**JAVA SWING BASED – VCE Network Connection System- SQL CONNECTIVITY USING JDBC**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

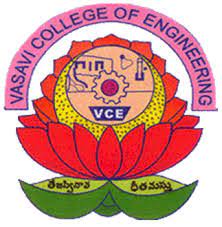
**IN**

**INFORMATION TECHNOLOGY**

**By**

**R.Harini <1602-21-737-017>**

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



**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Ibrahimbagh, Hyderabad-31**

**2022-2023**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled

‘***VCE Network Connection System***’

is a project work of ***R.Harini***

bearing roll no. 1602-21-737-017

who carried out

this project under my supervision

in the IV semester

of the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

**VCE NETWORK CONNECTION MANAGEMENT**

**SYSTEM**

**SUBJECT :** DATABASE MANAGEMENT SYSTEM

**ABSTRACT**

The proposed DBMS project aims to develop a system for managing the network connections of Vasavi College of Engineering located in Ibrahimbagh, Hyderabad. The college has a large number of students and faculty members who require internet access for their academic and research work. However, the current system for managing network connections is outdated and ineffective. The new system will use a database to manage the network connections of users and devices within the college. It will allow for efficient management of user accounts, device registrations, and network usage monitoring. The system will also include features for troubleshooting network issues and providing real-time updates on network status .The proposed system will improve the quality of network services provided to users, reduce downtime due to network issues, and ensure better utilization of network resources. It will also enhance the security of the network by providing a centralized system for managing user access and monitoring network activity. The system will be built using a relational database management system (RDBMS), with MySQL being the preferred choice. The system will also be designed to be scalable, able to handle an increasing number of users and devices as the college grows .In addition to managing network connections, the system will also provide valuable data insights through its reporting and analytics features. It will generate reports on network usage, bandwidth consumption, and device connectivity, allowing for informed decision-making in network management and infrastructure planning. Overall, the proposed system will be a valuable addition to the college's IT infrastructure, providing a robust and efficient solution for managing network connections.

**Requirments Analysis**

**List of Tables:**

1. Departments
2. Users
3. Devices
4. Connection Types
5. Network Connections

**List of Attributes :**

**1. Departments Table:**

**Attributes:**

**-**departmentID

-departmentName

-HODName

Domain Types:

-departmentID:number

-departmentName,HODName:varchar

Constraints:

-departmentName:unique key

-departmentID-primary key

**2.Users Table:**

Attributes:

-userID

-fname

-lname

-email

-phonenumber

-role

-departmentID

Domain Types:

-userID-integer

-fname,lname,email,phonenumber,role-varchar

**Constraints:**

**-**email:unique key

-userID-primary key

-departmentID-foreign key

**3.Devices Table:**

Attributes:

* DID
* Dname
* DType
* MACAddress
* IPAddress
* Subnetmask
* gatewayIP
* DNSServerIP

Domain Type:

-DID:integer

-Dname,MACAddress,IPAddress,subnetmask,gatewayIP,DNSServerIP:varchar

**Constraints:**

**-**MACAddress,IPAddress:unique key

-DID:primary key

**4.ConnectionTypes Table:**

Attributes:

-connectionTypeID

-connectionTypeName

Domain Types:

-connectionTypeID:number

-connectionTypeName:varchar

Constraints:

-connectionTypeName:unique key

-connectionTypeID:primary key

5.**Network Connections Table:**

**Attribute:**

**-**userID

-DID

-connectionID

-connectionTypeID

-connectionStatus

Domain Type:

-userID,DID,connectionID,connectionTypeID:Integer

-connectionstatus:varchar

Constraints:

-connectionID:primary key

-userID,DID,connectionTypeID:foreign key

**DATABASE DESIGN**

**Mapping Cardinalities:**

1. One user can have many network connections(1:M).

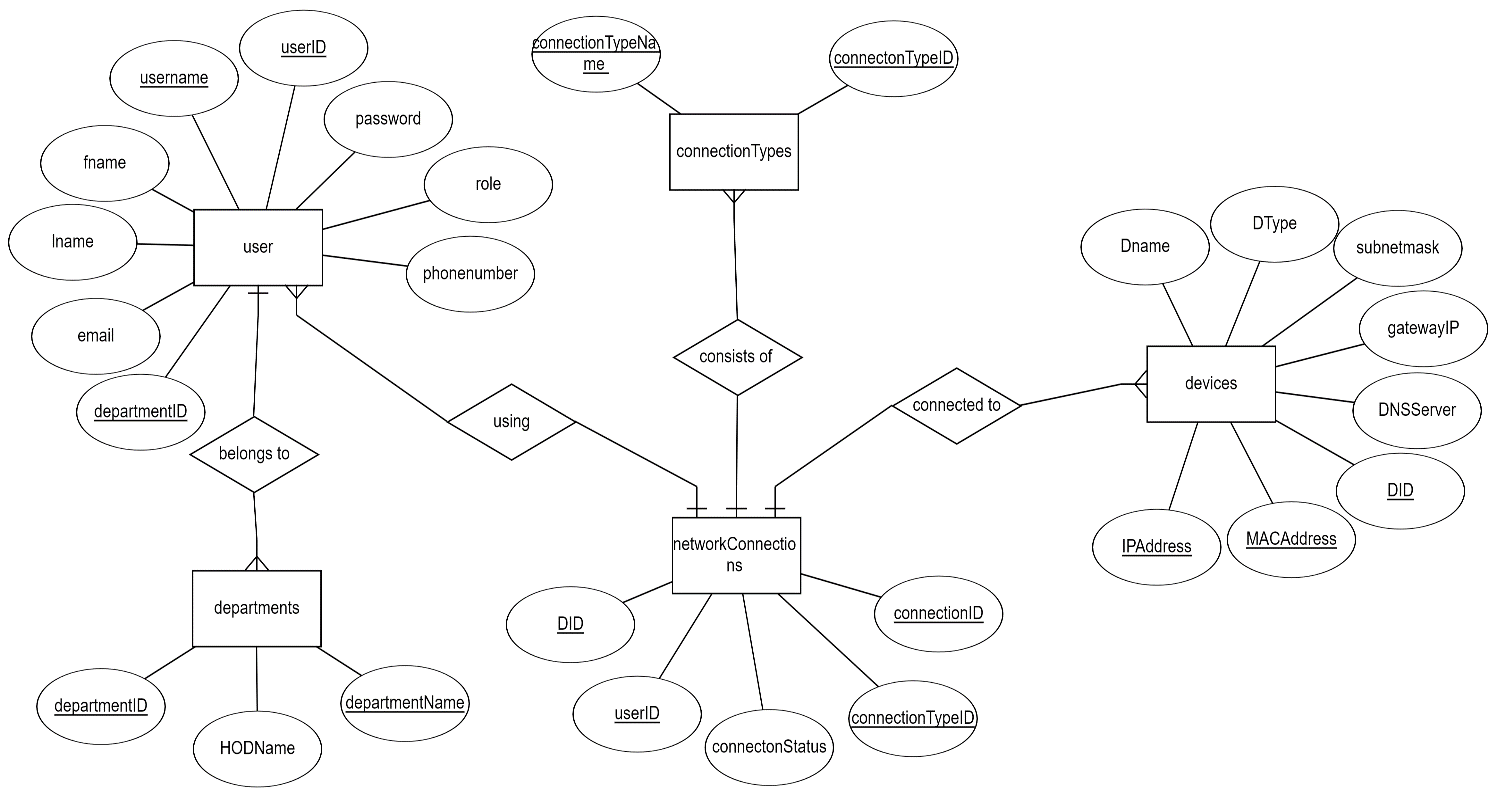
**2.**One device can have many network connections(1:M).

3. One connection type can have many network connections(1:M).

4.One department can have many users(1:M).

5. One network connection can be associated with only one user, device, and connection type (1:1).

**Entity Relationship Diagram**

****

**DDL Commands:**

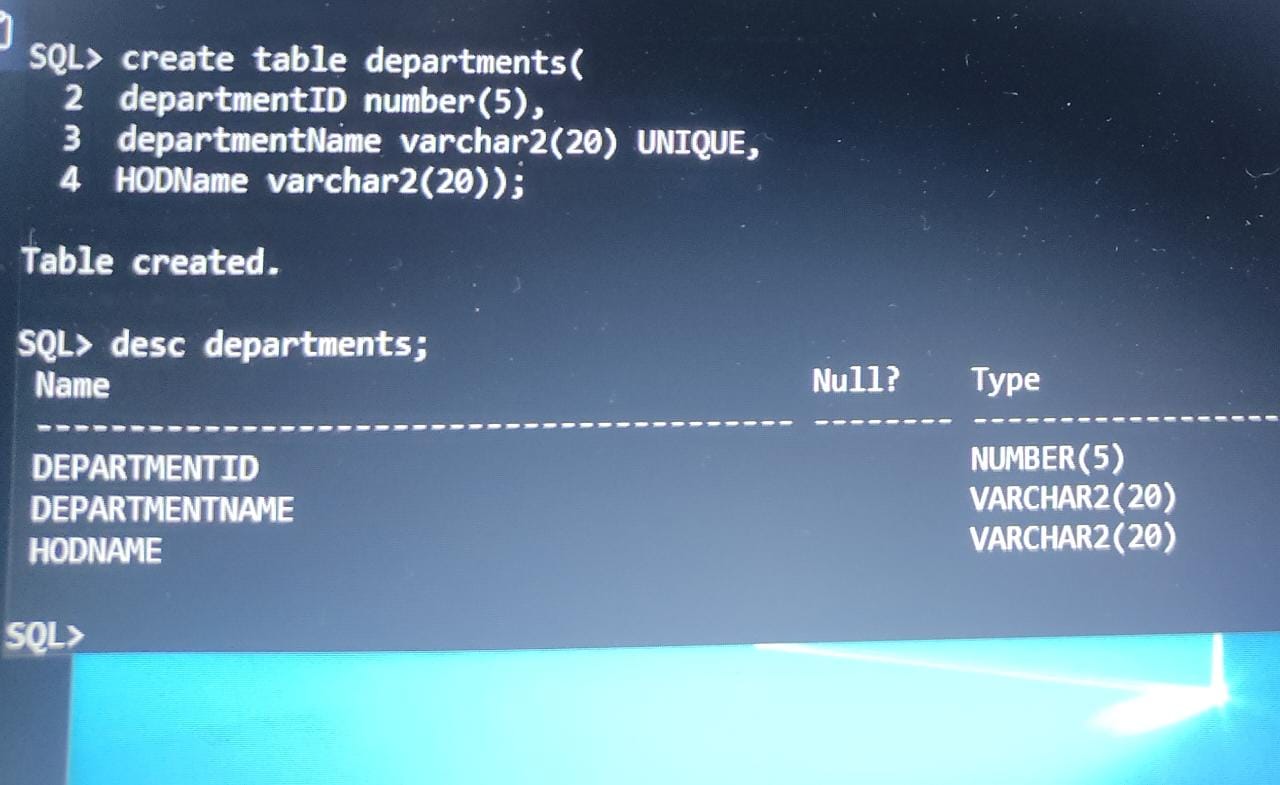
**Table 1** :**Departments**:

SQL> create table departments(

2 departmentID number(5),

3 departmentName varchar2(20) UNIQUE,

4 HODName varchar2(20));



**Table 2**: Users:

SQL> create table users(

2 userID number(5),

3 fname varchar2(20),

4 lname varchar2(20),

5 email varchar2(20) UNIQUE,

6 phonenumber varchar2(20),

7 username varchar2(20) UNIQUE,

8 password varchar2(20),

9 role varchar2(20),

10 departmentID number(5),

11 primary key (userID),

12 foreign key (departmentID) references departments on delete cascade);



**Table 3**: Devices:

SQL> create table devices(

2 DID number(5),

3 Dname varchar2(20),

4 DType varchar2(20),

5 MACAddress varchar2(20) UNIQUE,

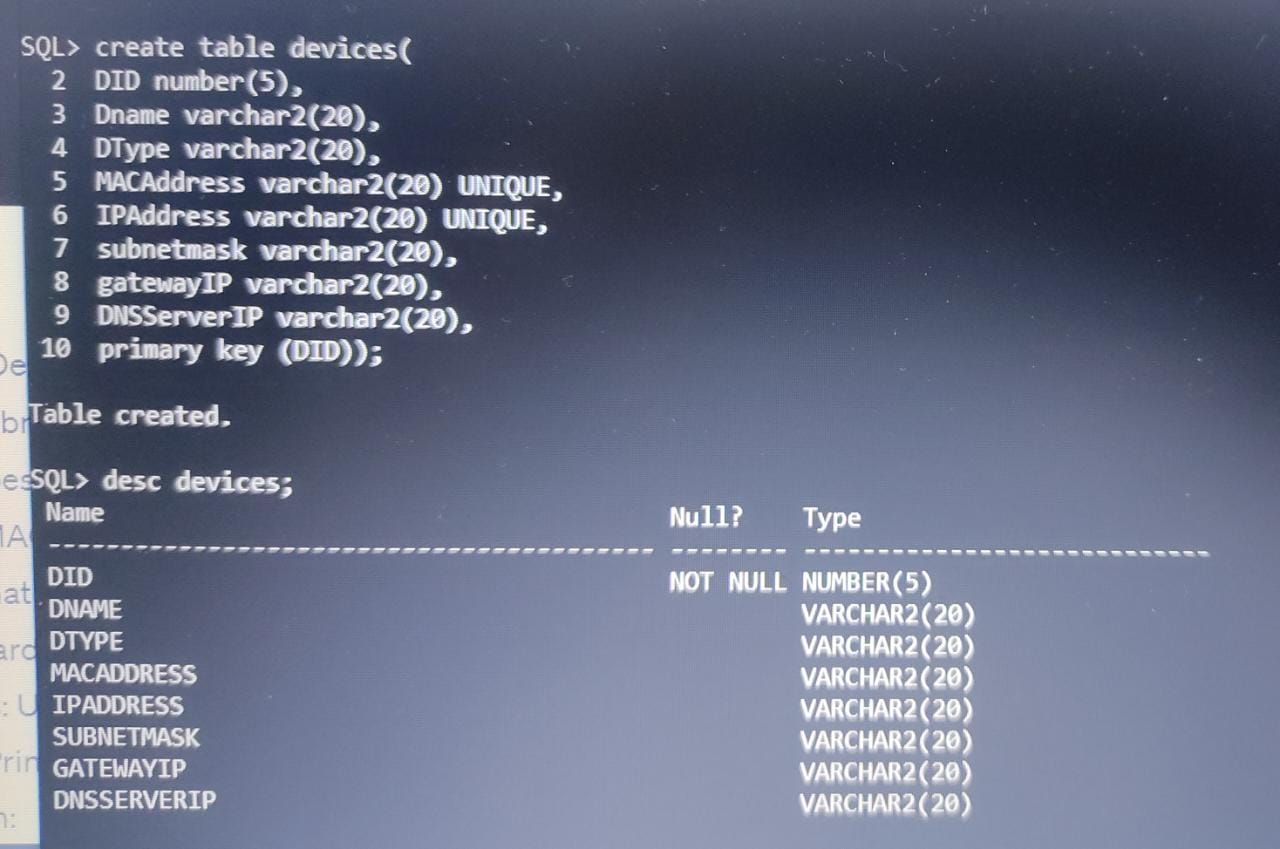
6 IPAddress varchar2(20) UNIQUE,

7 subnetmask varchar2(20),

8 gatewayIP varchar2(20),

9 DNSServerIP varchar2(20),

10 primary key (DID));



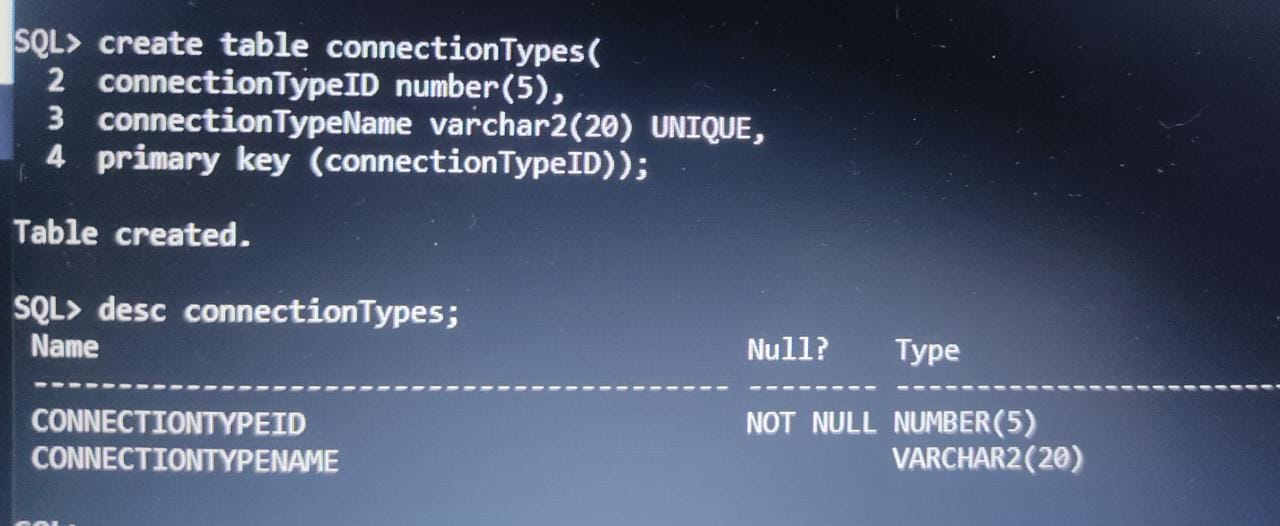
**Table 4**:**ConnectionsTypes**:

SQL> create table connectionTypes(

2 connectionTypeID number(5),

3 connectionTypeName varchar2(20) UNIQUE,

4 primary key (connectionTypeID));



**Table 5**: **NetworkConnections**:

SQL> create table networkConnections(

2 userID number(5),

3 DID number(5),

4 connectionID number(5),

5 connectionTypeID number(5),

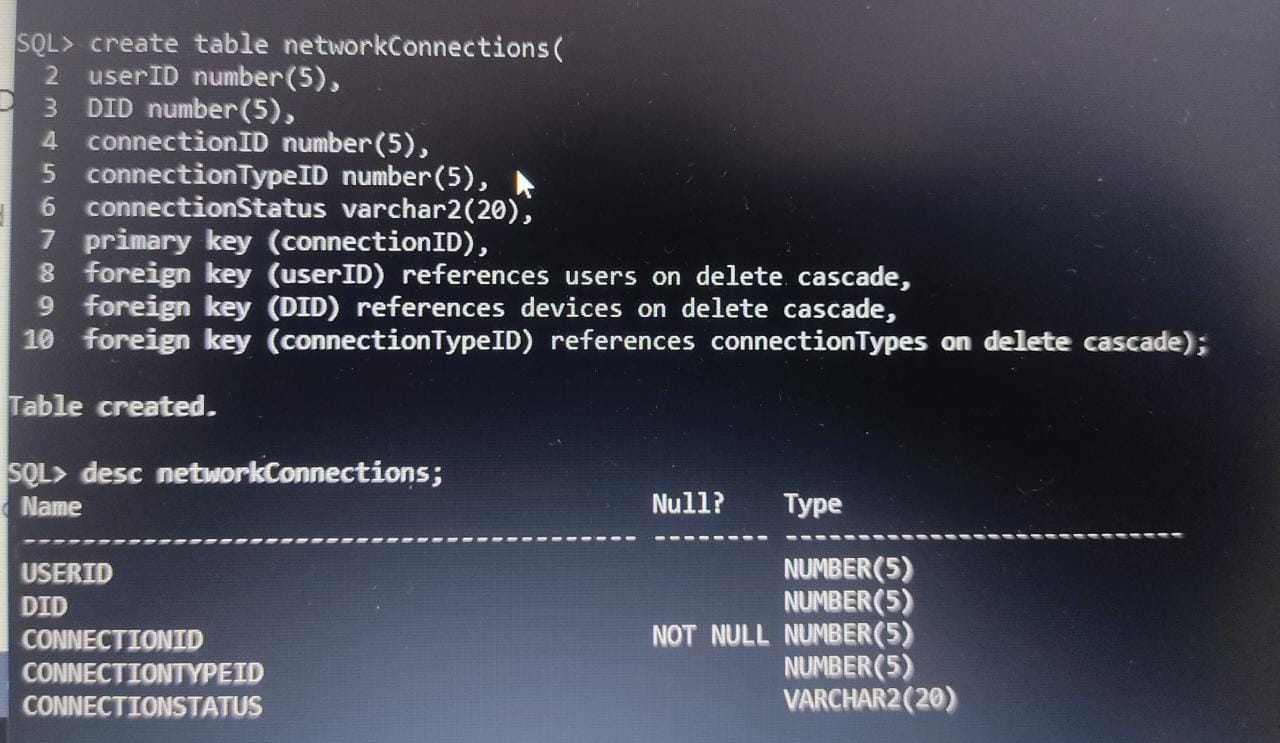
6 connectionStatus varchar2(20),

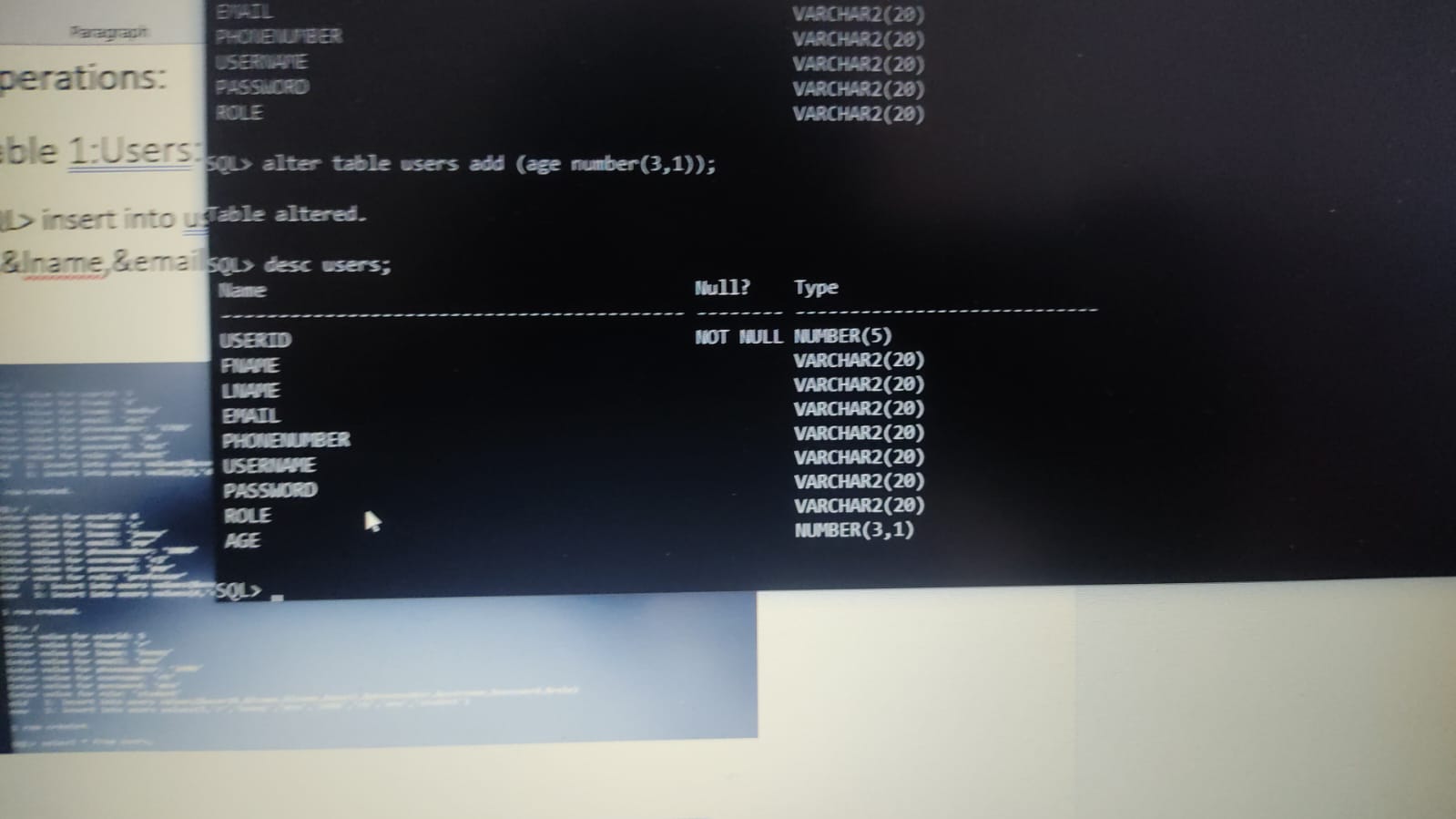
7 primary key (connectionID),

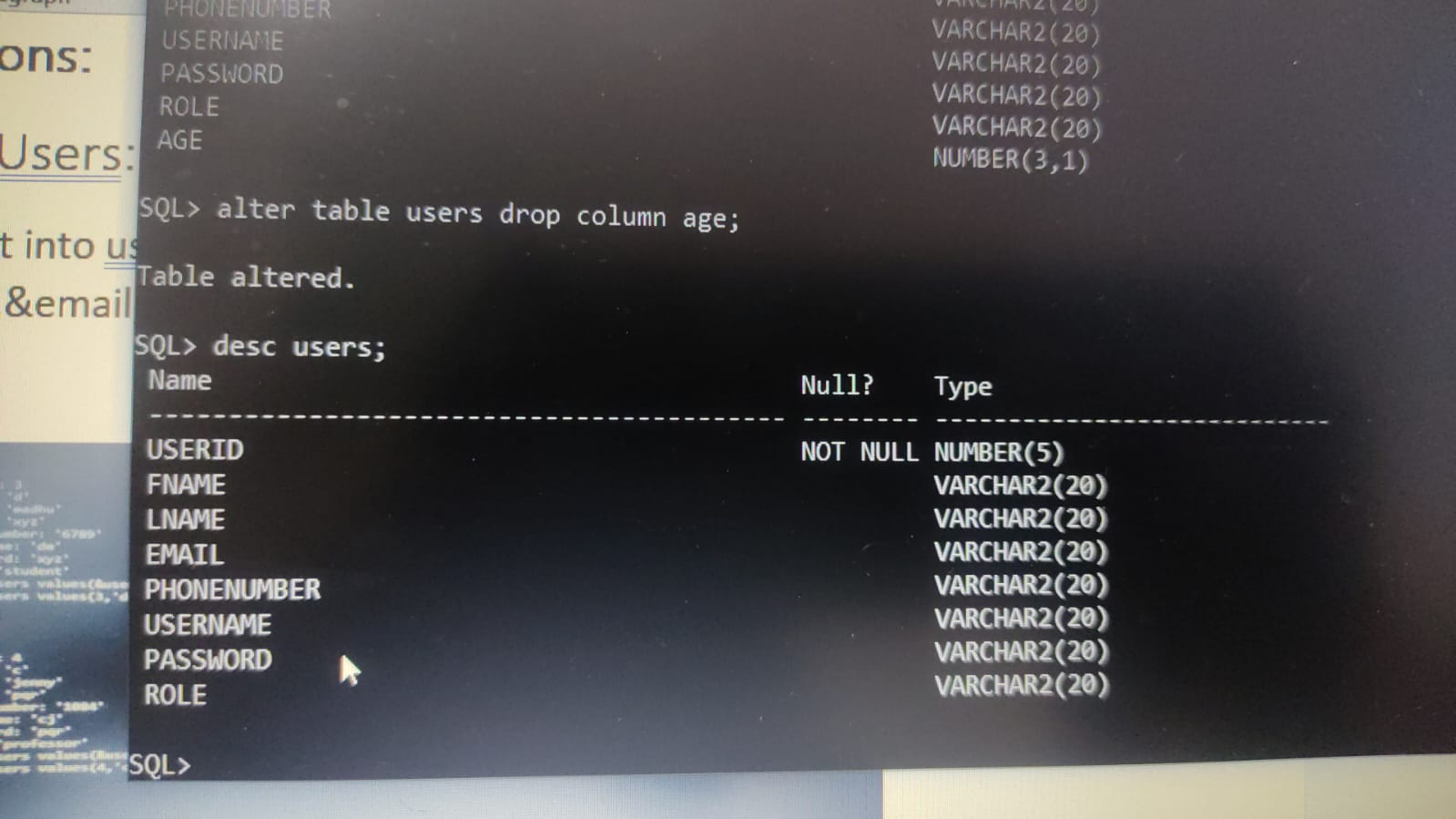
8 foreign key (userID) references users on delete cascade,

9 foreign key (DID) references devices on delete cascade,

10 foreign key (connectionTypeID) references connectionTypes on delete cascade);



****

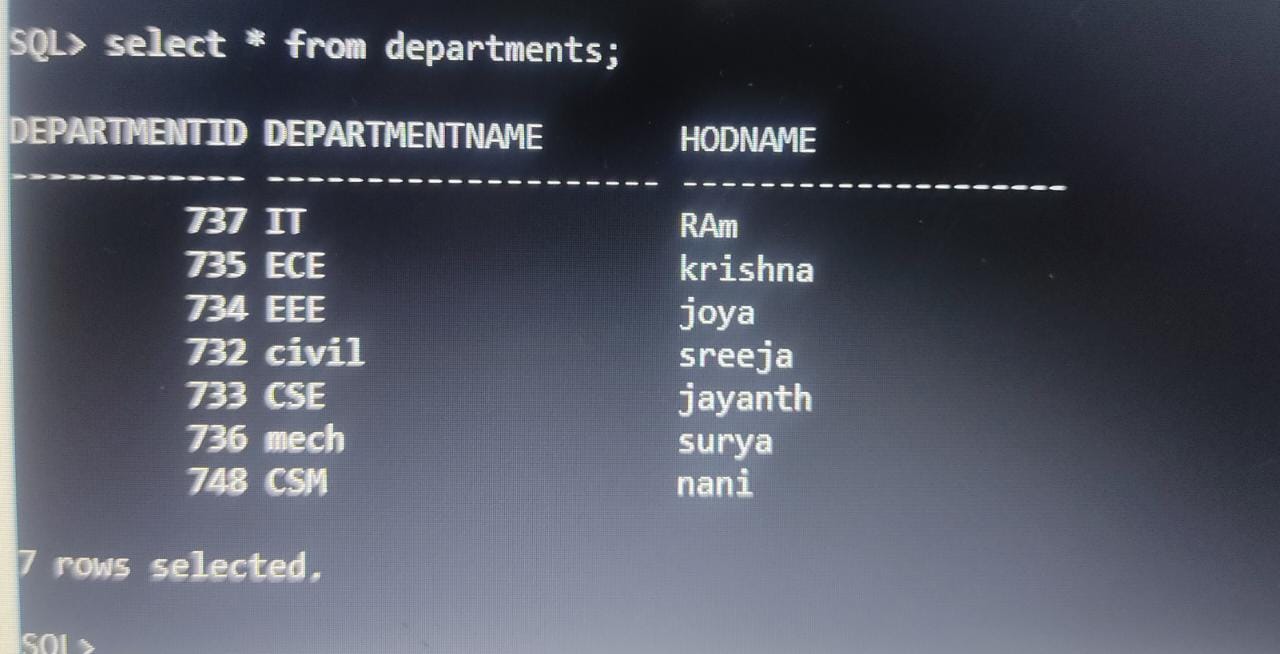
****

**DML Operations:**

Table 1:Departments:

SQL> insert into departments values(&departmentID,

‘&departmentName’,’&HODName’);



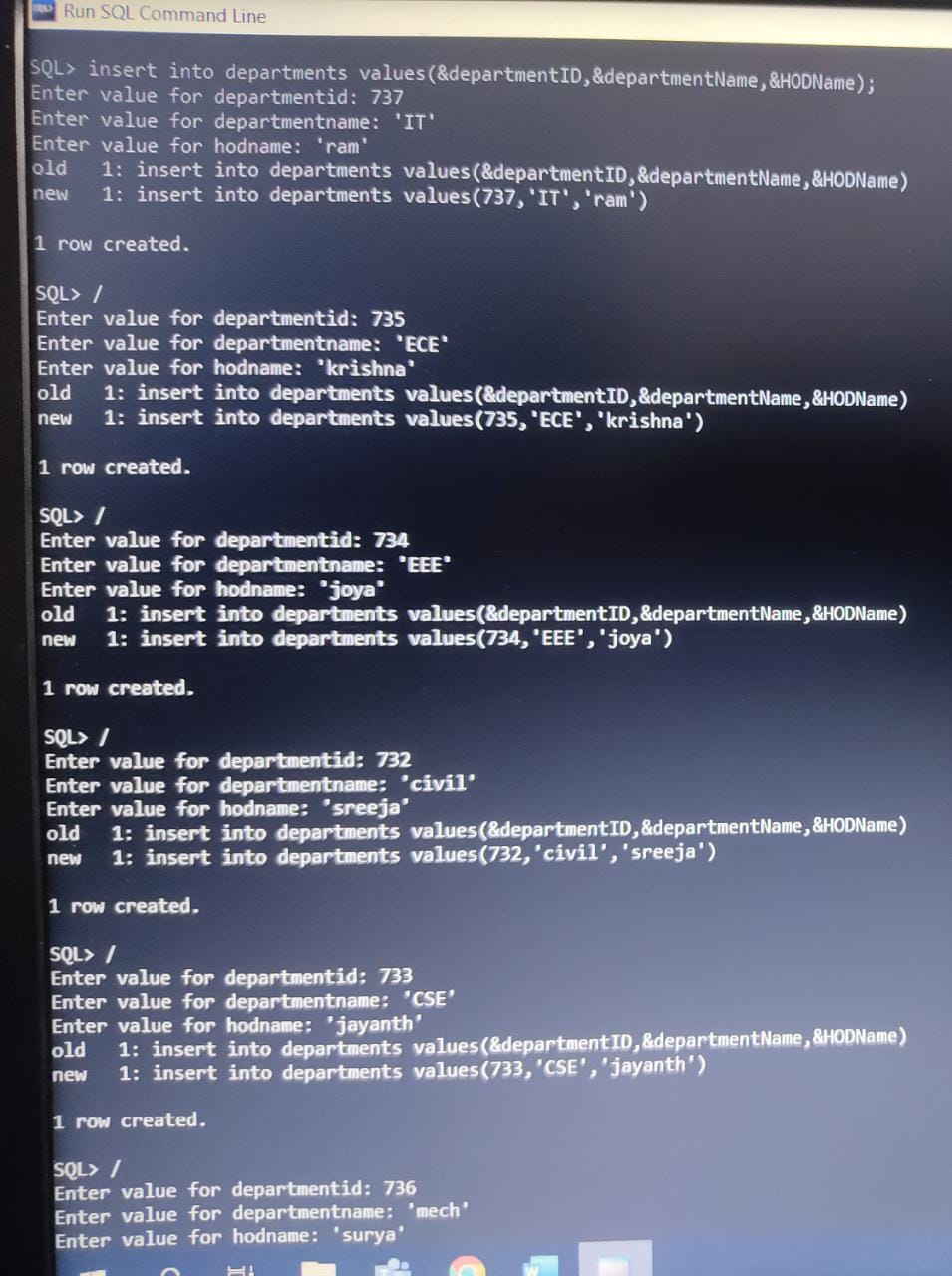
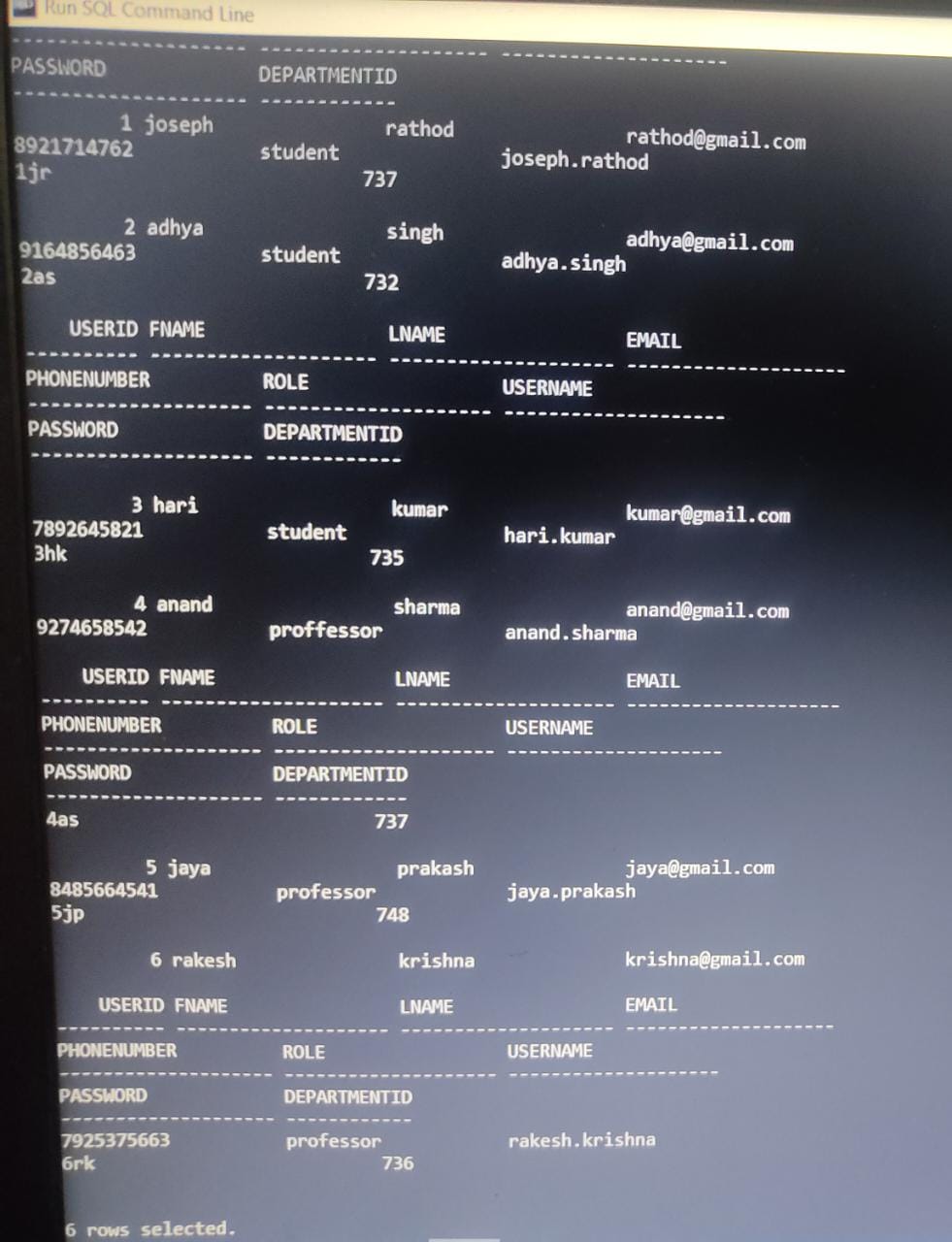


Table 2:Users:

SQL> insert into users values(&userID,& fname’,'&lname'

,'&email','&phonenumber','&role','&username','&password','&departmentID');



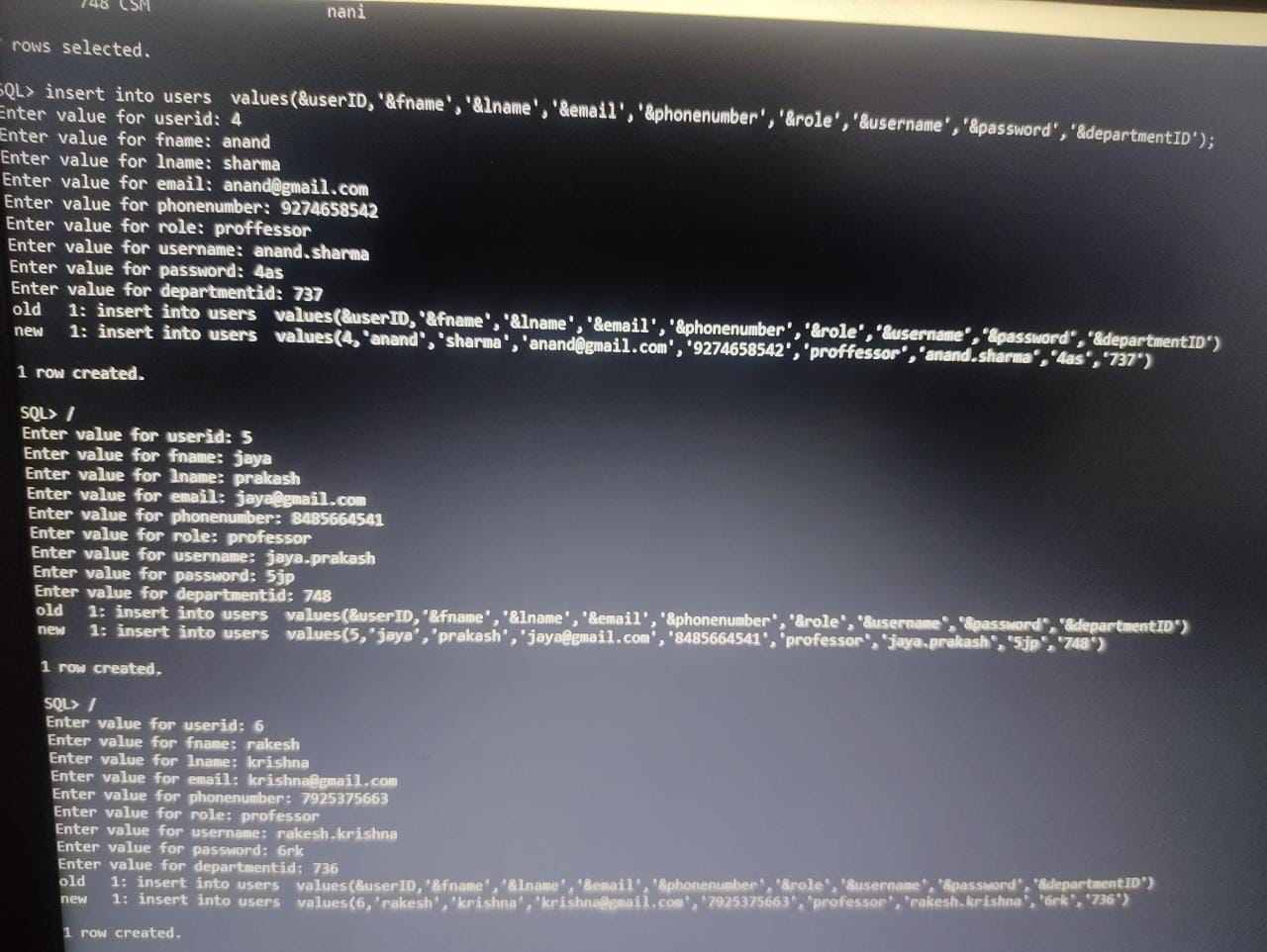
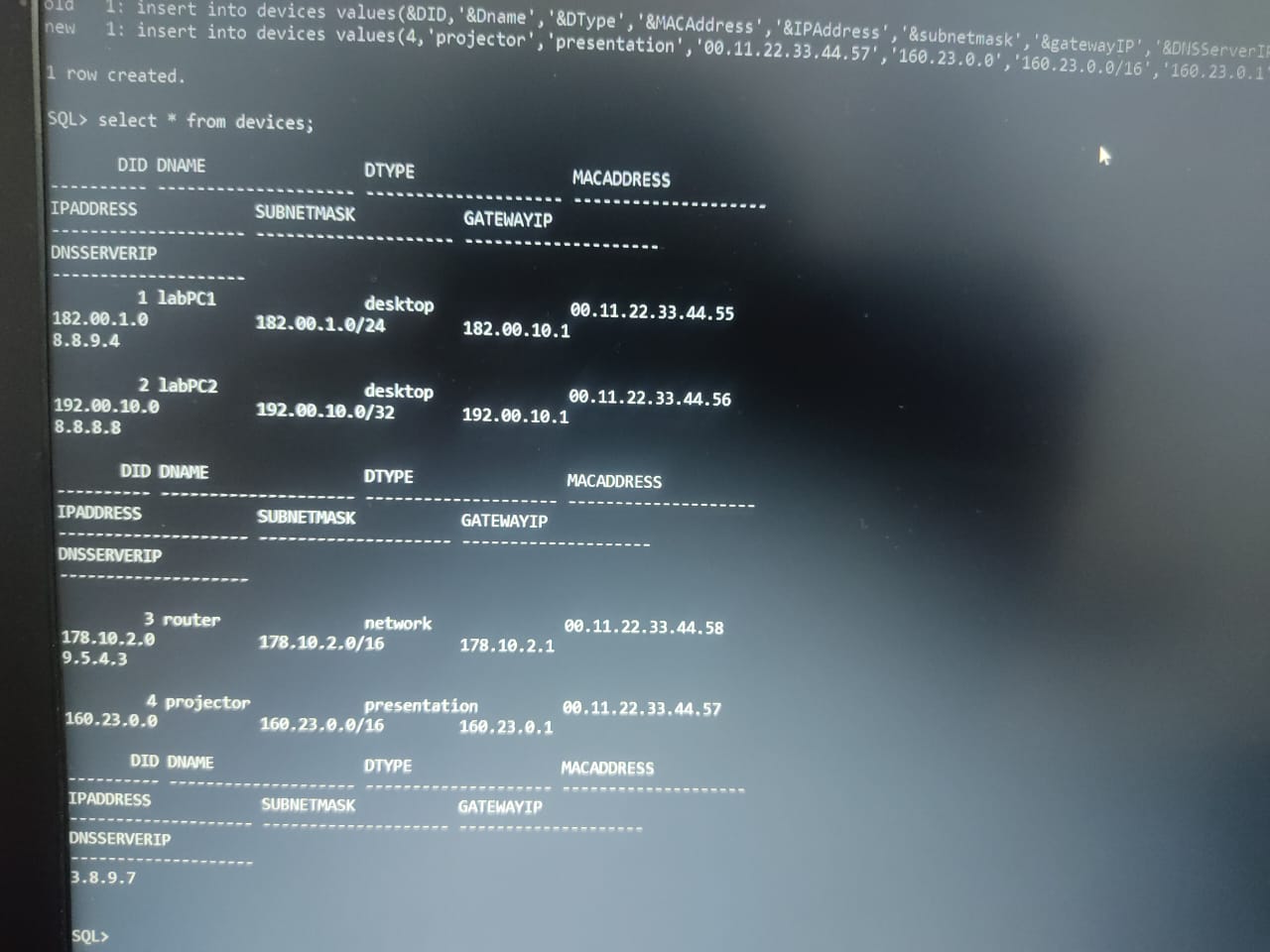


Table 3:Devices:

SQL> insert into devices values(&DID, ‘&Dname’,

&DType’,’&MACAddress’,’&IPAddress’,’&subnetmask’,’&gatewayIP’); 

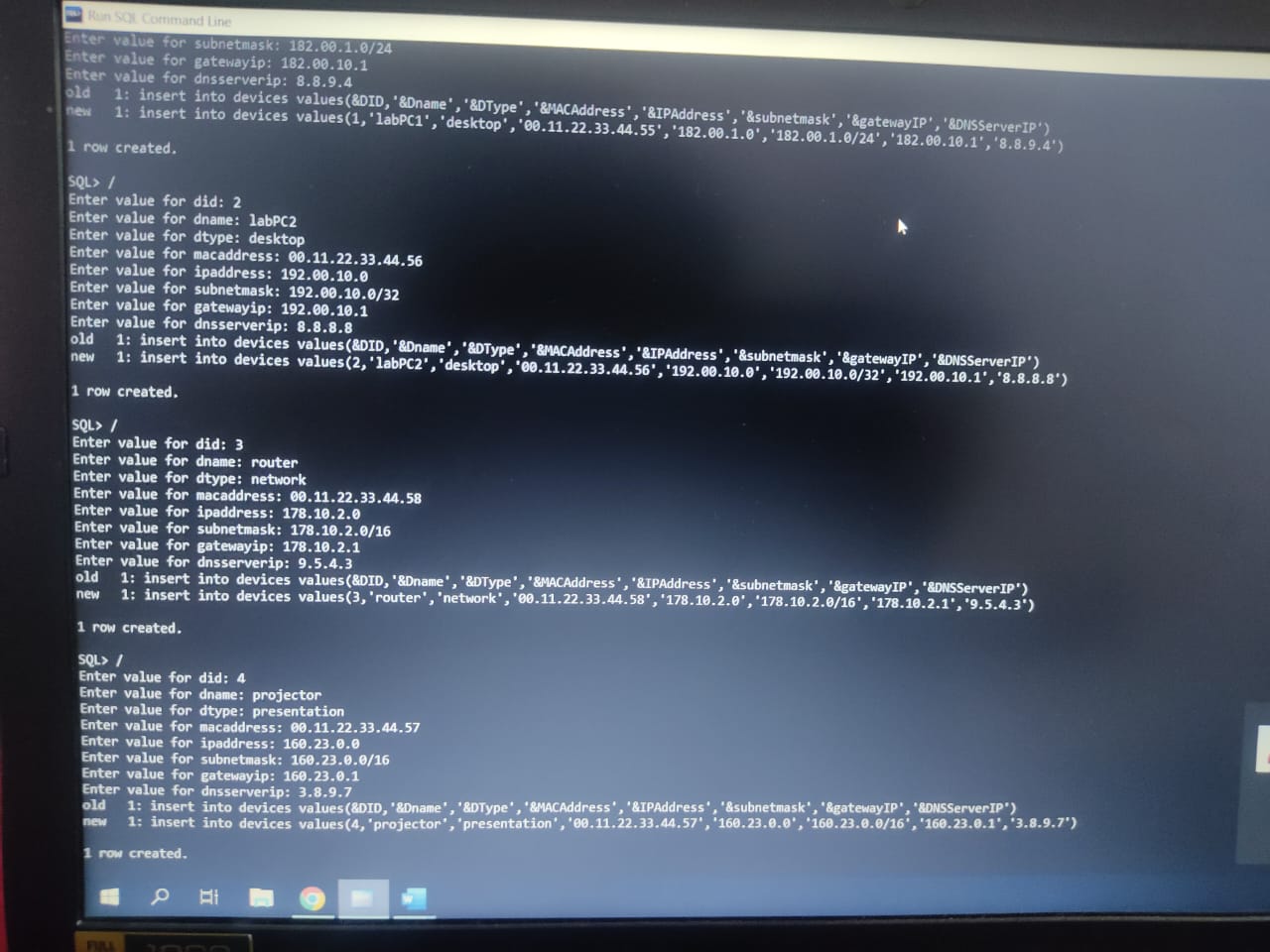


Table 4:ConnectionTypes:

SQL> insert into connectionTypes values(&connectionTypeID,

&connectionTypeName);

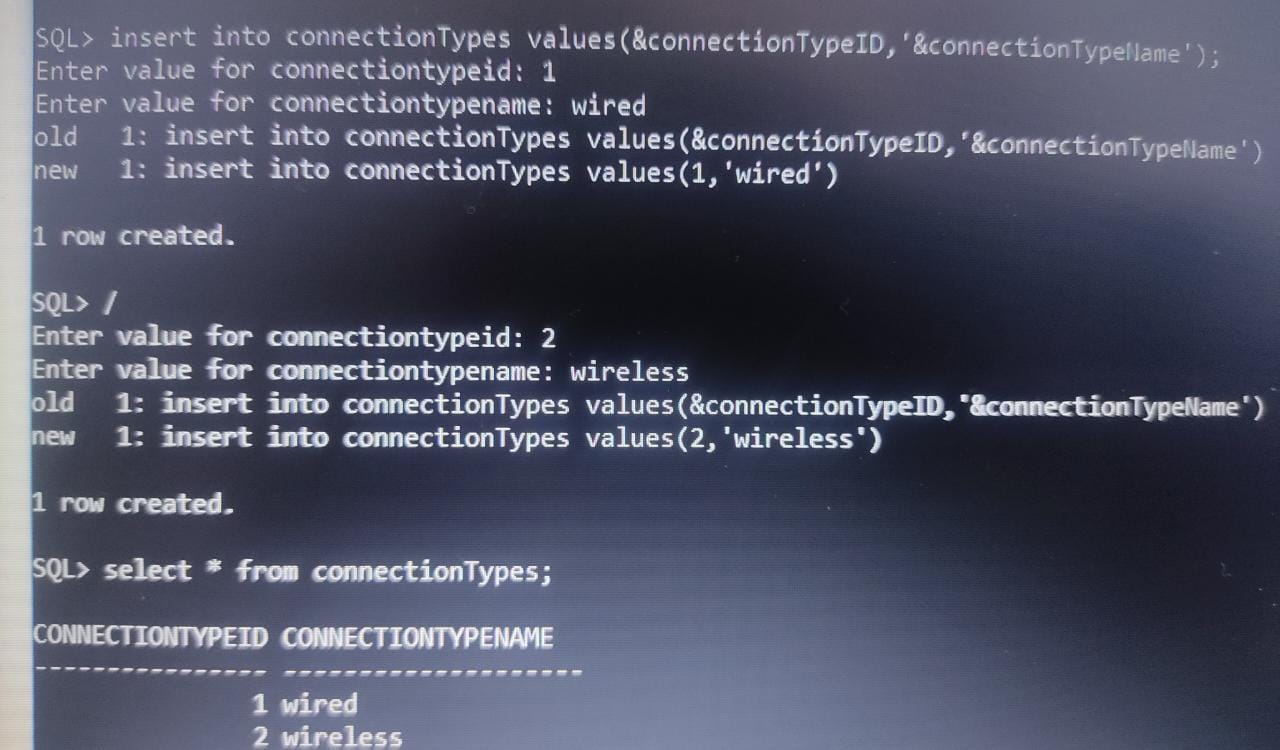
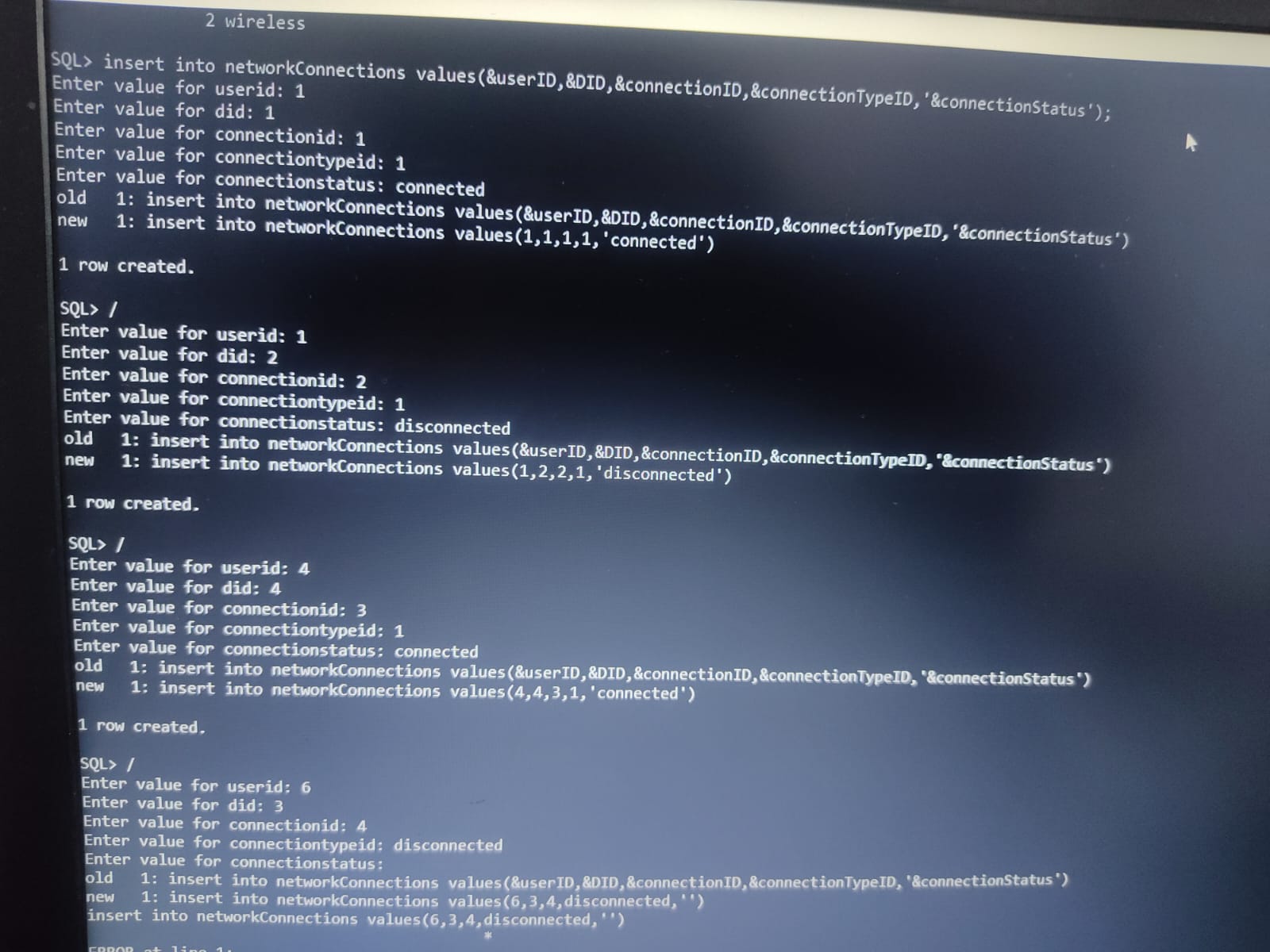
s

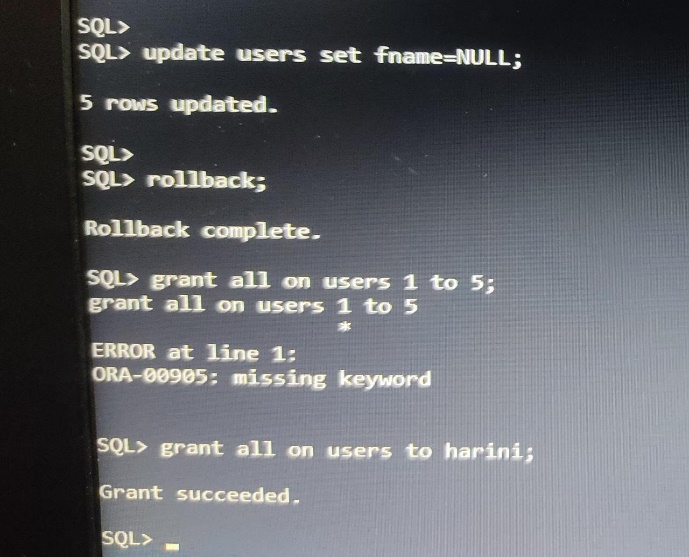
Table 5:NetworkConnections:

SQL> insert into networkConnections values(&userID,&DID,

&connectionID,&connectionTypeID,&connectionStatus);







**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:

package main;

import java.sql.\*;

public class ConnectionManager (

private static String url = "jdbc:oracle: thin:@localhost:1521:xe"; private static String username = "harini";

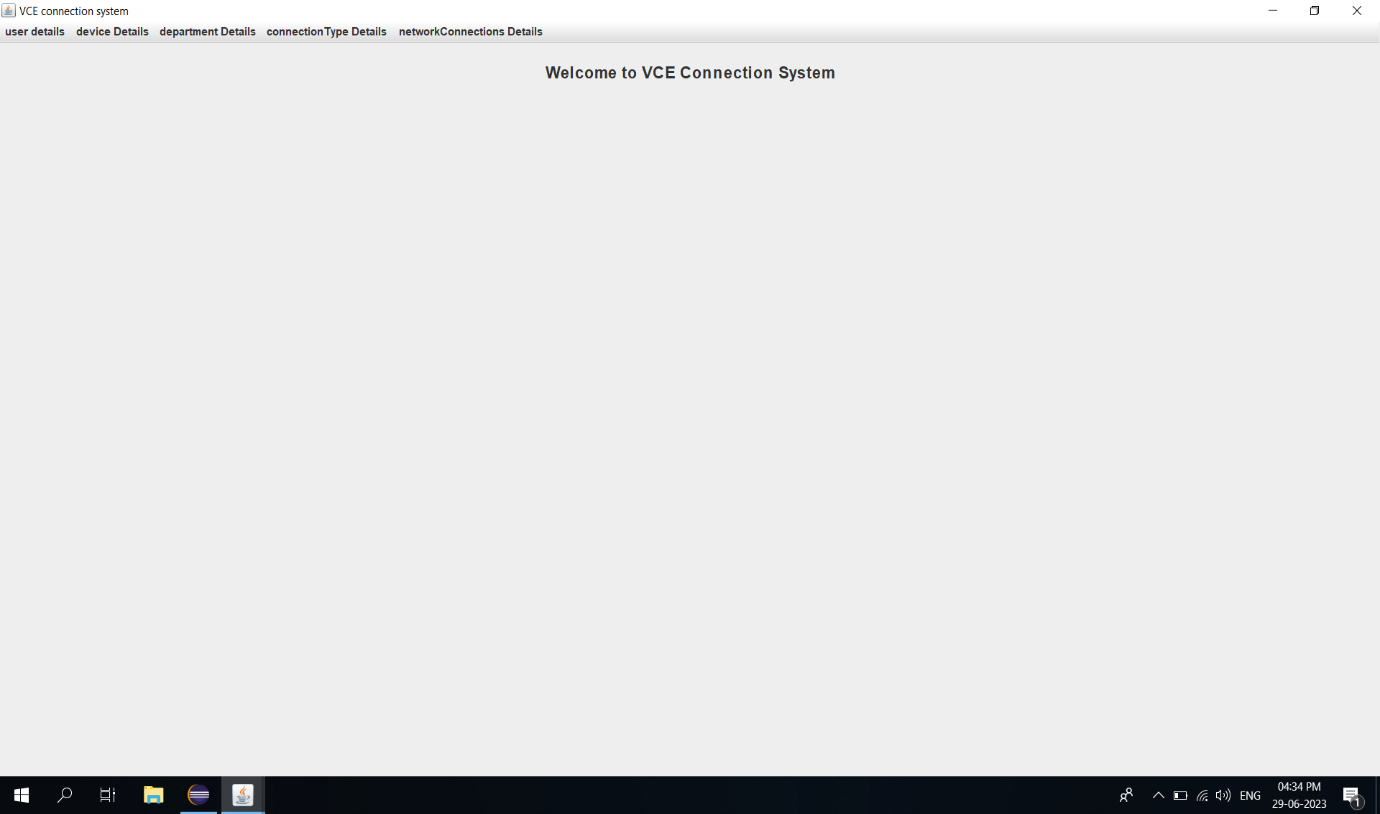
private static String password = "harini18";

private static Connection con;

public static Connection getConnection () throws Exception [ con = DriverManager.getConnection (url, username, password);

return con;

**Main Page:**

****

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class MainPage extends JFrame {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

//private JButton retrieveMarksButton;

public MainPage() {

// Set frame properties

setTitle("VCE connection system");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create label

JLabel welcomeLabel = new JLabel("Welcome to VCE Connection System");

welcomeLabel.setFont(new Font("Arial", Font.BOLD, 18));

welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);

welcomeLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));

add(welcomeLabel, BorderLayout.NORTH);

// Create panel for the button

//JPanel buttonPanel = new JPanel();

//retrieveMarksButton = new JButton("Retrieve Marks");

//buttonPanel.add(retrieveMarksButton);

// Create menu bar

JMenuBar menuBar = new JMenuBar();

// Create menus

JMenu userMenu = new JMenu("user details");

JMenu deviceMenu = new JMenu("device Details");

JMenu departmentMenu = new JMenu("department Details");

JMenu connectionTypeMenu = new JMenu("connectionType Details");

JMenu networkConnectionsMenu = new JMenu("networkConnections Details");

// Create menu item for student menu

JMenuItem viewUserDetails = new JMenuItem("View User Details");

viewUserDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new UserManagementSystem();

}

});

// Create menu item for course menu

JMenuItem viewDeviceDetails = new JMenuItem("View Devicee Details");

viewDeviceDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new DeviceManagementSystem();

}

});

// Create menu item for enrollment menu

JMenuItem viewDepartmentDetails = new JMenuItem("View Departmentment Details");

viewDepartmentDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new DepartmentManagementSystem();

}

});

// Create menu item for semester menu

JMenuItem viewconnectionTypeDetails = new JMenuItem("View connectionType Details");

viewconnectionTypeDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ConnectionTypeManagementSystem();

}

});

// Create menu item for grade menu

JMenuItem viewnetworkConnectionsDetails = new JMenuItem("View networkConnections Details");

viewnetworkConnectionsDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new NetworkConnectionManagementSystem();

}

});

// Add menu items to respective menus

userMenu.add(viewUserDetails);

deviceMenu.add(viewDeviceDetails);

departmentMenu.add(viewDepartmentDetails);

connectionTypeMenu.add(viewconnectionTypeDetails);

networkConnectionsMenu.add(viewnetworkConnectionsDetails);

// Add menus to the menu bar

menuBar.add(userMenu);

menuBar.add(deviceMenu);

menuBar.add(departmentMenu);

menuBar.add(connectionTypeMenu);

menuBar.add(networkConnectionsMenu);

// Set the menu bar

setJMenuBar(menuBar);

// Add the button panel to the frame

//add(buttonPanel, BorderLayout.CENTER);

// Set button action for "Retrieve Marks"

// retrieveMarksButton.addActionListener(new ActionListener() {

// public void actionPerformed(ActionEvent e) {

// new Retreive();

//}

//});

// Add window listener to handle maximizing the window

addWindowStateListener(new WindowStateListener() {

public void windowStateChanged(WindowEvent e) {

if ((e.getNewState() & Frame.MAXIMIZED\_BOTH) == Frame.MAXIMIZED\_BOTH) {

System.out.println("Window maximized");

} else {

System.out.println("Window not maximized");

}

}

});

// Set frame size and visibility

setSize(800, 600);

setVisible(true);

}

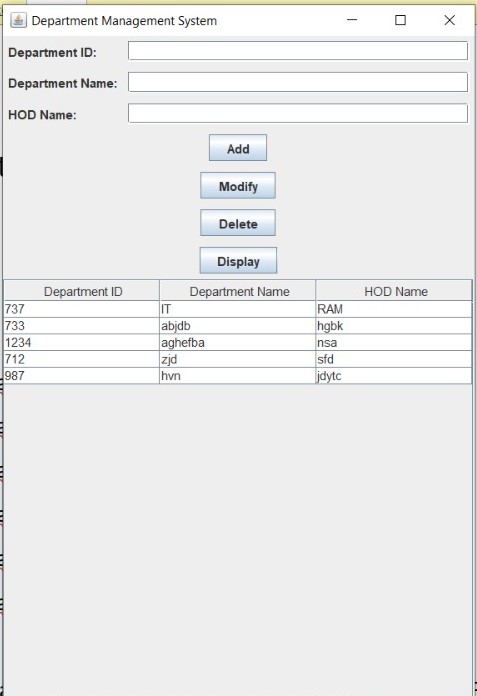
public static void main(String[] args) {

new MainPage();

}

}

**Department Page:**

****

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class DepartmentManagementSystem extends JFrame {

private JTextField txtDepartmentID, txtDepartmentName, txtHODName;

private JTable tblDepartments;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public DepartmentManagementSystem() {

initializeUI();

connectToDatabase();

displayDepartments();

}

private void initializeUI() {

txtDepartmentID = new JTextField();

txtDepartmentName = new JTextField();

txtHODName = new JTextField();

tblDepartments = new JTable();

tblDepartments.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblDepartments.getSelectionModel().addListSelectionListener(e -> selectDepartment());

JScrollPane scrollPane = new JScrollPane(tblDepartments);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("Department ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Department Name:"), gbc);

gbc.gridy++;

panel.add(new JLabel("HOD Name:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtDepartmentID, gbc);

gbc.gridy++;

panel.add(txtDepartmentName, gbc);

gbc.gridy++;

panel.add(txtHODName, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertDepartment());

btnModify.addActionListener(e -> modifyDepartment());

btnDelete.addActionListener(e -> deleteDepartment());

btnDisplay.addActionListener(e -> displayDepartments());

setTitle("Department Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "harini";

String password = "harini18";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertDepartment() {

String departmentID = txtDepartmentID.getText();

String departmentName = txtDepartmentName.getText();

String hodName = txtHODName.getText();

try {

String query = "INSERT INTO departments (departmentID, departmentName, HODName) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, departmentID);

statement.setString(2, departmentName);

statement.setString(3, hodName);

statement.executeUpdate();

clearFields();

displayDepartments();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyDepartment() {

int selectedRow = tblDepartments.getSelectedRow();

if (selectedRow >= 0) {

String departmentID = txtDepartmentID.getText();

String departmentName = txtDepartmentName.getText();

String hodName = txtHODName.getText();

try {

String query = "UPDATE departments SET departmentName=?, HODName=? WHERE departmentID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, departmentName);

statement.setString(2, hodName);

statement.setString(3, departmentID);

statement.executeUpdate();

clearFields();

displayDepartments();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a department to modify.");

}

}

private void deleteDepartment() {

int selectedRow = tblDepartments.getSelectedRow();

if (selectedRow >= 0) {

String departmentID = tblDepartments.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this department?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM departments WHERE departmentID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, departmentID);

statement.executeUpdate();

clearFields();

displayDepartments();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a department to delete.");

}

}

private void displayDepartments() {

try {

String query = "SELECT \* FROM departments";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Department> departments = new ArrayList<>();

while (resultSet.next()) {

String departmentID = resultSet.getString("departmentID");

String departmentName = resultSet.getString("departmentName");

String hodName = resultSet.getString("HODName");

departments.add(new Department(departmentID, departmentName, hodName));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Department ID", "Department Name", "HOD Name"});

for (Department department : departments) {

model.addRow(new String[]{department.getDepartmentID(), department.getDepartmentName(), department.getHODName()});

}

tblDepartments.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectDepartment() {

int selectedRow = tblDepartments.getSelectedRow();

if (selectedRow >= 0) {

String departmentID = tblDepartments.getValueAt(selectedRow, 0).toString();

String departmentName = tblDepartments.getValueAt(selectedRow, 1).toString();

String hodName = tblDepartments.getValueAt(selectedRow, 2).toString();

txtDepartmentID.setText(departmentID);

txtDepartmentName.setText(departmentName);

txtHODName.setText(hodName);

}

}

private void clearFields() {

txtDepartmentID.setText("");

txtDepartmentName.setText("");

txtHODName.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(DepartmentManagementSystem::new);

}

private class Department {

private String departmentID;

private String departmentName;

private String hodName;

public Department(String departmentID, String departmentName, String hodName) {

this.departmentID = departmentID;

this.departmentName = departmentName;

this.hodName = hodName;

}

public String getDepartmentID() {

return departmentID;

}

public String getDepartmentName() {

return departmentName;

}

public String getHODName() {

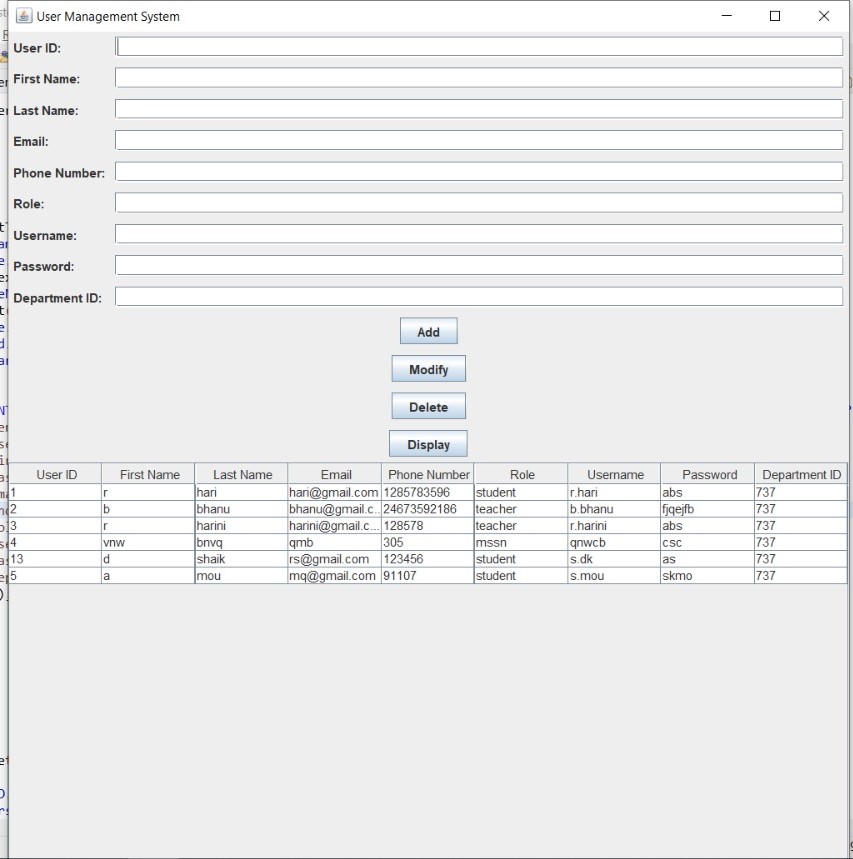
return hodName;

}

}

}

**User Page:**

****

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

//import java.awt.event.ActionEvent;

//import java.awt.event.ActionListener;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class UserManagementSystem extends JFrame {

private static final long serialVersionUID = 1L;

private JTextField txtUserID, txtFirstName, txtLastName, txtEmail, txtPhoneNumber, txtRole, txtUsername, txtPassword, txtDepartmentID;

private JTable tblUsers;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public UserManagementSystem() {

initializeUI();

connectToDatabase();

displayUsers();

}

private void initializeUI() {

txtUserID = new JTextField();

txtFirstName = new JTextField();

txtLastName = new JTextField();

txtEmail = new JTextField();

txtPhoneNumber = new JTextField();

txtRole = new JTextField();

txtUsername = new JTextField();

txtPassword = new JTextField();

txtDepartmentID = new JTextField();

tblUsers = new JTable();

tblUsers.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblUsers.getSelectionModel().addListSelectionListener(e -> selectUser());

JScrollPane scrollPane = new JScrollPane(tblUsers);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("User ID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("First Name:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Last Name:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Email:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Phone Number:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Role:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Username:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Password:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Department ID:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtUserID, gbc);

gbc.gridy++;

panel.add(txtFirstName, gbc);

gbc.gridy++;

panel.add(txtLastName, gbc);

gbc.gridy++;

panel.add(txtEmail, gbc);

gbc.gridy++;

panel.add(txtPhoneNumber, gbc);

gbc.gridy++;

panel.add(txtRole, gbc);

gbc.gridy++;

panel.add(txtUsername, gbc);

gbc.gridy++;

panel.add(txtPassword, gbc);

gbc.gridy++;

panel.add(txtDepartmentID, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertUser());

btnModify.addActionListener(e -> modifyUser());

btnDelete.addActionListener(e -> deleteUser());

btnDisplay.addActionListener(e -> displayUsers());

setTitle("User Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "harini";

String password = "harini18";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertUser() {

String userID = txtUserID.getText();

String firstName = txtFirstName.getText();

String lastName = txtLastName.getText();

String email = txtEmail.getText();

String phoneNumber = txtPhoneNumber.getText();

String role = txtRole.getText();

String username = txtUsername.getText();

String password = txtPassword.getText();

String departmentID = txtDepartmentID.getText();

try {

String query = "INSERT INTO users (userID, fname, lname, email, phonenumber, role, username, password, departmentID) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, userID);

statement.setString(2, firstName);

statement.setString(3, lastName);

statement.setString(4, email);

statement.setString(5, phoneNumber);

statement.setString(6, role);

statement.setString(7, username);

statement.setString(8, password);

statement.setString(9, departmentID);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

String userID = txtUserID.getText();

String firstName = txtFirstName.getText();

String lastName = txtLastName.getText();

String email = txtEmail.getText();

String phoneNumber = txtPhoneNumber.getText();

String role = txtRole.getText();

String username = txtUsername.getText();

String password = txtPassword.getText();

String departmentID = txtDepartmentID.getText();

try {

String query = "UPDATE users SET fname=?, lname=?, email=?, phonenumber=?, role=?, username=?, password=?, departmentID=? WHERE userID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, firstName);

statement.setString(2, lastName);

statement.setString(3, email);

statement.setString(4, phoneNumber);

statement.setString(5, role);

statement.setString(6, username);

statement.setString(7, password);

statement.setString(8, departmentID);

statement.setString(9, userID);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a user to modify.");

}

}

private void deleteUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

String userID = tblUsers.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this user?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM users WHERE userID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, userID);

statement.executeUpdate();

clearFields();

displayUsers();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a user to delete.");

}

}

private void displayUsers() {

try {

String query = "SELECT \* FROM users";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

ResultSetMetaData metaData = resultSet.getMetaData();

int columnCount = metaData.getColumnCount();

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

String columnName = metaData.getColumnName(i);

System.out.println("Column " + i + ": " + columnName);

}

List<User> users = new ArrayList<>();

while (resultSet.next()) {

String userID = resultSet.getString("userID");

String firstName = resultSet.getString("fname");

String lastName = resultSet.getString("lname");

String email = resultSet.getString("email");

String phoneNumber = resultSet.getString("phonenumber");

String role = resultSet.getString("role");

String username = resultSet.getString("username");

String password = resultSet.getString("password");

String departmentID = resultSet.getString("departmentID");

users.add(new User(userID, firstName, lastName, email, phoneNumber, role, username, password, departmentID));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"User ID", "First Name", "Last Name", "Email", "Phone Number", "Role", "Username", "Password", "Department ID"});

for (User user : users) {

model.addRow(new String[]{user.getUserID(), user.getFirstName(), user.getLastName(), user.getEmail(), user.getPhoneNumber(), user.getRole(), user.getUsername(), user.getPassword(), user.getDepartmentID()});

}

tblUsers.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectUser() {

int selectedRow = tblUsers.getSelectedRow();

if (selectedRow >= 0) {

String userID = tblUsers.getValueAt(selectedRow, 0).toString();

String firstName = tblUsers.getValueAt(selectedRow, 1).toString();

String lastName = tblUsers.getValueAt(selectedRow, 2).toString();

String email = tblUsers.getValueAt(selectedRow, 3).toString();

String phoneNumber = tblUsers.getValueAt(selectedRow, 4).toString();

String role = tblUsers.getValueAt(selectedRow, 5).toString();

String username = tblUsers.getValueAt(selectedRow, 6).toString();

String password = tblUsers.getValueAt(selectedRow, 7).toString();

String departmentID = tblUsers.getValueAt(selectedRow, 8).toString();

txtUserID.setText(userID);

txtFirstName.setText(firstName);

txtLastName.setText(lastName);

txtEmail.setText(email);

txtPhoneNumber.setText(phoneNumber);

txtRole.setText(role);

txtUsername.setText(username);

txtPassword.setText(password);

txtDepartmentID.setText(departmentID);

}

}

private void clearFields() {

txtUserID.setText("");

txtFirstName.setText("");

txtLastName.setText("");

txtEmail.setText("");

txtPhoneNumber.setText("");

txtRole.setText("");

txtUsername.setText("");

txtPassword.setText("");

txtDepartmentID.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(UserManagementSystem::new);

}

private class User {

private String userID;

private String firstName;

private String lastName;

private String email;

private String phoneNumber;

private String role;

private String username;

private String password;

private String departmentID;

public User(String userID, String firstName, String lastName, String email, String phoneNumber, String role, String username, String password, String departmentID) {

this.userID = userID;

this.firstName = firstName;

this.lastName = lastName;

this.email = email;

this.phoneNumber = phoneNumber;

this.role = role;

this.username = username;

this.password = password;

this.departmentID = departmentID;

}

public String getUserID() {

return userID;

}

public String getFirstName() {

return firstName;

}

public String getLastName() {

return lastName;

}

public String getEmail() {

return email;

}

public String getPhoneNumber() {

return phoneNumber;

public String getRole() {

return role;

}

public String getUsername() {

return username;

}

public String getPassword() {

return password;

}

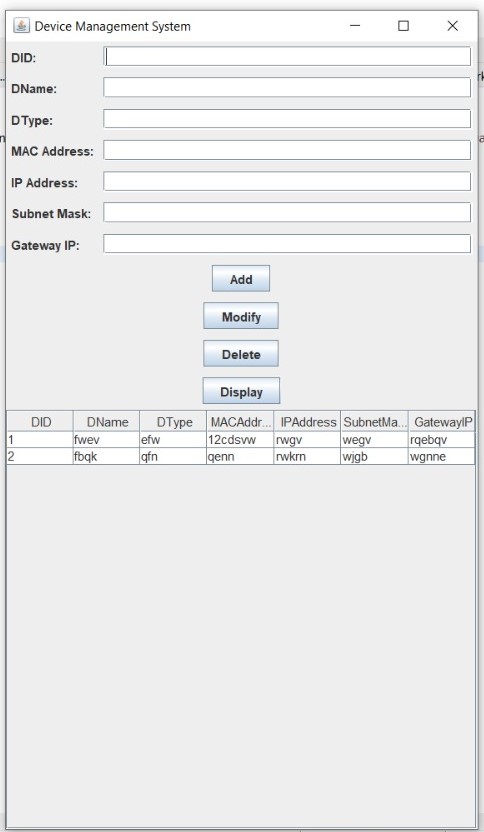
public String getDepartmentID() {

return departmentID;

}

}

}

**Device Page: **

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class DeviceManagementSystem extends JFrame {

private JTextField txtDID, txtDName, txtDType, txtMACAddress, txtIPAddress, txtSubnetMask, txtGatewayIP;

private JTable tblDevices;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public DeviceManagementSystem() {

initializeUI();

connectToDatabase();

displayDevices();

}

private void initializeUI() {

txtDID = new JTextField();

txtDName = new JTextField();

txtDType = new JTextField();

txtMACAddress = new JTextField();

txtIPAddress = new JTextField();

txtSubnetMask = new JTextField();

txtGatewayIP = new JTextField();

tblDevices = new JTable();

tblDevices.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblDevices.getSelectionModel().addListSelectionListener(e -> selectDevice());

JScrollPane scrollPane = new JScrollPane(tblDevices);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("DID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("DName:"), gbc);

gbc.gridy++;

panel.add(new JLabel("DType:"), gbc);

gbc.gridy++;

panel.add(new JLabel("MAC Address:"), gbc);

gbc.gridy++;

panel.add(new JLabel("IP Address:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Subnet Mask:"), gbc);

gbc.gridy++;

panel.add(new JLabel("Gateway IP:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtDID, gbc);

gbc.gridy++;

panel.add(txtDName, gbc);

gbc.gridy++;

panel.add(txtDType, gbc);

gbc.gridy++;

panel.add(txtMACAddress, gbc);

gbc.gridy++;

panel.add(txtIPAddress, gbc);

gbc.gridy++;

panel.add(txtSubnetMask, gbc);

gbc.gridy++;

panel.add(txtGatewayIP, gbc);

gbc.gridy++;

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertDevice());

btnModify.addActionListener(e -> modifyDevice());

btnDelete.addActionListener(e -> deleteDevice());

btnDisplay.addActionListener(e -> displayDevices());

setTitle("Device Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "harini";

String password = "harini18";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertDevice() {

String did = txtDID.getText();

String dname = txtDName.getText();

String dtype = txtDType.getText();

String macAddress = txtMACAddress.getText();

String ipAddress = txtIPAddress.getText();

String subnetMask = txtSubnetMask.getText();

String gatewayIP = txtGatewayIP.getText();

// String userID=txtuserID.getText();

try {

String query = "INSERT INTO devices (DID, Dname, Dtype, MACAddress, IPAddress, subnetmask, gatewayIP) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, did);

statement.setString(2, dname);

statement.setString(3, dtype);

statement.setString(4, macAddress);

statement.setString(5, ipAddress);

statement.setString(6, subnetMask);

statement.setString(7, gatewayIP);

//statement.setString(8, userID);

statement.executeUpdate();

clearFields();

displayDevices();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyDevice() {

int selectedRow = tblDevices.getSelectedRow();

if (selectedRow >= 0) {

String did = txtDID.getText();

String dname = txtDName.getText();

String dtype = txtDType.getText();

String macAddress = txtMACAddress.getText();

String ipAddress = txtIPAddress.getText();

String subnetMask = txtSubnetMask.getText();

String gatewayIP = txtGatewayIP.getText();

//String userID= txtuserID.getText();

try {

String query = "UPDATE devices SET Dname=?, Dtype=?, MACAddress=?, IPAddress=?, subnetmask=?, gatewayIP=? WHERE DID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, dname);

statement.setString(2, dtype);

statement.setString(3, macAddress);

statement.setString(4, ipAddress);

statement.setString(5, subnetMask);

statement.setString(6, gatewayIP);

statement.setString(7, did);

// statement.setString(8,userID);

statement.executeUpdate();

clearFields();

displayDevices();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a device to modify.");

}

}

private void deleteDevice() {

int selectedRow = tblDevices.getSelectedRow();

if (selectedRow >= 0) {

String did = tblDevices.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this device?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM devices WHERE DID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, did);

statement.executeUpdate();

clearFields();

displayDevices();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a device to delete.");

}

}

private void displayDevices() {

try {

String query = "SELECT \* FROM devices";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Device> devices = new ArrayList<>();

while (resultSet.next()) {

String did = resultSet.getString("DID");

String dname = resultSet.getString("Dname");

String dtype = resultSet.getString("Dtype");

String macAddress = resultSet.getString("MACAddress");

String ipAddress = resultSet.getString("IPAddress");

String subnetMask = resultSet.getString("subnetmask");

String gatewayIP = resultSet.getString("gatewayIP");

//String userID = resultSet.getString("userID");

devices.add(new Device(did, dname, dtype, macAddress, ipAddress, subnetMask, gatewayIP));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"DID", "DName", "DType", "MACAddress", "IPAddress", "SubnetMask", "GatewayIP"});

for (Device device : devices) {

model.addRow(new String[]{device.getDid(), device.getDname(), device.getDtype(), device.getMacAddress(), device.getIpAddress(), device.getSubnetMask(), device.getGatewayIP()});

}

tblDevices.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectDevice() {

int selectedRow = tblDevices.getSelectedRow();

if (selectedRow >= 0) {

String did = tblDevices.getValueAt(selectedRow, 0).toString();

String dname = tblDevices.getValueAt(selectedRow, 1).toString();

String dtype = tblDevices.getValueAt(selectedRow, 2).toString();

String macAddress = tblDevices.getValueAt(selectedRow, 3).toString();

String ipAddress = tblDevices.getValueAt(selectedRow, 4).toString();

String subnetMask = tblDevices.getValueAt(selectedRow, 5).toString();

String gatewayIP = tblDevices.getValueAt(selectedRow, 6).toString();

//String userID = tblDevices.getValueAt(selectedRow, 7).toString();

txtDID.setText(did);

txtDName.setText(dname);

txtDType.setText(dtype);

txtMACAddress.setText(macAddress);

txtIPAddress.setText(ipAddress);

txtSubnetMask.setText(subnetMask);

txtGatewayIP.setText(gatewayIP);

// txtuserID.setText(userID);

}

}

private void clearFields() {

txtDID.setText("");

txtDName.setText("");

txtDType.setText("");

txtMACAddress.setText("");

txtIPAddress.setText("");

txtSubnetMask.setText("");

txtGatewayIP.setText("");

// txtuserID.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(DeviceManagementSystem::new);

}

private class Device {

private String did;

private String dname;

private String dtype;

private String macAddress;

private String ipAddress;

private String subnetMask;

private String gatewayIP;

// private String userID;

public Device(String did, String dname, String dtype, String macAddress, String ipAddress, String subnetMask, String gatewayIP) {

this.did = did;

this.dname = dname;

this.dtype = dtype;

this.macAddress = macAddress;

this.ipAddress = ipAddress;

this.subnetMask = subnetMask;

this.gatewayIP = gatewayIP;

}

public String getDid() {

return did;

}

public String getDname() {

return dname;

}

public String getDtype() {

return dtype;

}

public String getMacAddress() {

return macAddress;

}

public String getIpAddress() {

return ipAddress;

}

public String getSubnetMask() {

return subnetMask;

}

public String getGatewayIP() {

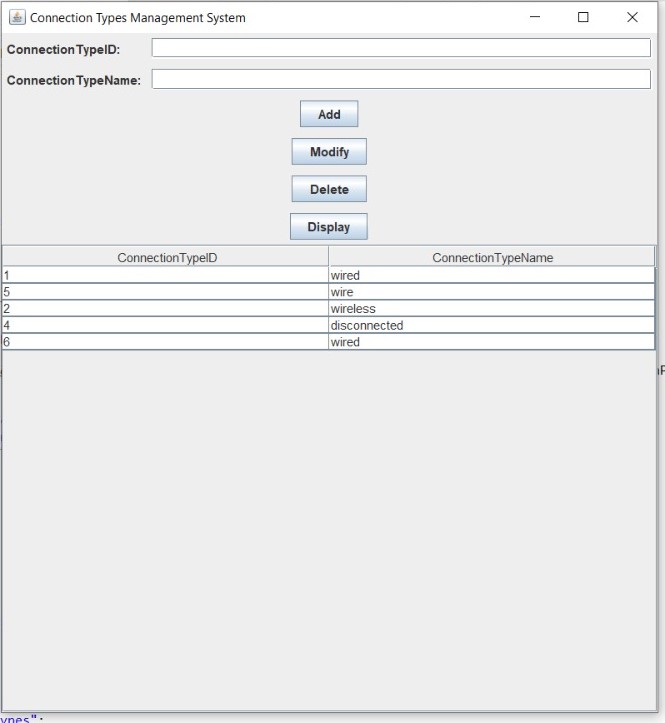
return gatewayIP;

}

}

}

**Connection Type Page:**

****

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class ConnectionTypeManagementSystem extends JFrame {

private JTextField txtConnectionTypeID, txtConnectionTypeName;

private JTable tblConnectionTypes;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public ConnectionTypeManagementSystem() {

initializeUI();

connectToDatabase();

displayConnectionTypes();

}

private void initializeUI() {

txtConnectionTypeID = new JTextField();

txtConnectionTypeName = new JTextField();

tblConnectionTypes = new JTable();

tblConnectionTypes.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblConnectionTypes.getSelectionModel().addListSelectionListener(e -> selectConnectionType());

JScrollPane scrollPane = new JScrollPane(tblConnectionTypes);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("ConnectionTypeID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("ConnectionTypeName:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtConnectionTypeID, gbc);

gbc.gridy++;

panel.add(txtConnectionTypeName, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertConnectionType());

btnModify.addActionListener(e -> modifyConnectionType());

btnDelete.addActionListener(e -> deleteConnectionType());

btnDisplay.addActionListener(e -> displayConnectionTypes());

setTitle("Connection Types Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "harini";

String password = "harini18";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertConnectionType() {

String connectionTypeID = txtConnectionTypeID.getText();

String connectionTypeName = txtConnectionTypeName.getText();

try {

String query = "INSERT INTO connectionTypes (connectionTypeID, connectionTypeName) VALUES (?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, connectionTypeID);

statement.setString(2, connectionTypeName);

statement.executeUpdate();

clearFields();

displayConnectionTypes();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyConnectionType() {

int selectedRow = tblConnectionTypes.getSelectedRow();

if (selectedRow >= 0) {

String connectionTypeID = txtConnectionTypeID.getText();

String connectionTypeName = txtConnectionTypeName.getText();

try {

String query = "UPDATE connectionTypes SET connectionTypeName=? WHERE connectionTypeID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, connectionTypeName);

statement.setString(2, connectionTypeID);

statement.executeUpdate();

clearFields();

displayConnectionTypes();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select a connection type to modify.");

}

}

private void deleteConnectionType() {

int selectedRow = tblConnectionTypes.getSelectedRow();

if (selectedRow >= 0) {

String connectionTypeID = tblConnectionTypes.getValueAt(selectedRow, 0).toString();

int option = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this connection type?", "Confirmation", JOptionPane.YES\_NO\_OPTION);

if (option == JOptionPane.YES\_OPTION) {

try {

String query = "DELETE FROM connectionTypes WHERE connectionTypeID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, connectionTypeID);

statement.executeUpdate();

clearFields();

displayConnectionTypes();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select a connection type to delete.");

}

}

private void displayConnectionTypes() {

try {

String query = "SELECT \* FROM connectionTypes";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<ConnectionType> connectionTypes = new ArrayList<>();

while (resultSet.next()) {

String connectionTypeID = resultSet.getString("connectionTypeID");

String connectionTypeName = resultSet.getString("connectionTypeName");

connectionTypes.add(new ConnectionType(connectionTypeID, connectionTypeName));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"ConnectionTypeID", "ConnectionTypeName"});

for (ConnectionType connectionType : connectionTypes) {

model.addRow(new String[]{connectionType.getConnectionTypeID(), connectionType.getConnectionTypeName()});

}

tblConnectionTypes.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectConnectionType() {

int selectedRow = tblConnectionTypes.getSelectedRow();

if (selectedRow >= 0) {

String connectionTypeID = tblConnectionTypes.getValueAt(selectedRow, 0).toString();

String connectionTypeName = tblConnectionTypes.getValueAt(selectedRow, 1).toString();

txtConnectionTypeID.setText(connectionTypeID);

txtConnectionTypeName.setText(connectionTypeName);

}

}

private void clearFields() {

txtConnectionTypeID.setText("");

txtConnectionTypeName.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(ConnectionTypeManagementSystem::new);

}

private class ConnectionType {

private String connectionTypeID;

private String connectionTypeName;

public ConnectionType(String connectionTypeID, String connectionTypeName) {

this.connectionTypeID = connectionTypeID;

this.connectionTypeName = connectionTypeName;

}

public String getConnectionTypeID() {

return connectionTypeID;

}

public String getConnectionTypeName() {

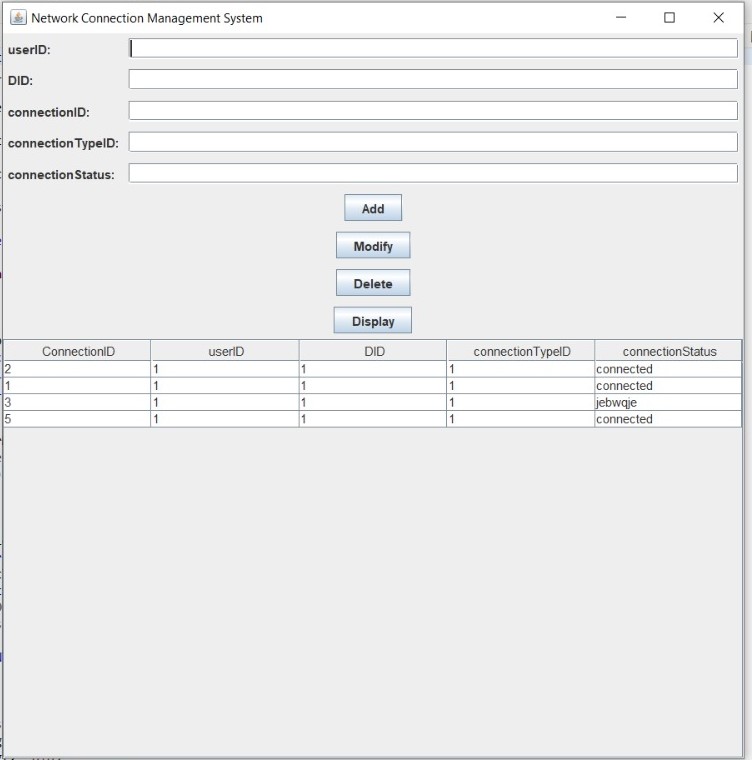
return connectionTypeName;

}

}

}

**Network Connections Page:**

****

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class NetworkConnectionManagementSystem extends JFrame {

private JTextField txtuserID, txtDID, txtconnectionID, txtconnectionTypeID, txtconnectionStatus;

private JTable tblConnections;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public NetworkConnectionManagementSystem() {

initializeUI();

connectToDatabase();

displayConnections();

}

private void initializeUI() {

txtuserID = new JTextField();

txtDID = new JTextField();

txtconnectionID = new JTextField();

txtconnectionTypeID = new JTextField();

txtconnectionStatus = new JTextField();

tblConnections = new JTable();

tblConnections.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblConnections.getSelectionModel().addListSelectionListener(e -> selectConnection());

JScrollPane scrollPane = new JScrollPane(tblConnections);

btnAdd = new JButton("Add");

btnModify = new JButton("Modify");

btnDelete = new JButton("Delete");

btnDisplay = new JButton("Display");

JPanel panel = new JPanel(new GridBagLayout());

GridBagConstraints gbc = new GridBagConstraints();

gbc.gridx = 0;

gbc.gridy = 0;

gbc.anchor = GridBagConstraints.WEST;

gbc.insets = new Insets(5, 5, 5, 5);

panel.add(new JLabel("userID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("DID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("connectionID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("connectionTypeID:"), gbc);

gbc.gridy++;

panel.add(new JLabel("connectionStatus:"), gbc);

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtuserID, gbc);

gbc.gridy++;

panel.add(txtDID, gbc);

gbc.gridy++;

panel.add(txtconnectionID, gbc);

gbc.gridy++;

panel.add(txtconnectionTypeID, gbc);

gbc.gridy++;

panel.add(txtconnectionStatus, gbc);

gbc.gridx = 0;

gbc.gridy++;

gbc.gridwidth = 2;

gbc.fill = GridBagConstraints.NONE;

gbc.anchor = GridBagConstraints.CENTER;

gbc.weightx = 0;

panel.add(btnAdd, gbc);

gbc.gridy++;

panel.add(btnModify, gbc);

gbc.gridy++;

panel.add(btnDelete, gbc);

gbc.gridy++;

panel.add(btnDisplay, gbc);

setLayout(new BorderLayout());

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

btnAdd.addActionListener(e -> insertConnection());

btnModify.addActionListener(e -> modifyConnection());

btnDelete.addActionListener(e -> deleteConnection());

btnDisplay.addActionListener(e -> displayConnections());

setTitle("Network Connection Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

String url = "jdbc:oracle:thin:@localhost:1521:xe";

String username = "harini";

String password = "harini18";

try {

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertConnection() {

String userID = txtuserID.getText();

String DID = txtDID.getText();

String connectionID = txtconnectionID.getText();

String connectionTypeID = txtconnectionTypeID.getText();

String connectionStatus = txtconnectionStatus.getText();

String sql = "INSERT INTO networkConnections (userID, DID, connectionID,connectionTypeID, connectionStatus) " +

"VALUES (?, ?, ?, ?, ?)";

try {

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, userID);

statement.setString(2, DID);

statement.setString(3, connectionID);

statement.setString(4, connectionTypeID);

statement.setString(5, connectionStatus);

int rowsInserted = statement.executeUpdate();

if (rowsInserted > 0) {

System.out.println("Connection inserted successfully.");

displayConnections();

clearFields();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyConnection() {

int selectedRow = tblConnections.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a connection to modify.");

return;

}

String connectionID = tblConnections.getValueAt(selectedRow, 0).toString();

String userID = txtuserID.getText();

String DID= txtDID.getText();

String connectionTypeID= txtconnectionTypeID.getText();

// String status = txtStatus.getText();

String connectionStatus = txtconnectionStatus.getText();

String sql = "UPDATE networkConnections SET userID = ?, DID = ?, connectionTypeID = ?, " +

"connectionStatus = ? WHERE connectionID = ?";

try {

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, userID);

statement.setString(2, DID);

statement.setString(3, connectionID);

statement.setString(4, connectionTypeID);

statement.setString(5,connectionStatus);

//statement.setString(6, connectionId);

int rowsUpdated = statement.executeUpdate();

if (rowsUpdated > 0) {

System.out.println("Connection modified successfully.");

displayConnections();

clearFields();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private void deleteConnection() {

int selectedRow = tblConnections.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a connection to delete.");

return;

}

int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete the selected connection?",

"Confirm Deletion", JOptionPane.YES\_NO\_OPTION);

if (confirm == JOptionPane.YES\_OPTION) {

String connectionId = tblConnections.getValueAt(selectedRow, 0).toString();

String sql = "DELETE FROM networkConnections WHERE connectionID = ?";

try {

PreparedStatement statement = connection.prepareStatement(sql);

statement.setString(1, connectionId);

int rowsDeleted = statement.executeUpdate();

if (rowsDeleted > 0) {

System.out.println("Connection deleted successfully.");

displayConnections();

clearFields();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

private void displayConnections() {

String sql = "SELECT \* FROM networkConnections";

try {

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(sql);

DefaultTableModel model = new DefaultTableModel();

model.addColumn("ConnectionID");

model.addColumn("userID");

model.addColumn("DID");

model.addColumn("connectionTypeID");

model.addColumn("connectionStatus");

//model.addColumn("Bandwidth");

while (resultSet.next()) {

String connectionId = resultSet.getString("connectionID");

String userID = resultSet.getString("userID");

String DID= resultSet.getString("DID");

String connectionTypeID= resultSet.getString("connectionTypeID");

//String status = resultSet.getString("status");

String connectionStatus = resultSet.getString("connectionStatus");

model.addRow(new Object[]{connectionId, userID, DID, connectionTypeID, connectionStatus});

}

tblConnections.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectConnection() {

int selectedRow = tblConnections.getSelectedRow();

if (selectedRow != -1) {

String userID = tblConnections.getValueAt(selectedRow, 1).toString();

String DID = tblConnections.getValueAt(selectedRow, 2).toString();

String connectionTypeID = tblConnections.getValueAt(selectedRow, 3).toString();

String connectionStatus = tblConnections.getValueAt(selectedRow, 4).toString();

//String bandwidth = tblConnections.getValueAt(selectedRow, 5).toString();

txtuserID.setText(userID);

txtDID.setText(DID);

txtconnectionTypeID.setText(connectionTypeID);

// txtStatus.setText(status);

txtconnectionStatus.setText(connectionStatus);

}

}

private void clearFields() {

txtuserID.setText("");

txtDID.setText("");

txtconnectionTypeID.setText("");

//txtStatus.setText("");

txtconnectionStatus.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(NetworkConnectionManagementSystem::new);

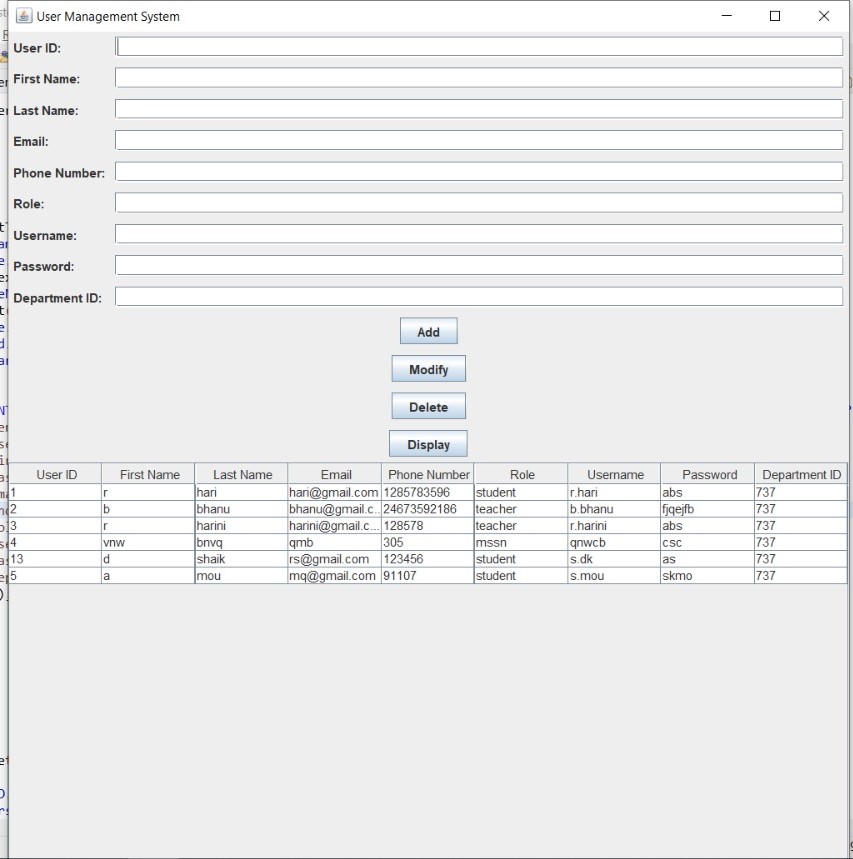
}

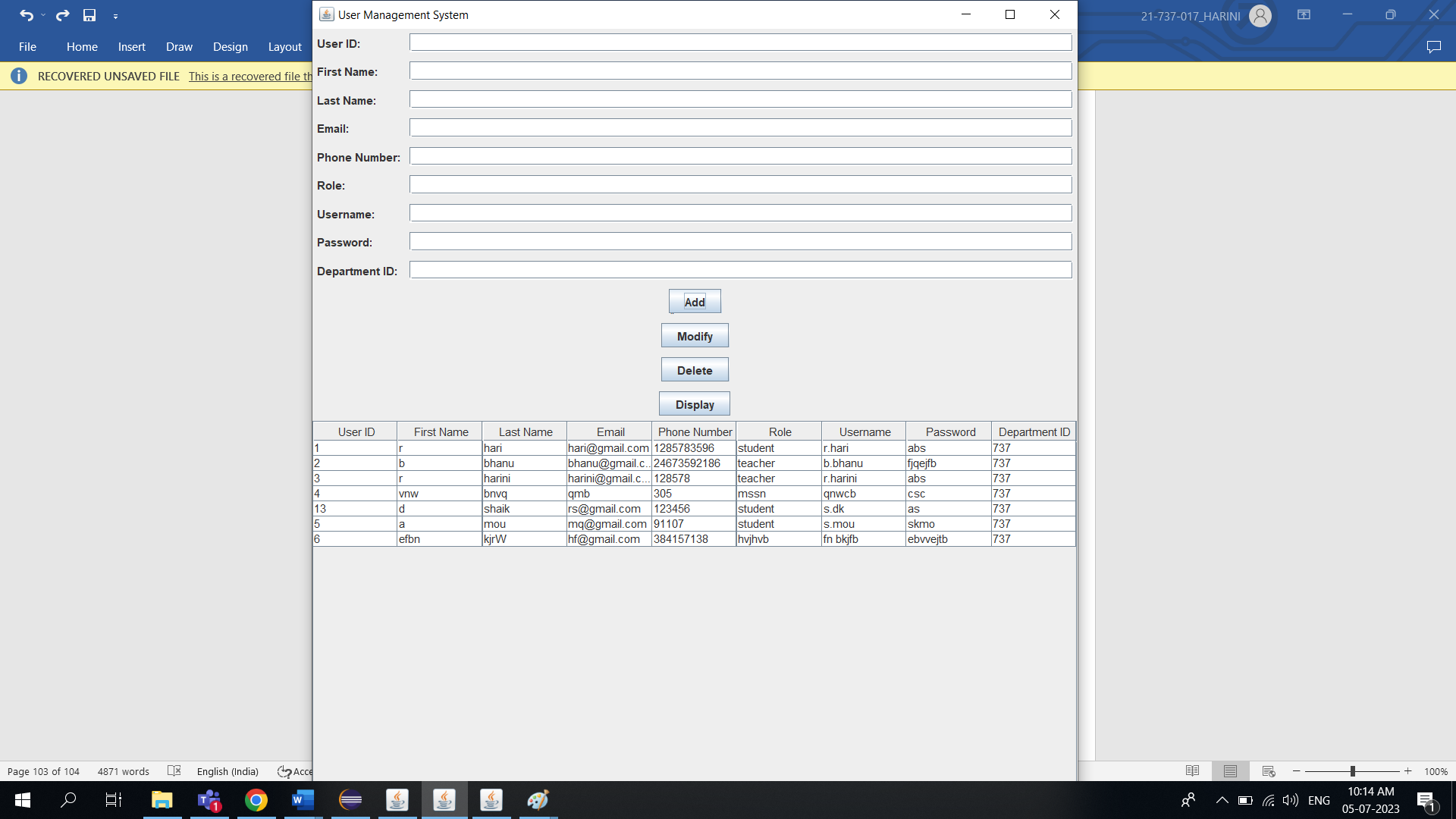
}

**TESTING**

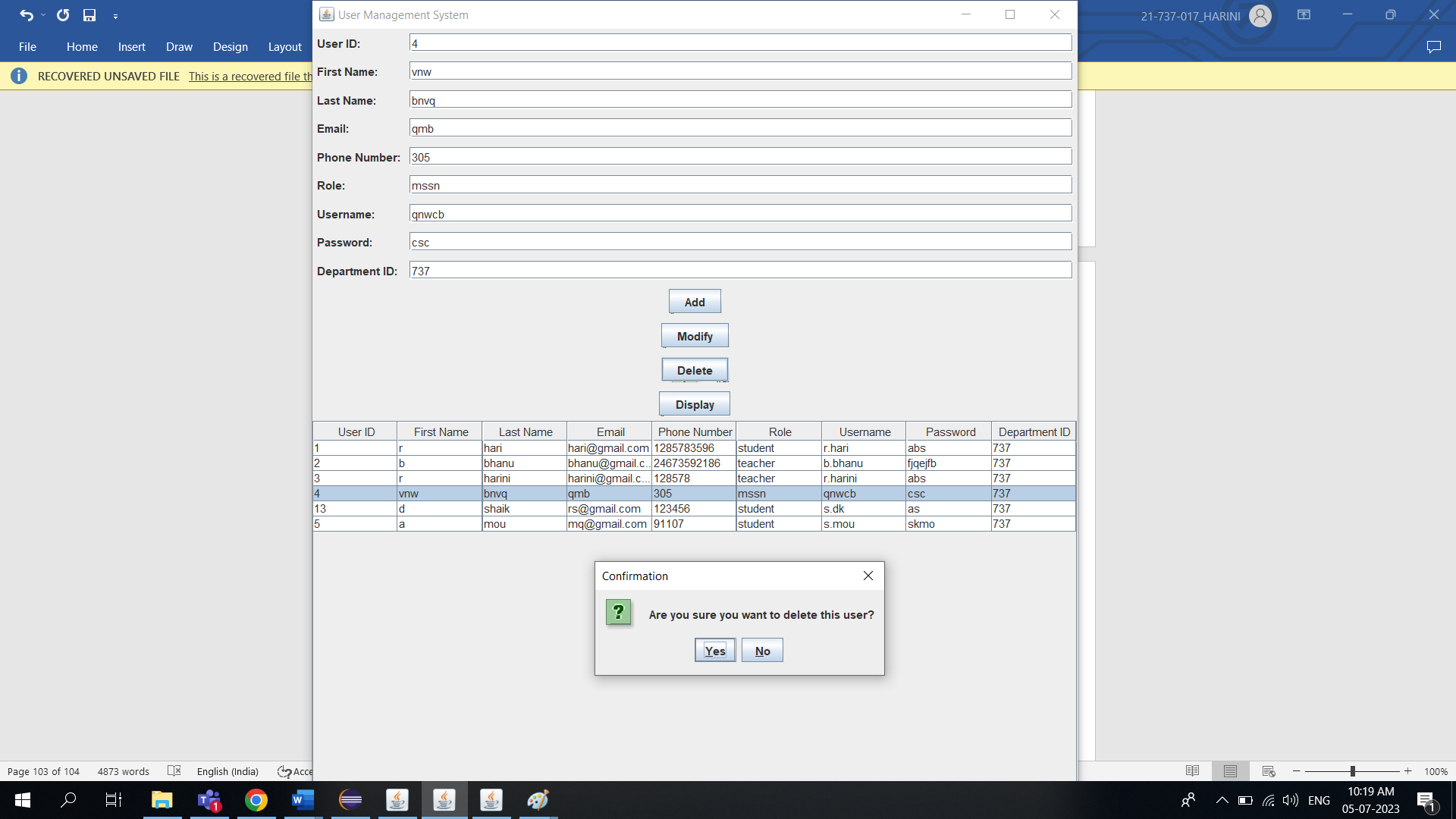
User page:

Before insertion into table :

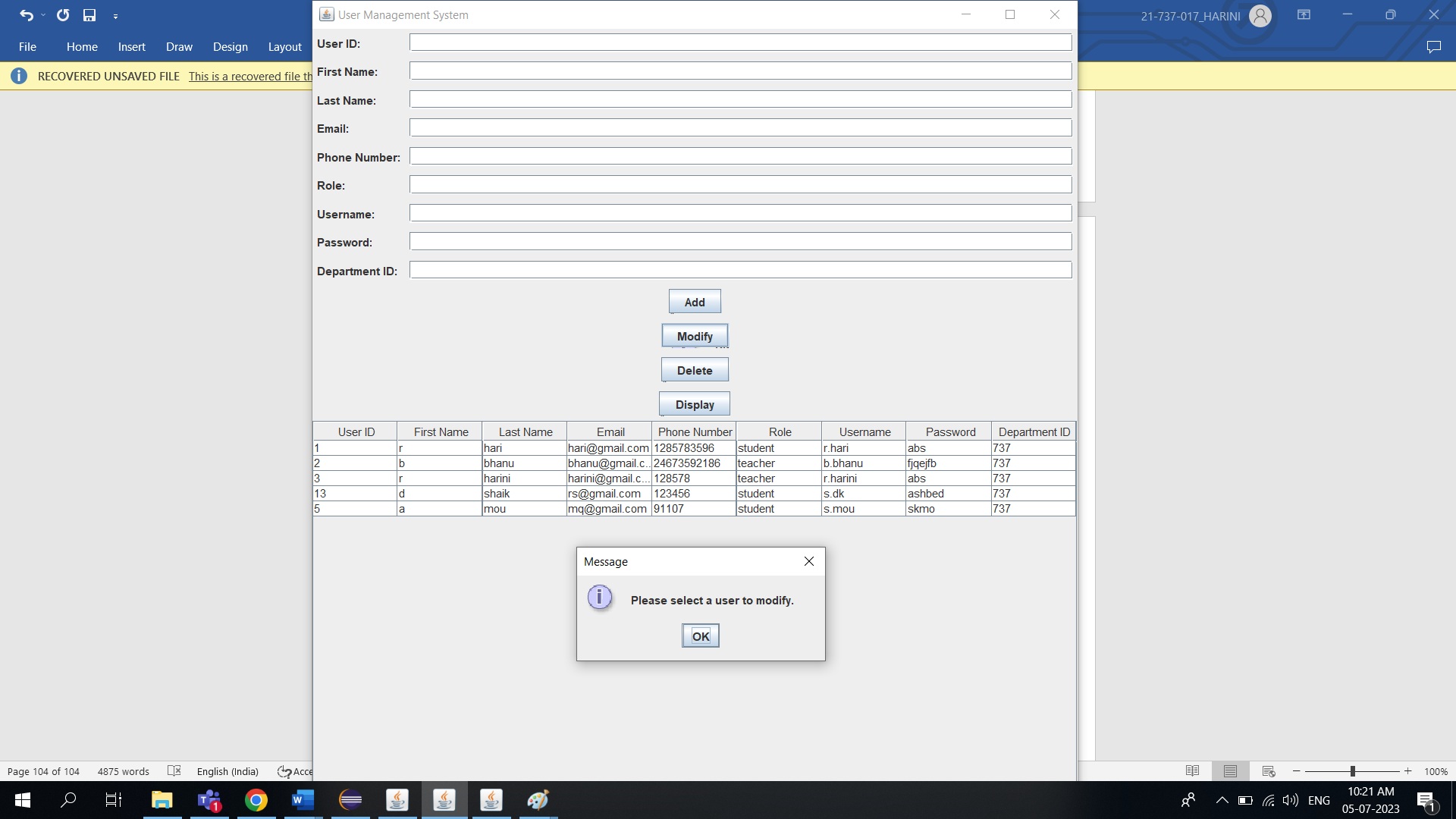
****

After insertion:

Deletion:

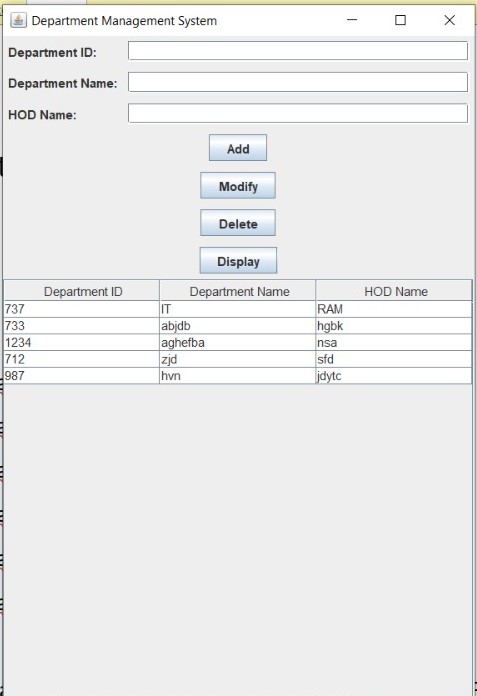
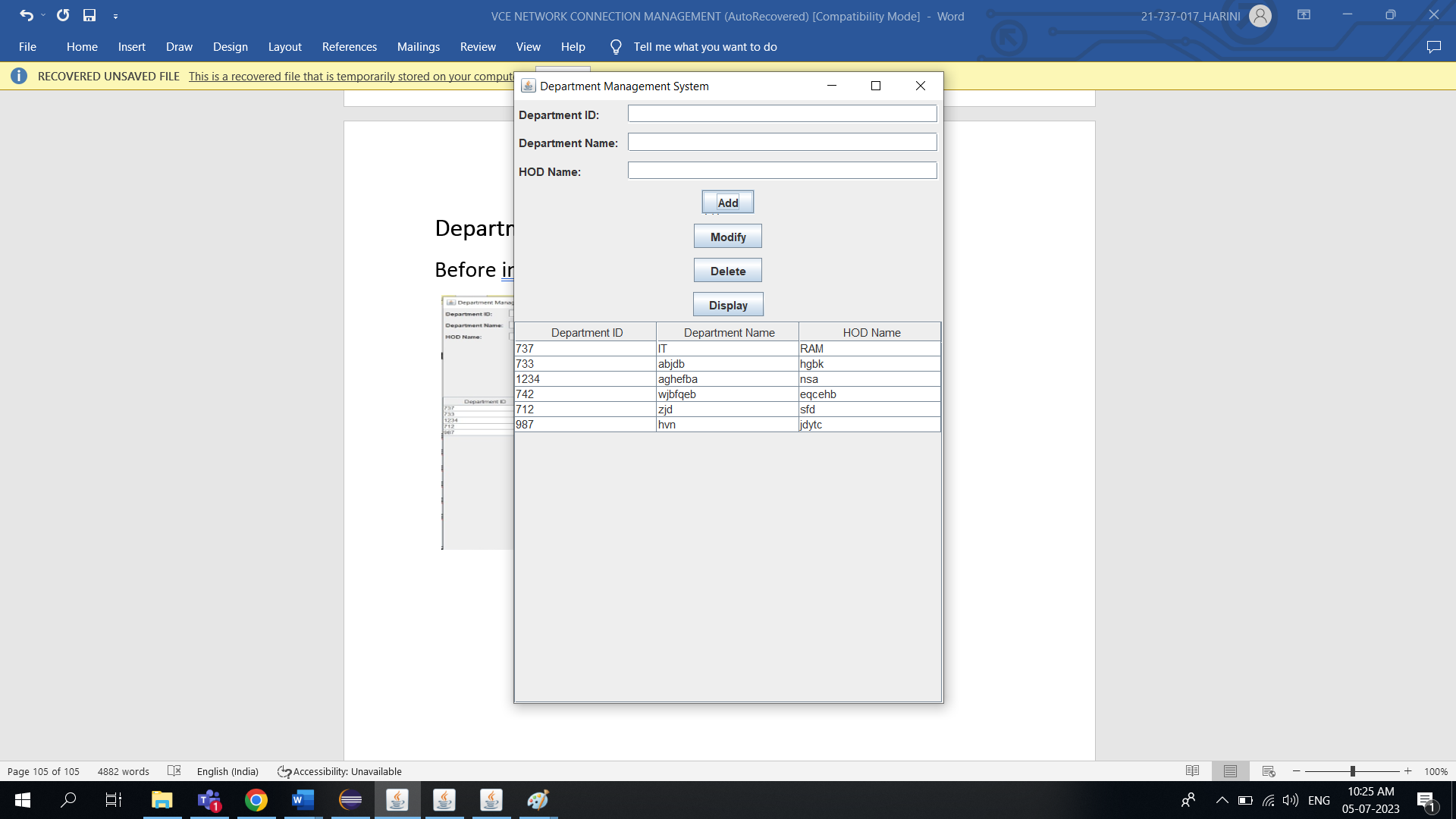


Modification:

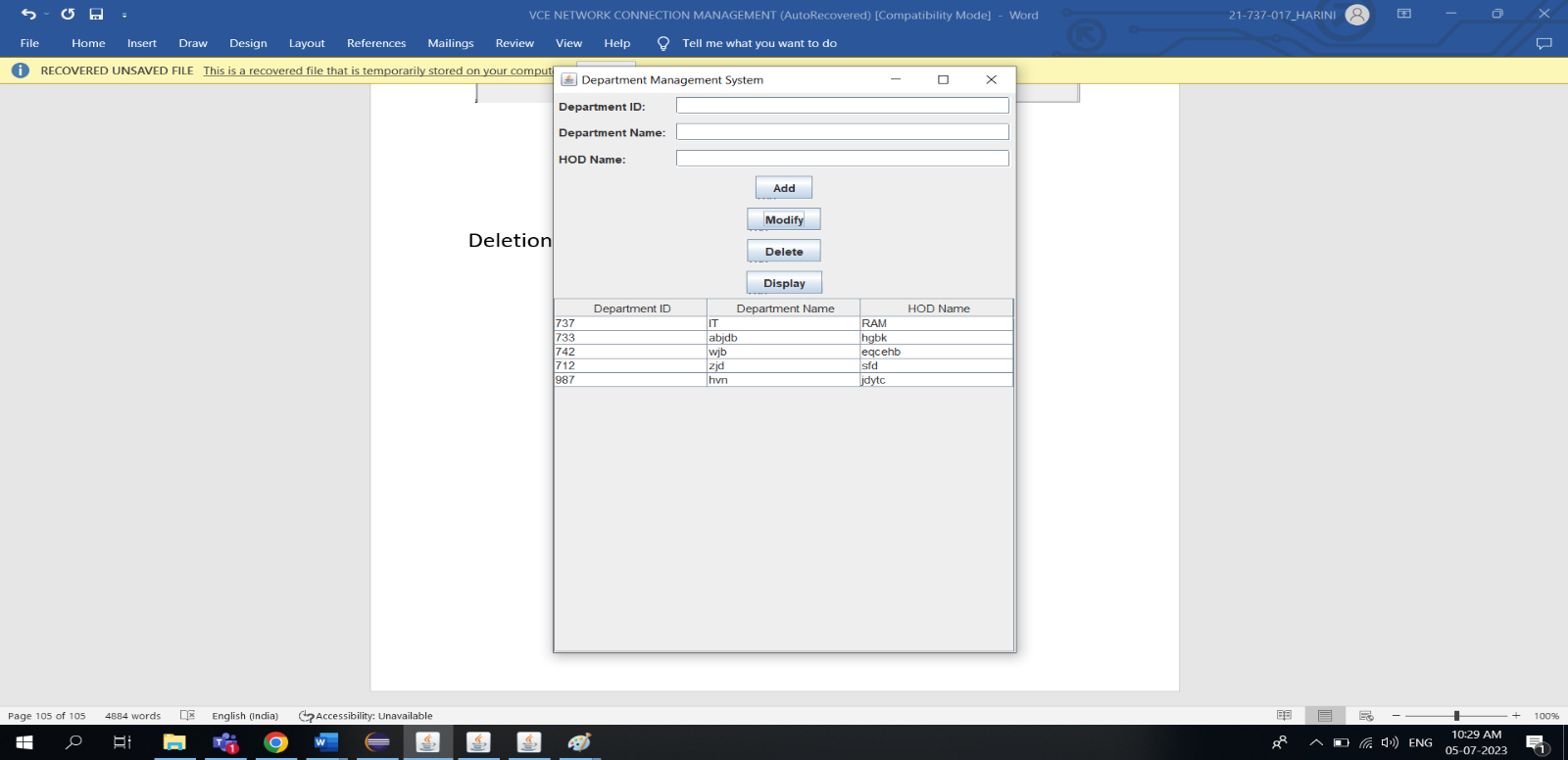
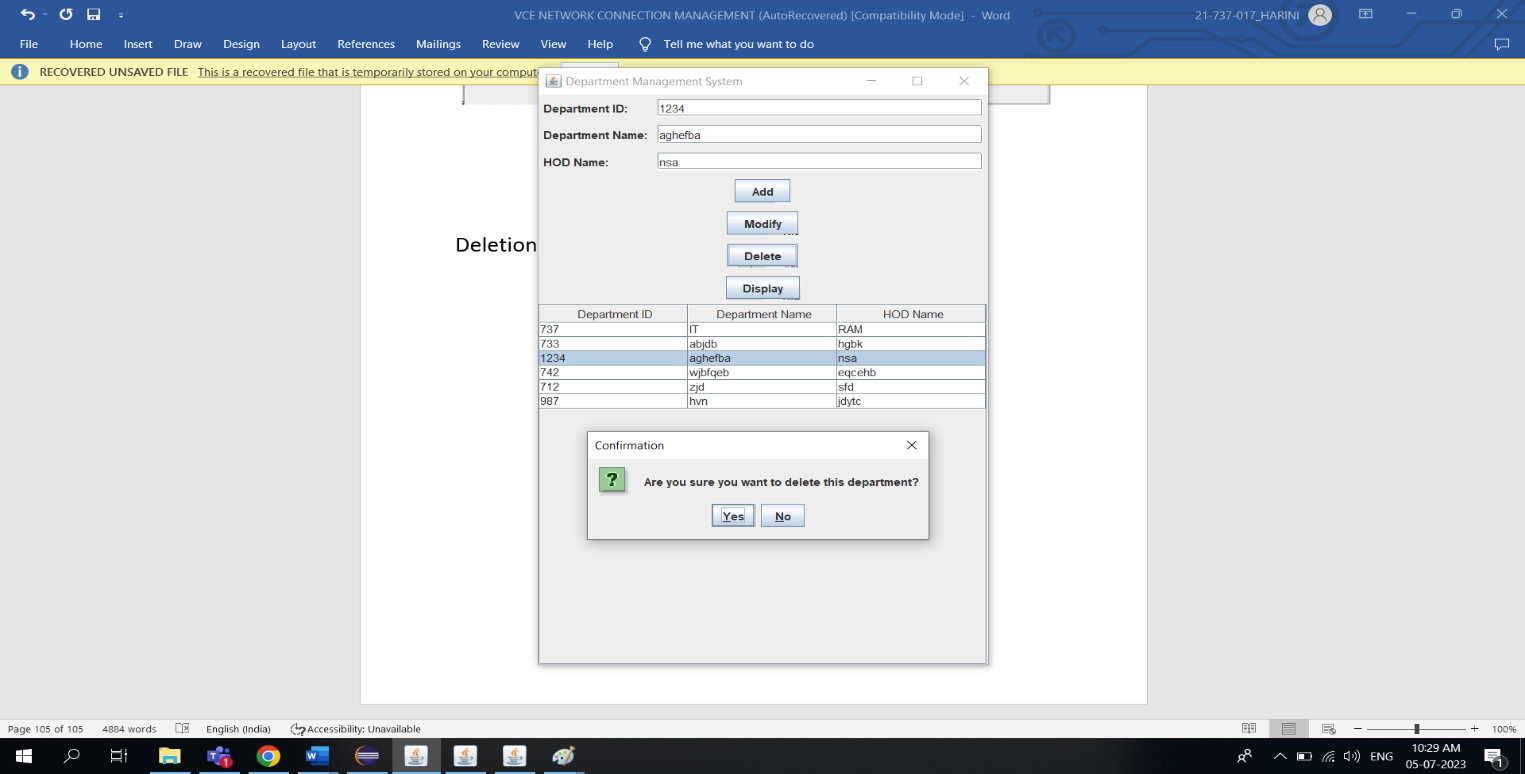


**Department Page:**

Before insertion : After insertion:

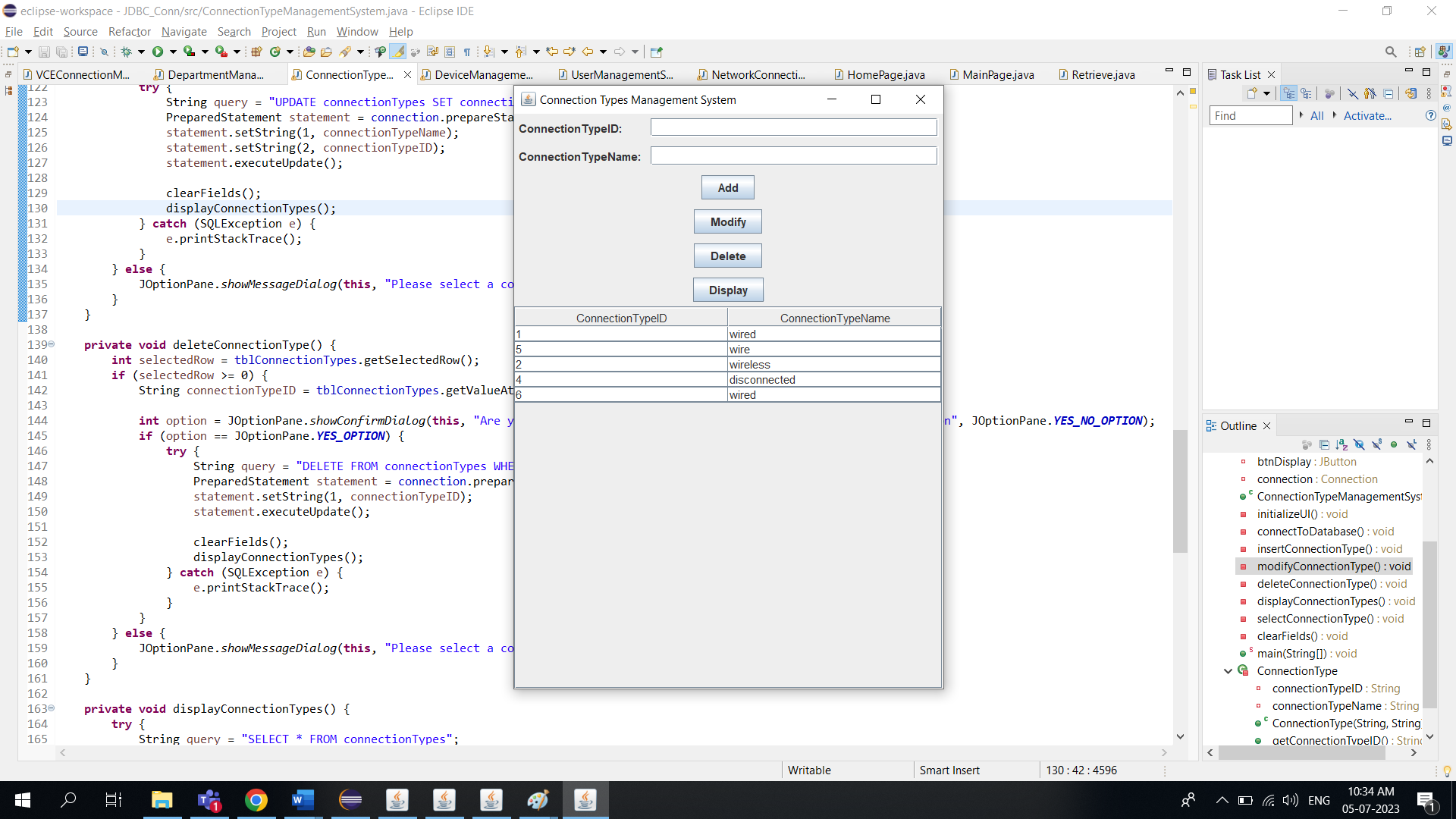
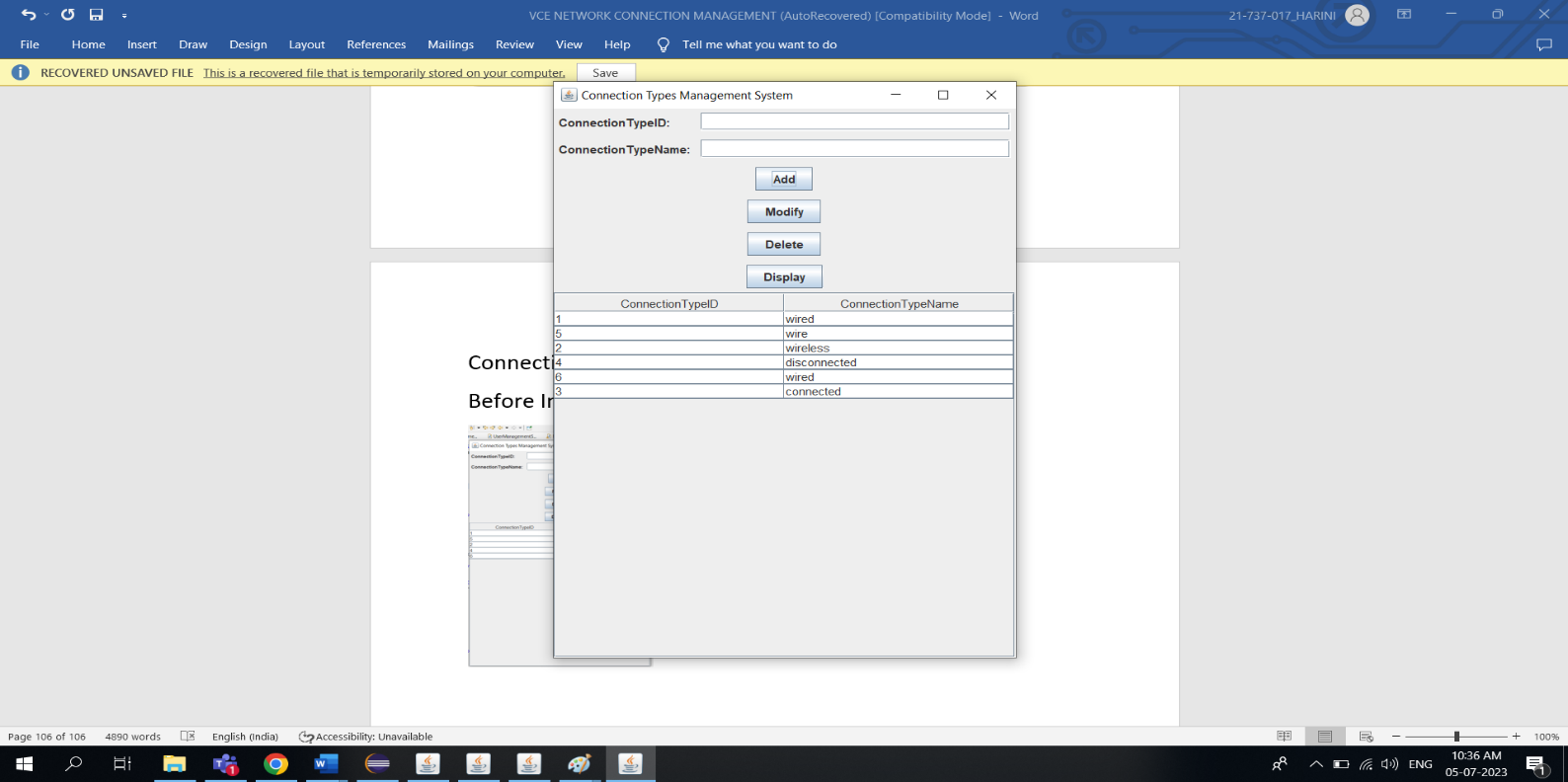
**** 

Modification: Deletion:

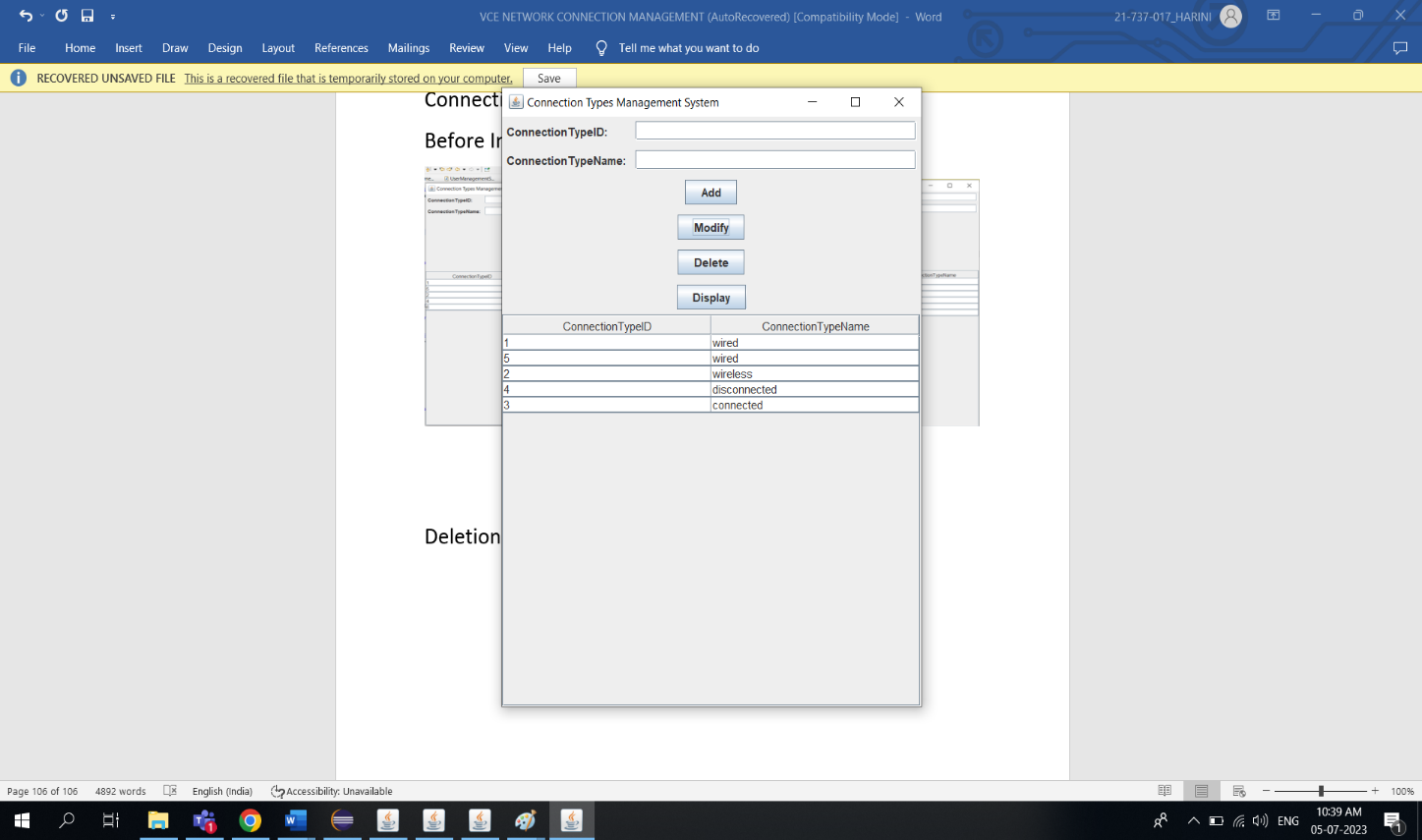
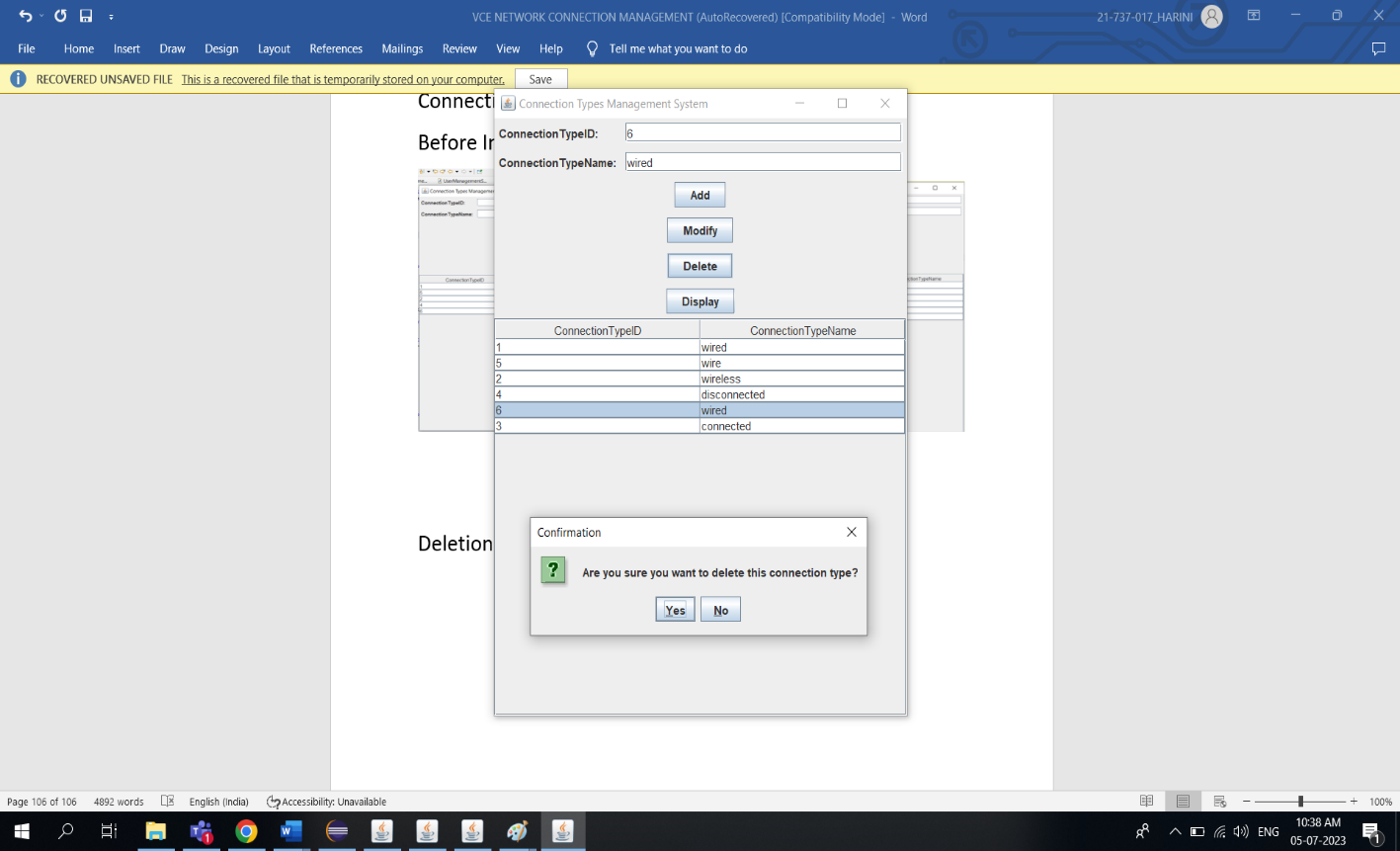
 

**ConnectionTypes Page:**

Before Insertion: After Insertion:

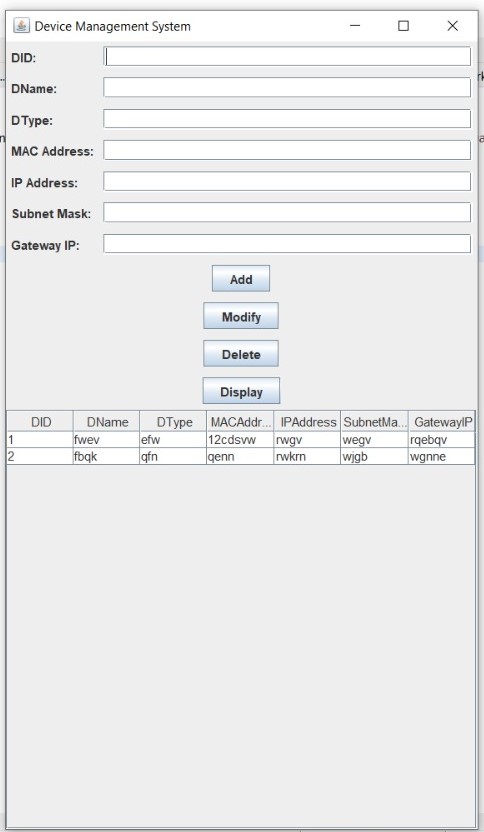
 

Modification: Deletion:

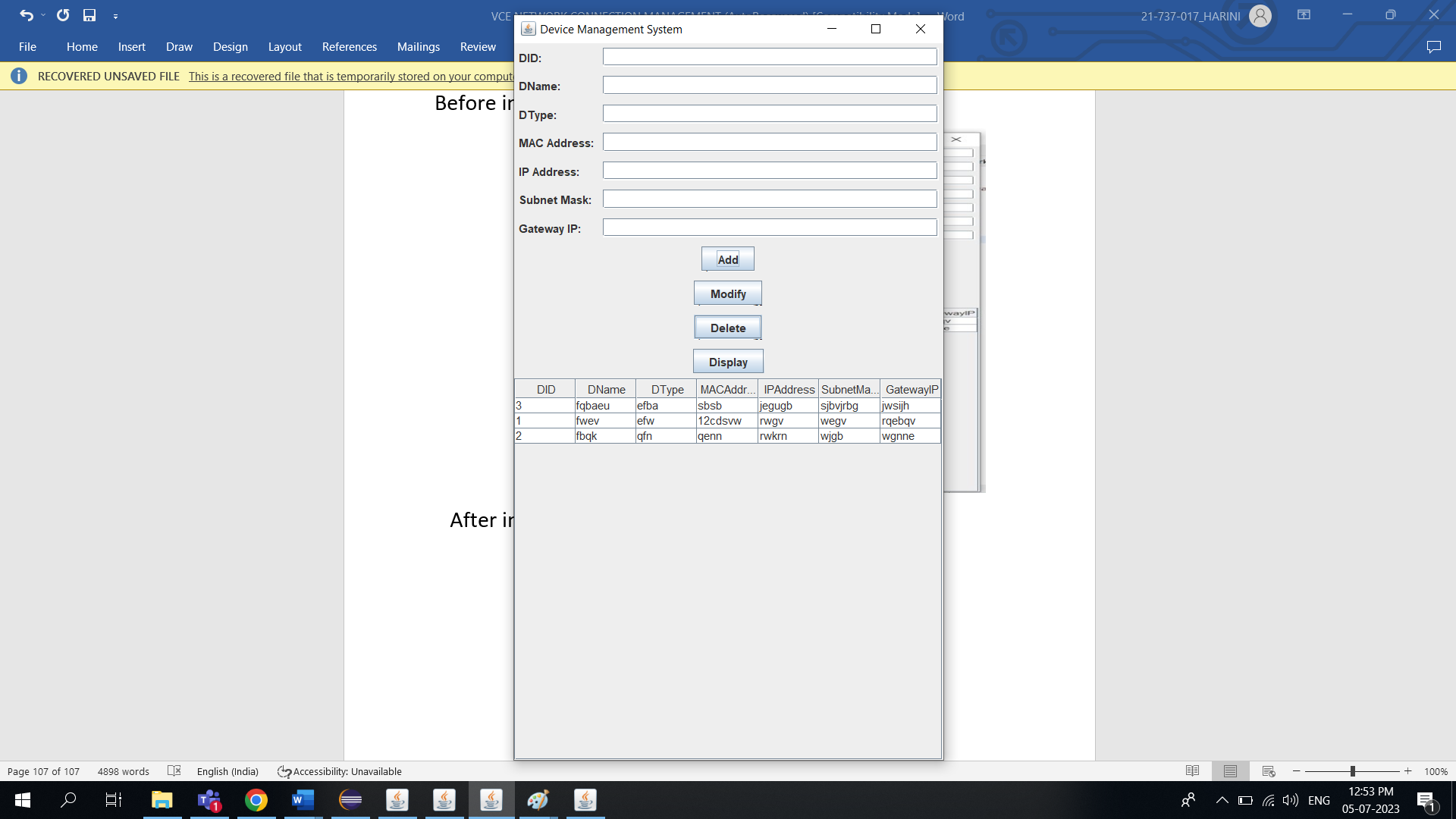
 

Devices Page:

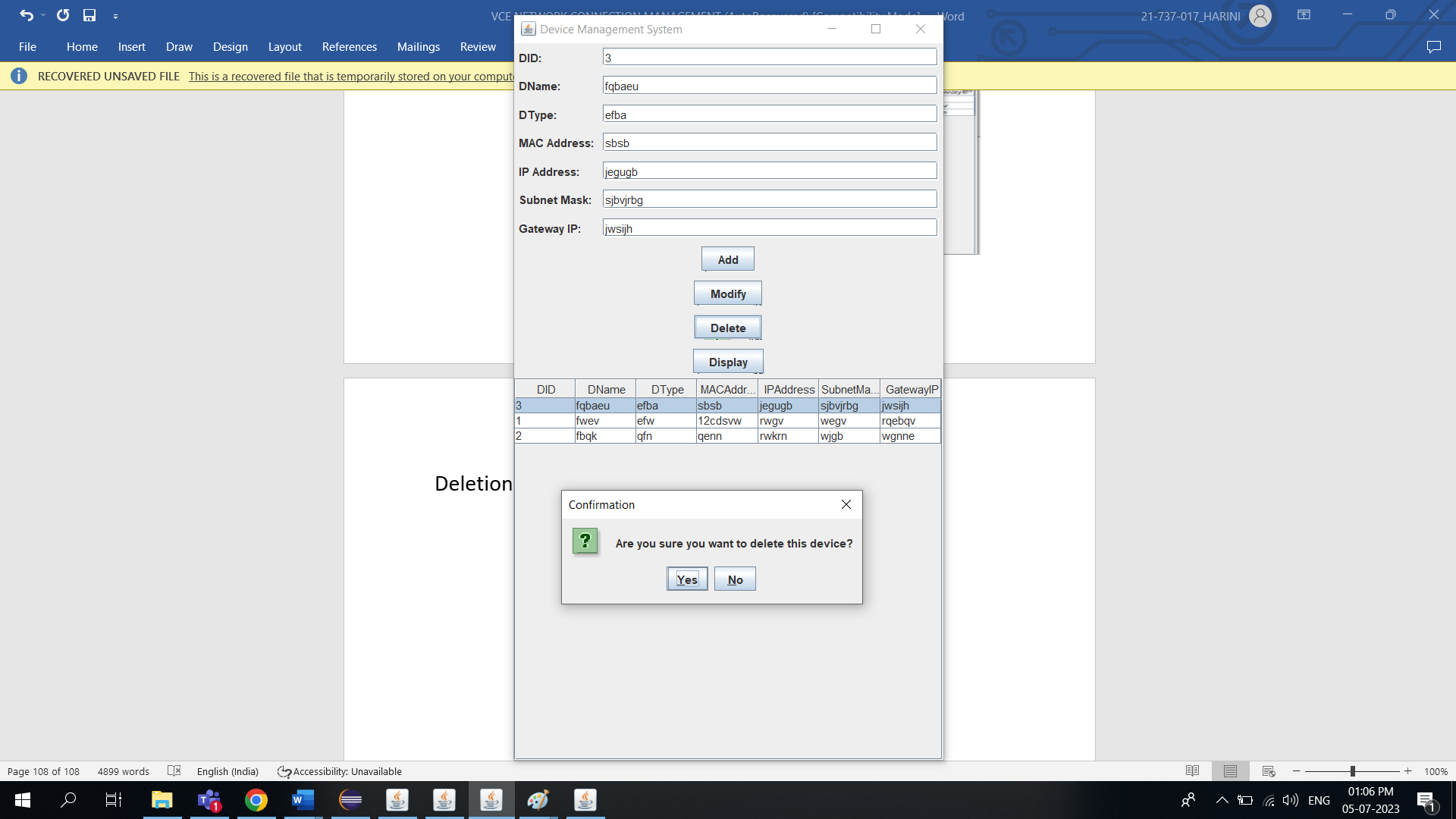
Before insertion:



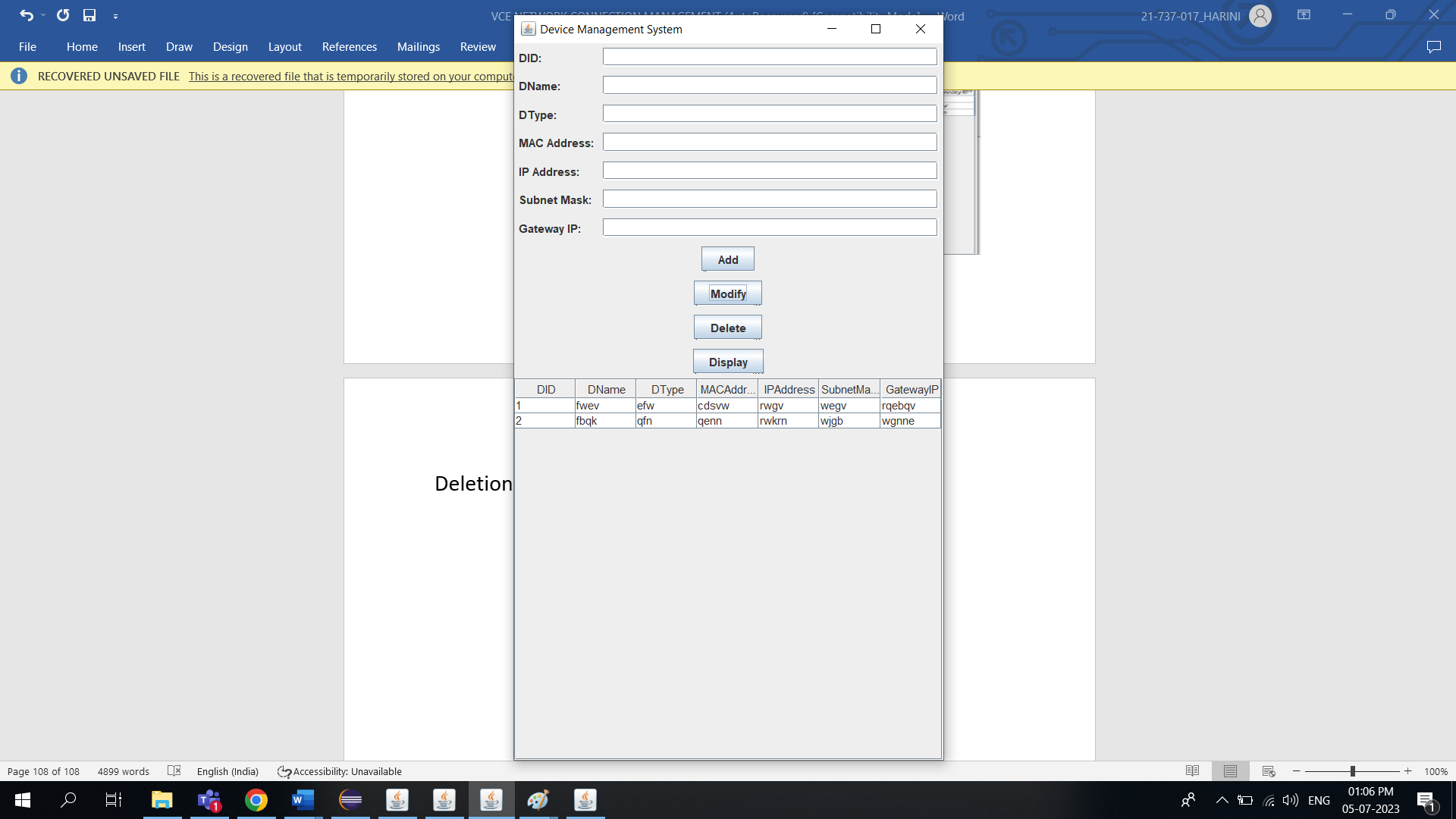
After insertion:



Deletion:

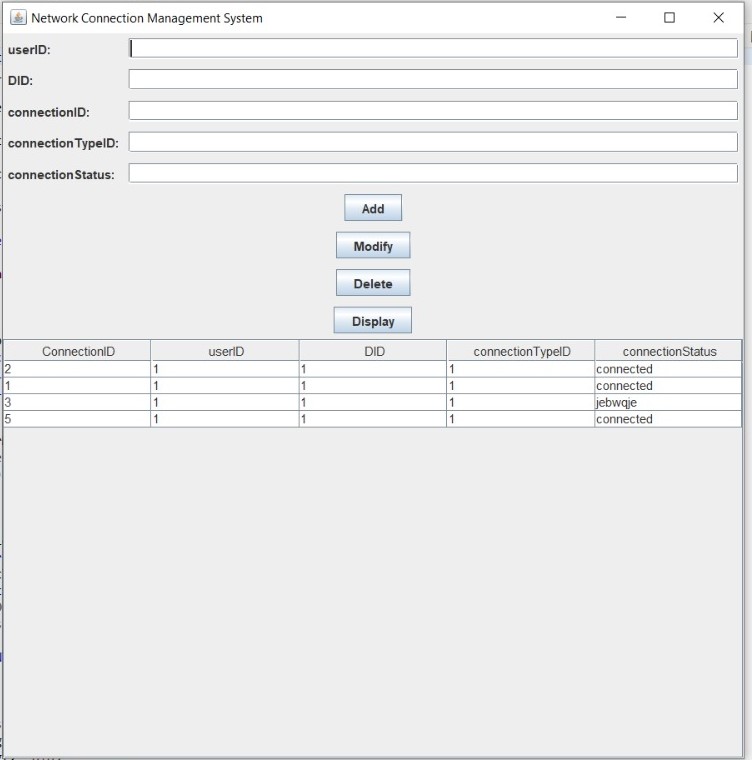


Modification:

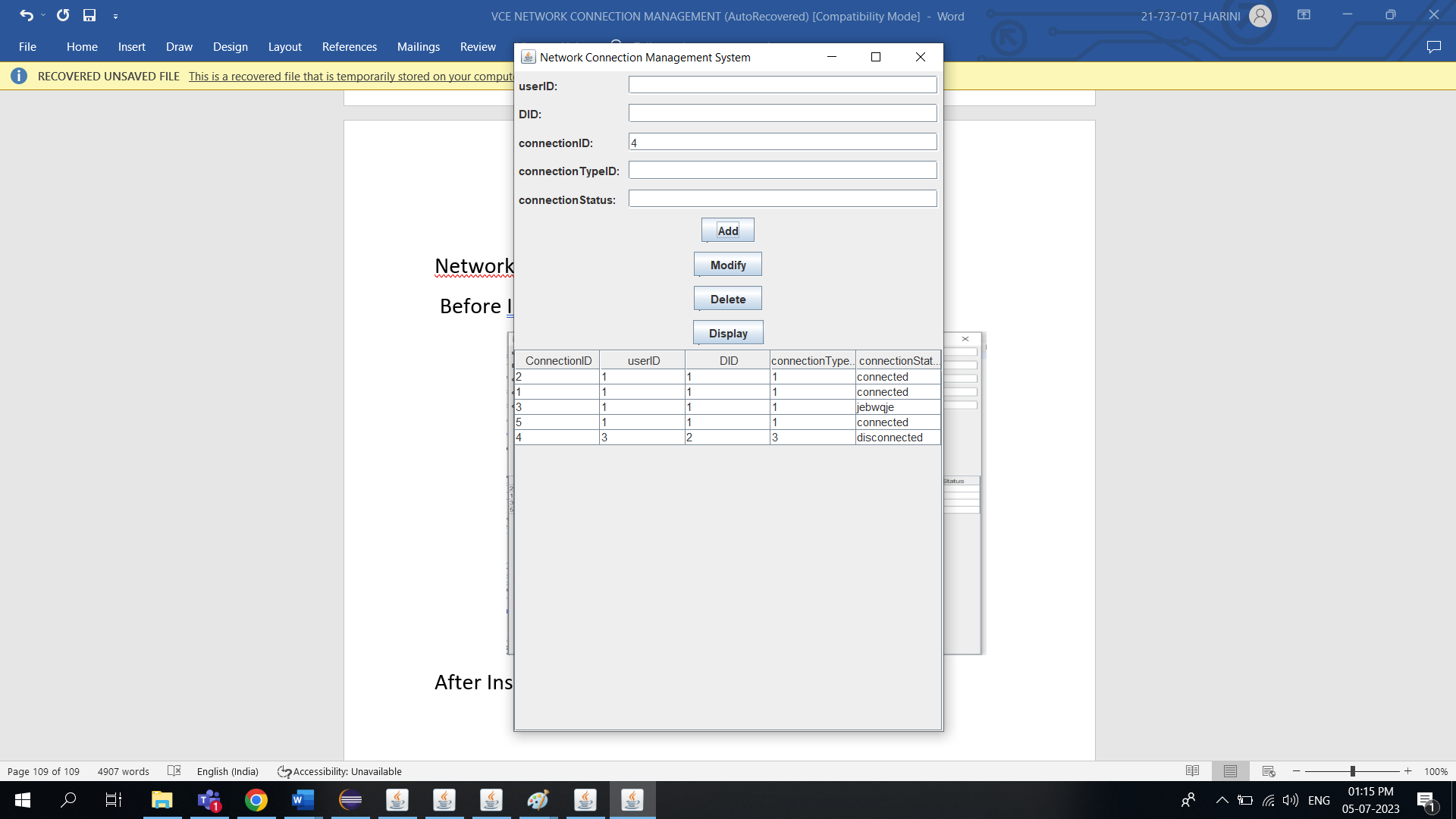


NetworkConnection Page:

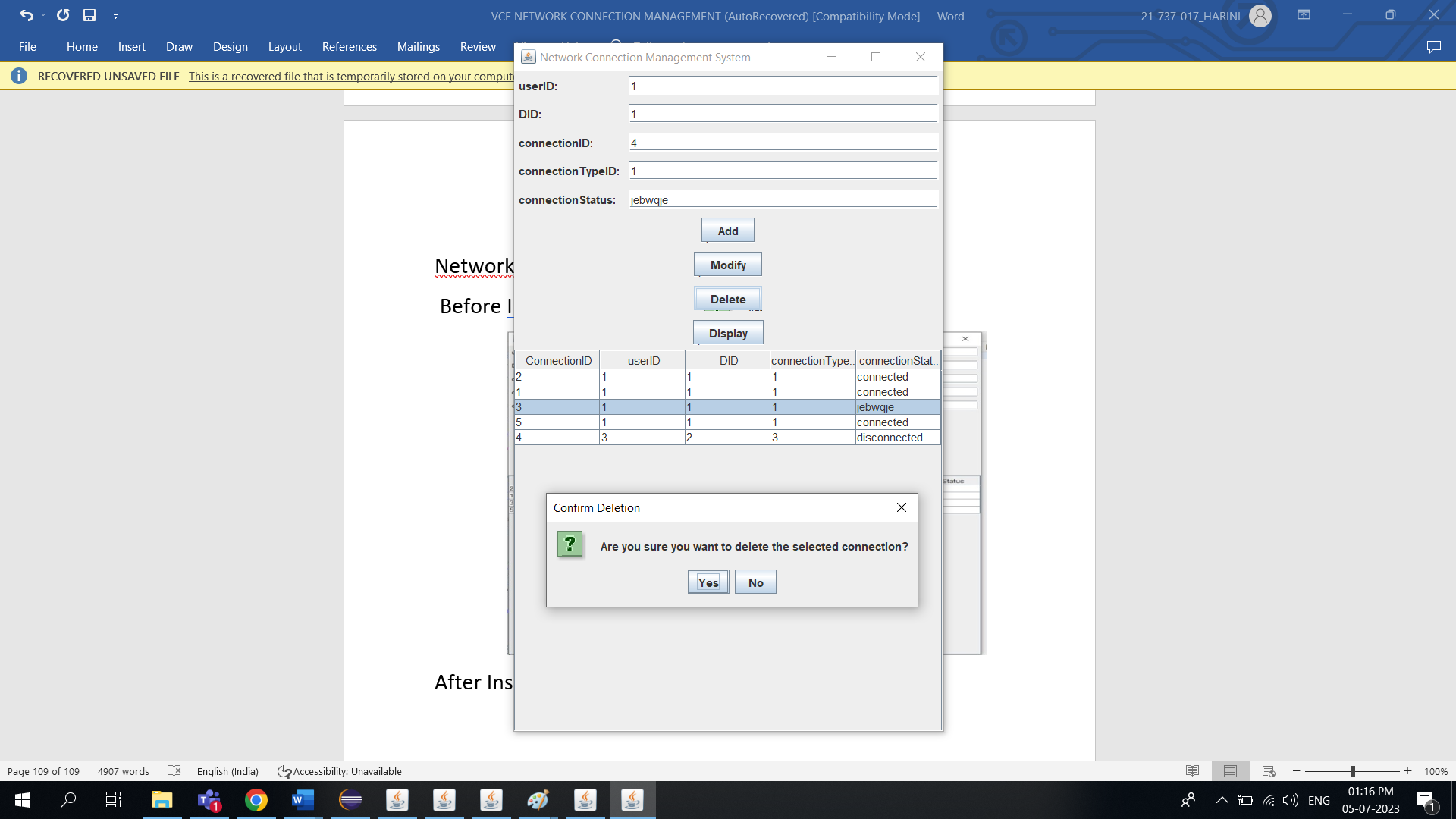
Before Insertion :

****

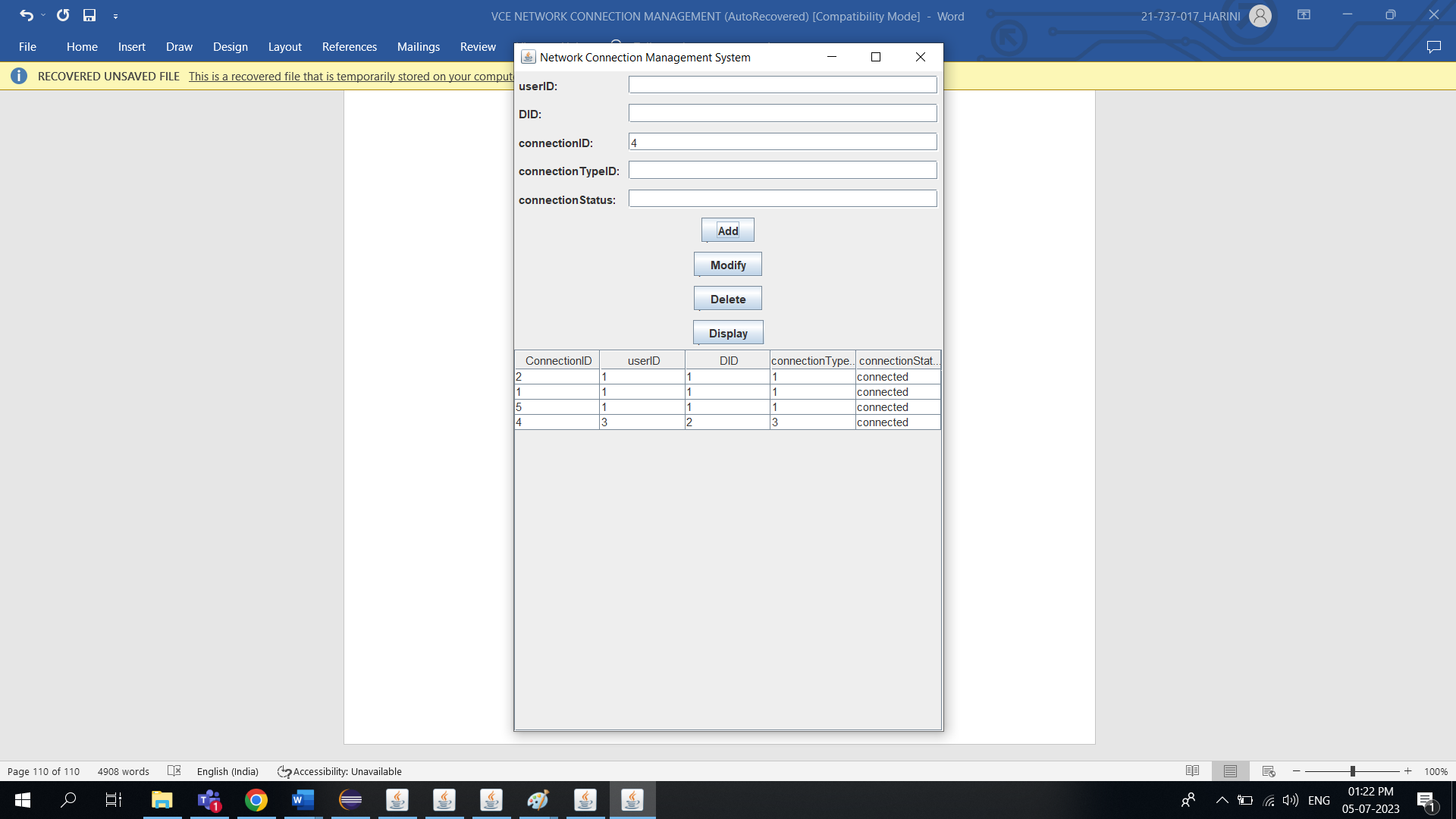
After Insertion:



Deletion:



Modification:



**RESULT**

I have successfully completed my DBMS project ‘***VCE Network Connection System***’.

**DISCUSSION AND FUTURE WORK**

The network connection system at Vasavi College of Engineering plays a vital role in facilitating communication and internet access for its stakeholders. To improve this system, several key areas need attention. Firstly, conducting a thorough assessment of the existing infrastructure will provide insights into its capacity, reliability, and security. Enhancing network security measures, such as regular audits and implementing firewalls, will protect against cyber threats. Expanding and optimizing Wi-Fi coverage throughout the campus will address the increasing demand for wireless connectivity. Collaborating with internet service providers to explore options for high-speed internet connections can further enhance the network system. Promoting digital literacy among the college community will encourage responsible and effective use of the network resources. Regular maintenance and monitoring of the network infrastructure will help identify and address issues proactively. Exploring the potential of emerging technologies, such as Internet of Things (IoT) devices, can bring innovative solutions to campus connectivity. Finally, seeking feedback from users and stakeholders to identify areas for improvement will contribute to the ongoing enhancement of the network connection system at Vasavi College of Engineering.

**References**

* <https://docs.oracle.com/javase/7/docs/api/>
* <https://www.javatpoint.com/java-swing>
* https://stackoverflow.com/

**Github Link:**

https://github.com/R-Harini2003/VCE-Connection-System