**Biometric based automated attendance management system**

*A*

*Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

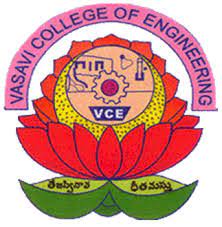
**IN**

**INFORMATION TECHNOLOGY**

**By**

**G.SURYATEJA <1602-21-737-059>**

**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

‘**Biometric based automated attendance management system**’

is a project work of ***G.SURYATEJA***

bearing roll no. 1602-21-737-059

who carried out

this project under my supervision

in the IV semester

of the academic year 2022- 2023

Signature Signature

External Examiner Internal Examiner

**DATABASE MANAGEMENT SYSTEM**

ASSIGNMENT-1

PROJECT TITLE: ‘BIOMETRIC BASED AUTOMATED ATTENDANCE MANAGEMENT SYSTEM’

**ABSTARCT**:

A biometric-based automated attendance management system is an innovative solution to the traditional manual attendance management methods that are prone to errors, time-consuming, and lack accuracy. This system aims to simplify the attendance tracking process and enhance the security of attendance data using biometric technology. The system is designed to capture and store biometric data of students or employees, such as fingerprint or facial recognition, and use it to automate attendance taking in real-time. The system provides several benefits such as reduced paperwork, timely and accurate attendance tracking, and improved security of attendance data. This paper presents an overview of a biometric-based automated attendance management system, including its key features, functionality, and benefits. It also discusses the challenges and limitations associated with the implementation of such a system and provides recommendations to overcome these challenges. Overall, the biometric-based automated attendance management system represents a significant advancement in attendance management, offering a more efficient, accurate, and secure method of tracking attendance.

**REQUIREMENTS:**

Over all 3 tables are required for the ER model representation, they are:

1.Employees

2.Attendance

3.BiometricData

**Table 1: EMPLOYEES**

**Attributes**

* emp\_id (primary key)
* emp\_name (varchar)
* emp\_email (varchar)
* emp\_dept (varchar)
* emp\_position (varchar)

**Domain Types:**

* emp\_id:integer
* emp\_name, emp\_email, emp\_dept, emp\_position:varchar

**constraints:**

primary key: emp\_id

**Table 2: ATTENDANCE**

**Attributes:**

* attendance\_id (number)
* emp\_id(number)
* date (date)

- time\_in (time)

- time\_out(time)

**Domain Types**:

- attendance\_id, emp\_id:integer

-date:date

-time\_in,time\_out:time

**Constraints:**

-prinmary key: attendance\_id

Foreign key: emp\_id

**Table 3:** **BiometricData**

**Attributes:**

**-** bio\_id(number)

- emp\_id(number)

- fingerprint\_data(varchar)

**Domain Types:**

- bio\_id, emp\_id:integer

- fingerprint\_data:varchar

**Constraints:**

Primary key: bio\_id

Foreign key: emp\_id

**Table 4:** **ShiftSchedules**

**Attributes:**

-ScheduleID INT PRIMARY KEY,

-Emp\_ID INT,

-StartTime varchar(30),

-EndTime varchar(30),

**Domain Types:**

ScheduleID , Emp\_ID:Integer

StartTime , EndTime:varchar

**Constraints:**

Primary key: ScheduleID

Foreign key: Emp\_ID

**Table 5:** **Departments**

**Attributes:**

-DepartmentID INT ,

-DepartmentName VARCHAR(255) ,

-Description VARCHAR(255)

**Domain Types:**

-DepartmentID :integer ,

-DepartmentName ,Description : VARCHAR

**Constraints:**

Primary key: DepartmentID

**Mapping Cardinalities:**

One Employee can have many Attendance records (1:M)

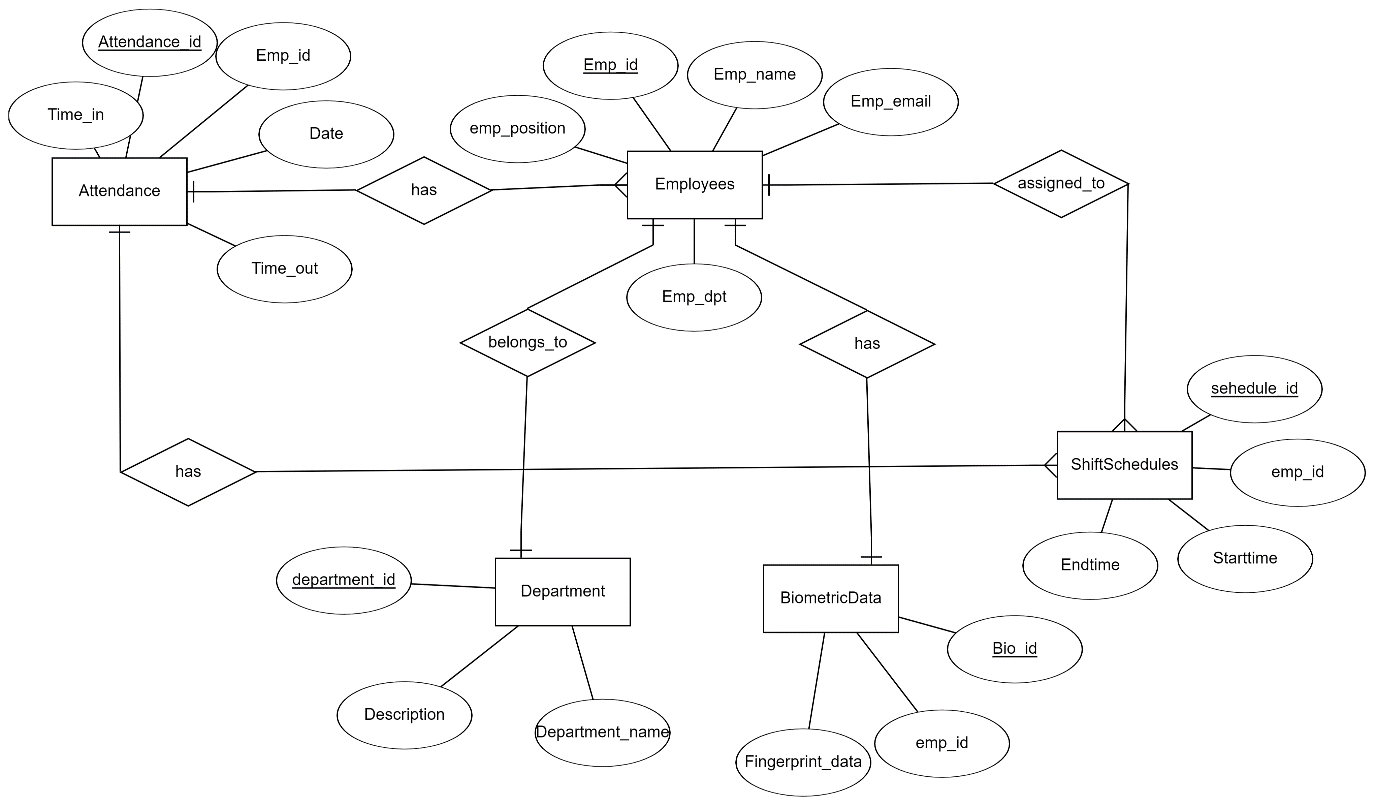
One Employee can have one Biometric Data record (1:1)

One Employee can belong to only one Department (1:1)

One Shift Schedule can have many Attendance records (1:M)

One Shift Schedule can be assigned to many Employees (M:M)

**ER DAIGRAM:**



**DDL COMMANDS:**

**EMPLOYEES TABLE:**

**a)create():**

SQL> create table Employees()

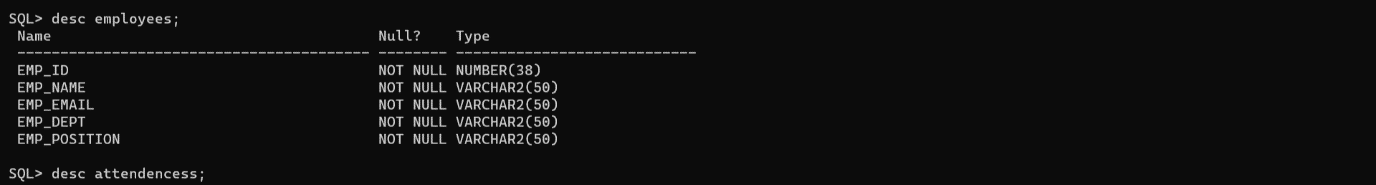
2 emp\_id number,

3 emp\_name varchar(50),

4emp\_email varchar(50),

5 emp\_dept varchar(50),

6 emp\_position varchar(50),

Table created. 

**ATTENDANCE TABLE:**

SQL> create table Attendance(

2 attendance number,

3 emp\_id number,

4 date date,

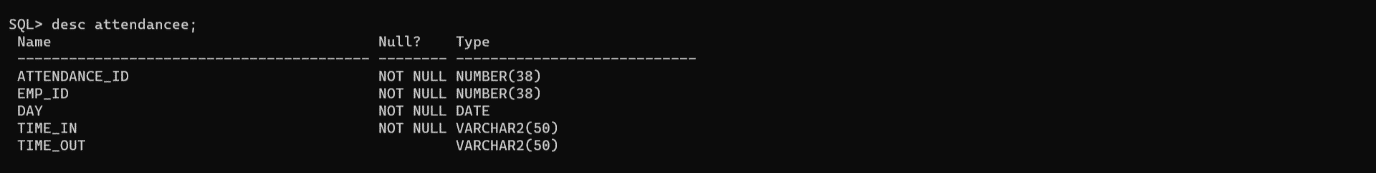
5 time\_in time,

6 time\_out time,

7 foreign key (emp\_id)references employees(emp\_id)

8 );

Table created.



****

**BIOMETRIC DATA TABLE:**

SQL> SQL> create table biometric data(

2 bio\_id number,

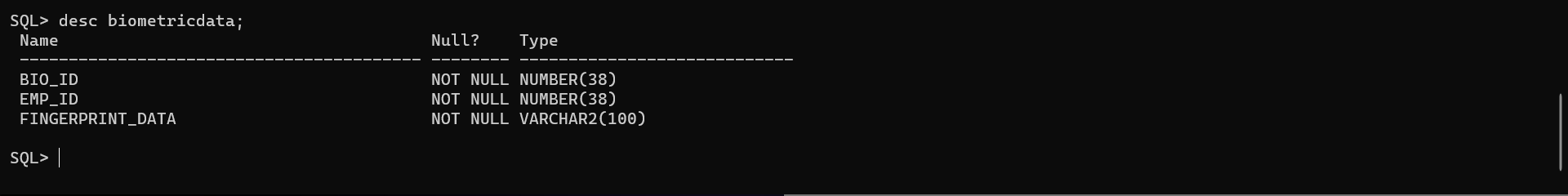
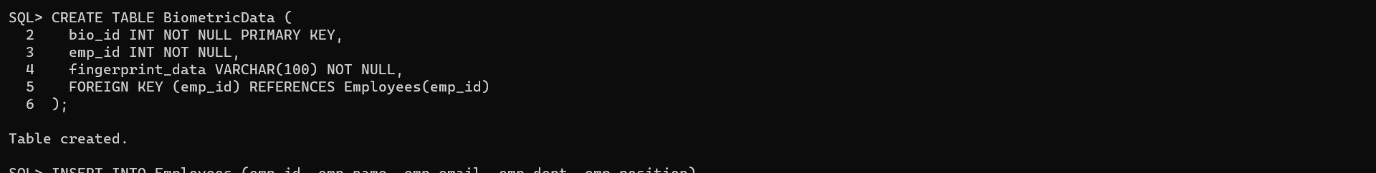
3 emp\_id number,

4 fingerprint\_data varchar,

5 foreign key(emp\_id)

6 );

Table created.

****

**SHIFTSCHEDULES TABLE:**

SQL> SQL> create table ShiftSchedules(

2 ScheduleID INT PRIMARY KEY,

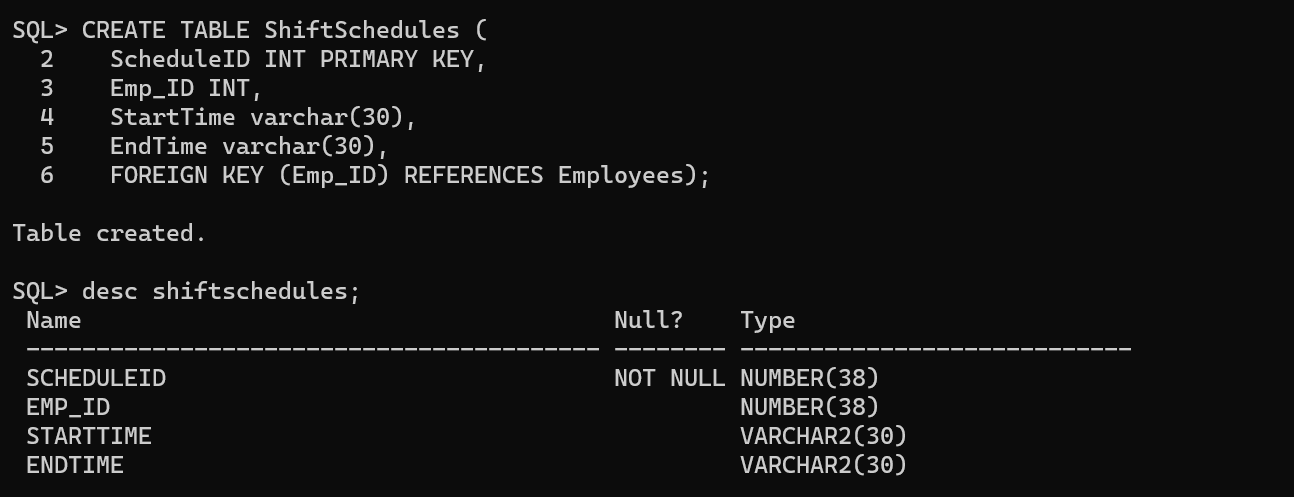
3 Emp\_ID INT,

4 StartTime varchar(30),

5 EndTime varchar(30),

6 FOREIGN KEY (Emp\_ID) REFERENCES Employees);

Table created.

****

**DEPARTMENT TABLE:**

SQL> SQL> create table Department(

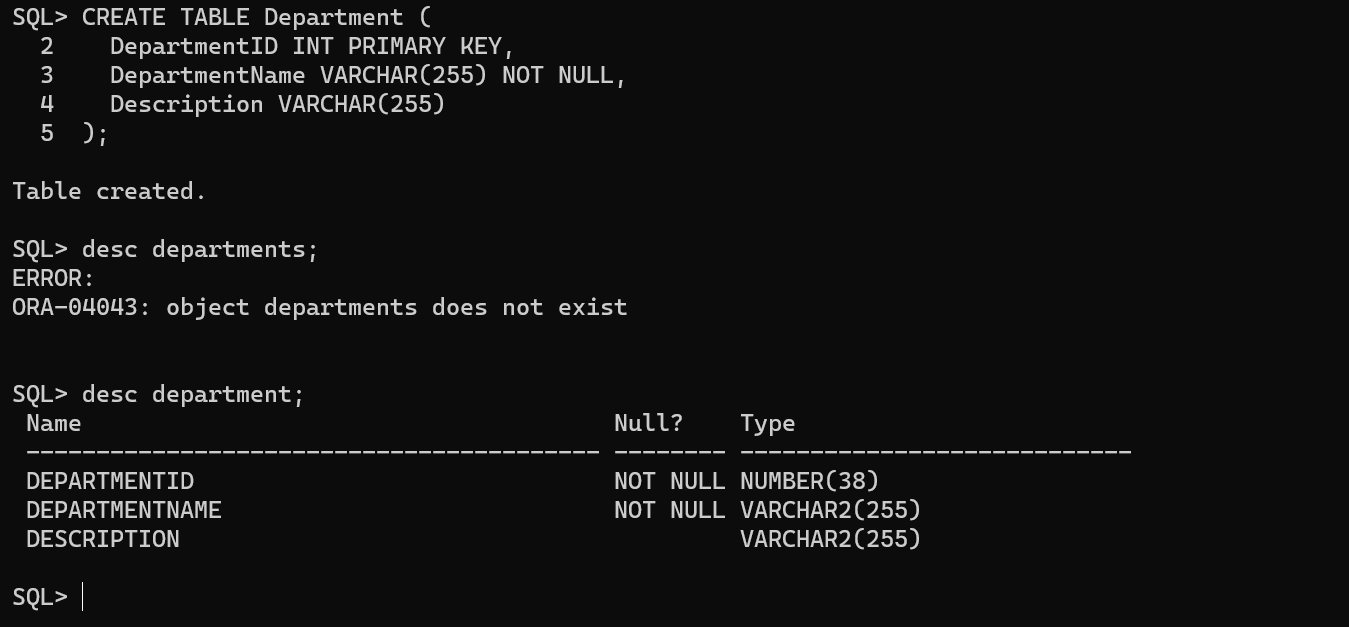
2 DepartmentID INT PRIMARY KEY,

3 DepartmentName VARCHAR(255) NOT NULL,

4 Description VARCHAR(255)

5 );

Table created.

****

**b)alter():**

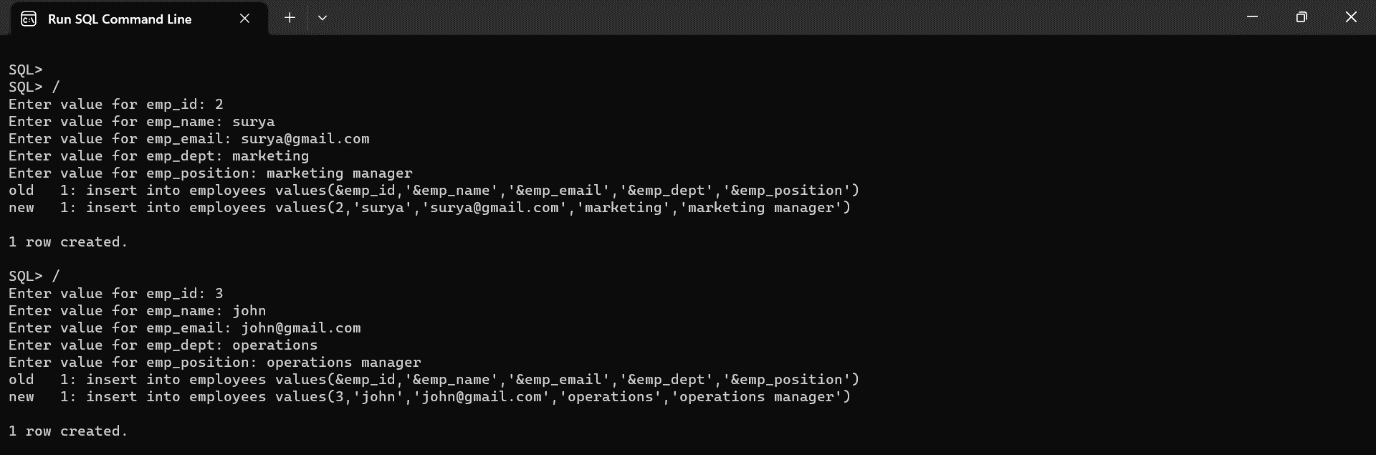
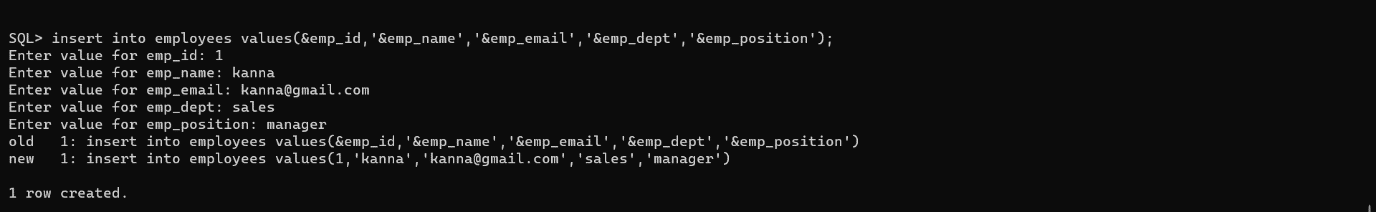
****

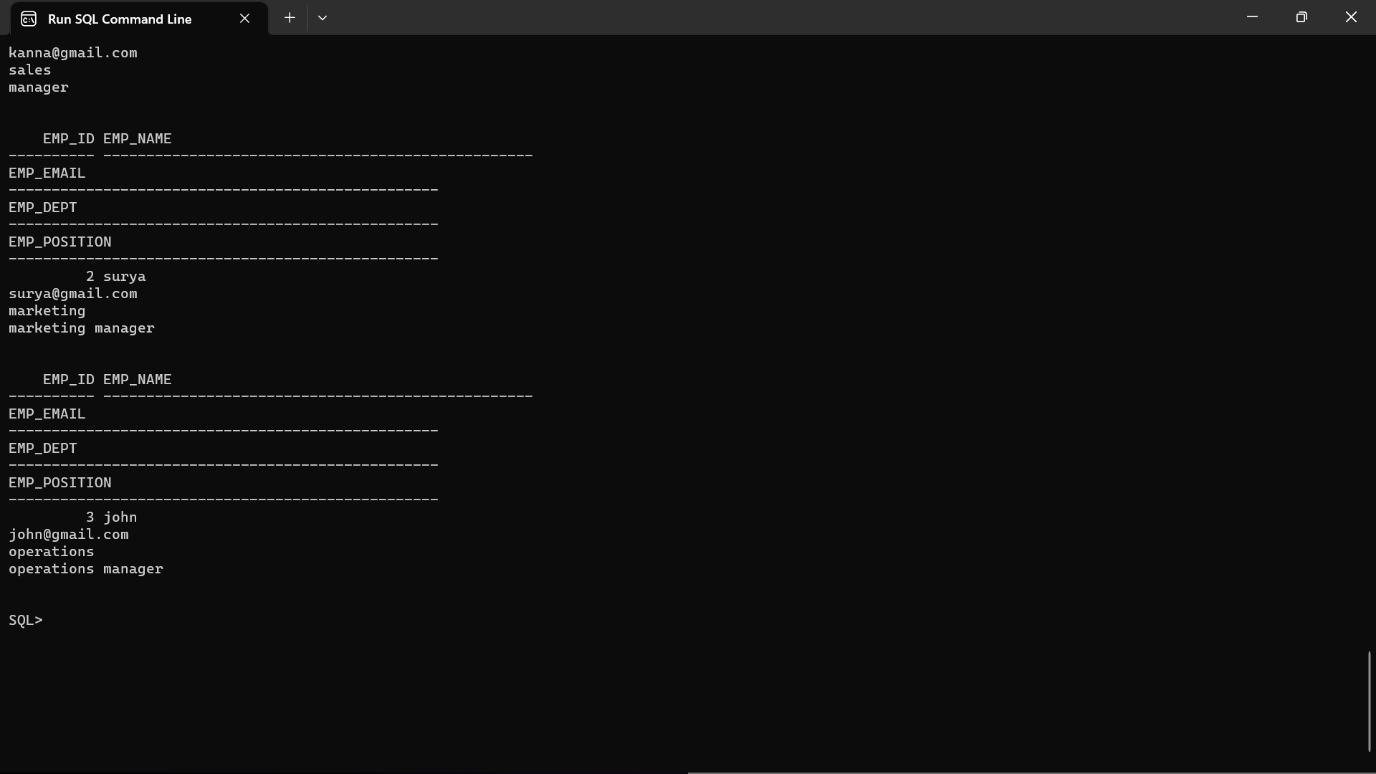
**C)drop:**

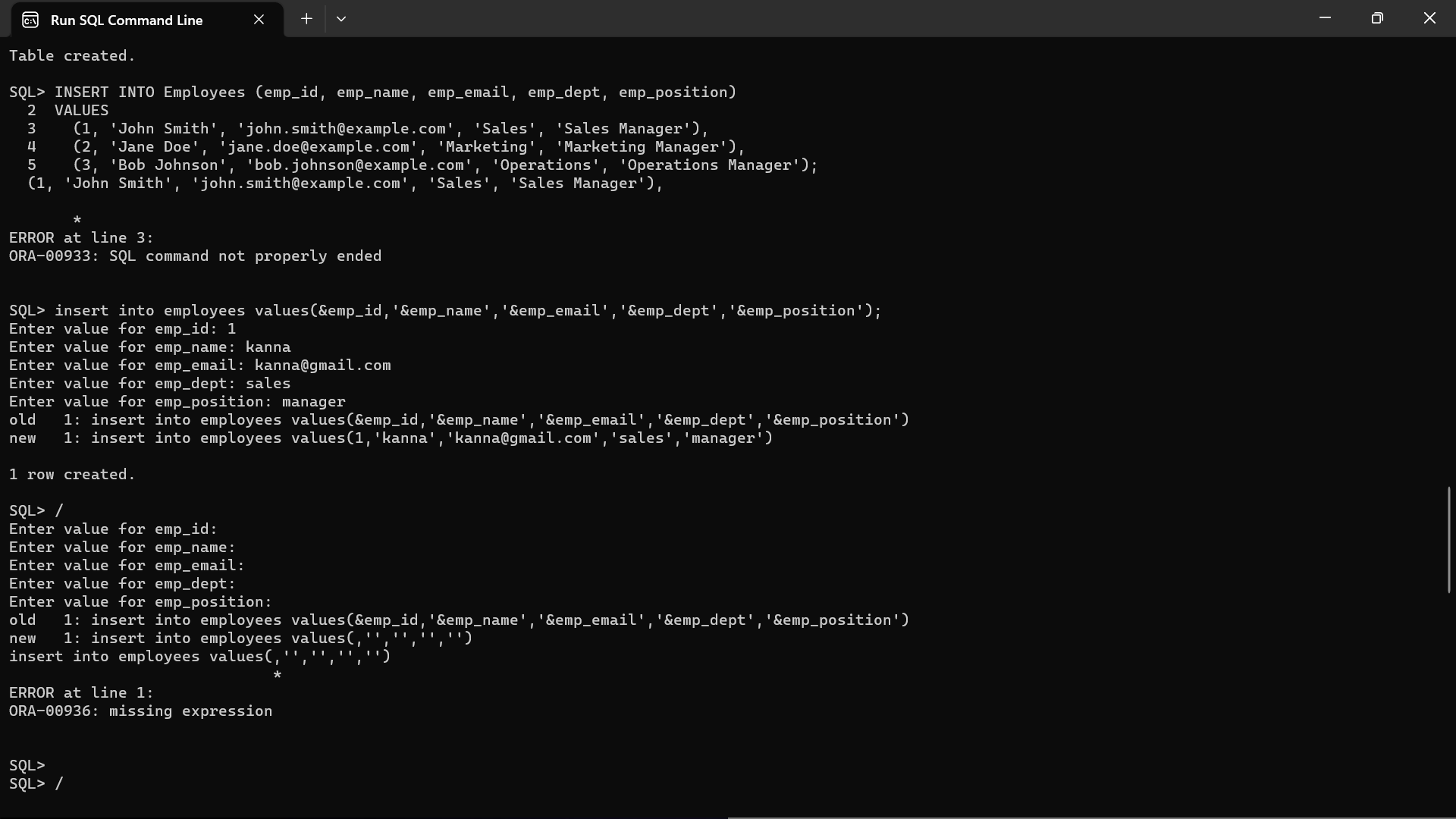
****

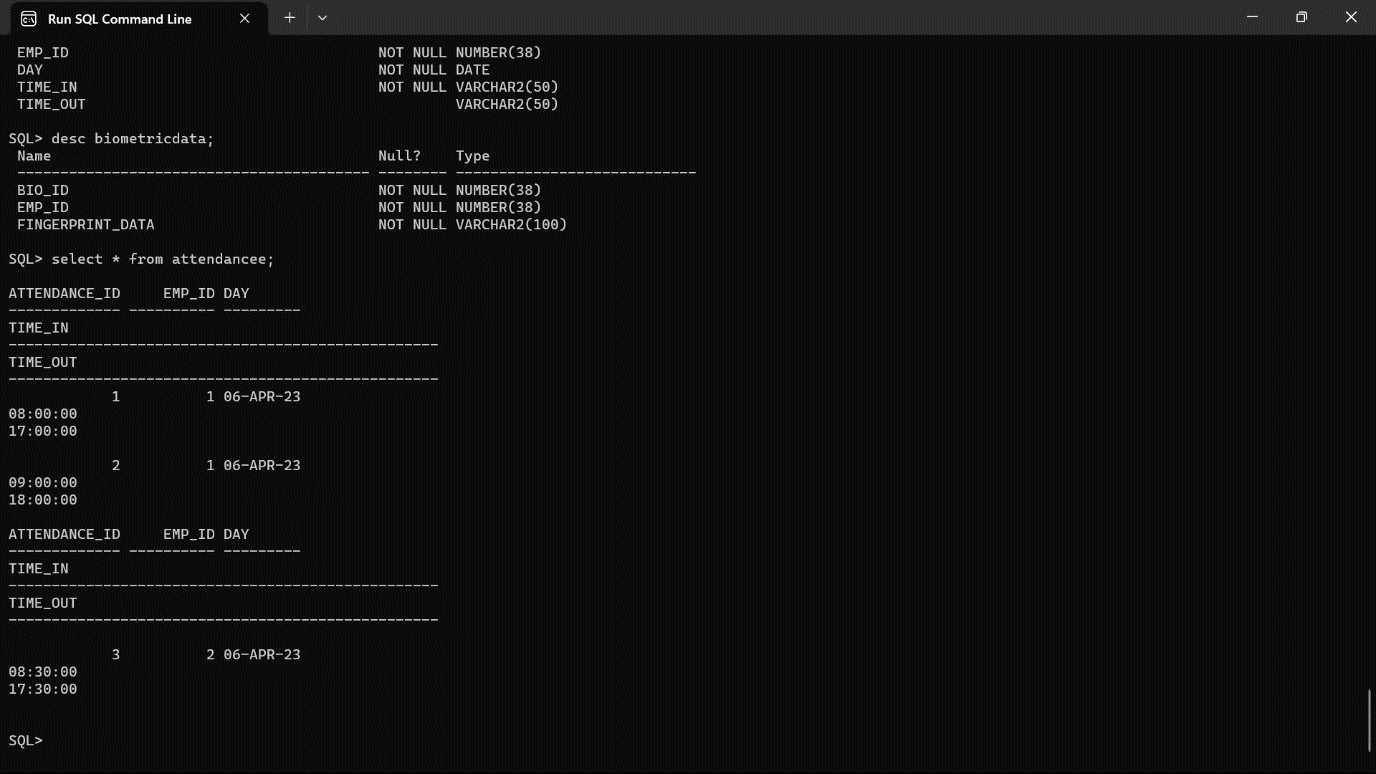
**DML COMMANDS:**

**EMPLOYEES TABLE:**

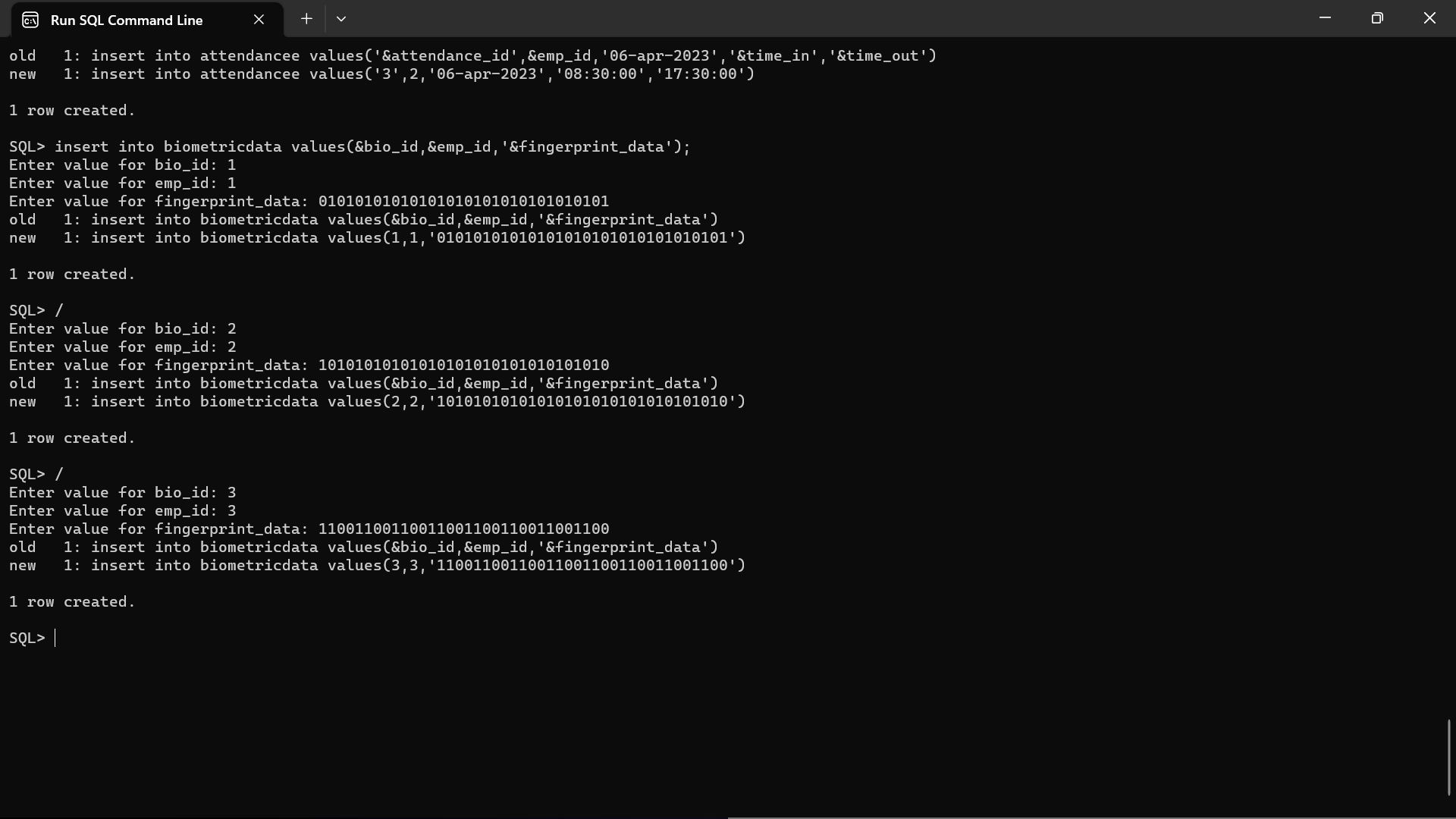


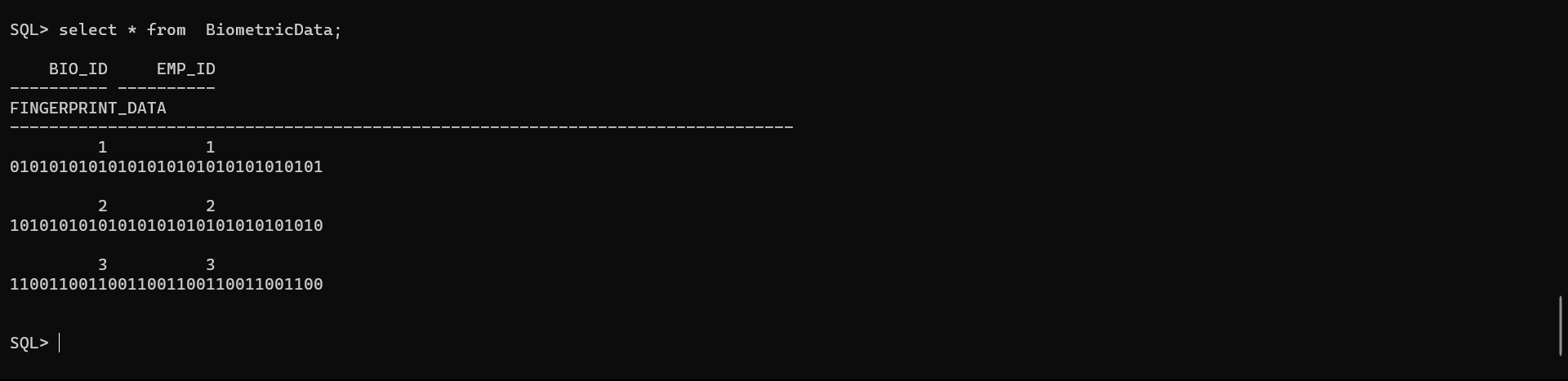
****

**ATTENDANCE TABLE:**

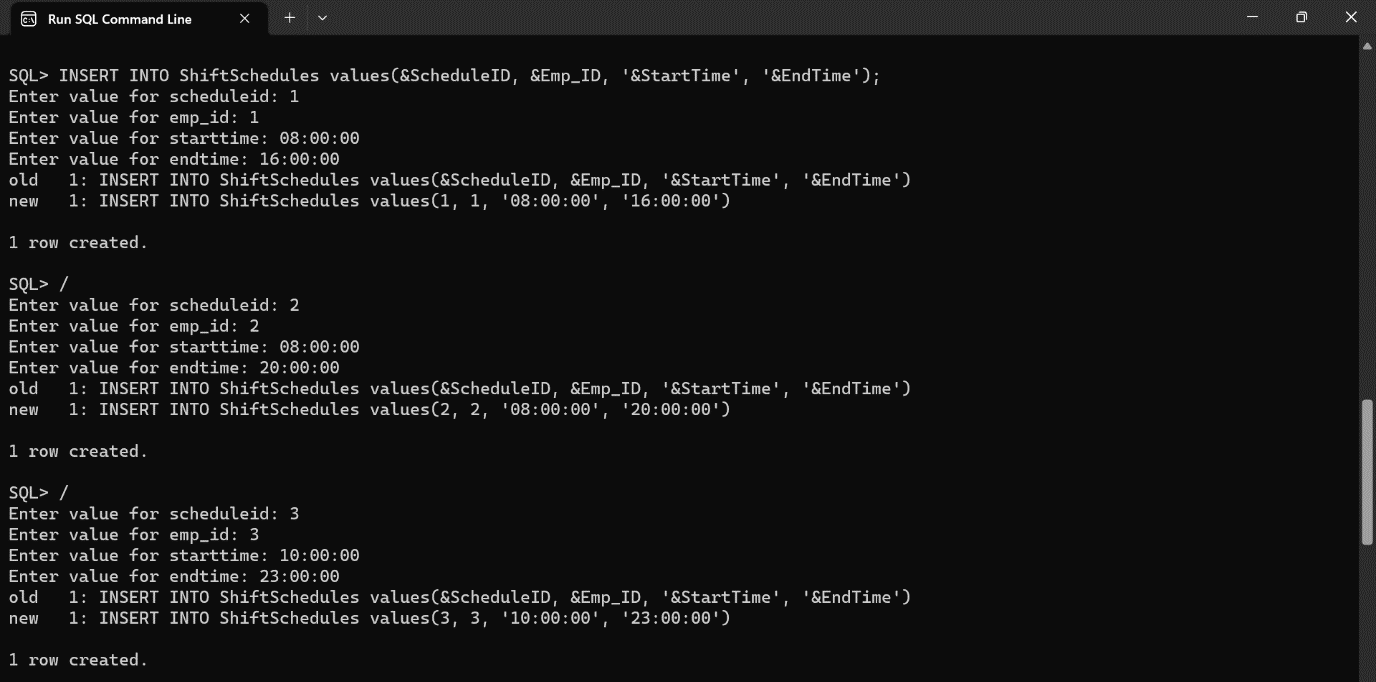
****

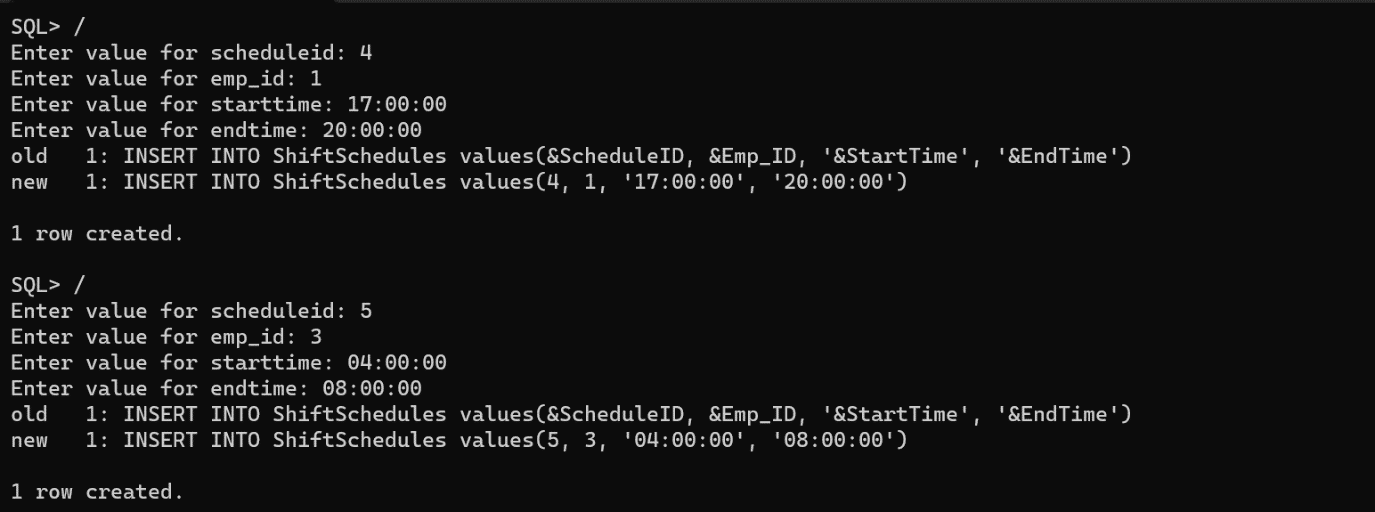
**BIOMETRIC DATA TABLE:**

****

****

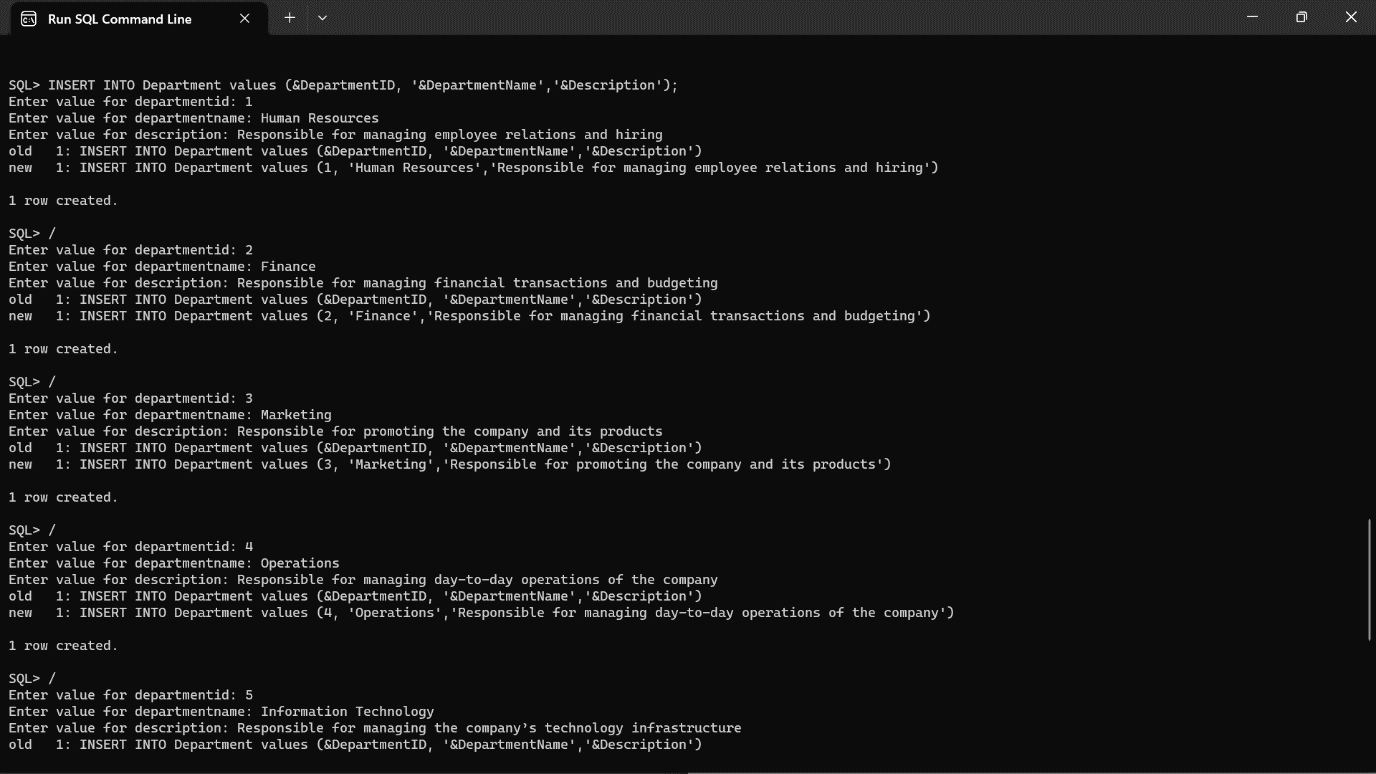
**SHIFTSCHEDULES TABLE:**

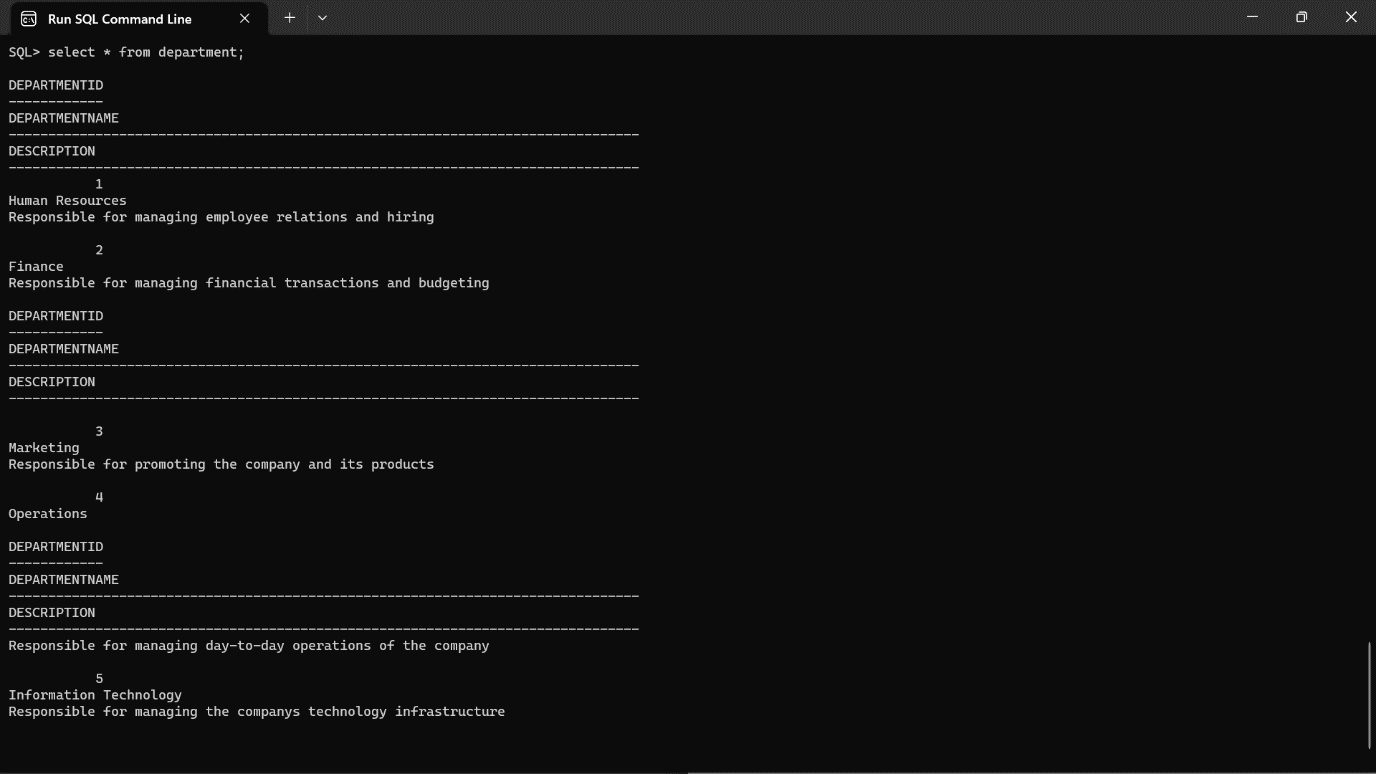
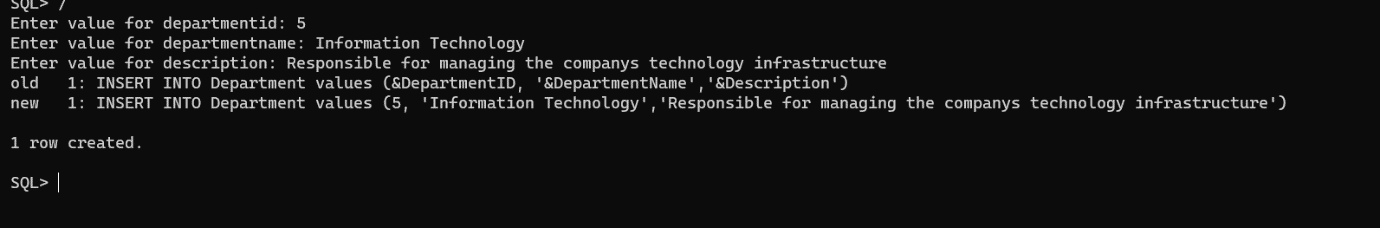
****

****

****

**DEPARTMENTS TABLE:**

****

****

**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 = "surya";

private static String password = "kanna";

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 retrievedetailsButton;

public MainPage() {

// Set frame properties

setTitle("biometric based automated attendance management system");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create label

JLabel welcomeLabel = new JLabel("biometric based automated attendance management 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();

retrievedetailsButton = new JButton("Retrieve Marks");

buttonPanel.add(retrievedetailsButton);

// Create menu bar

JMenuBar menuBar = new JMenuBar();

// Create menus

JMenu EmployeesMenu = new JMenu("Employees Details");

JMenu AttendanceMenu = new JMenu("Attendance Details");

JMenu biometricDataMenu = new JMenu("biometricData Details");

JMenu ShiftSchedulesMenu = new JMenu("ShiftSchedules Details");

JMenu DepartmentMenu = new JMenu("Department Details");

// Create menu item for student menu

JMenuItem viewEmployeesDetails = new JMenuItem("View Employees Details");

viewEmployeesDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new EmployeesTableGUI();

}

});

// Create menu item for course menu

JMenuItem viewAttendanceDetails = new JMenuItem("View Attendance Details");

viewAttendanceDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new AttendanceTableGUI();

}

});

// Create menu item for enrollment menu

JMenuItem viewbiometricDataDetails = new JMenuItem("View biometricData Details");

viewbiometricDataDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new BiometricDataGUI();

}

});

// Create menu item for semester menu

JMenuItem viewShiftSchedulesDetails = new JMenuItem("View ShiftSchedules Details");

viewShiftSchedulesDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new ShiftSchedulesGUI();

}

});

// Create menu item for grade menu

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

viewDepartmentDetails.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

new DepartmentTableGUI();

}

});

// Add menu items to respective menus

EmployeesMenu.add(viewEmployeesDetails);

AttendanceMenu.add(viewAttendanceDetails);

biometricDataMenu.add(viewbiometricDataDetails);

ShiftSchedulesMenu.add(viewShiftSchedulesDetails);

DepartmentMenu.add(viewDepartmentDetails);

// Add menus to the menu bar

menuBar.add(EmployeesMenu);

menuBar.add(AttendanceMenu);

menuBar.add(biometricDataMenu);

menuBar.add(ShiftSchedulesMenu);

menuBar.add(DepartmentMenu);

// Set the menu bar

setJMenuBar(menuBar);

// Add the button panel to the frame

add(buttonPanel, BorderLayout.CENTER);

// Set button action for "Retrieve Marks"

retrievedetailsButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Retrieve retrieveProgram=new Retrieve();

retrieveProgram.setVisible(true);

}

});

// 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);

}

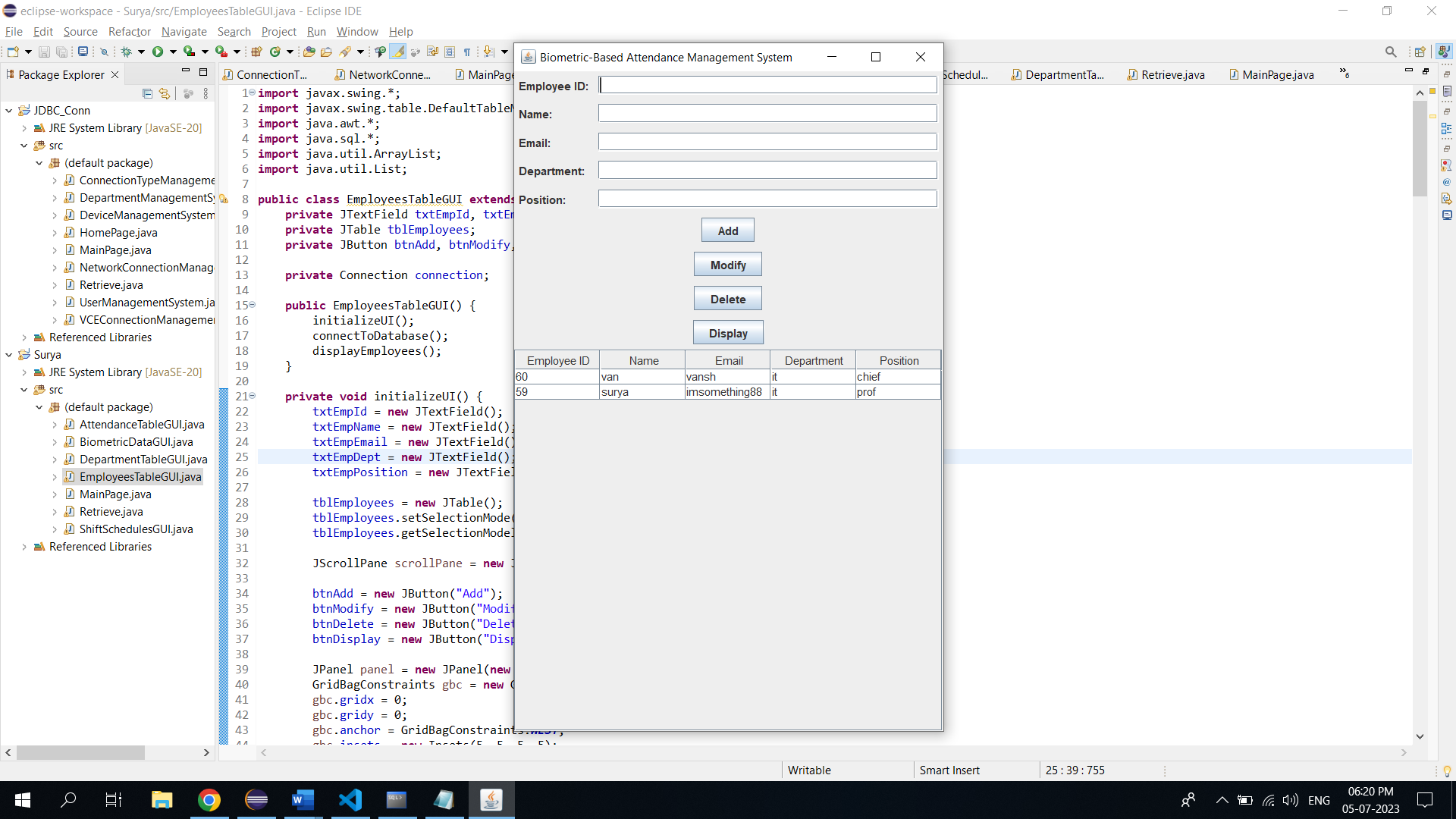
public static void main(String[] args) {

new MainPage();

}

}

**Employees 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 EmployeesTableGUI extends JFrame {

private JTextField txtEmpId, txtEmpName, txtEmpEmail, txtEmpDept, txtEmpPosition;

private JTable tblEmployees;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public EmployeesTableGUI() {

initializeUI();

connectToDatabase();

displayEmployees();

}

private void initializeUI() {

txtEmpId = new JTextField();

txtEmpName = new JTextField();

txtEmpEmail = new JTextField();

txtEmpDept = new JTextField();

txtEmpPosition = new JTextField();

tblEmployees = new JTable();

tblEmployees.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblEmployees.getSelectionModel().addListSelectionListener(e -> selectEmployee());

JScrollPane scrollPane = new JScrollPane(tblEmployees);

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("Employee ID:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtEmpId, gbc);

gbc.gridy++;

panel.add(txtEmpName, gbc);

gbc.gridy++;

panel.add(txtEmpEmail, gbc);

gbc.gridy++;

panel.add(txtEmpDept, gbc);

gbc.gridy++;

panel.add(txtEmpPosition, 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 -> insertEmployee());

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

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

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

setTitle("Biometric-Based Attendance Management System");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "surya";

String password = "kanna";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertEmployee() {

String empId = txtEmpId.getText();

String empName = txtEmpName.getText();

String empEmail = txtEmpEmail.getText();

String empDept = txtEmpDept.getText();

String empPosition = txtEmpPosition.getText();

try {

String query = "INSERT INTO EMPLOYEES (emp\_id, emp\_name, emp\_email, emp\_dept, emp\_position) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, empId);

statement.setString(2, empName);

statement.setString(3, empEmail);

statement.setString(4, empDept);

statement.setString(5, empPosition);

statement.executeUpdate();

clearFields();

displayEmployees();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyEmployee() {

int selectedRow = tblEmployees.getSelectedRow();

if (selectedRow >= 0) {

String empId = txtEmpId.getText();

String empName = txtEmpName.getText();

String empEmail = txtEmpEmail.getText();

String empDept = txtEmpDept.getText();

String empPosition = txtEmpPosition.getText();

try {

String query = "UPDATE EMPLOYEES SET emp\_name=?, emp\_email=?, emp\_dept=?, emp\_position=? WHERE emp\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, empName);

statement.setString(2, empEmail);

statement.setString(3, empDept);

statement.setString(4, empPosition);

statement.setString(5, empId);

statement.executeUpdate();

clearFields();

displayEmployees();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select an employee to modify.");

}

}

private void deleteEmployee() {

int selectedRow = tblEmployees.getSelectedRow();

if (selectedRow >= 0) {

String empId = tblEmployees.getValueAt(selectedRow, 0).toString();

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

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

try {

String query = "DELETE FROM EMPLOYEES WHERE emp\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, empId);

statement.executeUpdate();

clearFields();

displayEmployees();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select an employee to delete.");

}

}

private void displayEmployees() {

try {

String query = "SELECT \* FROM EMPLOYEES";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Employee> employees = new ArrayList<>();

while (resultSet.next()) {

String empId = resultSet.getString("emp\_id");

String empName = resultSet.getString("emp\_name");

String empEmail = resultSet.getString("emp\_email");

String empDept = resultSet.getString("emp\_dept");

String empPosition = resultSet.getString("emp\_position");

employees.add(new Employee(empId, empName, empEmail, empDept, empPosition));

}

DefaultTableModel model = new DefaultTableModel();

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

for (Employee employee : employees) {

model.addRow(new String[]{employee.getEmpId(), employee.getEmpName(), employee.getEmpEmail(), employee.getEmpDept(), employee.getEmpPosition()});

}

tblEmployees.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectEmployee() {

int selectedRow = tblEmployees.getSelectedRow();

if (selectedRow >= 0) {

String empId = tblEmployees.getValueAt(selectedRow, 0).toString();

String empName = tblEmployees.getValueAt(selectedRow, 1).toString();

String empEmail = tblEmployees.getValueAt(selectedRow, 2).toString();

String empDept = tblEmployees.getValueAt(selectedRow, 3).toString();

String empPosition = tblEmployees.getValueAt(selectedRow, 4).toString();

txtEmpId.setText(empId);

txtEmpName.setText(empName);

txtEmpEmail.setText(empEmail);

txtEmpDept.setText(empDept);

txtEmpPosition.setText(empPosition);

}

}

private void clearFields() {

txtEmpId.setText("");

txtEmpName.setText("");

txtEmpEmail.setText("");

txtEmpDept.setText("");

txtEmpPosition.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(EmployeesTableGUI::new);

}

private static class Employee {

private String empId;

private String empName;

private String empEmail;

private String empDept;

private String empPosition;

public Employee(String empId, String empName, String empEmail, String empDept, String empPosition) {

this.empId = empId;

this.empName = empName;

this.empEmail = empEmail;

this.empDept = empDept;

this.empPosition = empPosition;

}

public String getEmpId() {

return empId;

}

public String getEmpName() {

return empName;

}

public String getEmpEmail() {

return empEmail;

}

public String getEmpDept() {

return empDept;

}

public String getEmpPosition() {

return empPosition;

}

}

}

**Attendance 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 AttendanceTableGUI extends JFrame {

private JTextField txtAttendanceId, txtEmpId, txtDay, txtTimeIn, txtTimeOut;

private JTable tblAttendance;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public AttendanceTableGUI() {

initializeUI();

connectToDatabase();

displayAttendance();

}

private void initializeUI() {

txtAttendanceId = new JTextField();

txtEmpId = new JTextField();

txtDay = new JTextField();

txtTimeIn = new JTextField();

txtTimeOut = new JTextField();

tblAttendance = new JTable();

tblAttendance.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblAttendance.getSelectionModel().addListSelectionListener(e -> selectAttendance());

JScrollPane scrollPane = new JScrollPane(tblAttendance);

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("Attendance ID:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtAttendanceId, gbc);

gbc.gridy++;

panel.add(txtEmpId, gbc);

gbc.gridy++;

panel.add(txtDay, gbc);

gbc.gridy++;

panel.add(txtTimeIn, gbc);

gbc.gridy++;

panel.add(txtTimeOut, 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 -> insertAttendance());

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

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

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

setTitle("Biometric Attendance App");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "surya";

String password = "kanna";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertAttendance() {

int attendanceId = Integer.parseInt(txtAttendanceId.getText());

int empId = Integer.parseInt(txtEmpId.getText());

String day = txtDay.getText();

String timeIn = txtTimeIn.getText();

String timeOut = txtTimeOut.getText();

try {

String query = "INSERT INTO Attendance (attendance\_id, emp\_id, day, time\_in, time\_out) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, attendanceId);

statement.setInt(2, empId);

statement.setString(3, day);

statement.setString(4, timeIn);

statement.setString(5, timeOut);

statement.executeUpdate();

clearFields();

displayAttendance();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyAttendance() {

int selectedRow = tblAttendance.getSelectedRow();

if (selectedRow >= 0) {

int attendanceId = Integer.parseInt(txtAttendanceId.getText());

int empId = Integer.parseInt(txtEmpId.getText());

String day = txtDay.getText();

String timeIn = txtTimeIn.getText();

String timeOut = txtTimeOut.getText();

try {

String query = "UPDATE Attendance SET emp\_id=?, day=?, time\_in=?, time\_out=? WHERE attendance\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, empId);

statement.setString(2, day);

statement.setString(3, timeIn);

statement.setString(4, timeOut);

statement.setInt(5, attendanceId);

statement.executeUpdate();

clearFields();

displayAttendance();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

JOptionPane.showMessageDialog(this, "Please select an attendance record to modify.");

}

}

private void deleteAttendance() {

int selectedRow = tblAttendance.getSelectedRow();

if (selectedRow >= 0) {

int attendanceId = Integer.parseInt(tblAttendance.getValueAt(selectedRow, 0).toString());

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

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

try {

String query = "DELETE FROM Attendance WHERE attendance\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setInt(1, attendanceId);

statement.executeUpdate();

clearFields();

displayAttendance();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

JOptionPane.showMessageDialog(this, "Please select an attendance record to delete.");

}

}

private void displayAttendance() {

try {

String query = "SELECT \* FROM Attendance";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<Attendance> attendanceList = new ArrayList<>();

while (resultSet.next()) {

int attendanceId = resultSet.getInt("attendance\_id");

int empId = resultSet.getInt("emp\_id");

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

String timeIn = resultSet.getString("time\_in");

String timeOut = resultSet.getString("time\_out");

attendanceList.add(new Attendance(attendanceId, empId, day, timeIn, timeOut));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Attendance ID", "Employee ID", "Day", "Time In", "Time Out"});

for (Attendance attendance : attendanceList) {

model.addRow(new Object[]{attendance.getAttendanceId(), attendance.getEmpId(), attendance.getDay(), attendance.getTimeIn(), attendance.getTimeOut()});

}

tblAttendance.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectAttendance() {

int selectedRow = tblAttendance.getSelectedRow();

if (selectedRow >= 0) {

int attendanceId = Integer.parseInt(tblAttendance.getValueAt(selectedRow, 0).toString());

int empId = Integer.parseInt(tblAttendance.getValueAt(selectedRow, 1).toString());

String day = tblAttendance.getValueAt(selectedRow, 2).toString();

String timeIn = tblAttendance.getValueAt(selectedRow, 3).toString();

String timeOut = tblAttendance.getValueAt(selectedRow, 4).toString();

txtAttendanceId.setText(Integer.toString(attendanceId));

txtEmpId.setText(Integer.toString(empId));

txtDay.setText(day);

txtTimeIn.setText(timeIn);

txtTimeOut.setText(timeOut);

}

}

private void clearFields() {

txtAttendanceId.setText("");

txtEmpId.setText("");

txtDay.setText("");

txtTimeIn.setText("");

txtTimeOut.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(AttendanceTableGUI::new);

}

private class Attendance {

private int attendanceId;

private int empId;

private String day;

private String timeIn;

private String timeOut;

public Attendance(int attendanceId, int empId, String day, String timeIn, String timeOut) {

this.attendanceId = attendanceId;

this.empId = empId;

this.day = day;

this.timeIn = timeIn;

this.timeOut = timeOut;

}

public int getAttendanceId() {

return attendanceId;

}

public int getEmpId() {

return empId;

}

public String getDay() {

return day;

}

public String getTimeIn() {

return timeIn;

}

public String getTimeOut() {

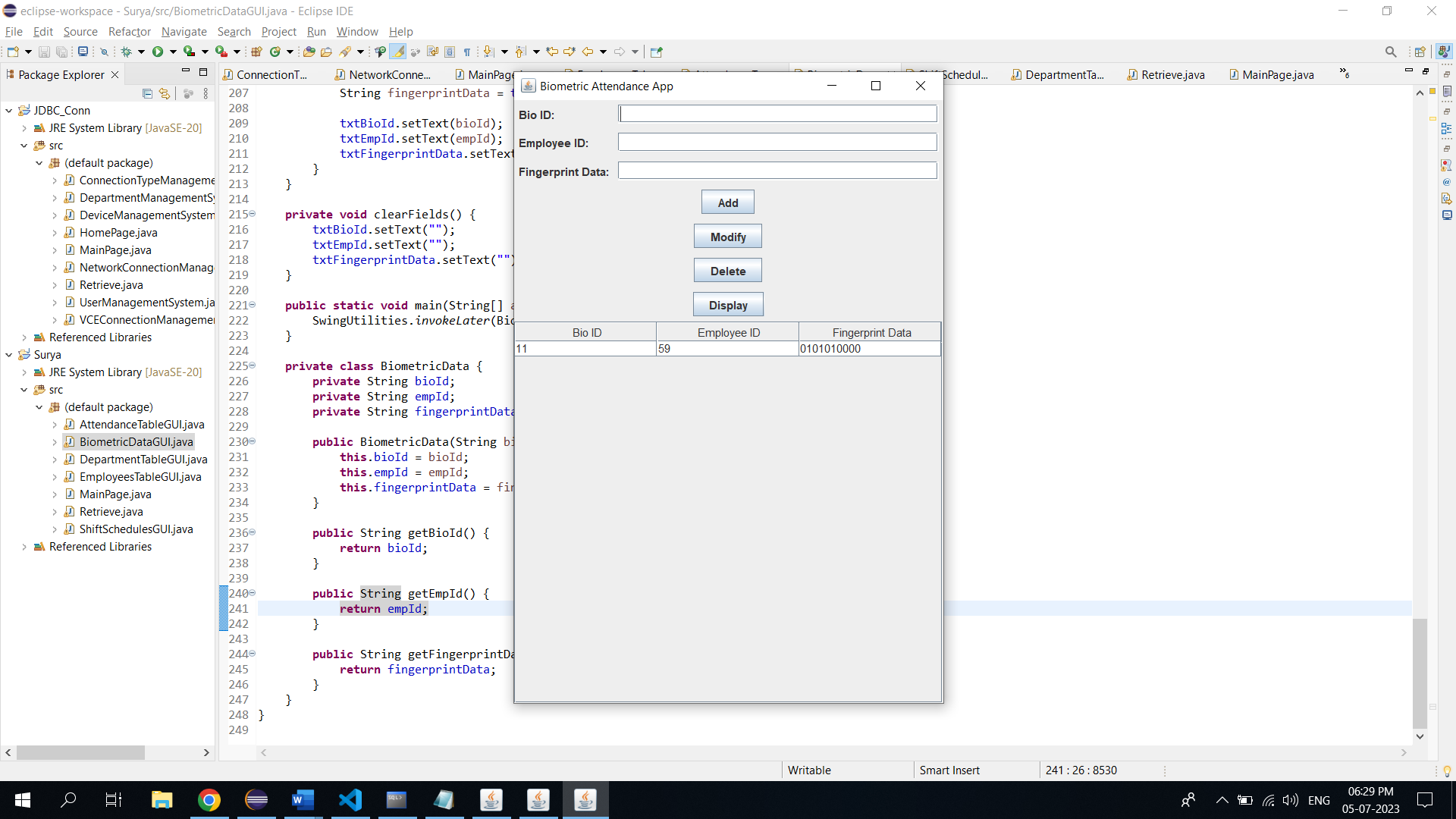
return timeOut;

}

}

}

**biometricData 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 BiometricDataGUI extends JFrame {

private JTextField txtBioId, txtEmpId, txtFingerprintData;

private JTable tblBiometricData;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public BiometricDataGUI() {

initializeUI();

connectToDatabase();

displayBiometricData();

}

private void initializeUI() {

txtBioId = new JTextField();

txtEmpId = new JTextField();

txtFingerprintData = new JTextField();

tblBiometricData = new JTable();

tblBiometricData.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblBiometricData.getSelectionModel().addListSelectionListener(e -> selectBiometricData());

JScrollPane scrollPane = new JScrollPane(tblBiometricData);

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("Bio ID:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtBioId, gbc);

gbc.gridy++;

panel.add(txtEmpId, gbc);

gbc.gridy++;

panel.add(txtFingerprintData, 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 -> insertBiometricData());

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

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

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

setTitle("Biometric Attendance App");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "surya";

String password = "kanna";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertBiometricData() {

String bioId = txtBioId.getText();

String empId = txtEmpId.getText();

String fingerprintData = txtFingerprintData.getText();

try {

String query = "INSERT INTO biometricData (bio\_id, emp\_id, fingerprint\_data) VALUES (?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, bioId);

statement.setString(2, empId);

statement.setString(3, fingerprintData);

statement.executeUpdate();

clearFields();

displayBiometricData();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifyBiometricData() {

int selectedRow = tblBiometricData.getSelectedRow();

if (selectedRow >= 0) {

String bioId = txtBioId.getText();

String empId = txtEmpId.getText();

String fingerprintData = txtFingerprintData.getText();

try {

String query = "UPDATE biometricData SET emp\_id=?, fingerprint\_data=? WHERE bio\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, empId);

statement.setString(2, fingerprintData);

statement.setString(3, bioId);

statement.executeUpdate();

clearFields();

displayBiometricData();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

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

}

}

private void deleteBiometricData() {

int selectedRow = tblBiometricData.getSelectedRow();

if (selectedRow >= 0) {

String bioId = tblBiometricData.getValueAt(selectedRow, 0).toString();

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

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

try {

String query = "DELETE FROM biometricData WHERE bio\_id=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, bioId);

statement.executeUpdate();

clearFields();

displayBiometricData();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

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

}

}

private void displayBiometricData() {

try {

String query = "SELECT \* FROM biometricData";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<BiometricData> biometricDataList = new ArrayList<>();

while (resultSet.next()) {

String bioId = resultSet.getString("bio\_id");

String empId = resultSet.getString("emp\_id");

String fingerprintData = resultSet.getString("fingerprint\_data");

biometricDataList.add(new BiometricData(bioId, empId, fingerprintData));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Bio ID", "Employee ID", "Fingerprint Data"});

for (BiometricData biometricData : biometricDataList) {

model.addRow(new String[]{biometricData.getBioId(), biometricData.getEmpId(), biometricData.getFingerprintData()});

}

tblBiometricData.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectBiometricData() {

int selectedRow = tblBiometricData.getSelectedRow();

if (selectedRow >= 0) {

String bioId = tblBiometricData.getValueAt(selectedRow, 0).toString();

String empId = tblBiometricData.getValueAt(selectedRow, 1).toString();

String fingerprintData = tblBiometricData.getValueAt(selectedRow, 2).toString();

txtBioId.setText(bioId);

txtEmpId.setText(empId);

txtFingerprintData.setText(fingerprintData);

}

}

private void clearFields() {

txtBioId.setText("");

txtEmpId.setText("");

txtFingerprintData.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(BiometricDataGUI::new);

}

private class BiometricData {

private String bioId;

private String empId;

private String fingerprintData;

public BiometricData(String bioId, String empId, String fingerprintData) {

this.bioId = bioId;

this.empId = empId;

this.fingerprintData = fingerprintData;

}

public String getBioId() {

return bioId;

}

public String getEmpId() {

return empId;

}

public String getFingerprintData() {

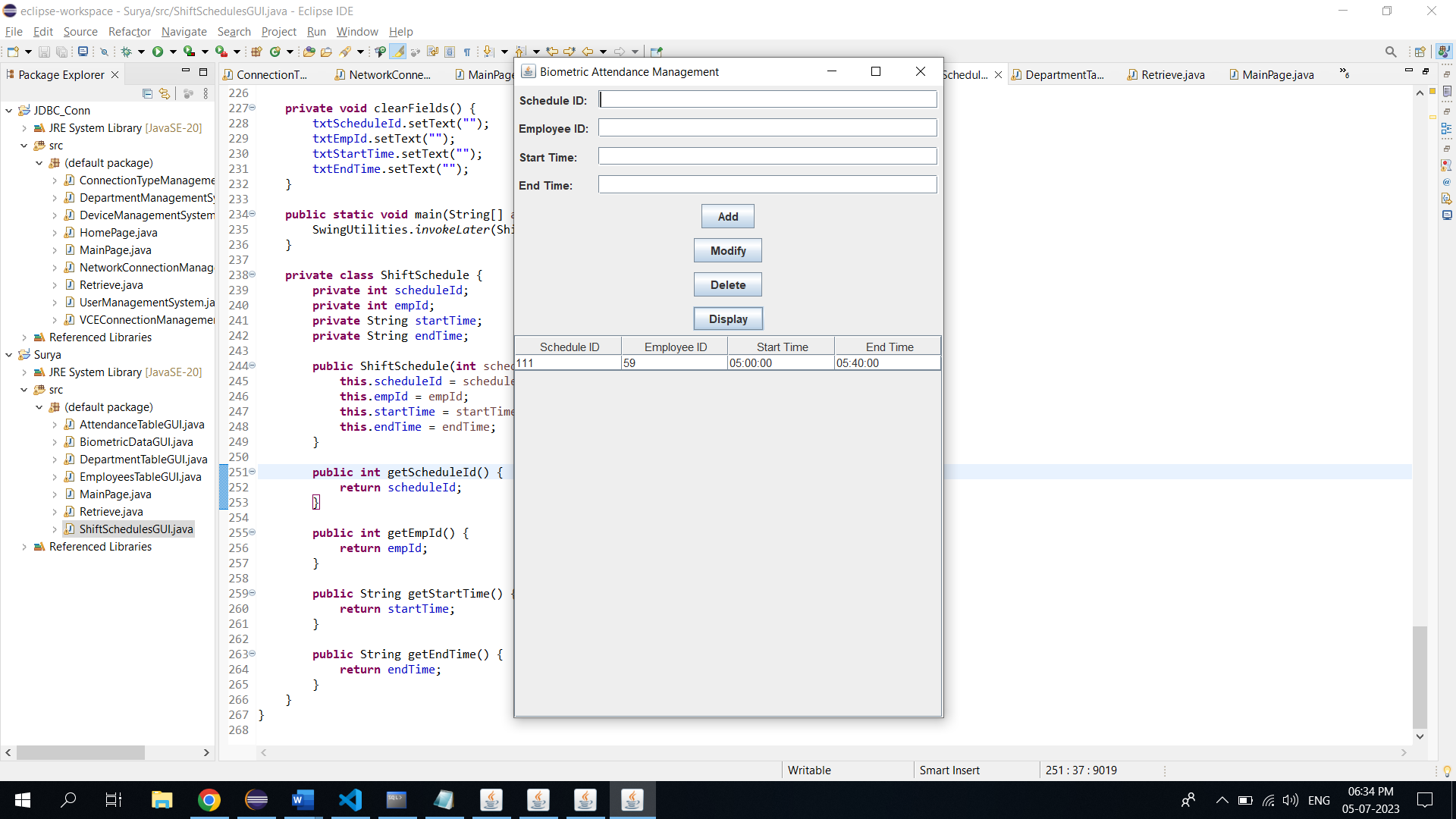
return fingerprintData;

}

}

}

**ShiftSchedules 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 ShiftSchedulesGUI extends JFrame {

private JTextField txtScheduleId, txtEmpId, txtStartTime, txtEndTime;

private JTable tblSchedules;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public ShiftSchedulesGUI() {

initializeUI();

connectToDatabase();

displaySchedules();

}

private void initializeUI() {

txtScheduleId = new JTextField();

txtEmpId = new JTextField();

txtStartTime = new JTextField();

txtEndTime = new JTextField();

tblSchedules = new JTable();

tblSchedules.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

tblSchedules.getSelectionModel().addListSelectionListener(e -> selectSchedule());

JScrollPane scrollPane = new JScrollPane(tblSchedules);

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("Schedule ID:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridy++;

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

gbc.gridx = 1;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.weightx = 1;

panel.add(txtScheduleId, gbc);

gbc.gridy++;

panel.add(txtEmpId, gbc);

gbc.gridy++;

panel.add(txtStartTime, gbc);

gbc.gridy++;

panel.add(txtEndTime, 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 -> insertSchedule());

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

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

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

setTitle("Biometric Attendance Management");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "surya";

String password = "kanna";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertSchedule() {

String scheduleId = txtScheduleId.getText();

String empId = txtEmpId.getText();

String startTime = txtStartTime.getText();

String endTime = txtEndTime.getText();

try {

String query = "INSERT INTO ShiftSchedules (ScheduleID, Emp\_ID, StartTime, EndTime) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, scheduleId);

statement.setString(2, empId);

statement.setString(3, startTime);

statement.setString(4, endTime);

statement.executeUpdate();

clearFields();

displaySchedules();

} catch (SQLException e) {

e.printStackTrace();

}

}

private void modifySchedule() {

int selectedRow = tblSchedules.getSelectedRow();

if (selectedRow >= 0) {

String scheduleId = txtScheduleId.getText();

String empId = txtEmpId.getText();

String startTime = txtStartTime.getText();

String endTime = txtEndTime.getText();

try {

String query = "UPDATE ShiftSchedules SET Emp\_ID=?, StartTime=?, EndTime=? WHERE ScheduleID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, empId);

statement.setString(2, startTime);

statement.setString(3, endTime);

statement.setString(4, scheduleId);

statement.executeUpdate();

clearFields();

displaySchedules();

} catch (SQLException e) {

e.printStackTrace();

}

} else {

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

}

}

private void deleteSchedule() {

int selectedRow = tblSchedules.getSelectedRow();

if (selectedRow >= 0) {

String scheduleId = tblSchedules.getValueAt(selectedRow, 0).toString();

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

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

try {

String query = "DELETE FROM ShiftSchedules WHERE ScheduleID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, scheduleId);

statement.executeUpdate();

clearFields();

displaySchedules();

} catch (SQLException e) {

e.printStackTrace();

}

}

} else {

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

}

}

private void displaySchedules() {

try {

String query = "SELECT \* FROM ShiftSchedules";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

List<ShiftSchedule> schedules = new ArrayList<>();

while (resultSet.next()) {

int scheduleId = resultSet.getInt("ScheduleID");

int empId = resultSet.getInt("Emp\_ID");

String startTime = resultSet.getString("StartTime");

String endTime = resultSet.getString("EndTime");

schedules.add(new ShiftSchedule(scheduleId, empId, startTime, endTime));

}

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(new String[]{"Schedule ID", "Employee ID", "Start Time", "End Time"});

for (ShiftSchedule schedule : schedules) {

model.addRow(new Object[]{schedule.getScheduleId(), schedule.getEmpId(), schedule.getStartTime(), schedule.getEndTime()});

}

tblSchedules.setModel(model);

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectSchedule() {

int selectedRow = tblSchedules.getSelectedRow();

if (selectedRow >= 0) {

int scheduleId = (int) tblSchedules.getValueAt(selectedRow, 0);

int empId = (int) tblSchedules.getValueAt(selectedRow, 1);

String startTime = tblSchedules.getValueAt(selectedRow, 2).toString();

String endTime = tblSchedules.getValueAt(selectedRow, 3).toString();

txtScheduleId.setText(String.valueOf(scheduleId));

txtEmpId.setText(String.valueOf(empId));

txtStartTime.setText(startTime);

txtEndTime.setText(endTime);

}

}

private void clearFields() {

txtScheduleId.setText("");

txtEmpId.setText("");

txtStartTime.setText("");

txtEndTime.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(ShiftSchedulesGUI::new);

}

private class ShiftSchedule {

private int scheduleId;

private int empId;

private String startTime;

private String endTime;

public ShiftSchedule(int scheduleId, int empId, String startTime, String endTime) {

this.scheduleId = scheduleId;

this.empId = empId;

this.startTime = startTime;

this.endTime = endTime;

}

public int getScheduleId() {

return scheduleId;

}

public int getEmpId() {

return empId;

}

public String getStartTime() {

return startTime;

}

public String getEndTime() {

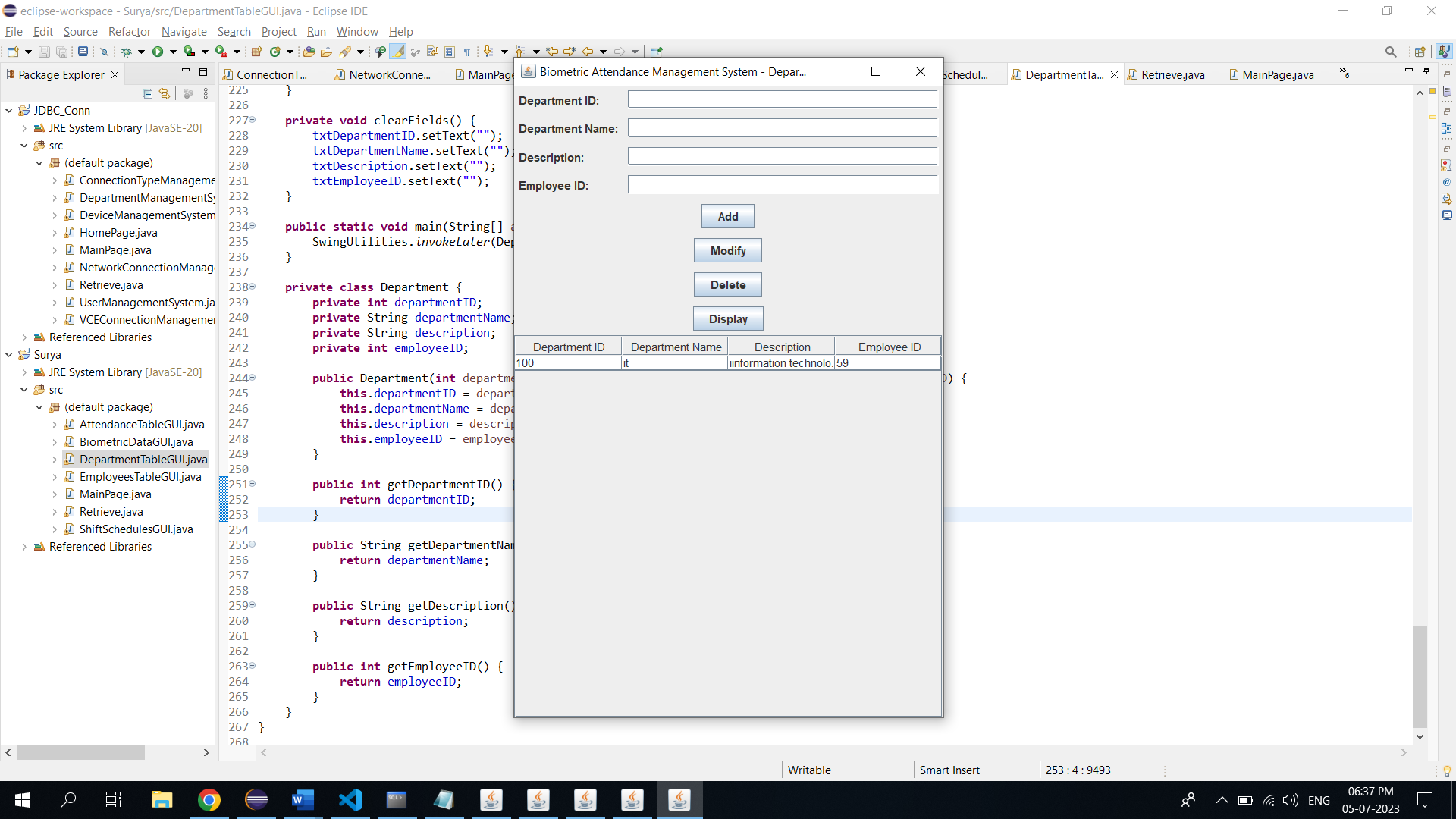
return endTime;

}

}

}

**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 DepartmentTableGUI extends JFrame {

private JTextField txtDepartmentID, txtDepartmentName, txtDescription, txtEmployeeID;

private JTable tblDepartments;

private JButton btnAdd, btnModify, btnDelete, btnDisplay;

private Connection connection;

public DepartmentTableGUI() {

initializeUI();

connectToDatabase();

displayDepartments();

}

private void initializeUI() {

txtDepartmentID = new JTextField();

txtDepartmentName = new JTextField();

txtDescription = new JTextField();

txtEmployeeID = 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("Description:"), gbc);

gbc.gridy++;

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

gbc.gridy++;

panel.add(txtEmployeeID, 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("Biometric Attendance Management System - Department");

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void connectToDatabase() {

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

String username = "surya";

String password = "kanna";

try {

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

} catch (SQLException e) {

e.printStackTrace();

}

}

private void insertDepartment() {

String departmentID = txtDepartmentID.getText();

String departmentName = txtDepartmentName.getText();

String description = txtDescription.getText();

String employeeID = txtEmployeeID.getText();

try {

String query = "INSERT INTO department (DepartmentID, DepartmentName, Description, emp\_id) VALUES (?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, departmentID);

statement.setString(2, departmentName);

statement.setString(3, description);

statement.setString(4, employeeID);

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 description = txtDescription.getText();

String employeeID = txtEmployeeID.getText();

try {

String query = "UPDATE department SET DepartmentName=?, Description=?, emp\_id=? WHERE DepartmentID=?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, departmentName);

statement.setString(2, description);

statement.setString(3, employeeID);

statement.setString(4, 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 department 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 department";

Statement statement = connection.createStatement();

ResultSet resultSet = statement.executeQuery(query);

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

while (resultSet.next()) {

int departmentID = resultSet.getInt("DepartmentID");

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

String description = resultSet.getString("Description");

int employeeID = resultSet.getInt("emp\_id");

departments.add(new Department(departmentID, departmentName, description, employeeID));

}

DefaultTableModel model = new DefaultTableModel();

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

for (Department department : departments) {

model.addRow(new Object[]{department.getDepartmentID(), department.getDepartmentName(), department.getDescription(), department.getEmployeeID()});

}

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 description = tblDepartments.getValueAt(selectedRow, 2).toString();

String employeeID = tblDepartments.getValueAt(selectedRow, 3).toString();

txtDepartmentID.setText(departmentID);

txtDepartmentName.setText(departmentName);

txtDescription.setText(description);

txtEmployeeID.setText(employeeID);

}

}

private void clearFields() {

txtDepartmentID.setText("");

txtDepartmentName.setText("");

txtDescription.setText("");

txtEmployeeID.setText("");

}

public static void main(String[] args) {

SwingUtilities.invokeLater(DepartmentTableGUI::new);

}

private class Department {

private int departmentID;

private String departmentName;

private String description;

private int employeeID;

public Department(int departmentID, String departmentName, String description, int employeeID) {

this.departmentID = departmentID;

this.departmentName = departmentName;

this.description = description;

this.employeeID = employeeID;

}

public int getDepartmentID() {

return departmentID;

}

public String getDepartmentName() {

return departmentName;

}

public String getDescription() {

return description;

}

public int getEmployeeID() {

return employeeID;

}

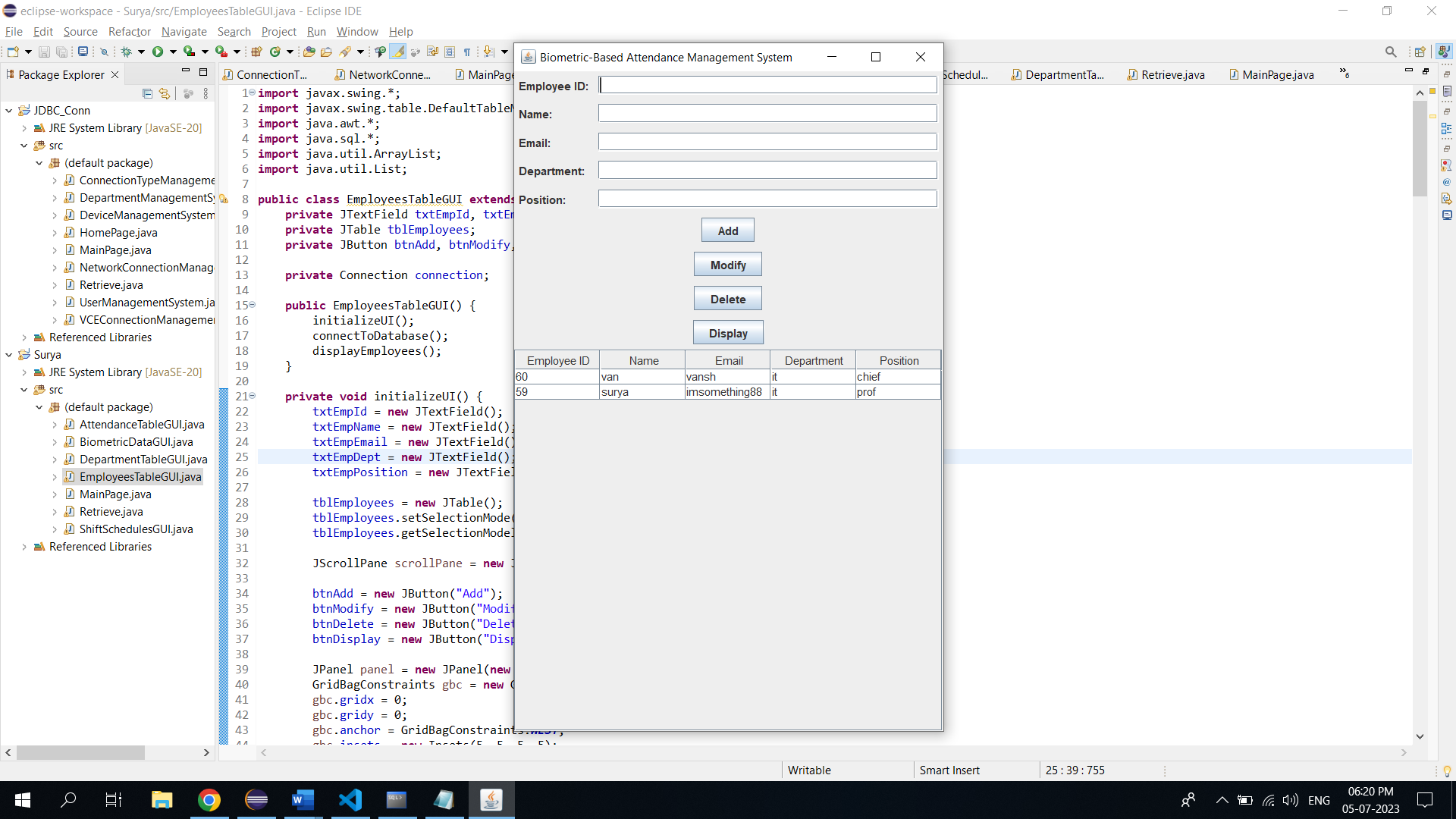
}

}

**TESTING**

Employees page:

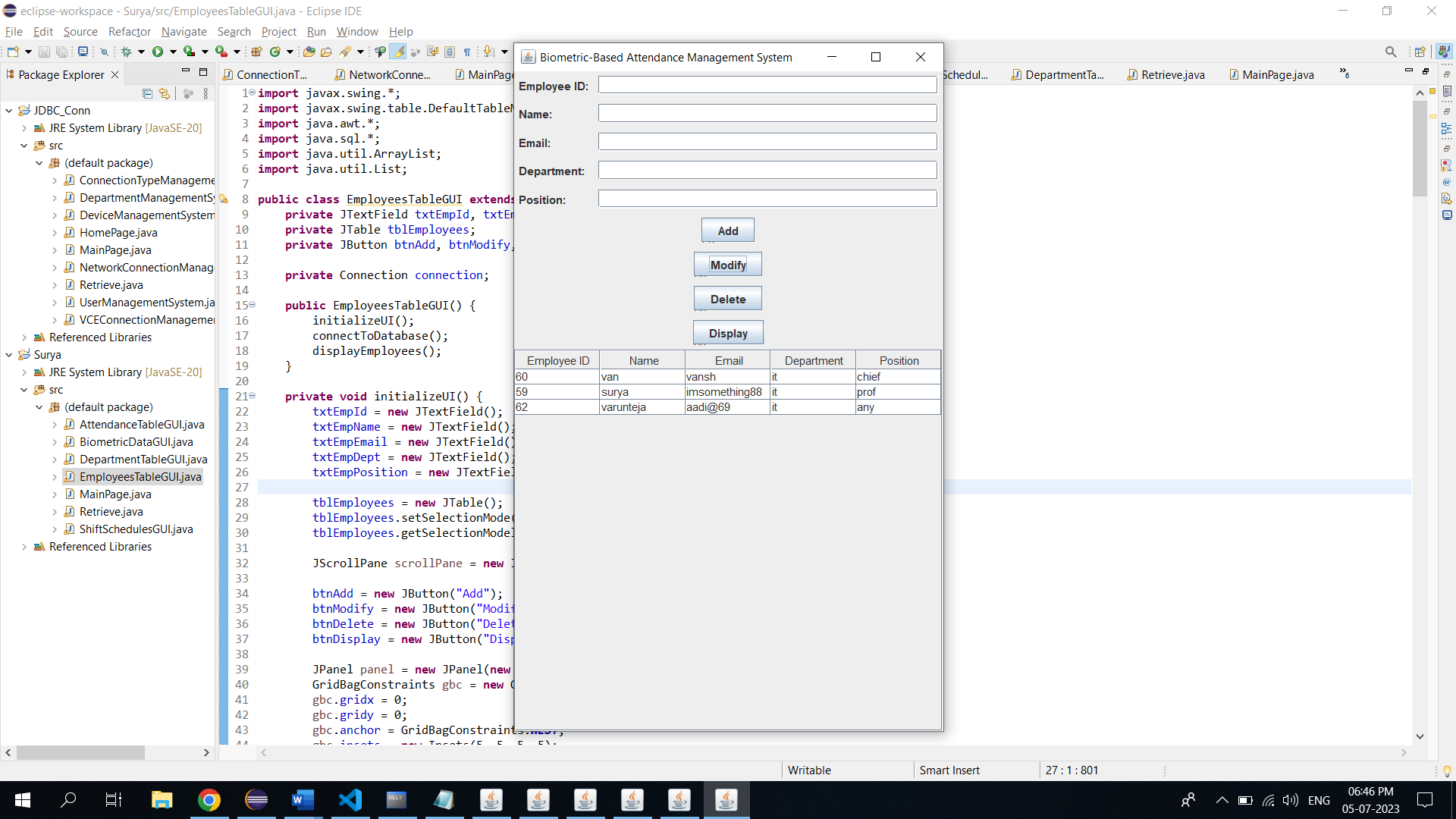
Before insertion into table :

****

After insertion:

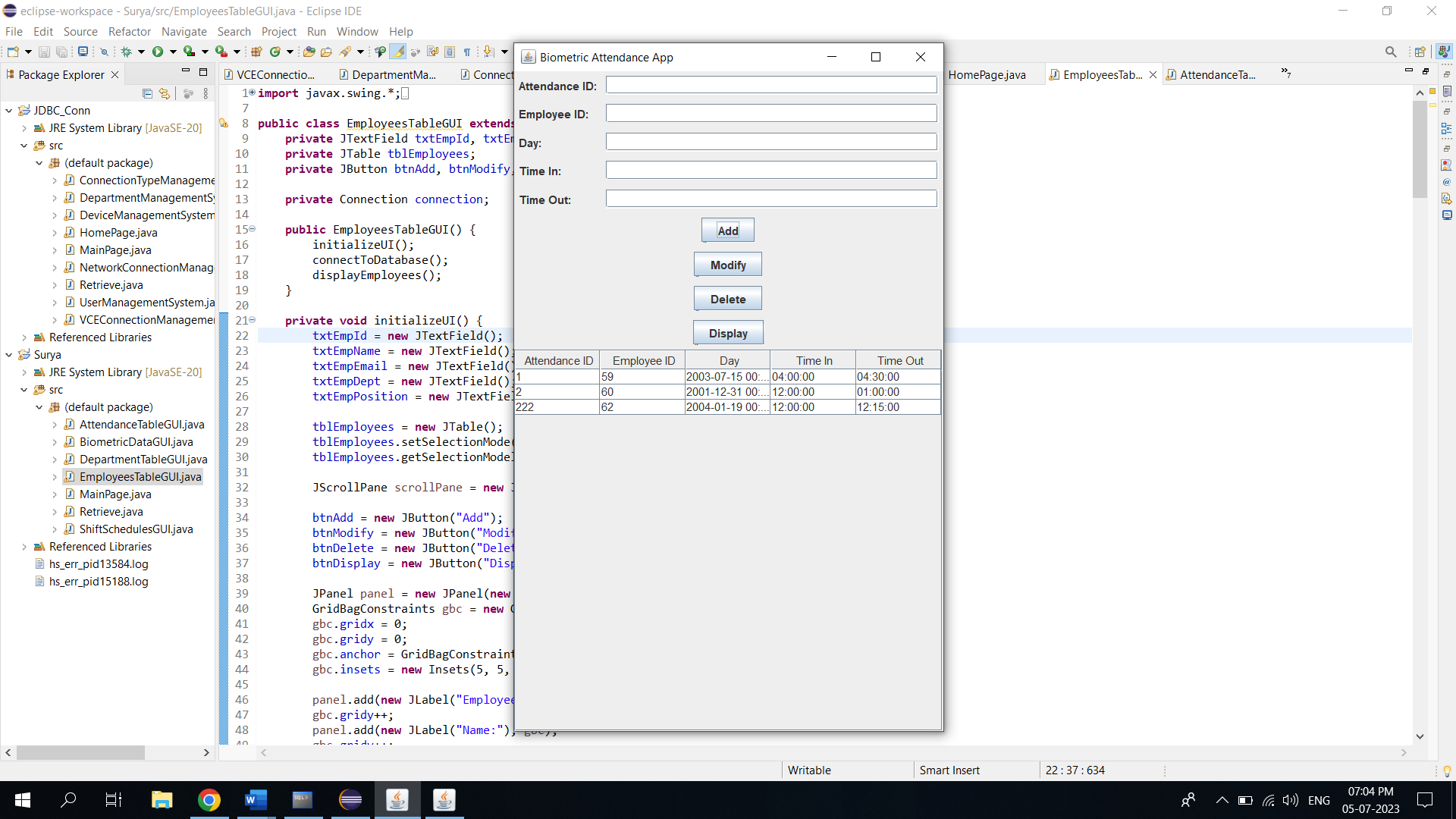
****

Modification:

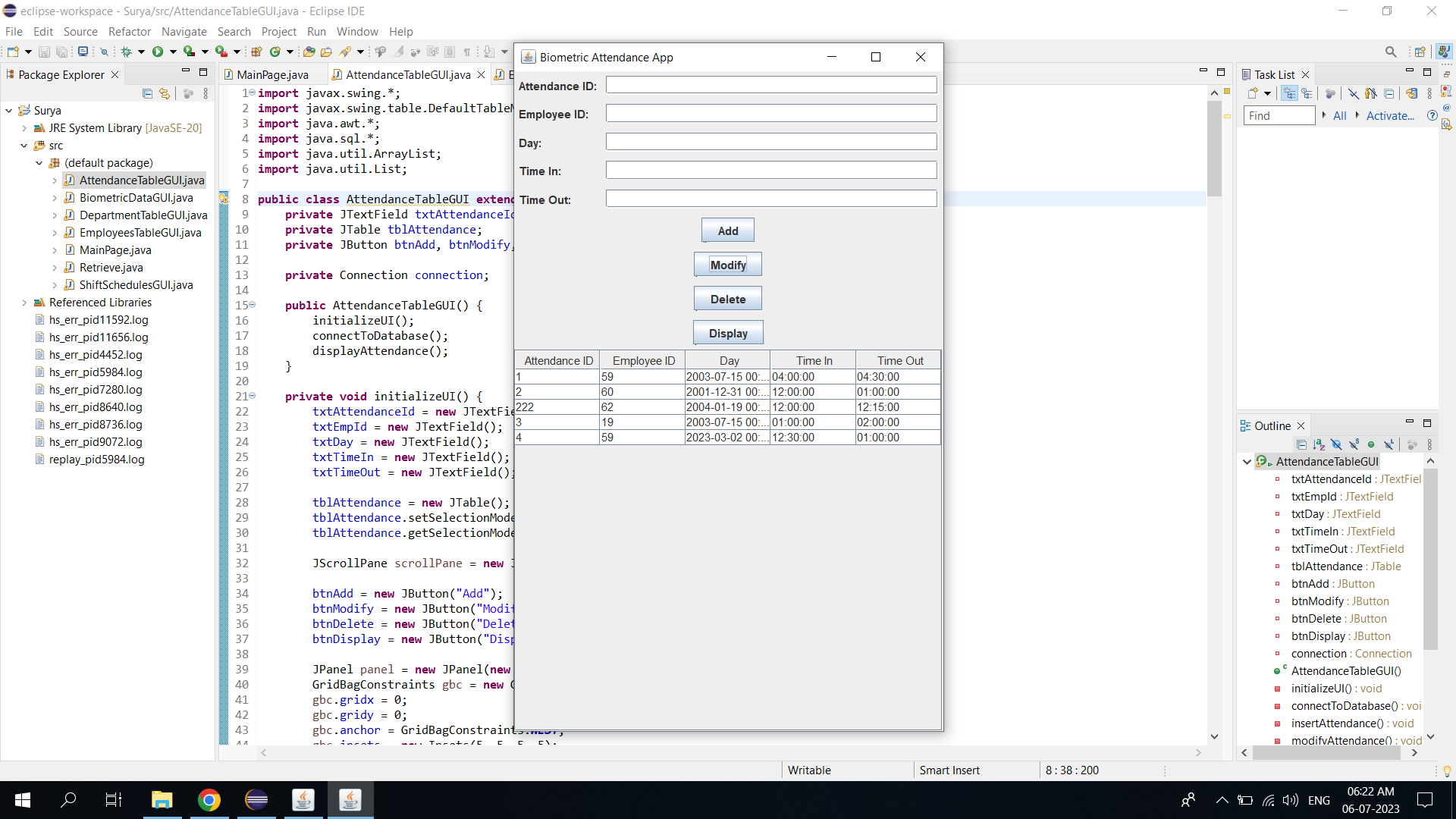
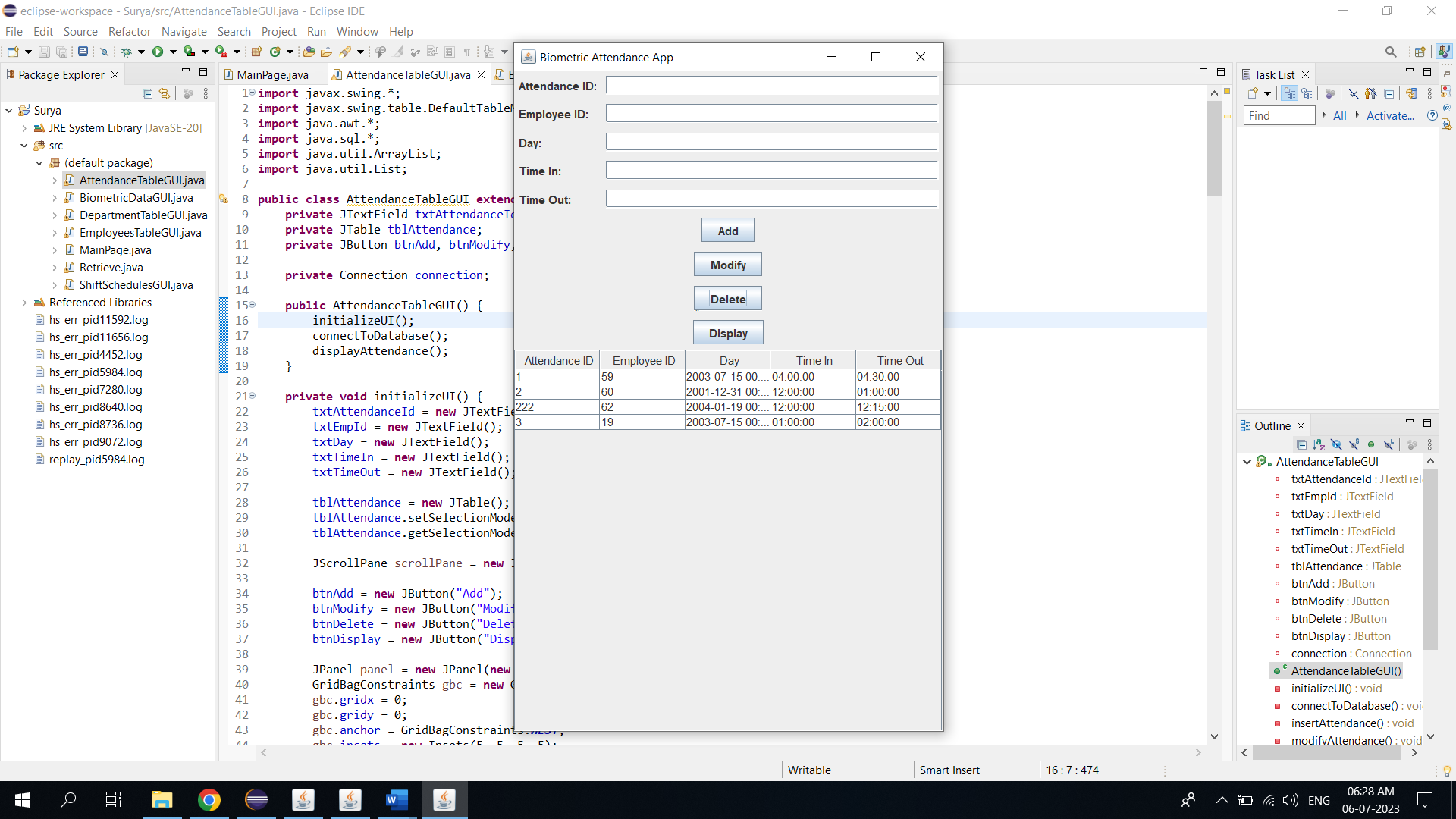


**Attendance Page:**

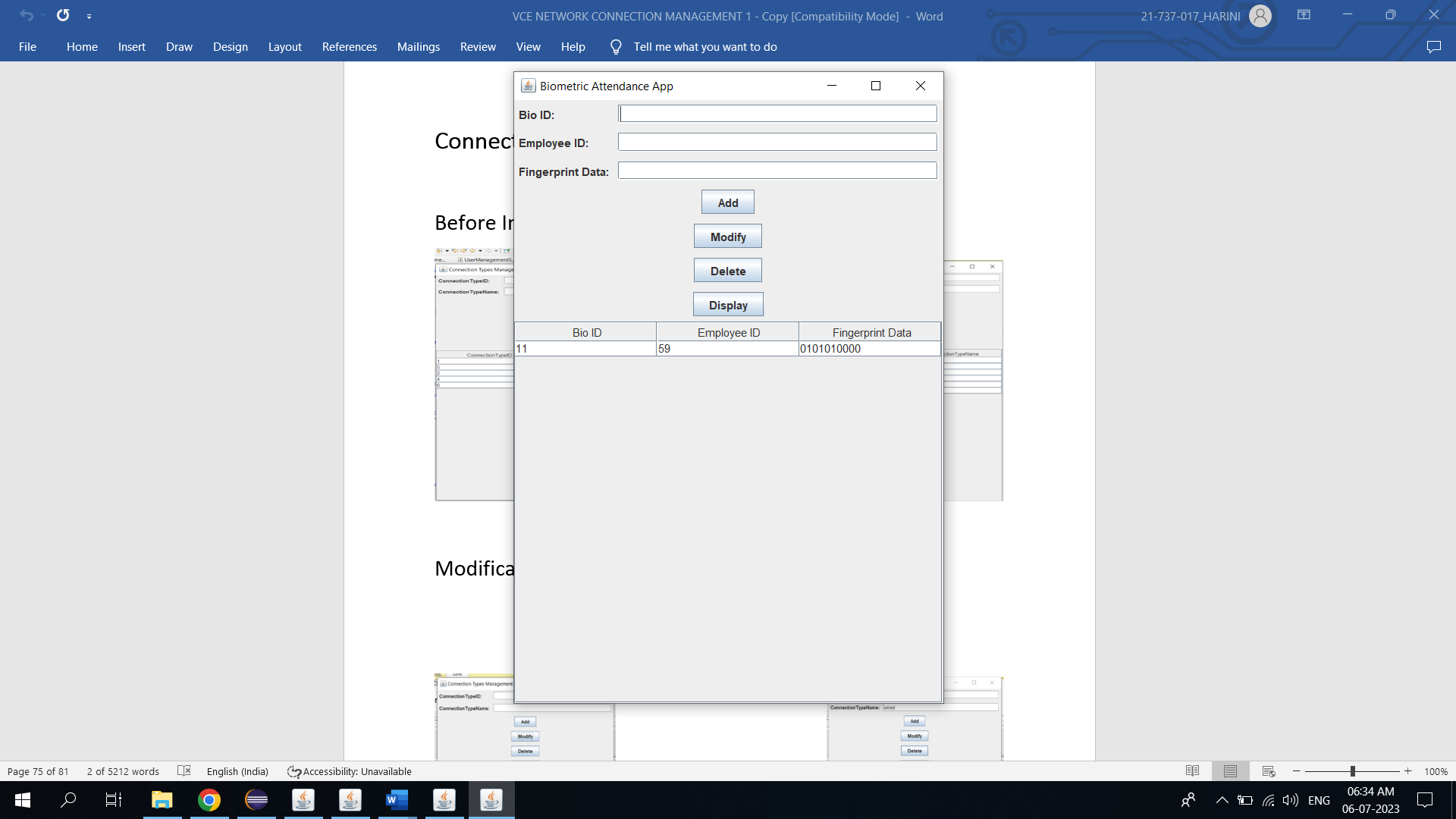
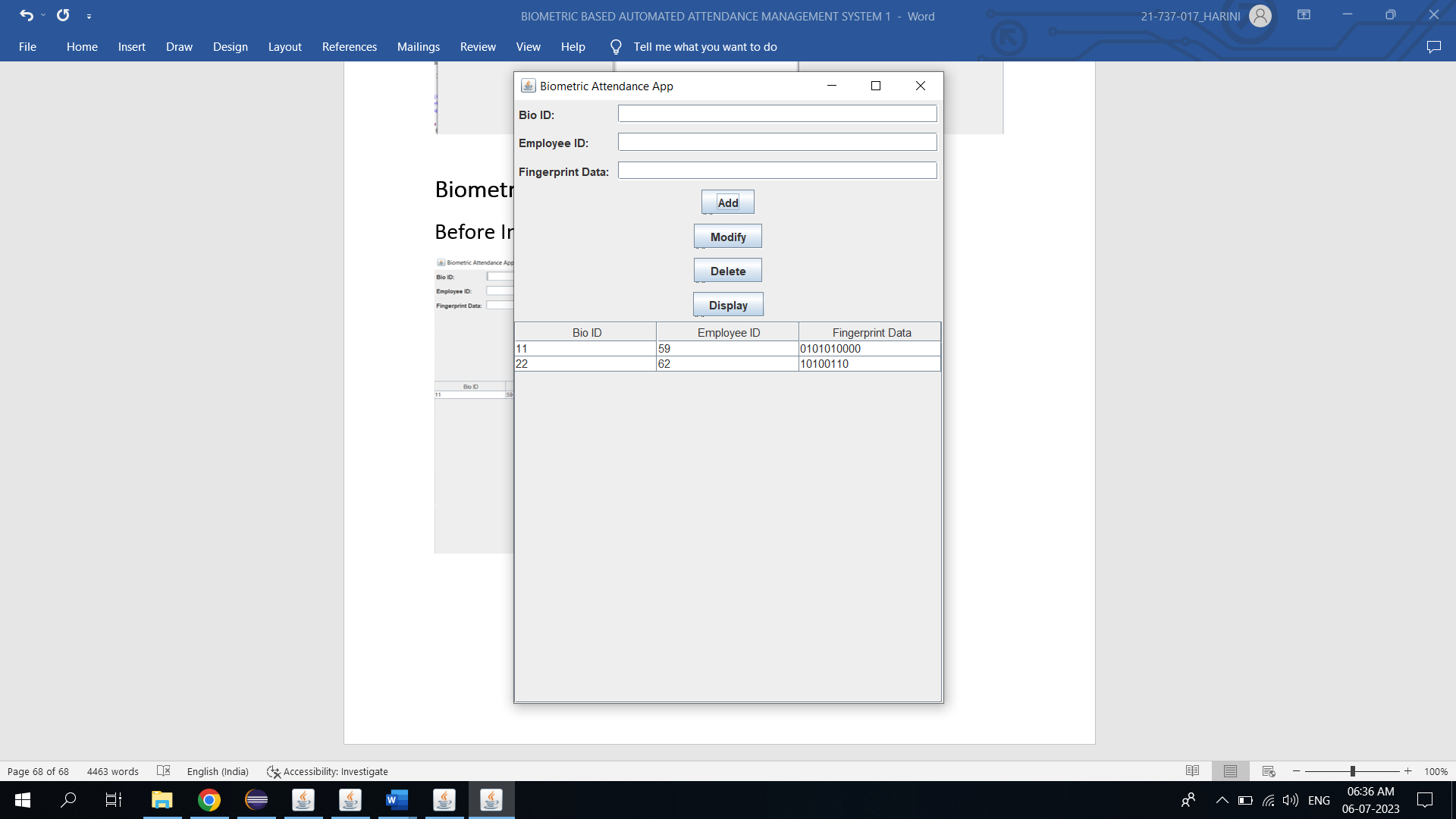
Before insertion : After insertion:

**** 

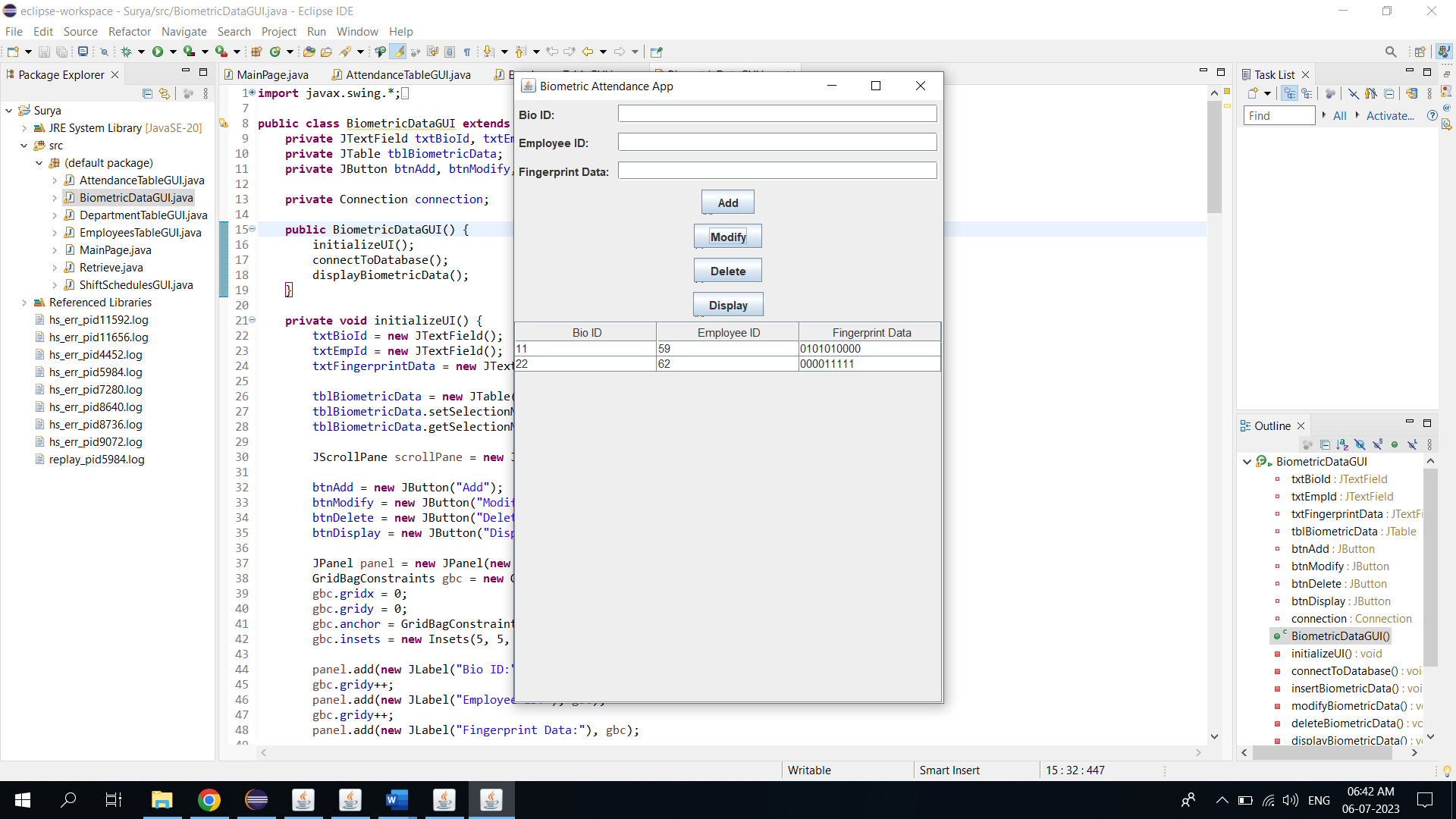
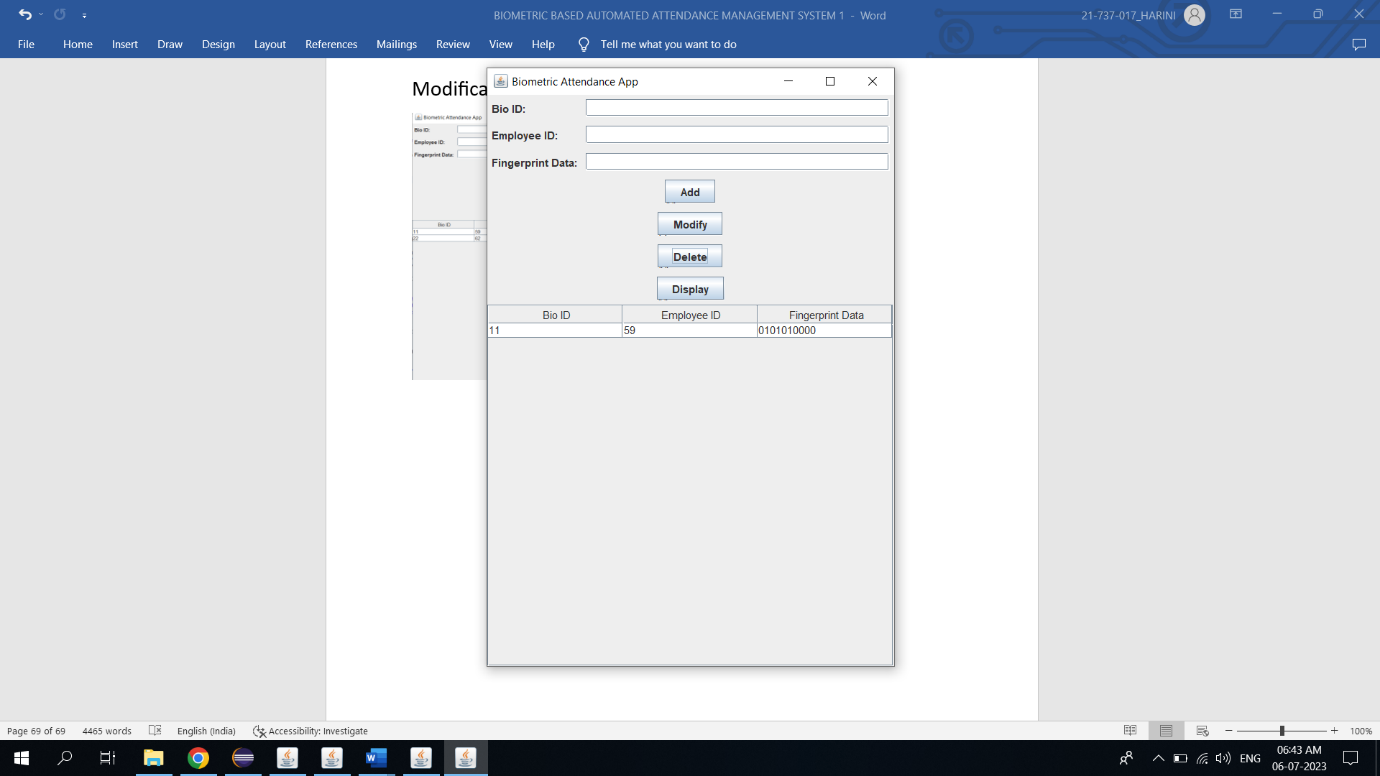
Modification: Deletion:

**BiometricData Page:**

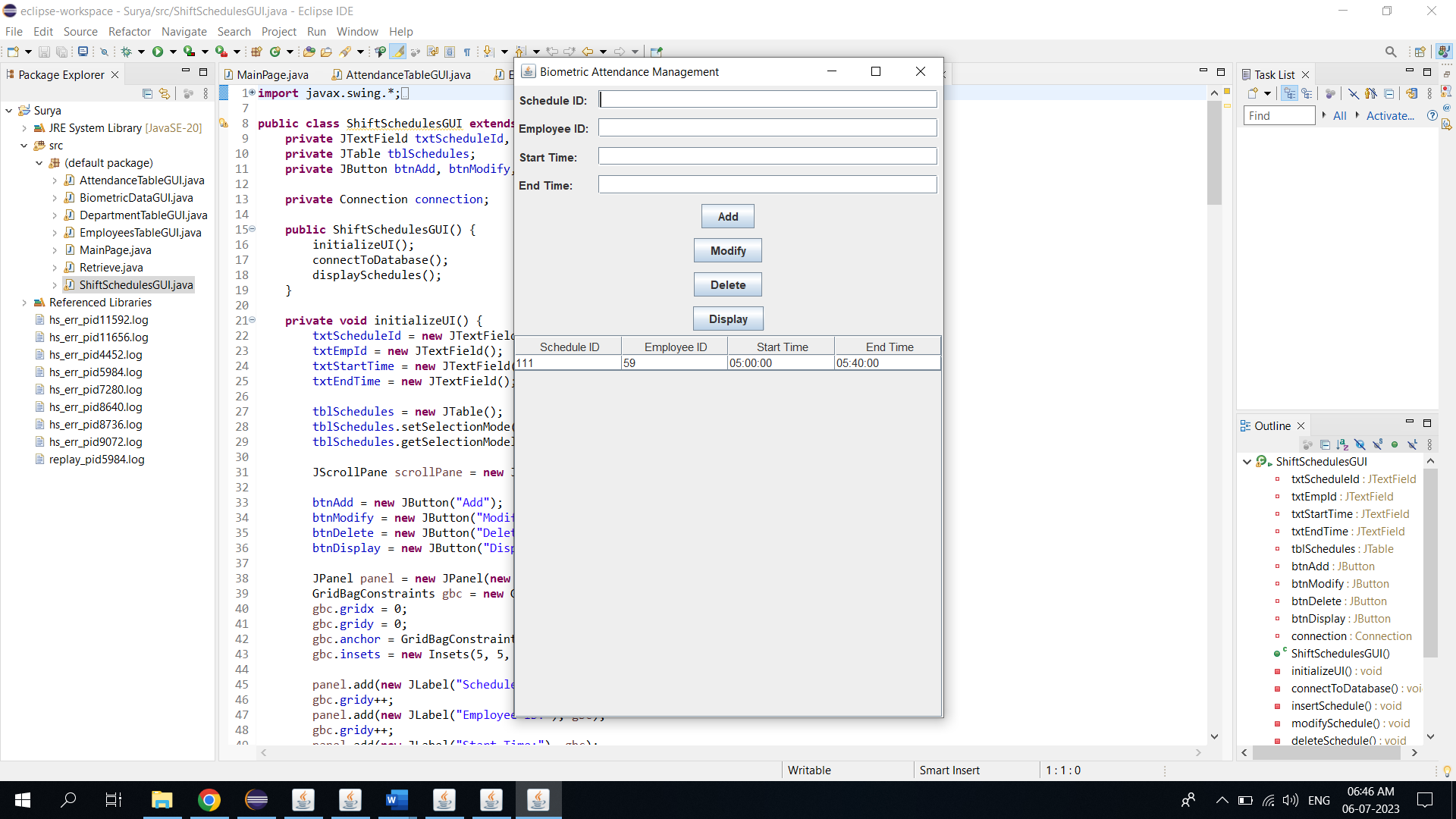
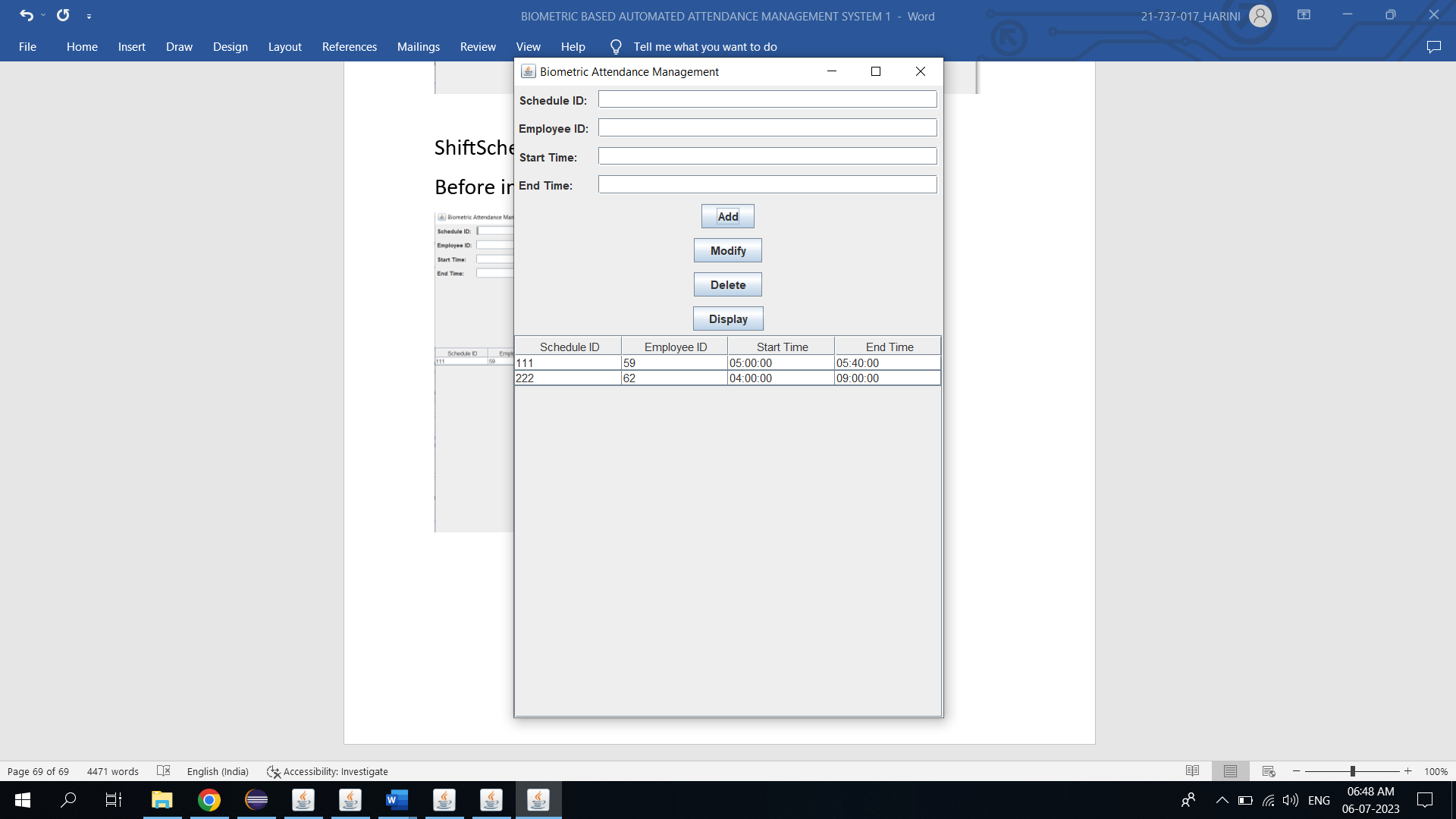
Before Insertion: After Insertion:**** 

Modification: Deletion:

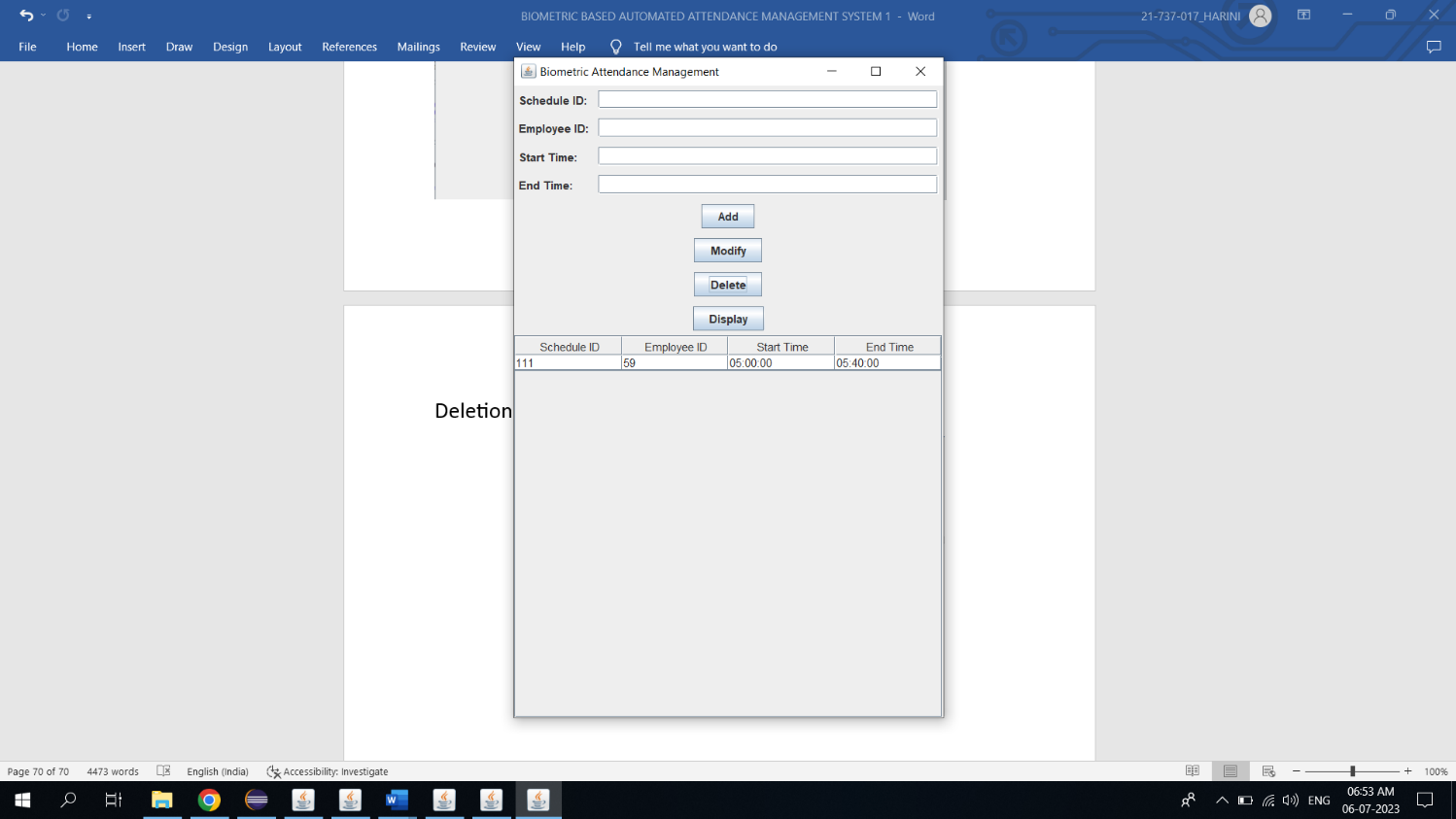
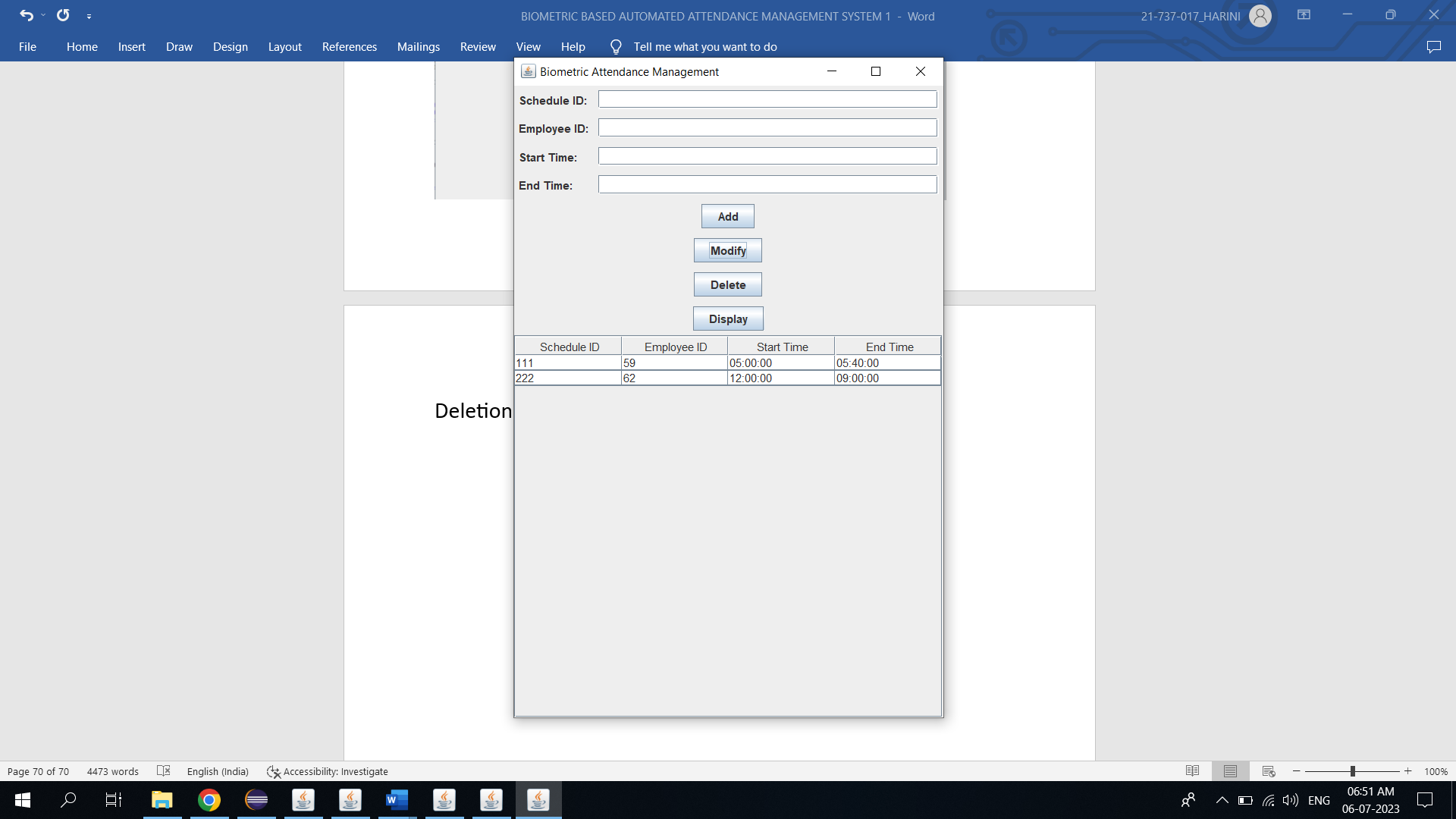
 

ShiftSchedules Page:

Before insertion: After insertion:

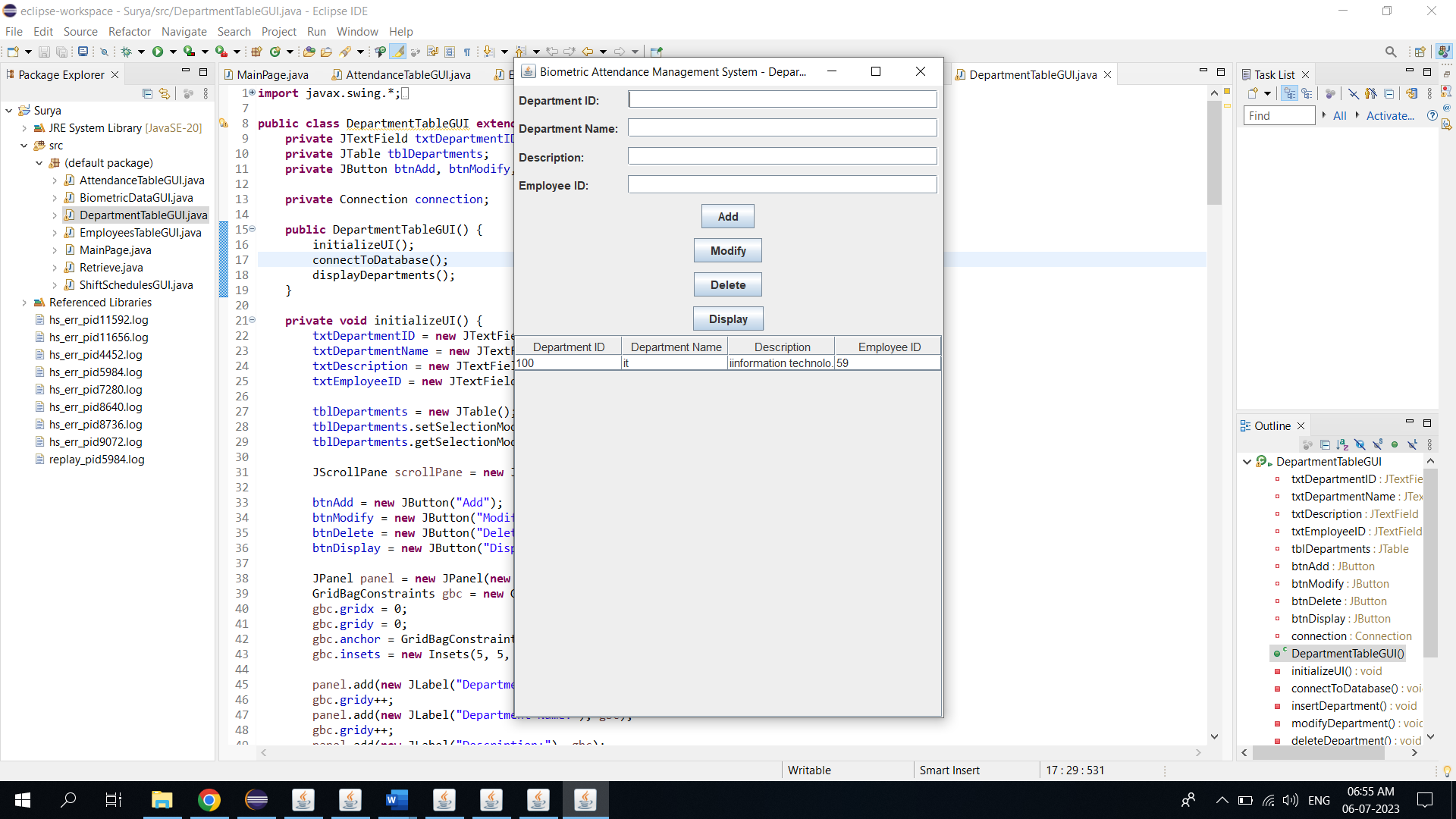
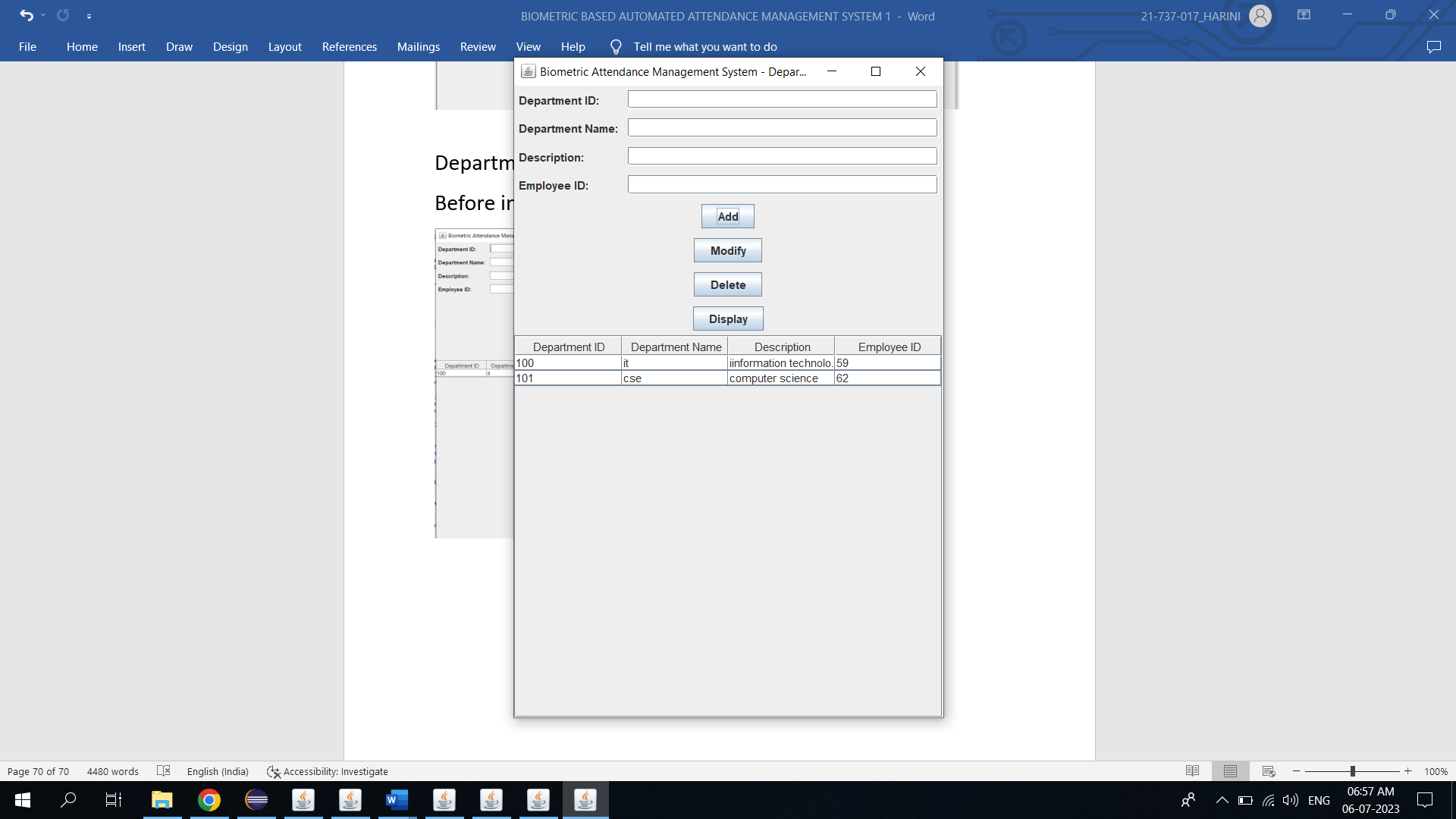
 

Deletion: Modification:

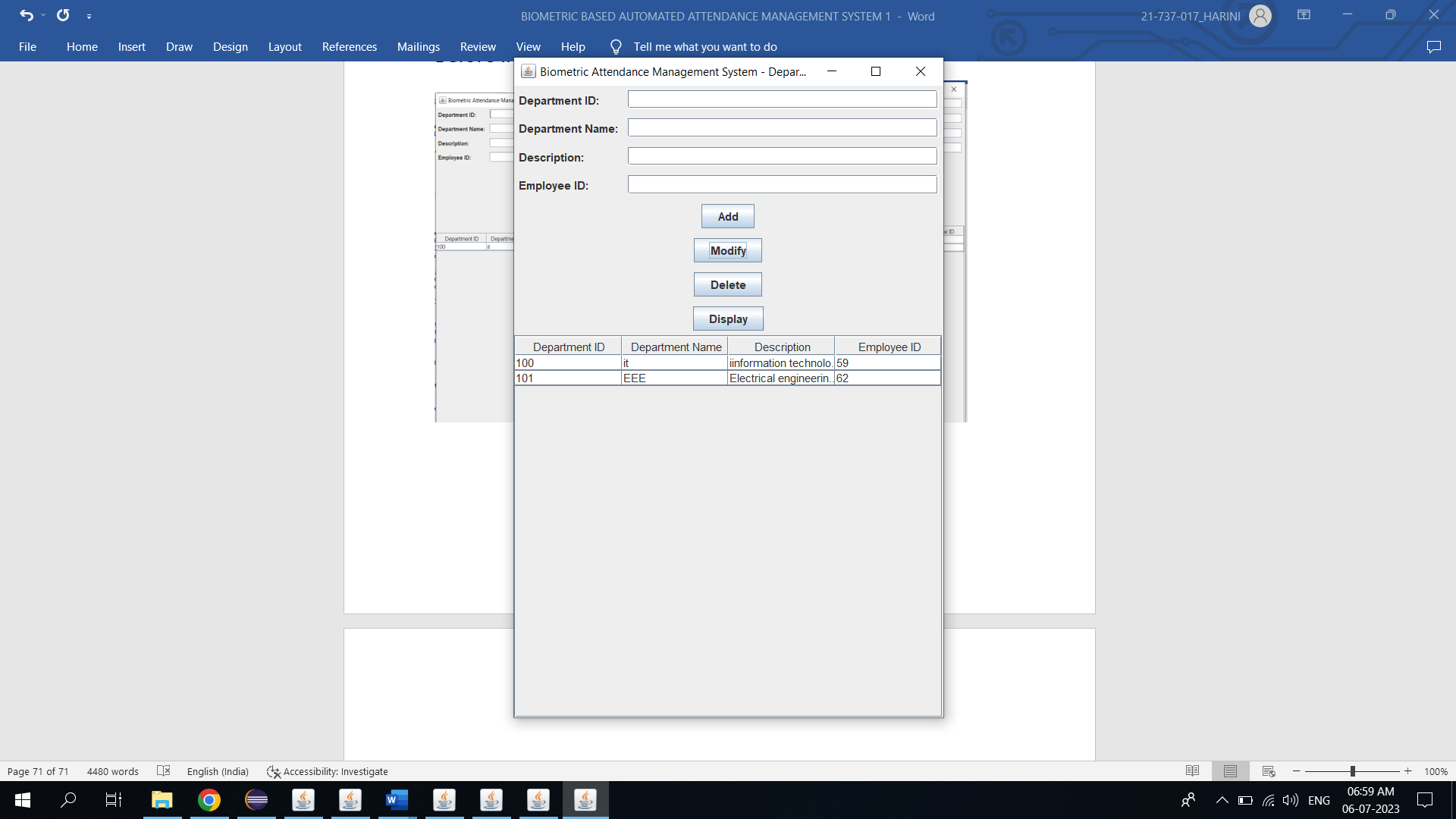
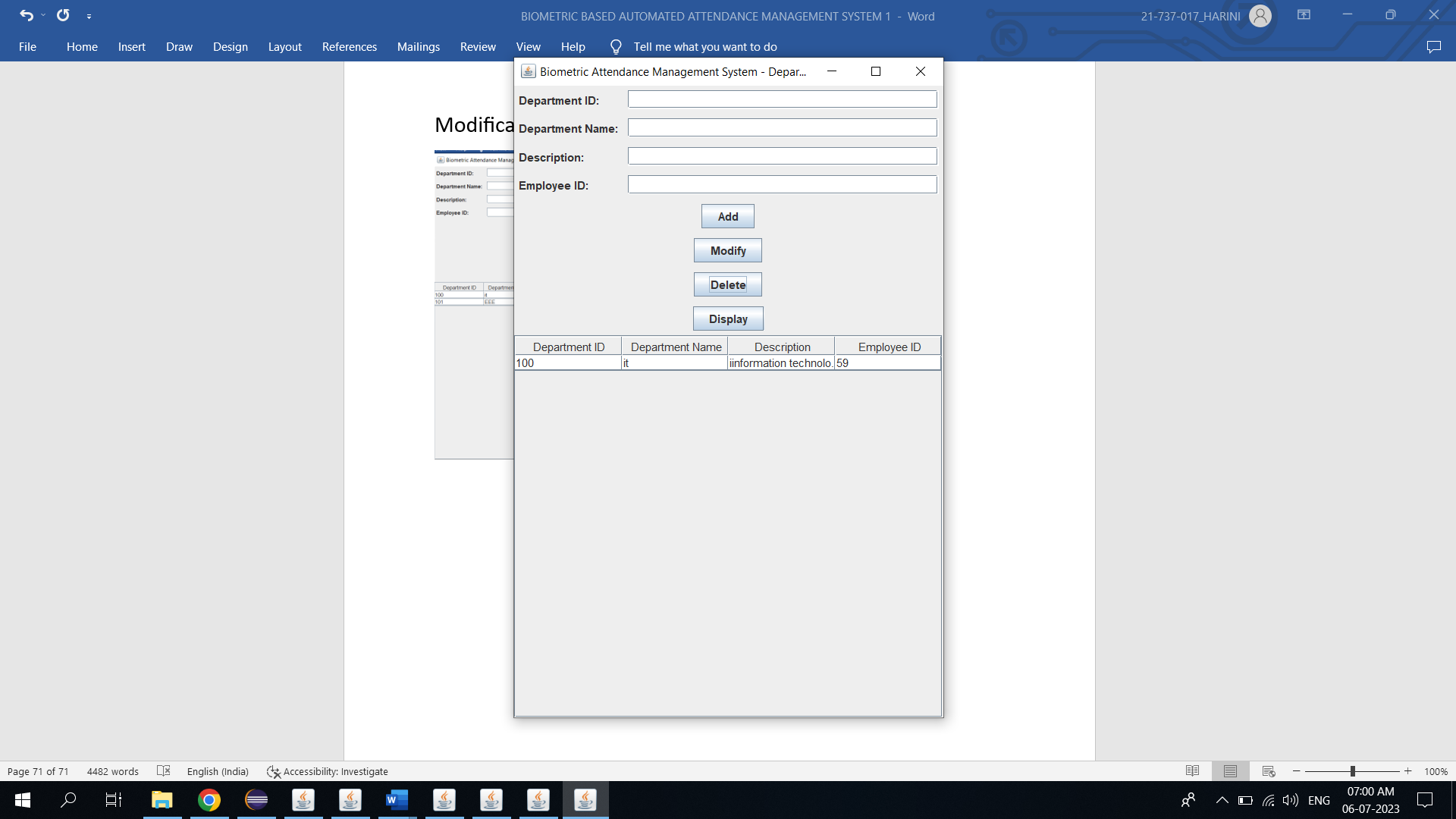
 

Department table Page:

Before insertion: After insertion:

Modification: Deletion:

**RESULT**

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

**DISCUSSION AND FUTURE WORK**

**Discussion:**

The Biometric Based Automated Attendance Management System is a reliable and efficient solution that utilizes biometric technology for accurate attendance tracking. It eliminates manual processes, reduces time theft, and ensures data accuracy. The system improves security and compliance, streamlines HR tasks, and increases overall productivity.

**Future Work:**

Integration with other HR systems for seamless data exchange.

Mobile application support for convenient attendance marking.

Real-time notifications and alerts for attendance anomalies.

Advanced analytics and reporting for data-driven decision making.

Integration with access control systems for enhanced security.

**References**

. <https://docs.oracle.com/javase/7/docs/api/>

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

. <https://stackoverflow.com/>