSTALLATION\_DATE = TO\_DATE('" + formattedIsd + "', 'YYYY-MM-DD'), " +

"WARRANTY\_PERIOD = " + wp +

" WHERE MAC\_ADDRESS = '" + mac + "'";

int no = stmt.executeUpdate(sql);

//conn.commit();

if (no > 0) {

JOptionPane.showMessageDialog(Network.this, "Record Modified successfully", "Updation status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the MAC address: " + device\_mac\_txt.getText(), "Updation status", JOptionPane.WARNING\_MESSAGE);

}

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple:" + sqle, "Updation status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple:" + ex, "Updation status", JOptionPane.ERROR\_MESSAGE);

}

}

});

delete2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

String macAddress = device\_mac\_txt.getText();

String sql = "DELETE FROM warranty WHERE mac\_address='" + macAddress + "'";

int rowsAffected = stmt.executeUpdate(sql);

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(Network.this, "Record deleted successfully", "Deletion status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the MAC address: " + macAddress, "Deletion status", JOptionPane.WARNING\_MESSAGE);

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + sqle.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + ex.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

}

}

});

submit3.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

String sql = "INSERT INTO connection (from\_ip\_address,to\_ip\_address ) " +

"VALUES ('" + device\_fromip\_txt.getText() + "', '" + device\_toip\_txt.getText() +"')";

stmt.executeUpdate(sql);

JOptionPane.showMessageDialog(Network.this,"New record inserted into connection table","Insertion status",JOptionPane.INFORMATION\_MESSAGE);

//conn.commit();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+sqle,"Insertion status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+ex,"Insertion status",JOptionPane.ERROR\_MESSAGE);}

}

});

modify3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

if (table3.isEditing()) {

table3.getCellEditor().stopCellEditing(); // Stop cell editing and commit changes

}

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

int[] selectedRows = table3.getSelectedRows(); // Get selected rows

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(Network.this, "No rows selected", "Update Error", JOptionPane.WARNING\_MESSAGE);

return; // Return if no rows are selected

}

for (int row : selectedRows) {

String fromIP = table3.getValueAt(row, 0).toString();

String toIP = table3.getValueAt(row, 1).toString();

String sql = "UPDATE connection SET " +

"from\_ip\_address = '" + fromIP + "', " +

"to\_ip\_address = '" + toIP + "' "+

"WHERE cid = '" + row + "' and from\_ip\_address='" + fromIP + "'";

int no = stmt.executeUpdate(sql);

if (no > 0) {

JOptionPane.showMessageDialog(Network.this, "Record modified successfully", "Updation status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the From IP address: " + fromIP, "Updation status", JOptionPane.WARNING\_MESSAGE);

}

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple: " + sqle, "Updation status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple: " + ex, "Updation status", JOptionPane.ERROR\_MESSAGE);

}

}

});

delete3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

String fipAddress = device\_fromip\_txt.getText();

String sql = "DELETE FROM connection WHERE from\_ip\_address='" + fipAddress + "'";

int rowsAffected = stmt.executeUpdate(sql);

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(Network.this, "Record deleted successfully", "Deletion status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the From ip-address: " + fipAddress, "Deletion status", JOptionPane.WARNING\_MESSAGE);

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + sqle.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + ex.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

}

}

});

submit4.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

String sql = "INSERT INTO users (ip\_address,user\_id,department) " +

"VALUES ('" + user\_ip\_txt.getText() + "', '" + user\_id\_txt.getText()+ "', '"+dept\_txt.getText() +"')";

stmt.executeUpdate(sql);

JOptionPane.showMessageDialog(Network.this,"New record inserted into user table","Insertion status",JOptionPane.INFORMATION\_MESSAGE);

//conn.commit();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+sqle,"Insertion status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+ex,"Insertion status",JOptionPane.ERROR\_MESSAGE);}

}

});

modify4.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

if (table4.isEditing()) {

table4.getCellEditor().stopCellEditing(); // Stop cell editing and commit changes

}

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

int[] selectedRows = table4.getSelectedRows(); // Get selected rows

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(Network.this, "No rows selected", "Update Error", JOptionPane.WARNING\_MESSAGE);

return; // Return if no rows are selected

}

for (int row : selectedRows) {

String IP = table4.getValueAt(row, 0).toString();

String uid= table4.getValueAt(row, 1).toString();

String dept= table4.getValueAt(row, 2).toString();

//String manf= table1.getValueAt(row, 3).toString();

String sql = "UPDATE users SET " +

"ip\_address = '" + IP + "', " +

"department = '" + dept + "', " +

"user\_id = '" + uid + "' "+

"WHERE ip\_address='" + IP + "'";

int no=stmt.executeUpdate(sql);

if(no>0){

JOptionPane.showMessageDialog(Network.this,"Record Modified successfully","Updation status",JOptionPane.INFORMATION\_MESSAGE);}

else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the ip-address: " + user\_ip\_txt.getText(), "Updation status", JOptionPane.WARNING\_MESSAGE);

}

}

conn.close();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not update tuple:"+sqle,"Updation status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not update tuple:"+ex,"Updation status",JOptionPane.ERROR\_MESSAGE);}

}

});

delete4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

String sql = "DELETE FROM users WHERE ip\_address='" + user\_ip\_txt.getText() + "'";

int rowsAffected = stmt.executeUpdate(sql);

if (rowsAffected > 0) {

JOptionPane.showMessageDialog(Network.this, "Record deleted successfully", "Deletion status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the ip-address: " + user\_ip\_txt.getText(), "Deletion status", JOptionPane.WARNING\_MESSAGE);

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + sqle.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not delete tuple: " + ex.getMessage(), "Deletion status", JOptionPane.ERROR\_MESSAGE);

}

}

});

submit5.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

try{

Connection conn= DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063","venkat","dbmsra");

Statement stmt=conn.createStatement();

String sql = "INSERT INTO service (mac\_address,serviced\_by,date\_of\_service,type\_of\_service) " +

"VALUES ('" + device\_macs\_txt.getText() + "', '" + device\_lastby\_txt.getText()+ "', '"+device\_last\_txt.getText() +"', '"+device\_lastser\_txt.getText() +"')";

stmt.executeUpdate(sql);

JOptionPane.showMessageDialog(Network.this,"New record inserted into service table","Insertion status",JOptionPane.INFORMATION\_MESSAGE);

//conn.commit();

}

catch(SQLException sqle)

{

JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+sqle,"Insertion status",JOptionPane.ERROR\_MESSAGE);

}

catch(Exception ex){JOptionPane.showMessageDialog(Network.this,"Could not insert tuple:"+ex,"Insertion status",JOptionPane.ERROR\_MESSAGE);}

}

});

modify5.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

if (table5.isEditing()) {

table5.getCellEditor().stopCellEditing(); // Stop cell editing and commit changes

}

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

int[] selectedRows = table5.getSelectedRows(); // Get selected rows

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(Network.this, "No rows selected", "Update Error", JOptionPane.WARNING\_MESSAGE);

return; // Return if no rows are selected

}

for (int row : selectedRows) {

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");

String mac = table5.getValueAt(row, 0).toString();

String serby = table5.getValueAt(row, 1).toString();

String serd = table5.getValueAt(row, 2).toString();

String tpser = table5.getValueAt(row, 3).toString();

String formattedSerd = dateFormat.format(dateFormat.parse(serd));

String sql = "UPDATE service SET " +

"serviced\_by = '" + serby + "', " +

"date\_of\_service = TO\_DATE('" + formattedSerd + "', 'YYYY-MM-DD'), " +

"type\_of\_service = '" + tpser + "' " +

"WHERE mac\_address = '" + mac + "'";

int no = stmt.executeUpdate(sql);

if (no > 0) {

JOptionPane.showMessageDialog(Network.this, "Record Modified successfully", "Updation status", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(Network.this, "No record found matching the MAC address: " + device\_macs\_txt.getText(), "Updation status", JOptionPane.WARNING\_MESSAGE);

}

}

conn.close();

} catch (SQLException sqle) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple:" + sqle, "Updation status", JOptionPane.ERROR\_MESSAGE);

} catch (Exception ex) {

JOptionPane.showMessageDialog(Network.this, "Could not update tuple:" + ex, "Updation status", JOptionPane.ERROR\_MESSAGE);

}

}

});

getContentPane().add(contentPanel);

// Show the JFrame

setVisible(true);

setLocationRelativeTo(null);

refresh1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

table1.setModel(displayTableContents("device"));

//System.out.println(getSize());

}

});

refresh2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

table2.setModel(displayTableContents("warranty"));

}

});

refresh3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

table3.setModel(displayTableContents("connection"));

}

});

refresh4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

table4.setModel(displayTableContents("users"));

}

});

refresh5.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

table5.setModel(displayTableContents("service"));

}

});

}

private DefaultTableModel displayTableContents(String menuOption) {

DefaultTableModel model = new DefaultTableModel();

try {

Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:Ven063", "venkat", "dbmsra");

Statement stmt = conn.createStatement();

String query = "";

switch (menuOption) {

case "warranty":

try {

query = "SELECT \* FROM warranty";

ResultSet resultSet = stmt.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

// Set column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

columnNames.add(metaData.getColumnName(i));

}

model.setColumnIdentifiers(columnNames);

// Set table data

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

rowData.add(resultSet.getString(i));

}

model.addRow(rowData);

}

stmt.close();

return model;

} catch (SQLException e) {

e.printStackTrace();

}

return model;

case "device":

try {

query = "SELECT \* FROM device";

ResultSet resultSet = stmt.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

// Set column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

columnNames.add(metaData.getColumnName(i));

}

model.setColumnIdentifiers(columnNames);

// Set table data

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

rowData.add(resultSet.getString(i));

}

model.addRow(rowData);

}

stmt.close();

return model;

} catch (SQLException e) {

e.printStackTrace();

}

return model;

case "connection":

try {

query = "SELECT \* FROM connection";

ResultSet resultSet = stmt.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

// Set column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

columnNames.add(metaData.getColumnName(i));

}

model.setColumnIdentifiers(columnNames);

// Set table data

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

rowData.add(resultSet.getString(i));

}

model.addRow(rowData);

}

stmt.close();

return model;

} catch (SQLException e) {

e.printStackTrace();

}

return model;

case "service":

try {

query = "SELECT \* FROM service";

ResultSet resultSet = stmt.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

// Set column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

columnNames.add(metaData.getColumnName(i));

}

model.setColumnIdentifiers(columnNames);

// Set table data

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

rowData.add(resultSet.getString(i));

}

model.addRow(rowData);

}

stmt.close();

return model;

} catch (SQLException e) {

e.printStackTrace();

}

return model;

case "users":

try {

query = "SELECT \* FROM users";

ResultSet resultSet = stmt.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

// Set column names

Vector<String> columnNames = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

columnNames.add(metaData.getColumnName(i));

}

model.setColumnIdentifiers(columnNames);

// Set table data

while (resultSet.next()) {

Vector<String> rowData = new Vector<>();

for (int i = 1; i <= columnCount; i++) {

rowData.add(resultSet.getString(i));

}

model.addRow(rowData);

}

stmt.close();

return model;

} catch (SQLException e) {

e.printStackTrace();

}

return model;

}

} catch (SQLException e) {

e.printStackTrace();

}

return model;

}

public static void main(String args[]) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new Network();

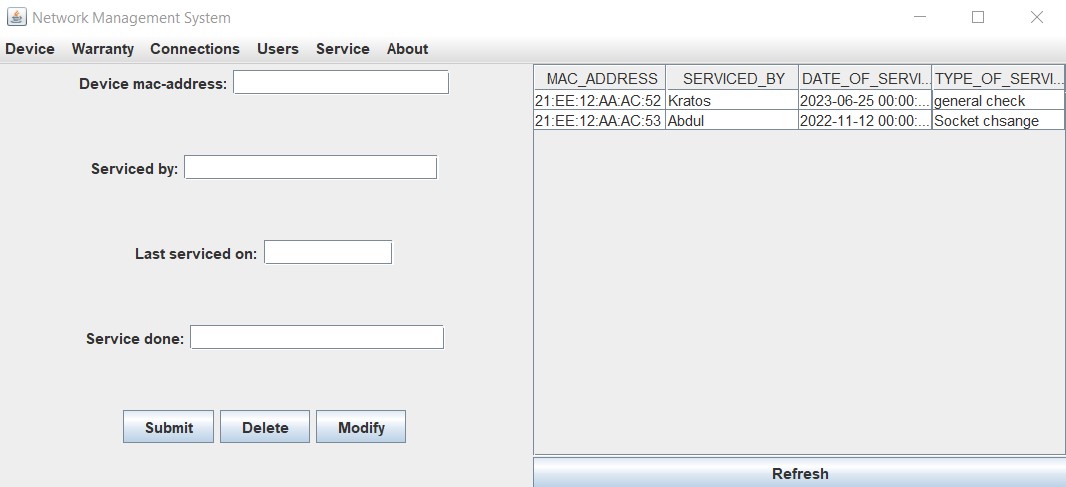
}

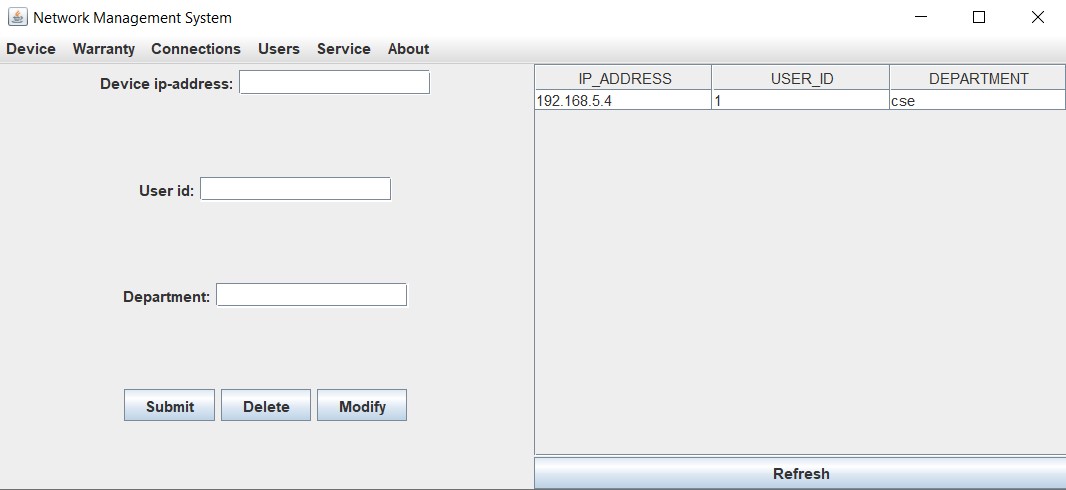
});

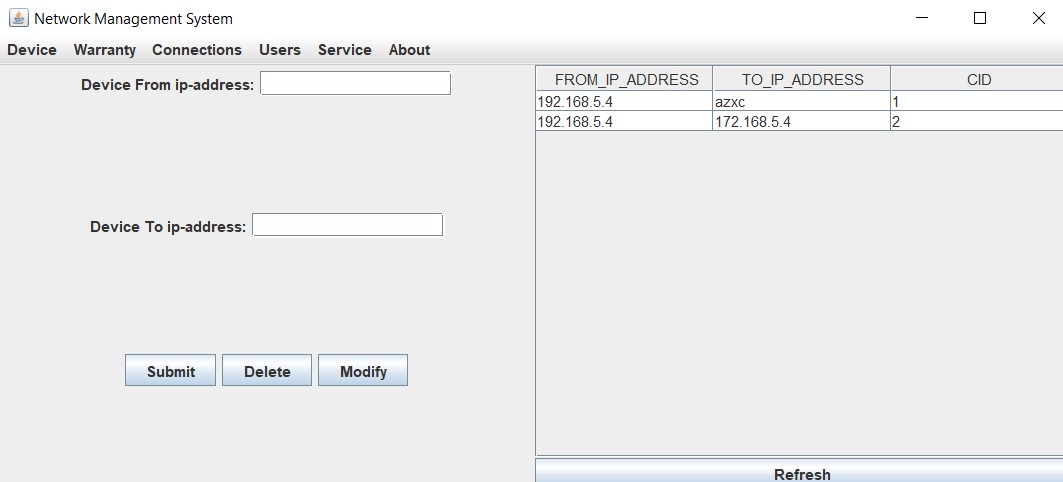
}

}

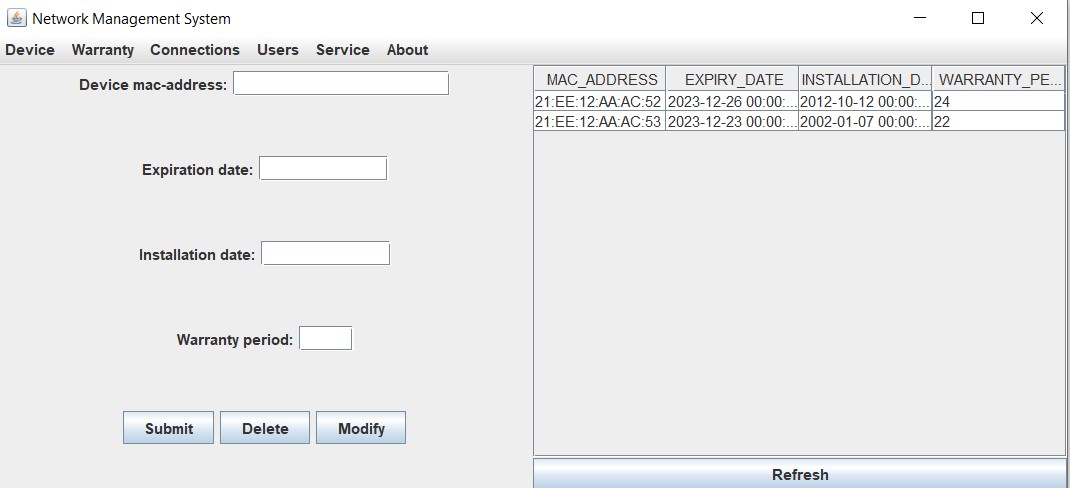
Service table:



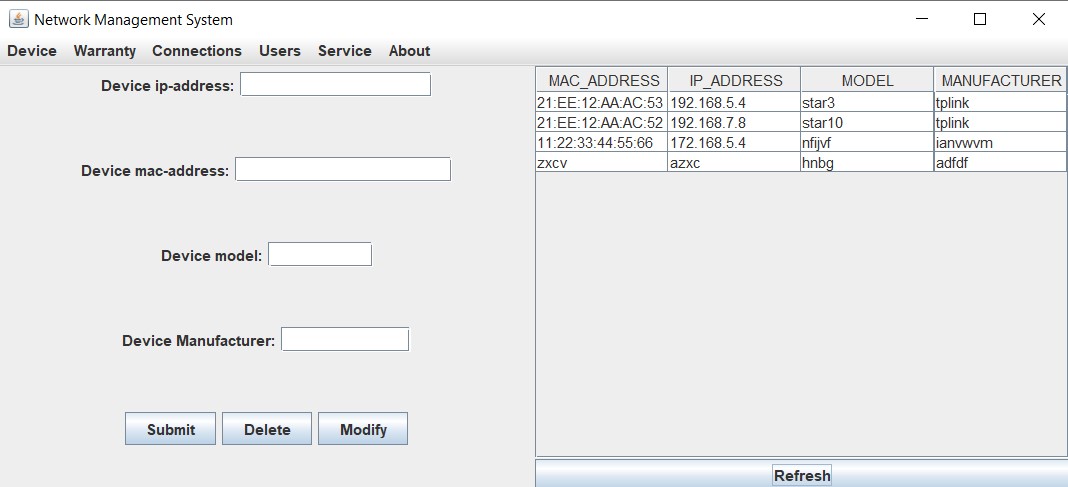
User table:

Connection table:

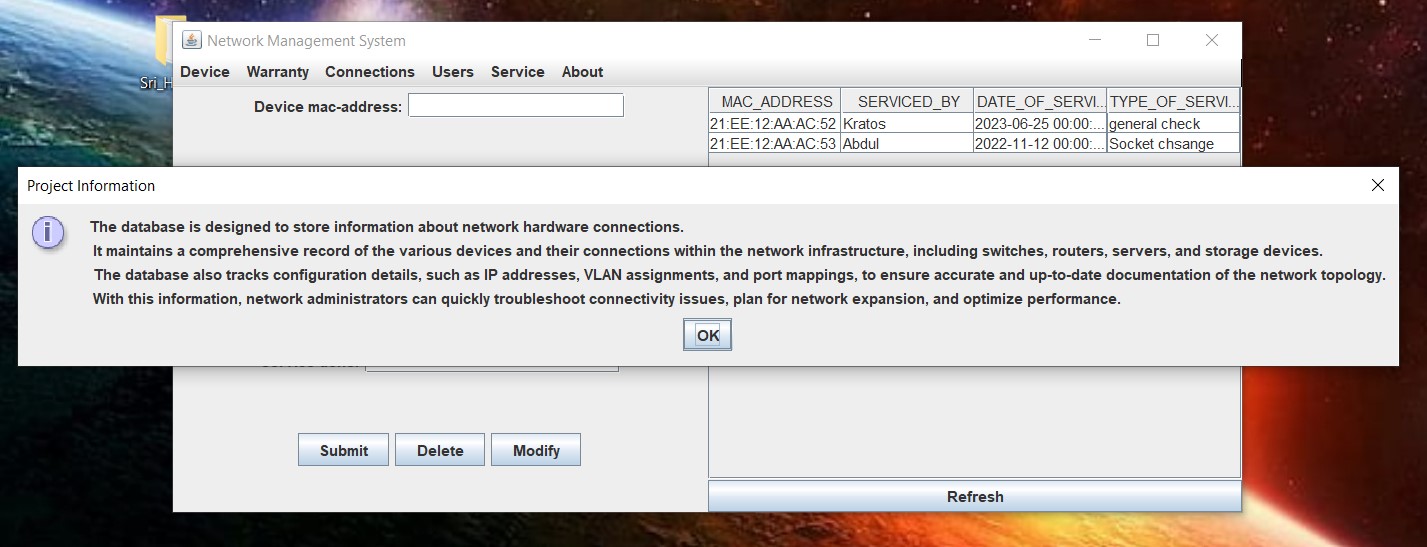
Warranty table:



Device table:



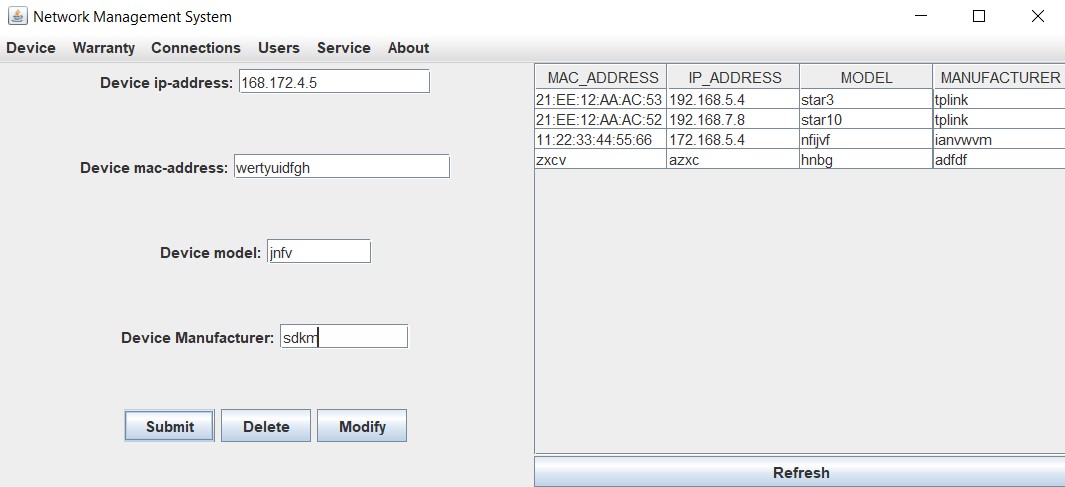
About pane:



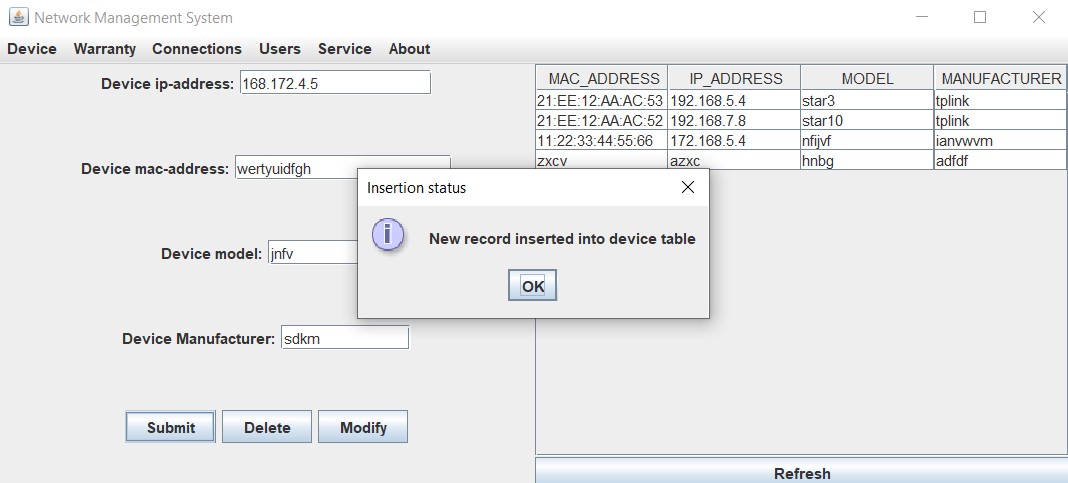
**TESTING:**

**Inserting Device Details:**

Before insertion:



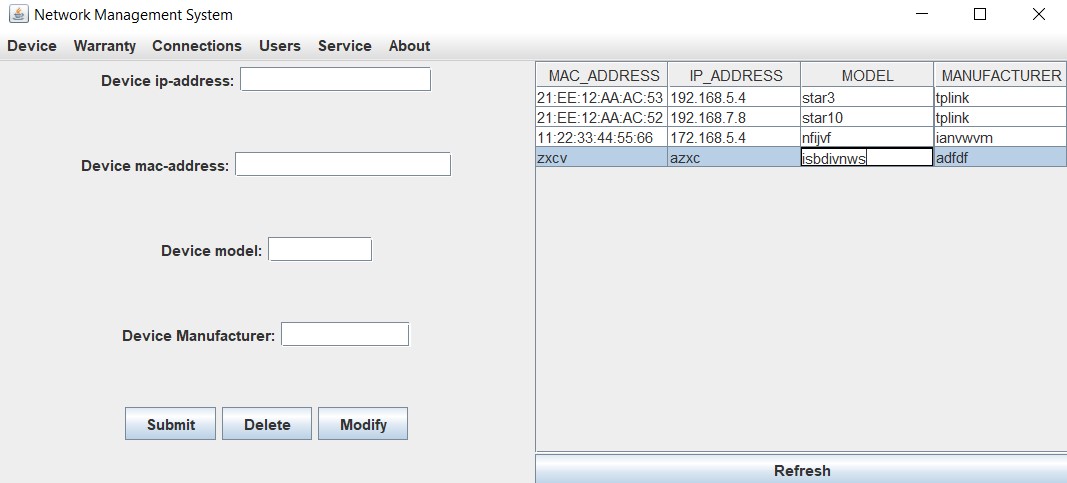
After insertion:



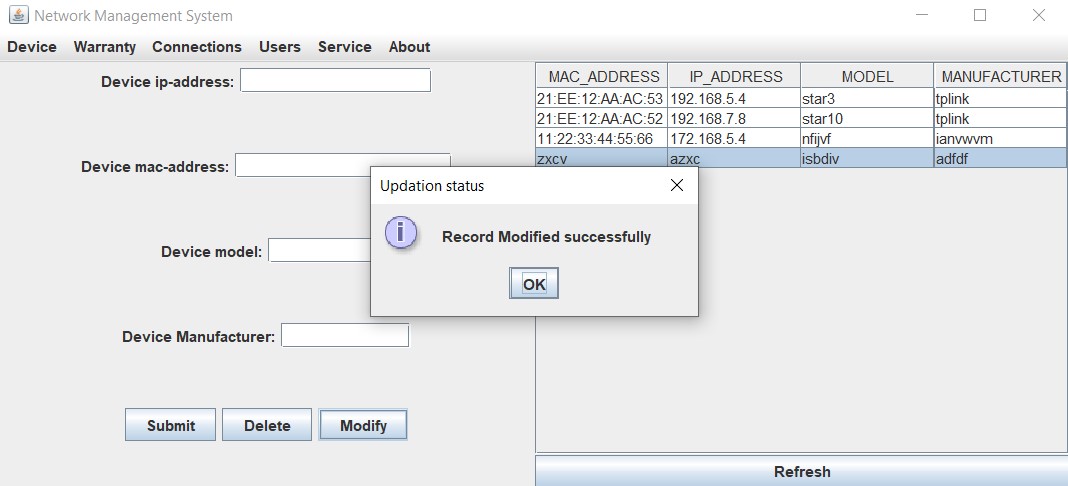
Value gets updated and saved in the table automatically.

**Updating Device Details:**

Before Update:

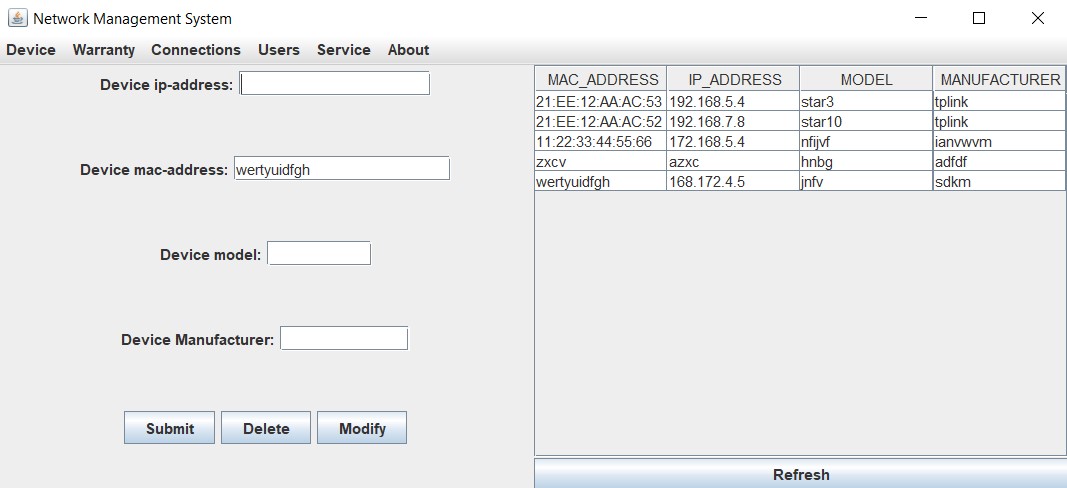


After Update:

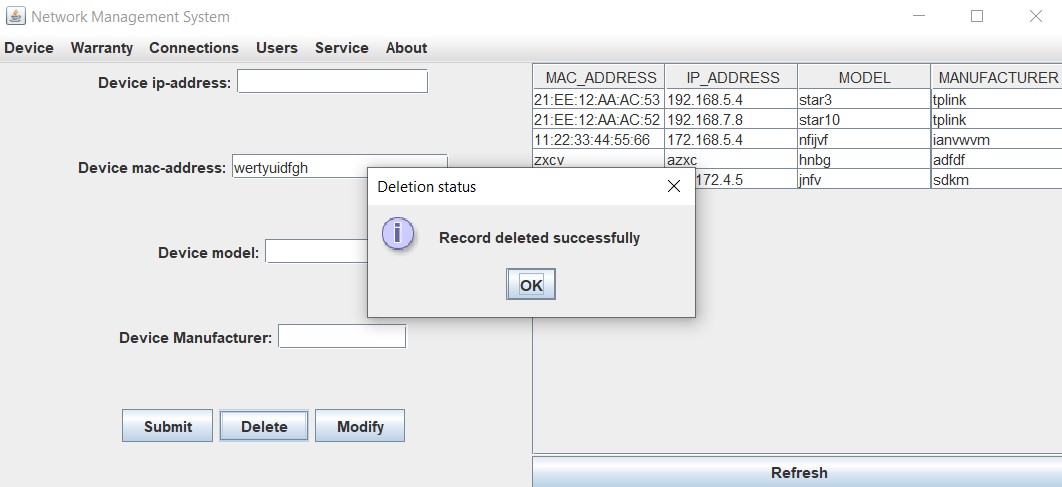


**Deleting Device Details:**

Before Delete:

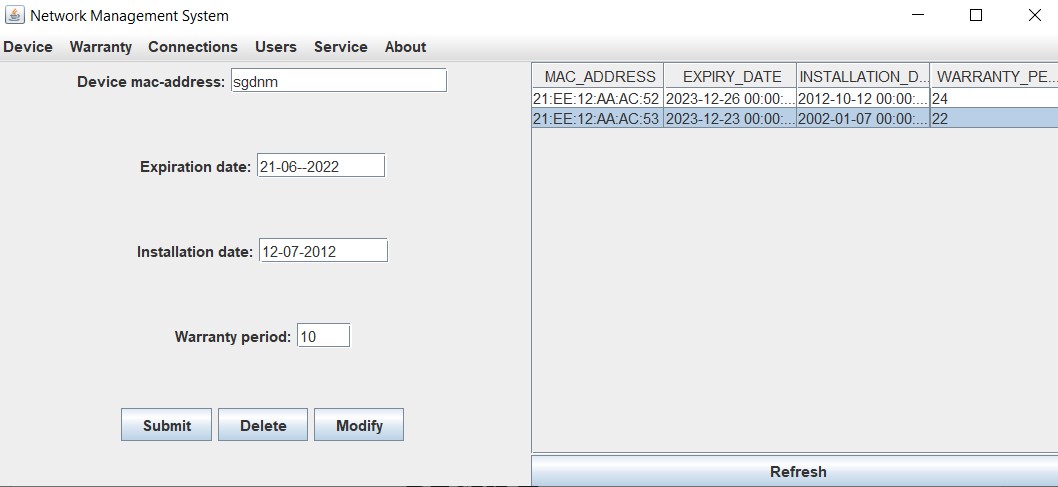


After Delete:

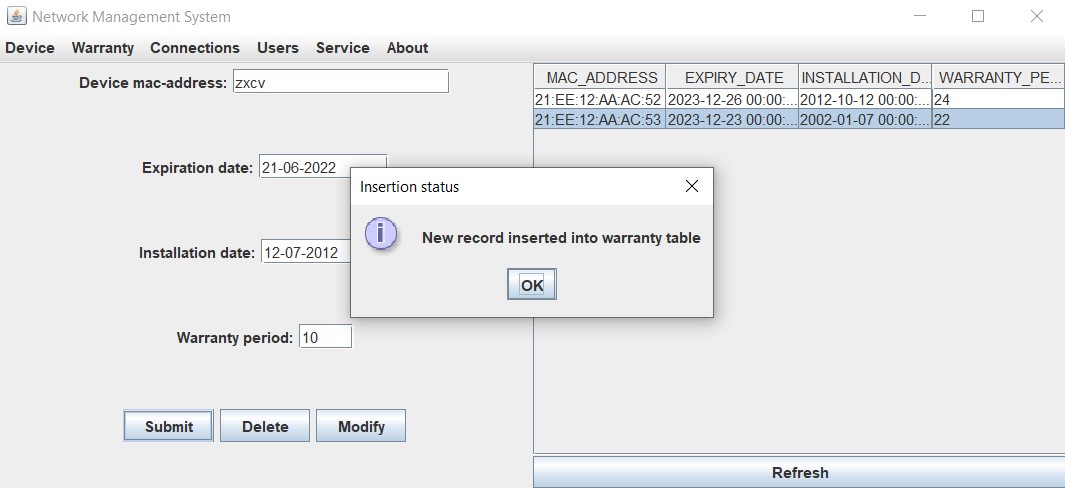


**Inserting Warranty Details:**

Before Insert:

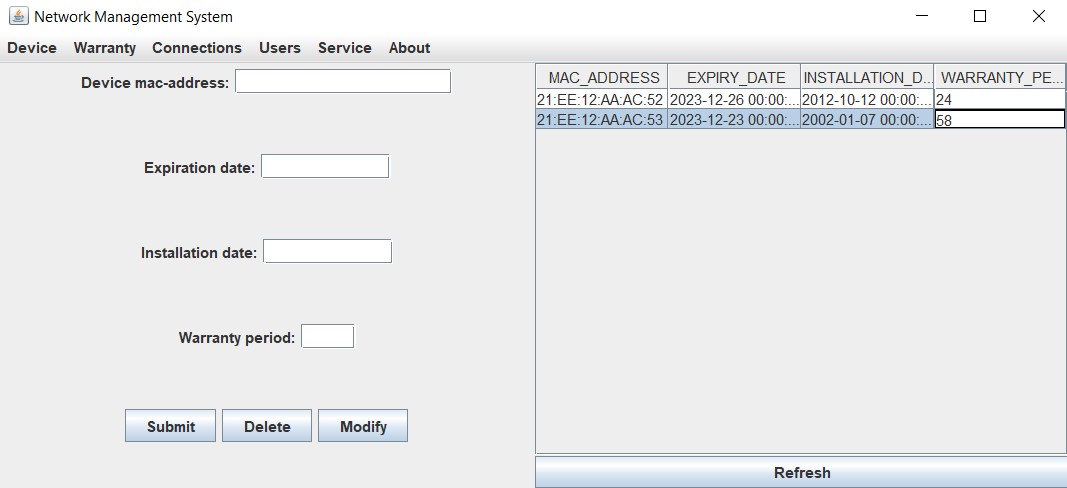


After Insert:

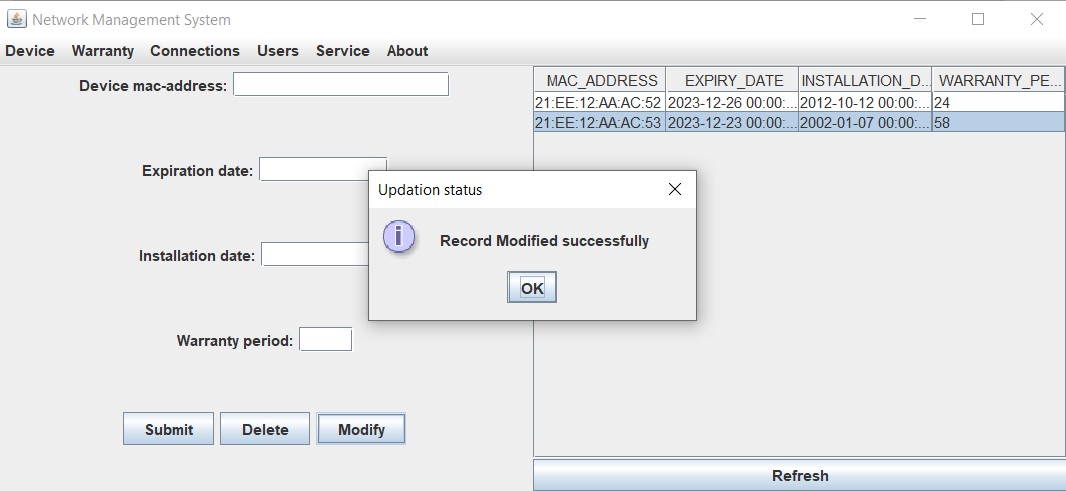


**Updating Warranty Details:**

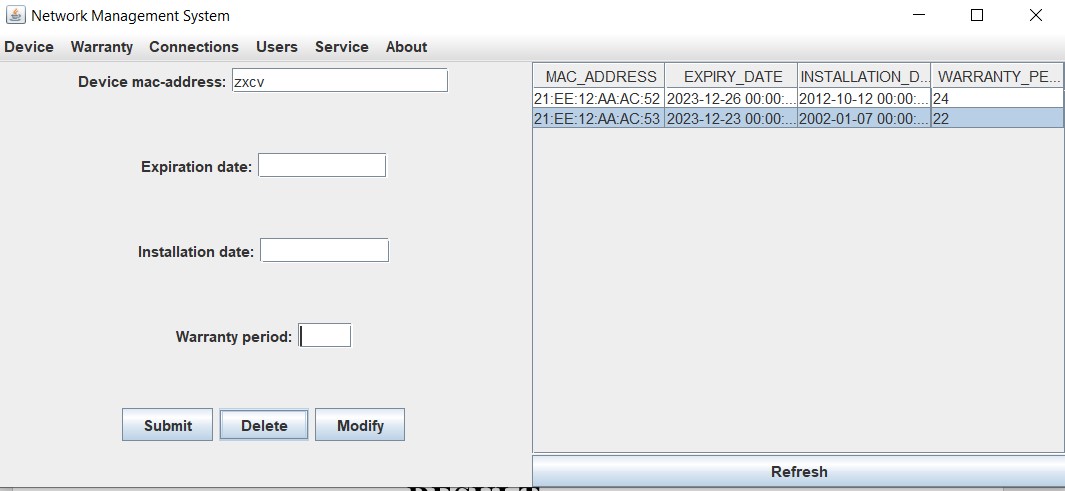
Before Update:



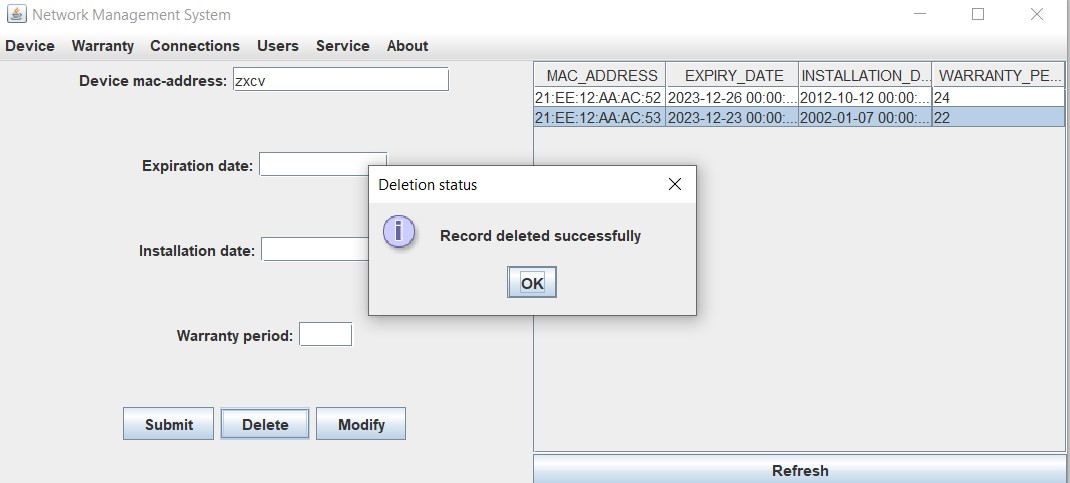
After Update:



**Deleting Warranty Details:**

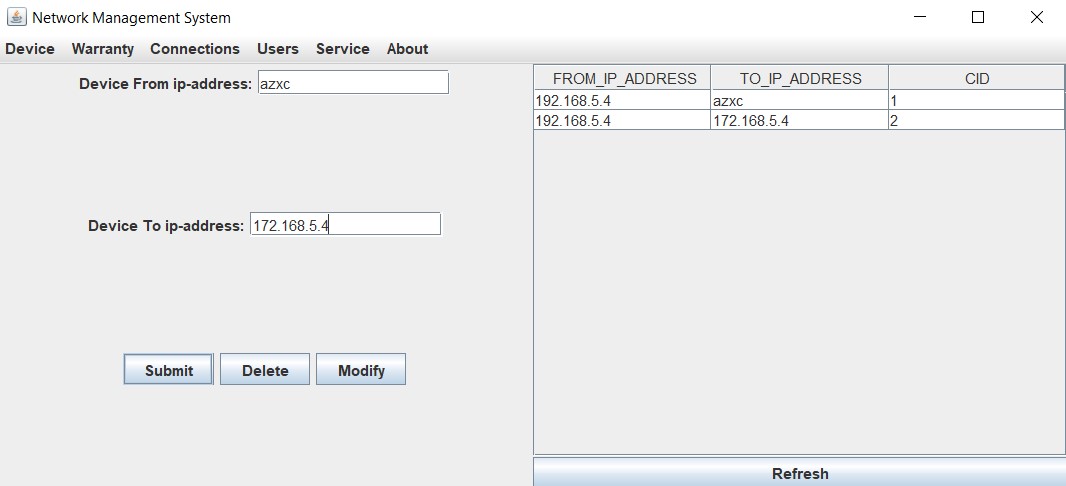
Before Delete:

After Delete:

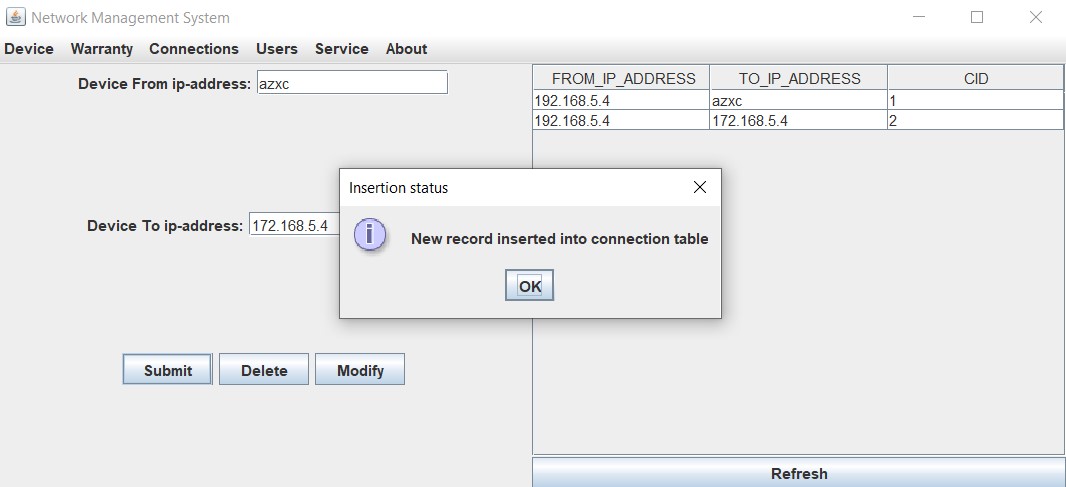


**Inserting Connection Details:**

Before Insert:

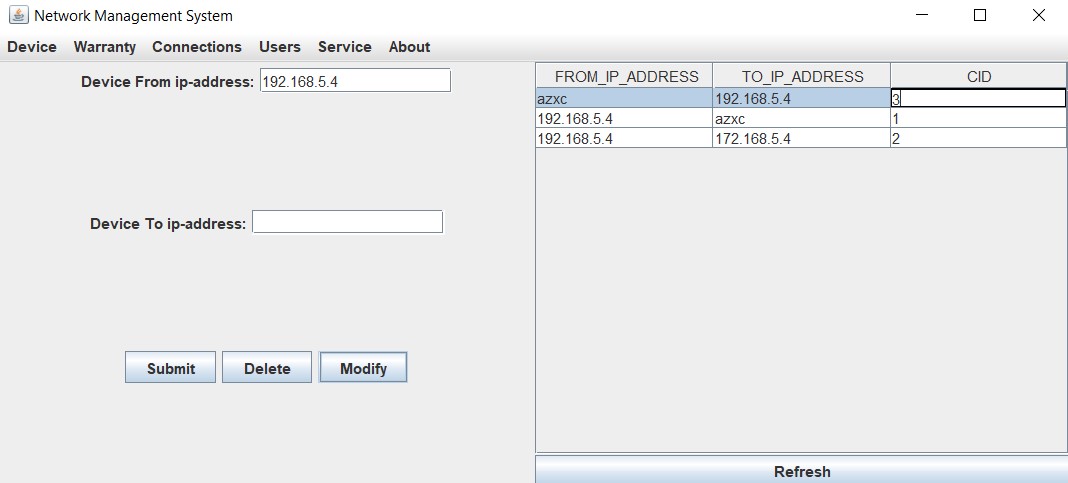


After Insert:



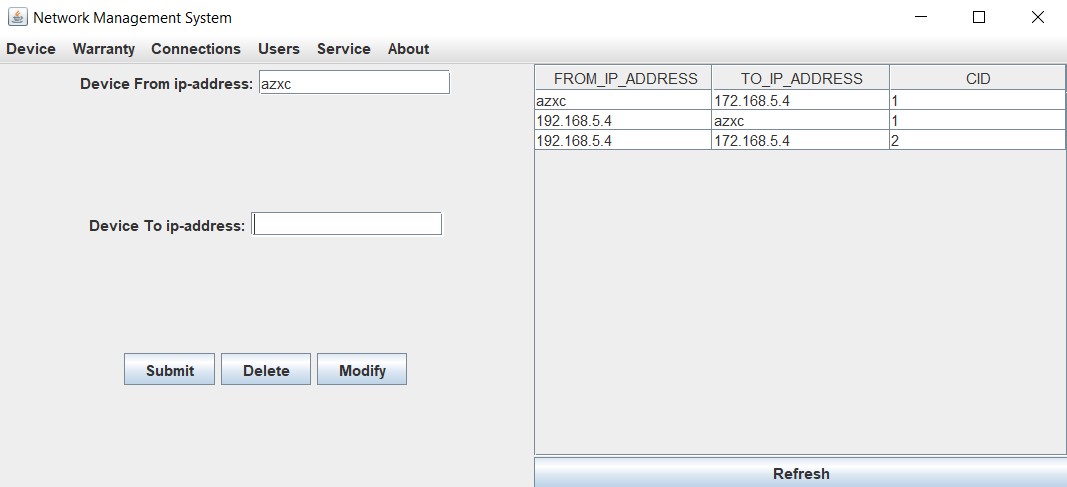
**Updating Connection Details:**

Before Update:

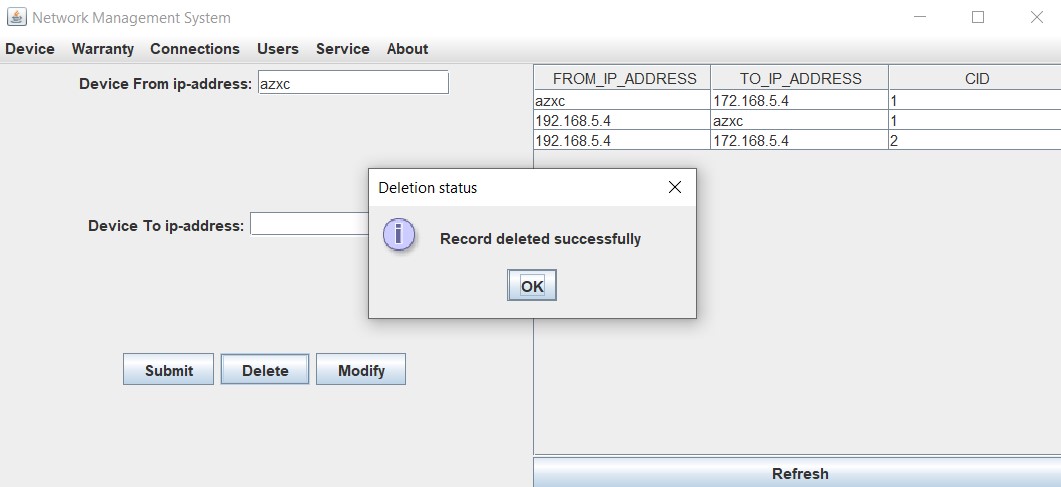


**Delete Connection Details:**

Before Delete:

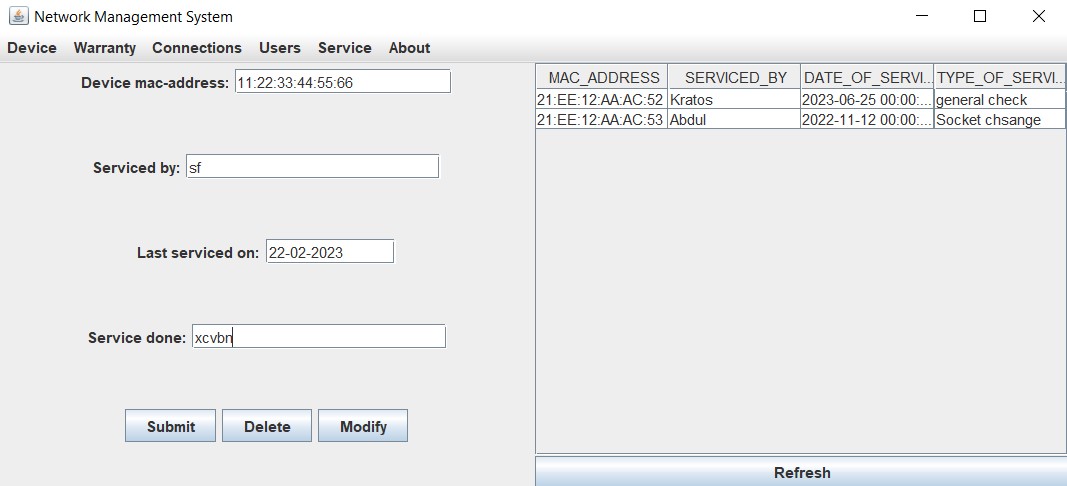


After Delete:

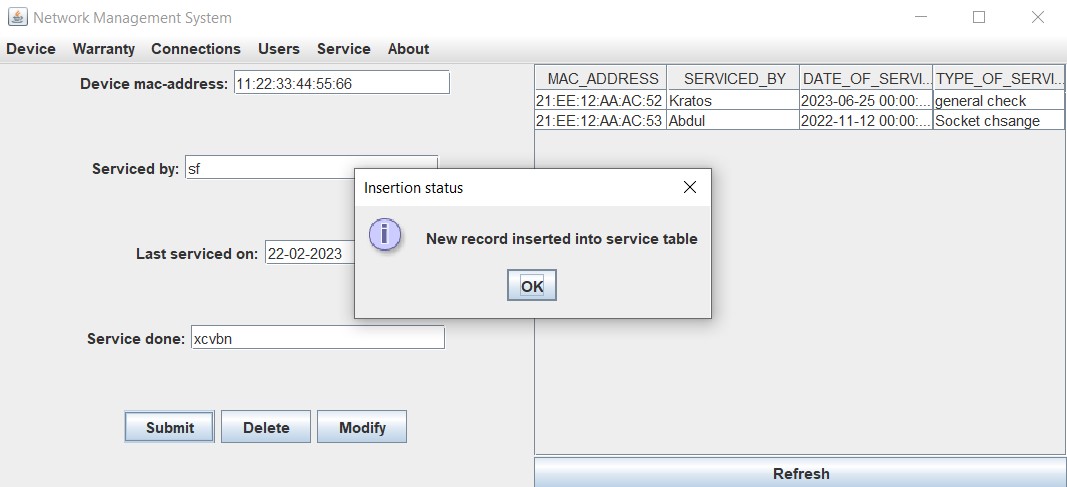


**Insert Service Details:**

Before Score:

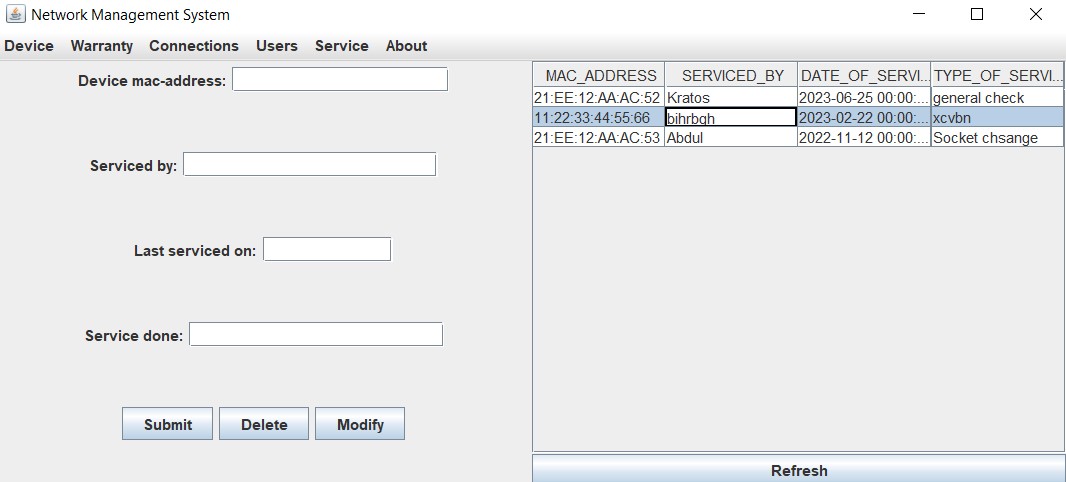


After Score:

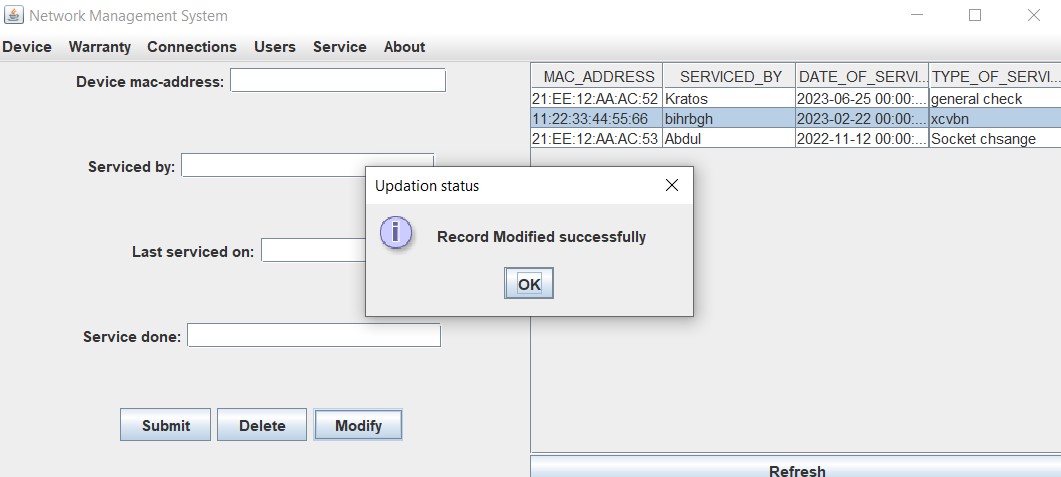


**Update Service Details:**

Before Update:

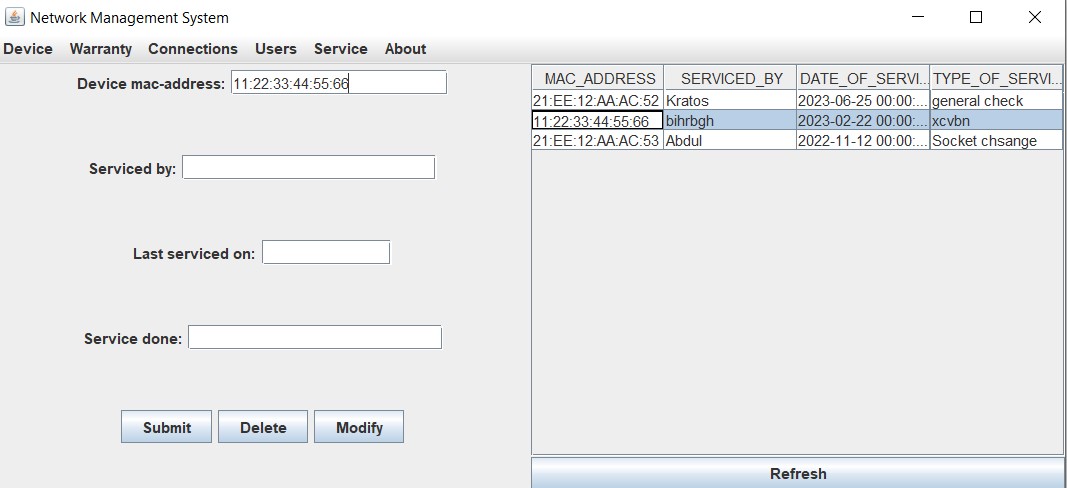


After Update:

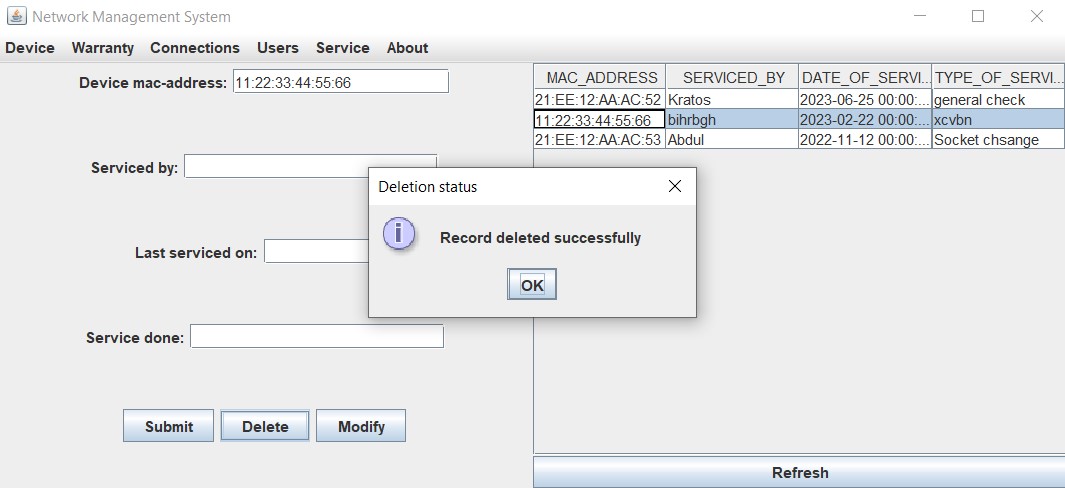


**Delete Service Details:**

Before Delete:

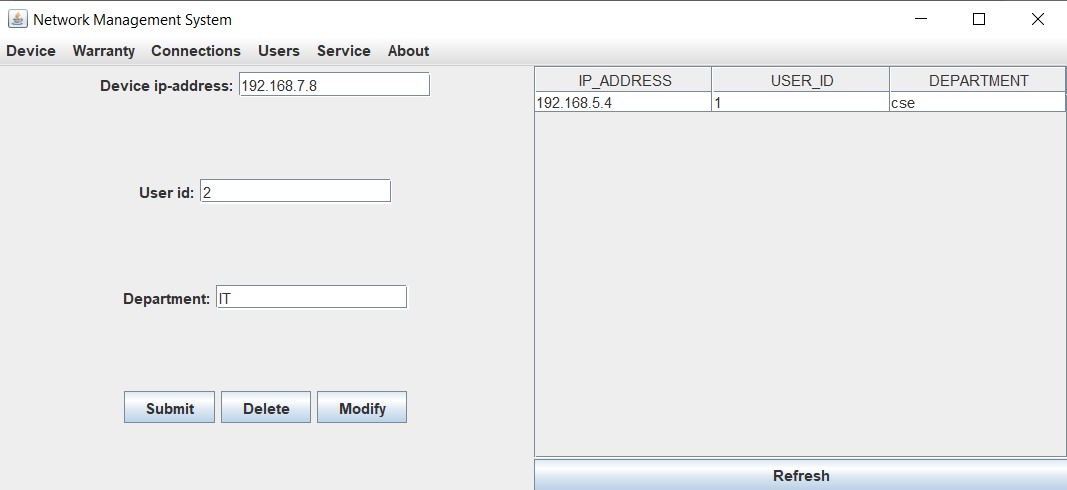


After Delete:

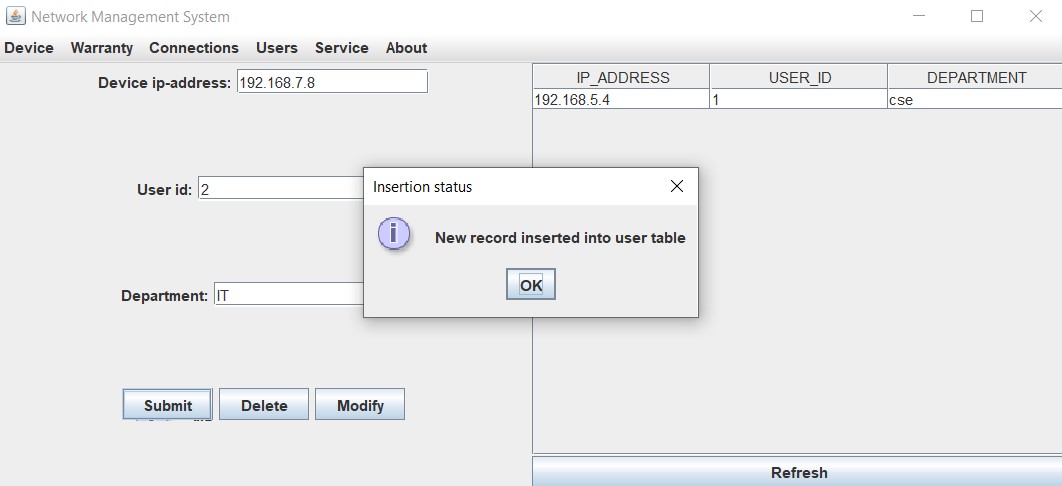


**Insert Users Details:**

Before Insert:

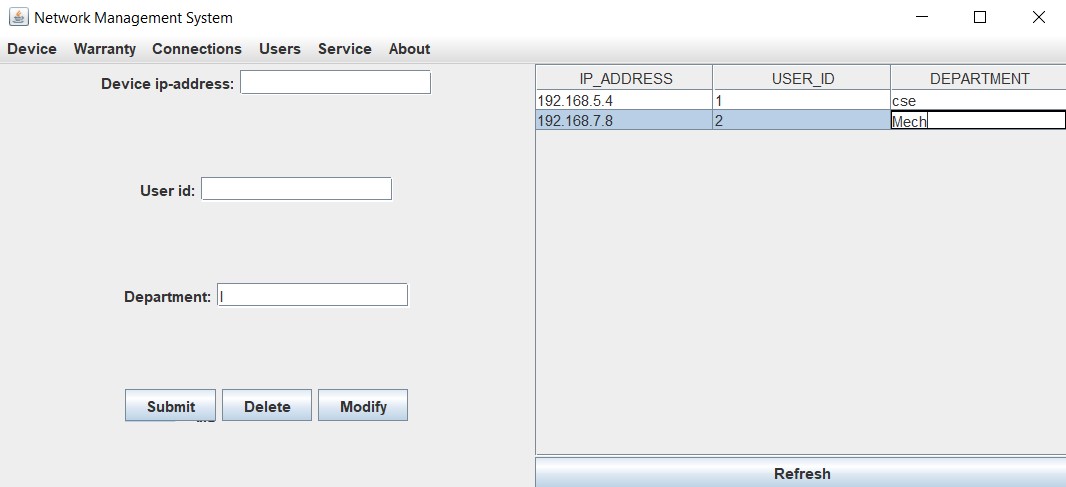


After Insert:

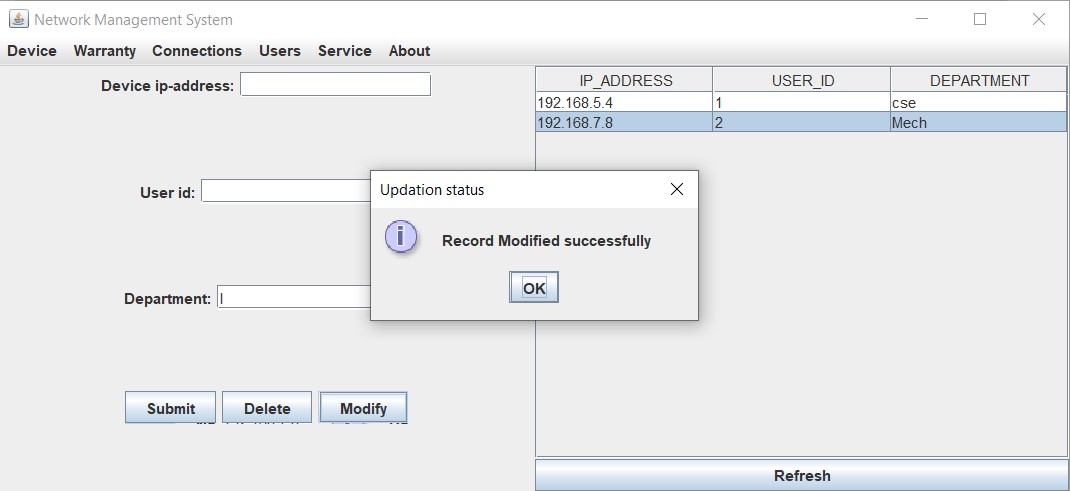


**Update Users Details:**

Before Insert:

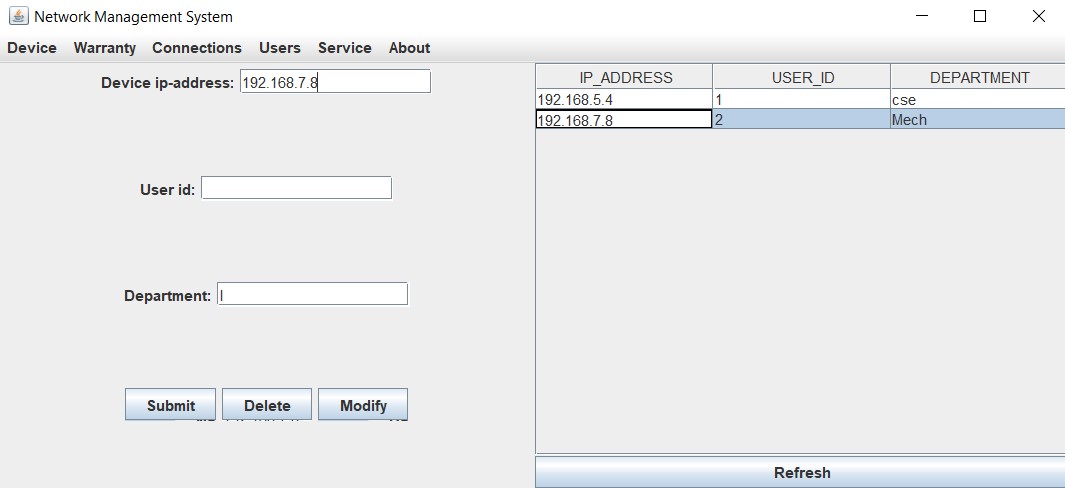


After Insert:

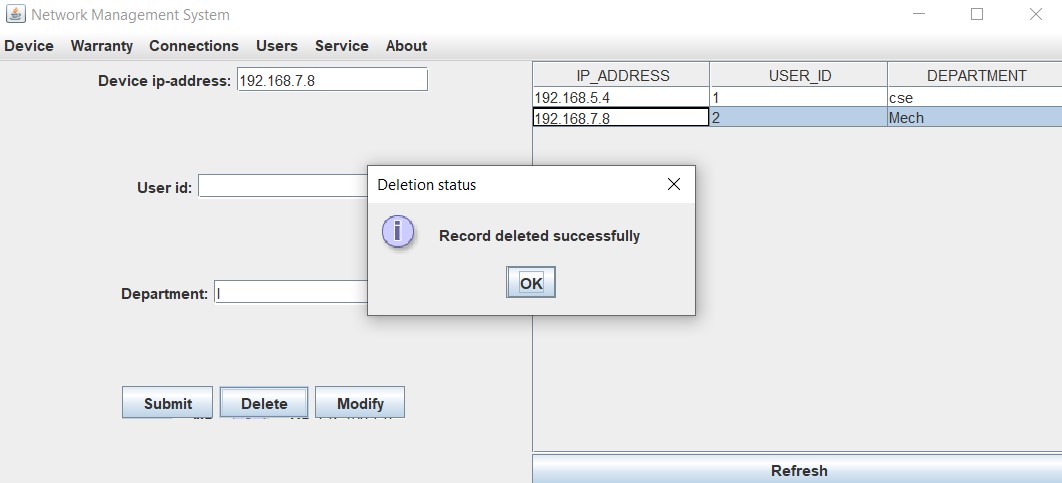


**Delete Users Details:**

Before Insert:



After Insert:



**RESULT**

I have successfully completed my DBMS project ‘***College Network hardware Database***’

**DISCUSSION AND FUTURE WORK**

This project is an application which helps the students access their highest-grade points from all the courses they have been certified for, contains the basic interaction of applying for internships by students and hiring of interns by project managers. It has a very basic user interface. Future scope would be to make the UI more appealing by using graphics. One more feature would be to allow student-users to upload their resumes and official documents required so that the project managers can take a look while they hire them. We can also think of including a feedback system to allow the users to leave their valuable feedback after using this app. Making this feedback to be publicly viewable, would attract many more users to use this app.

**REFERENCES**

● https://docs.oracle.com/javase/7/docs/api/

● <https://www.javatpoint.com/java-swing>

● https://stackoverflow.com/