JAVA AWT BASED – BIOMETRIC BASED AUTOMATED METRO RAIL SYSTEM -SQL CONNECTIVITY USING JDBC

A report submitted in partial fulfillment of the Requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By- B. Sai Manisha (1602-18-737-097)

Under the guidance

Of

B.Leelavathy

Department of Information Technology

VASAVI COLLEGE OF ENGINEERING

AUTONOMOUS (AFFILIATED TO O.U)

IBRAHIMBAGH, HYDERABAD-500031

2019-20

BONAFIDE CERTIFICATE

This is to certify that this project report titled

"BIOMETRIC BASED AUTOMATED METRO RAIL SYSTEM" is the bonafide mini project work of **Ms. B. Sai Manisha**

Bearing hall ticket number **1602-18-737-097** under the guidance of **B. Leelavathy** during 4th semester B.E for the academic year **2019-2020.**

External Examiner

Internal Examiner
B.LEELAVATHY
Assistant professor
Department of Information Technology

AIM AND PRIORITY OF THE PROJECT:

To create a GUI based form for the project of **BIOMETRIC BASED AUTOMATED METRO RAIL SYSTEM** where passengers can reserve their tickets using biometric system which is faster and safer.

The values entered (insertion, updating and deletion) by the user for

Respective table in **GUI** should be updated in the database

using **JDBC.**

ABSTRACT:

The project is an application software developed for monitoring the biometric system which mainly focuses on basic operations like scanning the fingerprint, updating information, and generating online transactions. It is mainly developed for the sake of passengers who need to wait in the long queues for the ticket near the counter. This project helps to solve the problem by providing biometric system. The biometric scanner identifies the passengers who have subscribed for a metro card and directly deducts the corresponding amount from their card which makes the transactions easier and safer. This article aims to provide a structured approach to minimize the time duration of waiting in queues and also makes the system more secure.

A biometric scanner can scan any number of passengers and any number of passengers can also scan biometric scanner and hence many to many mapping cardinality is established between passengers and biometric scanners.

A passenger can reserve only one metro card and vice versa and hence one to one mapping cardinality is established between passengers and metro card.

Any number of transactions can be made from metro card and a transaction can be generated from many number of metro cards and hence it is a many to many mapping cardinality.

A. REQUIREMENTS:

Tables Required: (5) Users, Views, Videos, Uploads, Admin.

TABLE	DSECRPTION	ATTRIBUTE	DATATYPE
BIOMETRIC SCANNERS	Scanner ID	sid	NUMBER(20)
	Scanner Name	name	VARCHAR2(20)
	Cost	cost	NUMBER(20)
	Accuracy	accuracy	NUMBER(3)
PASSENGERS	Passenger ID	pid	NUMBER(20)
	Passenger Name	name	VARCHAR2(20)
	Mail ID	mail_id	VARCHAR2(20)
	Contact	contact_number	NUMBER(20)

SCANS	Scanner ID Passenger ID Subscription plan Time	usid pid subscription when	NUMBER(20) NUMBER(20) VARCHAR(20) VARCHAR2(20)
METRO_ CARD	Card ID Balance Expiry date	cardid balance age	NUMBER(20) NUMBER(20) VARCHAR2(20)
TRANSACTI ON	Transaction ID Amount deducted Available balance	tid amount_deducted available_balance	NUMBER(20) NUMBER(20) NUMBER(20)

C.ARCHITECTURE AND TECHNOLOGY USED:

Java Eclipse, Oracle 11g Database, java SE version 8, SQL *plus, java AWT

Eclipse: It is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug in system for customizing the environment. The Eclipse software development kit (SDK), which includes java development tools is meant for java developers.

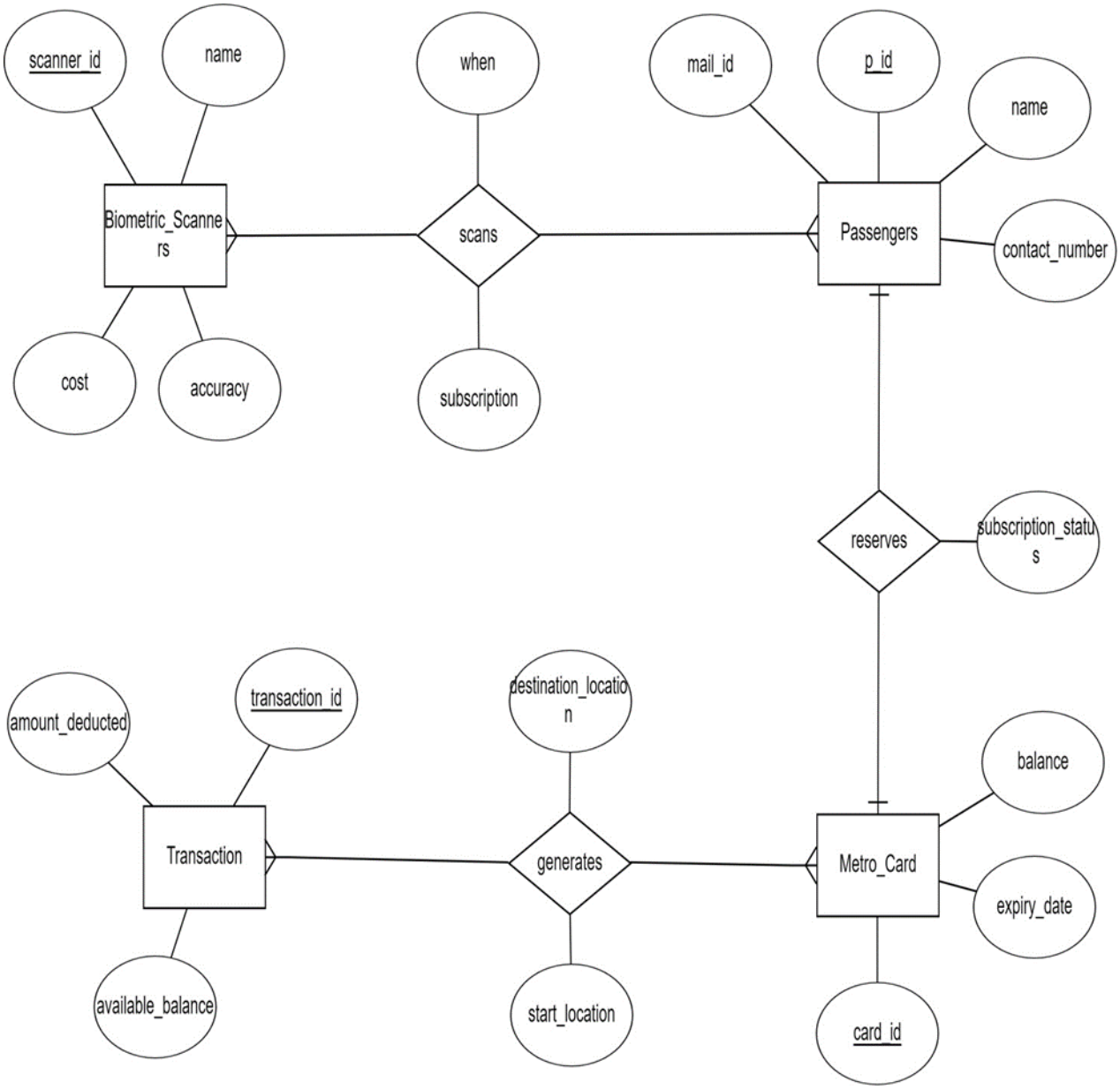
SQL *plus: SQL *plus is a command line tool proprietary to oracle. You can send SQL Queries to the server using the tool. It can also help you format the result of a query. SQL is the query language that is used to communicate with the oracle server to access and modify data.

JAVA AWT: Abstract window toolkit is an API to develop GUI or Window based applications in java. Java AWT components are platform dependent i.e., components are displayed according to the

view of the operating system. AWT is a heavy weight that is components are using the resources of O.S.

JDBC: Java Database Connectivity is an application programming interface (API) for the programming language java**,** which defines how a client may access a database. It is a java based data access technology used for java database connectivity. It is part of the java Standard Edition Platform, from Oracle Corporation.

D.ER DIAGRAM OF THIS PROJECT:



OUTPUTS

DDL COMMANDS

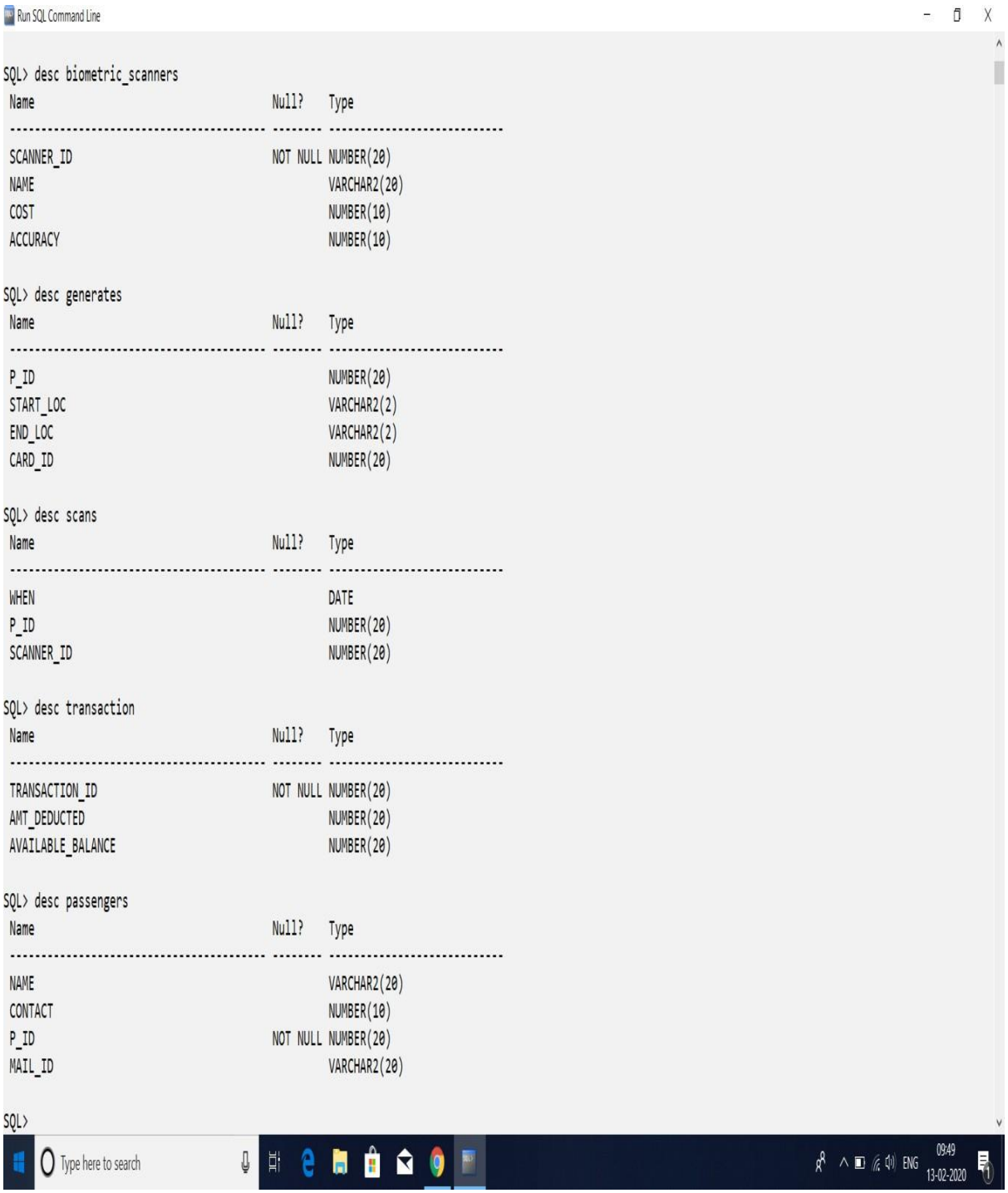
1. **Biometric_Scanners**
2. **Scans**

3. **Passengers**
4. **Reserves**
5. **Metro_card**
6. **Generates**
7. **Transaction**

```
SQL> create table Passengers(
  2  name varchar2(20),
  3  contact number(10),
  4  p_id number(20) primary key,
  5  mail id varchar2(20));
mail id varchar2(20))
       *
ERROR at line 5:
ORA-00907: missing right parenthesis


SQL> create table Passengers(
  2  name varchar2(20),
  3  contact number(10),
  4  p_id number(20) primary key,
  5  mail_id varchar2(20));

Table created.

SQL> create table Biometric_Scanners(
  2  scanner_id number(20),
  3  name varchar2(20),
  4  scanner_id number(20) primary key
  5
SQL> create table Biometric_Scanners(
  2  scanner_id number(20) primary key,
  3  name varchar2(20),
  4  cost number(10),
  5  accuracy number(10));

Table created.

SQL> create table Metro_Card(
  2  card_id number(20) primary key,
  3  validity date,
  4  balance number(20));

Table created.

SQL> create table transaction(
  2  transaction_id number(20) primary key,
  3  amt_deducted number(20),
  4  available_balance number(20));

Table created.
```

```
SQL> create table Scanned_by(
  2  when date,
  3  p_id number(20) foreign key references Passengers,
  4  scanner_id number(20) foreign key references Biometric_Scanners);
p_id number(20) foreign key references Passengers,
              *
ERROR at line 3:
ORA-00907: missing right parenthesis


SQL> create table Scanned_by(
  2  when date,
  3  p_id number(20),
  4  scanner_id number(20),
  5  foreign key(p_id) references Passengers,
  6  foreign key(scanner_id) references Biometric_Scanners);

Table created.

SQL> create table reserves(
  2  subscription_status varchar2(20),
  3  card_id number(20),
  4  p_id number(20),
  5  foreign key(p_id) references Passengers,
  6  foreign key(card_id) references Metro_Card);

Table created.

SQL> create table generates(
  2  p_id number(20),
  3  start_loc varchar(2),
  4  end_loc varchar(2),
  5  card_id number(20),
  6  foreign key(p_id) references Passengers,
  7  foreign key(card_id) references Metro_Card);

Table created.
```

DESCRIPTION OF TABLES


Run SQL Command Line
SQL> desc biometric_scanners
Name Null? Type
SCANNER_ID NOT NULL NUMBER(20)
NAME VARCHAR2(20)
COST NUMBER(10)
ACCURACY NUMBER(10)
SQL> desc generates
Name Null? Type
P_ID NUMBER(20)
START_LOC VARCHAR2(2)
END_LOC VARCHAR2(2)
CARD_ID NUMBER(20)
SQL> desc scans
Name Null? Type
WHEN DATE
P_ID NUMBER(20)
SCANNER_ID NUMBER(20)
SQL> desc transaction
Name Null? Type
TRANSACTION_ID NOT NULL NUMBER(20)
AMT_DEDUCTED NUMBER(20)
AVAILABLE_BALANCE NUMBER(20)
SQL> desc passengers
Name Null? Type
NAME VARCHAR2(20)
CONTACT NUMBER(10)
P_ID NOT NULL NUMBER(20)
MAIL_ID VARCHAR2(20)
SQL>
Type here to search
ENG
09:49
13-02-2020

```
SQL> desc reserves
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SUBSCRIPTION_STATUS                                VARCHAR2(20)
 CARD_ID                                            NUMBER(20)
 P_ID                                               NUMBER(20)

SQL> desc metro_card
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CARD_ID                                   NOT NULL NUMBER(20)
 VALIDITY                                           DATE
 BALANCE                                            NUMBER(20)

SQL>
```

E. JAVA-SQL CONNECTIVITY USING JDBC:

I) FRONT END PROGRAMS AND CONNECTIVITY

The connection to the database can be performed using java programming **(JDBC API)** as**:**

```
public void connectToDB() {
try {
      connection=DriverManager.getConnection("jdbc:oracle:thin:@l
ocalhost:1521:ORCL","msr","vasavi");
            statement = connection.createStatement();
}
catch (SQLException connectException) {
    System.out.println(connectException.getMessage());
    System.out.println(connectException.getSQLState());
    System.out.println(connectException.getErrorCode());
    System.exit(1);
}
catch(Exception e){
   System.out.println("Unable to find and load driver");
   System.exit(1);} }
```

AS THIS PROJECT CONTAINS 5 TABLES

i.**e. BIOMETRIC_SCANNERS, SCANS, PASSENGERS, RESERVES, METRO_CARD, GENERATES, TRANSACTION**

BELOW IS THE CODE FOR ALL **DML OPERATIONS** ON THE TABLE **BIOMETRIC_SCANNERS**

INSERT BIOMETRIC_SCANNER:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class InsertScanner extends Frame
{
	Button insertScannerButton;
	TextField sidText, snameText, costText, accuracyText;
	TextArea errorText;
	Connection connection;
	Statement statement;
	public InsertScanner()
	{
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");
		}
		catch (Exception e)
		{
```

```
                System.err.println("Unable to find and load driver");
                System.exit(1);
            }
            connectToDB();
        }

        public void connectToDB()
    {
            try
            {
              connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"manisha","vasavi");
              statement = connection.createStatement();


            }
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
    }
        public void buildGUI()
```

```
        {
                //Handle Insert Account Button
                insertScannerButton = new Button("Insert");
                insertScannerButton.addActionListener(new
ActionListener()
                {
                        public void actionPerformed(ActionEvent e)
                        {
                                try
                                {


                                  String query= "INSERT INTO
biometric_scanners VALUES(" + sidText.getText() + ", " + "'" +
snameText.getText() + "'," + costText.getText() + "," +
accuracyText.getText() + ")";
                                  int i = statement.executeUpdate(query);
                                  errorText.append("\nInserted " + i + " rows
successfully");
                                }
                                catch (SQLException insertException)
                                {
                                  displaySQLErrors(insertException);
                                }
                        }
                });
```

```
        sidText = new TextField(15);
        snameText = new TextField(15);
        costText = new TextField(15);
        accuracyText = new TextField(15);


        errorText = new TextArea(10, 40);
        errorText.setEditable(false);

        Panel first = new Panel();
        first.setLayout(new GridLayout(4, 2));
        first.add(new Label("Scanner ID:"));
        first.add(sidText);
        first.add(new Label("Name:"));
        first.add(snameText);
        first.add(new Label("Cost:"));
        first.add(costText);
        first.add(new Label("Accuracy:"));
        first.add(accuracyText);
        first.setBounds(125,90,200,100);

        Panel second = new Panel(new GridLayout(4, 1));
        second.add(insertScannerButton);
second.setBounds(125,220,150,100);
```

```
        Panel third = new Panel();
        third.add(errorText);
        third.setBounds(125,320,300,200);

        setLayout(null);

        add(first);
        add(second);
        add(third);

        setTitle("INSERT BIOMETRIC SCANNER");
        setSize(500, 600);
        setVisible(true);
    }



    private void displaySQLErrors(SQLException e)
    {
        errorText.append("\nSQLException: " + e.getMessage() +
"\n");
        errorText.append("SQLState:     " + e.getSQLState() +
"\n");
        errorText.append("VendorError:  " + e.getErrorCode() +
"\n");
```

```
    }



    public static void main(String[] args)
    {
        InsertScanner s = new InsertScanner();

        s.addWindowListener(new WindowAdapter(){
          public void windowClosing(WindowEvent e)
          {
              System.exit(0);
          }
        });

        s.buildGUI();
    }
}
```

UPDATE BIOMETRIC_SCANNER:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateScanner extends Frame
{
    Button updateScannerButton;
```

```
    List scannerIDList;
    TextField sidText, snameText, costText, accuracyText;
    TextArea errorText;
    Connection connection;
    Statement statement;
    ResultSet rs;

    public UpdateScanner()
    {
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        }
        catch (Exception e)
        {
            System.err.println("Unable to find and load driver");
            System.exit(1);
        }
        connectToDB();
    }

    public void connectToDB()
{
        try
        {
```

```
                connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"manisha","vasavi");
                statement = connection.createStatement();


            }
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
    }


        private void loadScanners()
        {
                try
                {
                  rs = statement.executeQuery("SELECT SID FROM
biometric_scanners");
                  while (rs.next())
                  {
                        scannerIDList.add(rs.getString("SID"));
                  }
                }
```

```
            catch (SQLException e)
            {
              displaySQLErrors(e);
            }
        }

        public void buildGUI()
        {
           scannerIDList = new List(10);
            loadScanners();
            add(scannerIDList);

            //When a list item is selected populate the text fields
            scannerIDList.addItemListener(new ItemListener()
            {
                  public void itemStateChanged(ItemEvent e)
                  {
                        try
                        {
                              rs = statement.executeQuery("SELECT *
FROM biometric_scanners where SID
="+scannerIDList.getSelectedItem());
                              rs.next();
                              sidText.setText(rs.getString("SID"));

        snameText.setText(rs.getString("NAME"));
```

```
                        costText.setText(rs.getString("COST"));

        accuracyText.setText(rs.getString("ACCURACY"));

                    }
                    catch (SQLException selectException)
                    {
                        displaySQLErrors(selectException);
                    }
                }
            });


            //Handle Update Sailor Button
            updateScannerButton = new Button("Update");
            updateScannerButton.addActionListener(new
ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    try
                    {
                        Statement statement =
connection.createStatement();
                        int i =
statement.executeUpdate("UPDATE biometric_scanners "
```

```
                        + "SET name='" + snameText.getText() +
"', "
                        + "cost=" + costText.getText() + ", "
                        + "accuracy ="+ accuracyText.getText()
+ " WHERE sid = "
                        + scannerIDList.getSelectedItem());
                        errorText.append("\nUpdated " + i + "
rows successfully");
                        scannerIDList.removeAll();
                        loadScanners();
                    }
                    catch (SQLException insertException)
                    {
                        displaySQLErrors(insertException);
                    }
                }
            });

            sidText = new TextField(15);
            sidText.setEditable(false);
            snameText = new TextField(15);
            costText = new TextField(15);
            accuracyText = new TextField(15);

            errorText = new TextArea(10, 40);
            errorText.setEditable(false);
```

```
Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label("Scanner ID:"));
first.add(sidText);
first.add(new Label("Name:"));
first.add(snameText);
first.add(new Label("Cost:"));
first.add(costText);
first.add(new Label("Accuracy:"));
first.add(accuracyText);

Panel second = new Panel(new GridLayout(4, 1));
second.add(updateScannerButton);

Panel third = new Panel();
third.add(errorText);

add(first);
add(second);
add(third);

setTitle("Update Biometric Scanners");
setSize(500, 600);
setLayout(new FlowLayout());
```

```
            setVisible(true);

        }

        private void displaySQLErrors(SQLException e)
        {
            errorText.append("\nSQLException: " + e.getMessage() +
"\n");
            errorText.append("SQLState:     " + e.getSQLState() +
"\n");
            errorText.append("VendorError:  " + e.getErrorCode() +
"\n");
        }

        public static void main(String[] args)
        {
            UpdateScanner ups = new UpdateScanner();

            ups.addWindowListener(new WindowAdapter(){
              public void windowClosing(WindowEvent e)
              {
                  System.exit(0);
              }
            });

            ups.buildGUI();
```

```
        }
}
```

DELETE BIOMETRIC_SCANNER:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteScanner extends Frame
{
        Button deleteScannerButton;
        List scannerIDList;
        TextField sidText, snameText, costText, accuracyText;
        TextArea errorText;
        Connection connection;
        Statement statement;
        ResultSet rs;

        public DeleteScanner()
        {
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                }
                catch (Exception e)
```

```
            {
                System.err.println("Unable to find and load driver");
                System.exit(1);
            }
            connectToDB();
        }

        public void connectToDB()
    {
            try
            {
              connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe"
,"manisha","vasavi");
              statement = connection.createStatement();


            }
            catch (SQLException connectException)
            {
              System.out.println(connectException.getMessage());
              System.out.println(connectException.getSQLState());
              System.out.println(connectException.getErrorCode());
              System.exit(1);
            }
    }
```

```
    private void loadScanners()
    {
        try
        {
          rs = statement.executeQuery("SELECT * FROM
biometric_scanners");
          while (rs.next())
          {
              scannerIDList.add(rs.getString("SID"));
          }
        }
        catch (SQLException e)
        {
          displaySQLErrors(e);
        }
    }

    public void buildGUI()
    {
       scannerIDList = new List(10);
        loadScanners();
        add(scannerIDList);

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

```
            scannerIDList.addItemListener(new ItemListener()
            {
                  public void itemStateChanged(ItemEvent e)
                  {
                        try
                        {
                              rs = statement.executeQuery("SELECT *
FROM biometric_scanners");
                              while (rs.next())
                              {
                                    if
(rs.getString("SID").equals(scannerIDList.getSelectedItem()))
                                    break;
                              }
                              if (!rs.isAfterLast())
                              {

      sidText.setText(rs.getString("SID"));

      snameText.setText(rs.getString("NAME"));

      costText.setText(rs.getString("COST"));

      accuracyText.setText(rs.getString("ACCURACY"));
                              }
                        }
                        catch (SQLException selectException)
```

```
                {
                    displaySQLErrors(selectException);
                }
            }
        });


        //Handle Delete Sailor Button
        deleteScannerButton = new Button("Delete");
        deleteScannerButton.addActionListener(new
ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {
                    Statement statement =
connection.createStatement();
                    int i =
statement.executeUpdate("DELETE FROM biometric_scanners
WHERE SID = "
                                        +
scannerIDList.getSelectedItem());
                    errorText.append("\nDeleted " + i + "
rows successfully");
                    sidText.setText(null);
                    snameText.setText(null);
```

```
                costText.setText(null);
                accuracyText.setText(null);
                scannerIDList.removeAll();
                loadScanners();
            }
            catch (SQLException insertException)
            {
                displaySQLErrors(insertException);
            }
        }
});

sidText = new TextField(15);
snameText = new TextField(15);
costText = new TextField(15);
accuracyText = new TextField(15);

errorText = new TextArea(10, 40);
errorText.setEditable(false);

Panel first = new Panel();
first.setLayout(new GridLayout(4, 2));
first.add(new Label(" Biometric Scanner ID:"));
first.add(sidText);
first.add(new Label("Name:"));
```

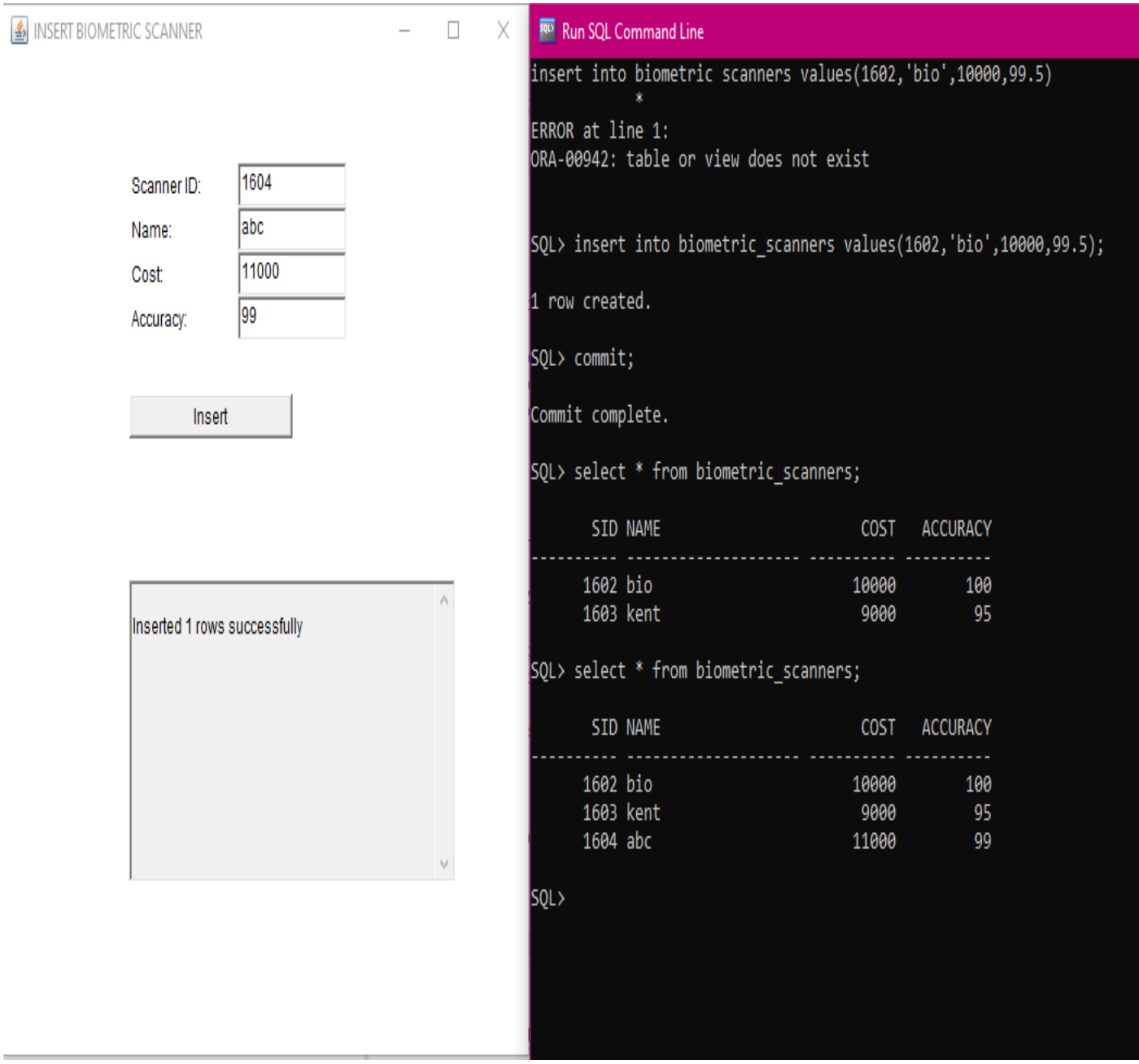
```
        first.add(snameText);
        first.add(new Label("Cost:"));
        first.add(costText);
        first.add(new Label("Accuracy:"));
        first.add(accuracyText);

        Panel second = new Panel(new GridLayout(4, 1));
        second.add(deleteScannerButton);

        Panel third = new Panel();
        third.add(errorText);

        add(first);
        add(second);
        add(third);

        setTitle("DELETE BIOMETRIC SCANNER");
        setSize(450, 600);
        setLayout(new FlowLayout());
        setVisible(true);

}
```

```
    private void displaySQLErrors(SQLException e)
    {
        errorText.append("\nSQLException: " + e.getMessage() +
"\n");
        errorText.append("SQLState:     " + e.getSQLState() +
"\n");
        errorText.append("VendorError:  " + e.getErrorCode() +
"\n");
    }



    public static void main(String[] args)
    {
        DeleteScanner dels = new DeleteScanner();

        dels.addWindowListener(new WindowAdapter(){
          public void windowClosing(WindowEvent e)
          {
              System.exit(0);
          }
        });

        dels.buildGUI();
    }
}
```

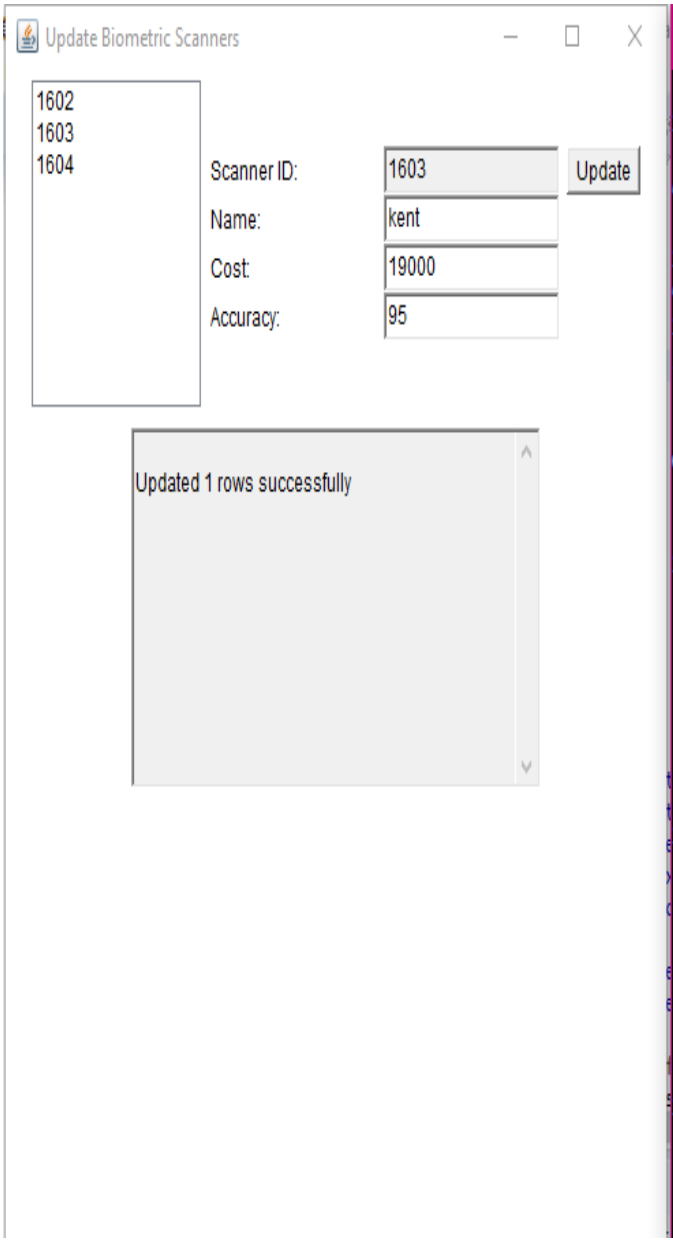
GITHUB LINK:

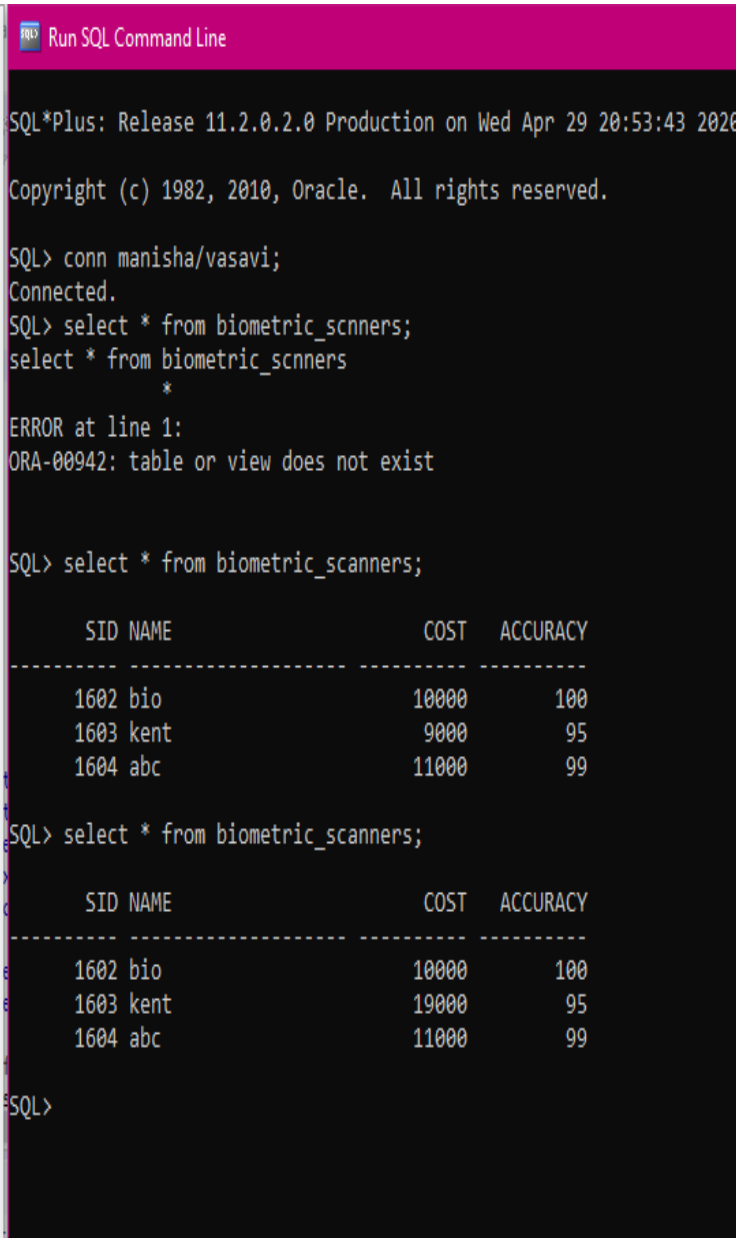
https://github.com/manisha1405/DBMS-ASSIGNMENT

OUTPUT OF INSERT SCANNER:

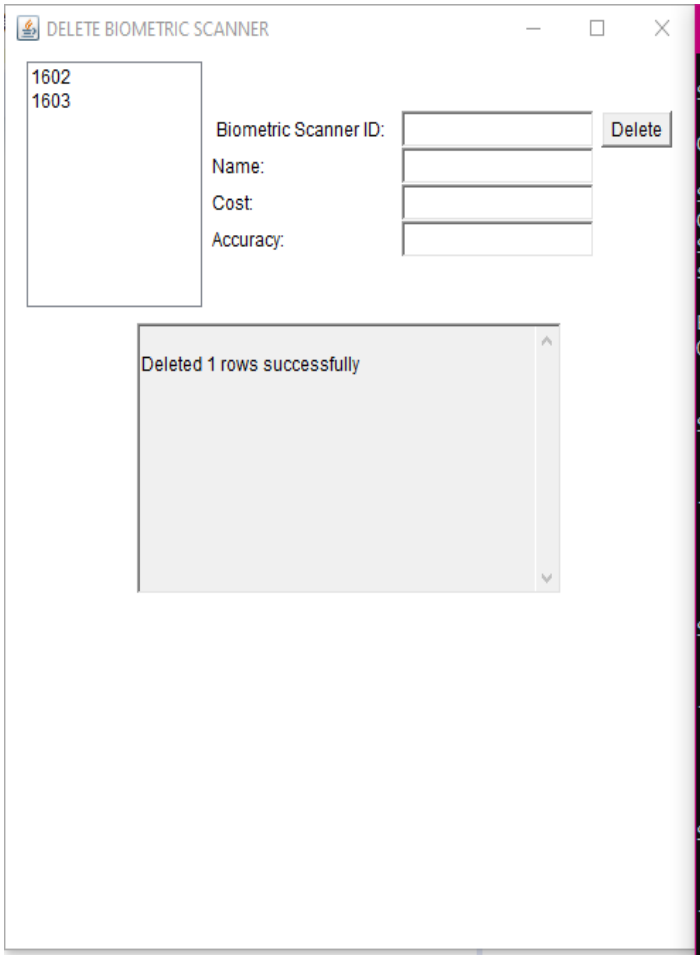


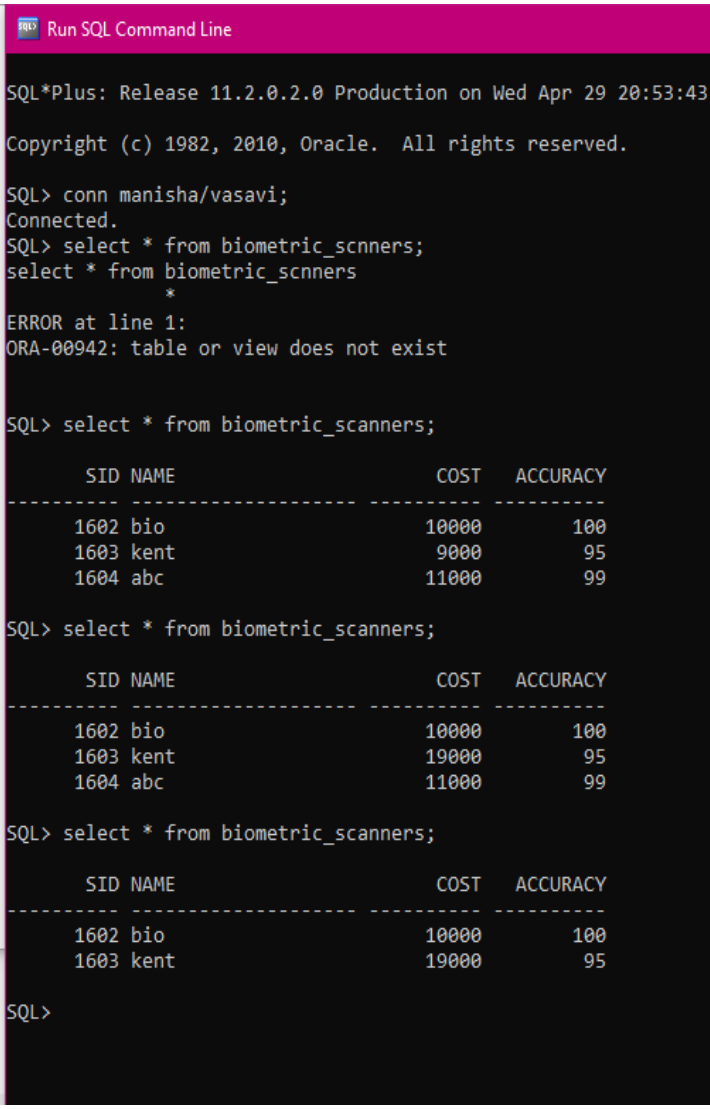
OUTPUT OF UPDATE SCANNER:





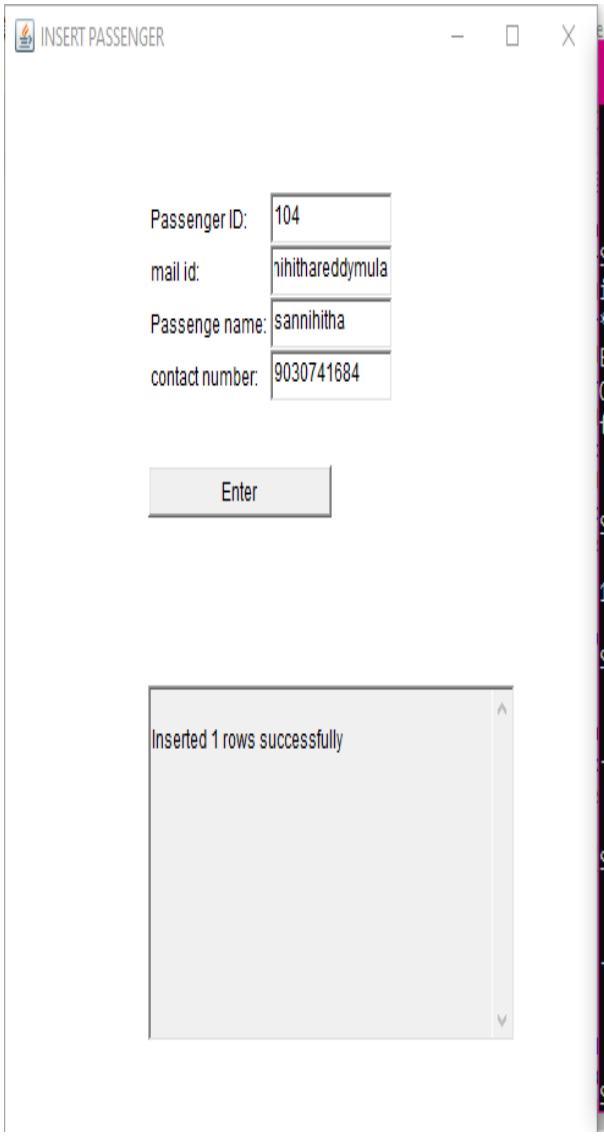
OUTPUT OF DELETE SCANNER

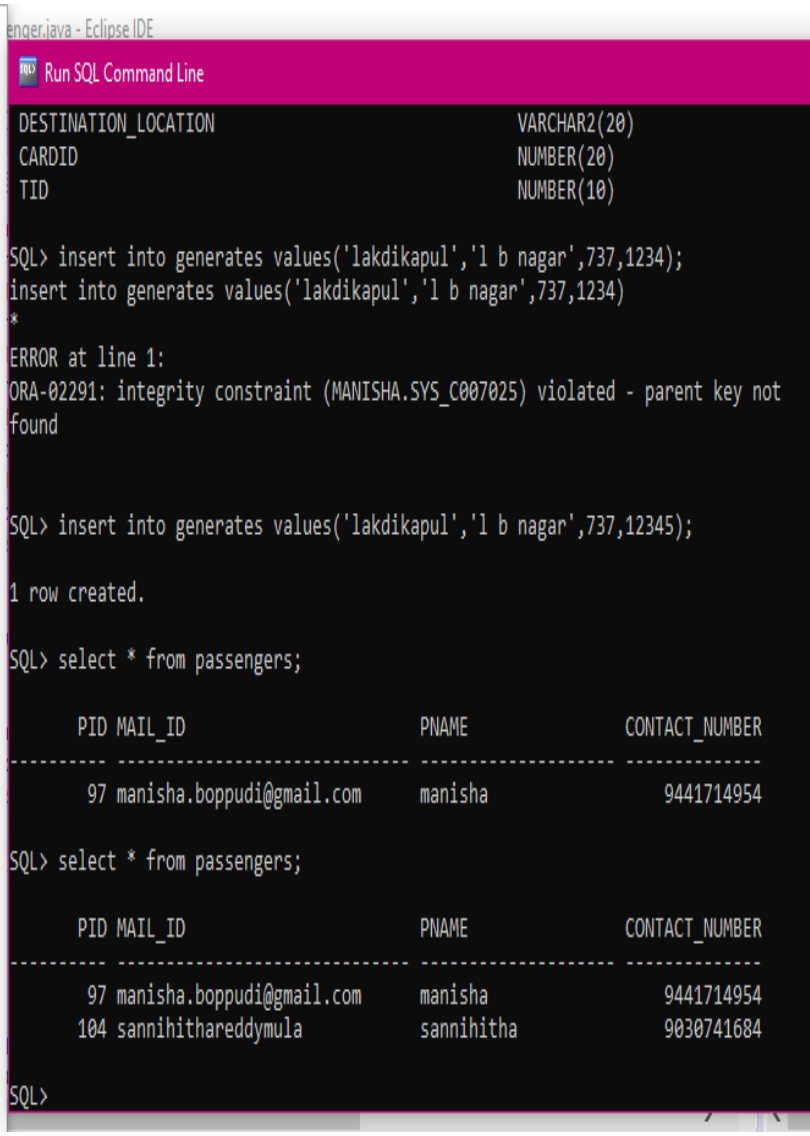




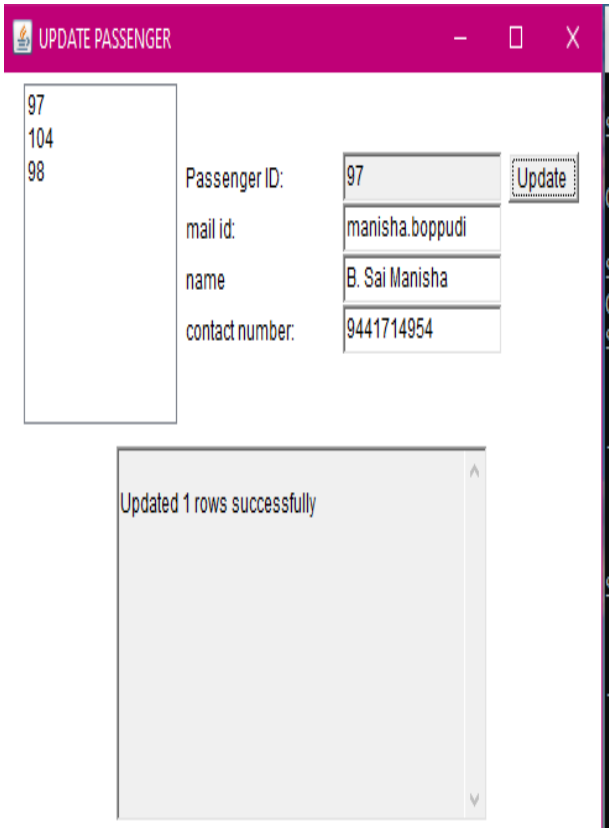
:

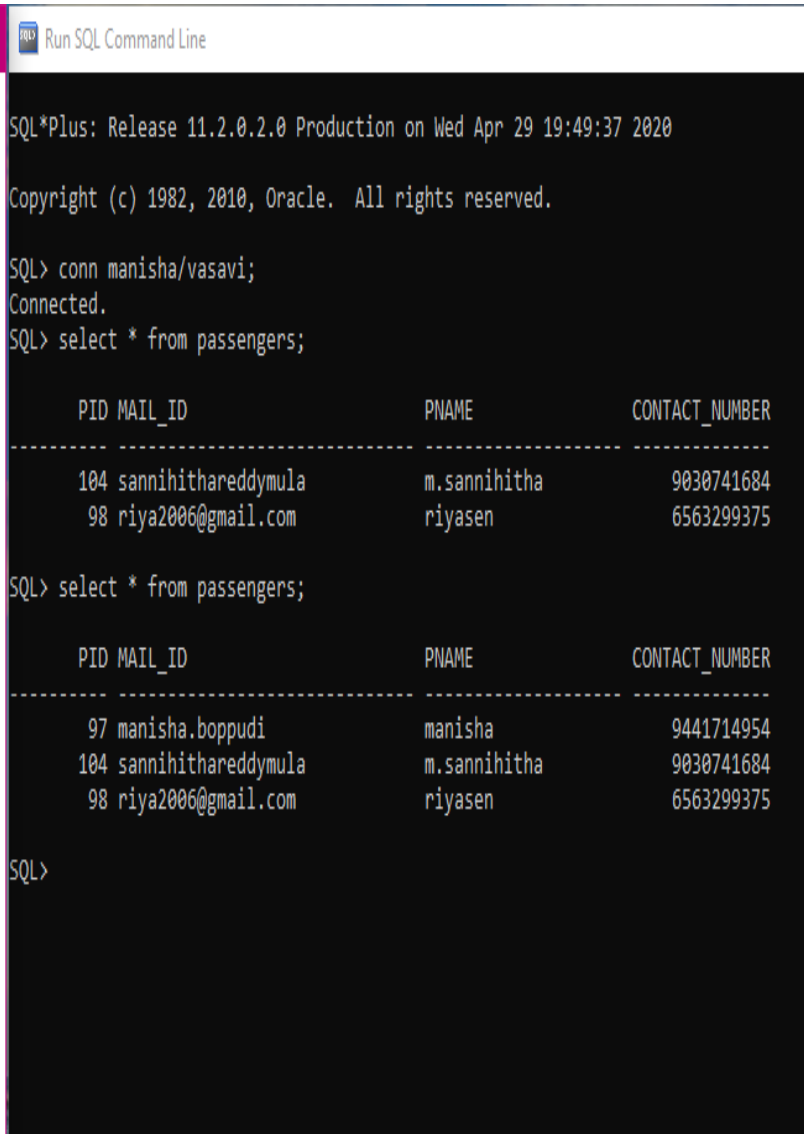
OUTPUT OF INSERT PASSENGER:



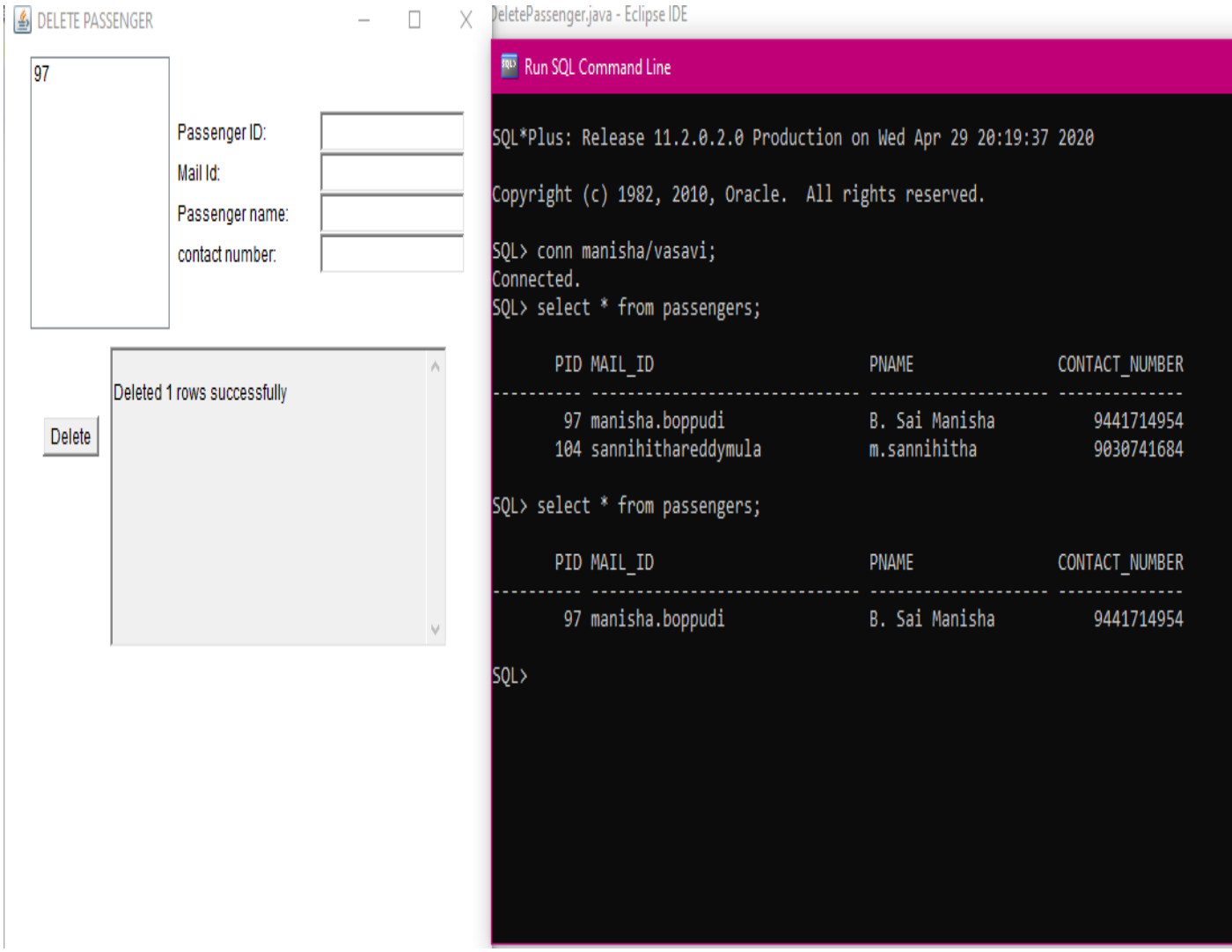


OUTPUT OF UPDATE PASSENGER:



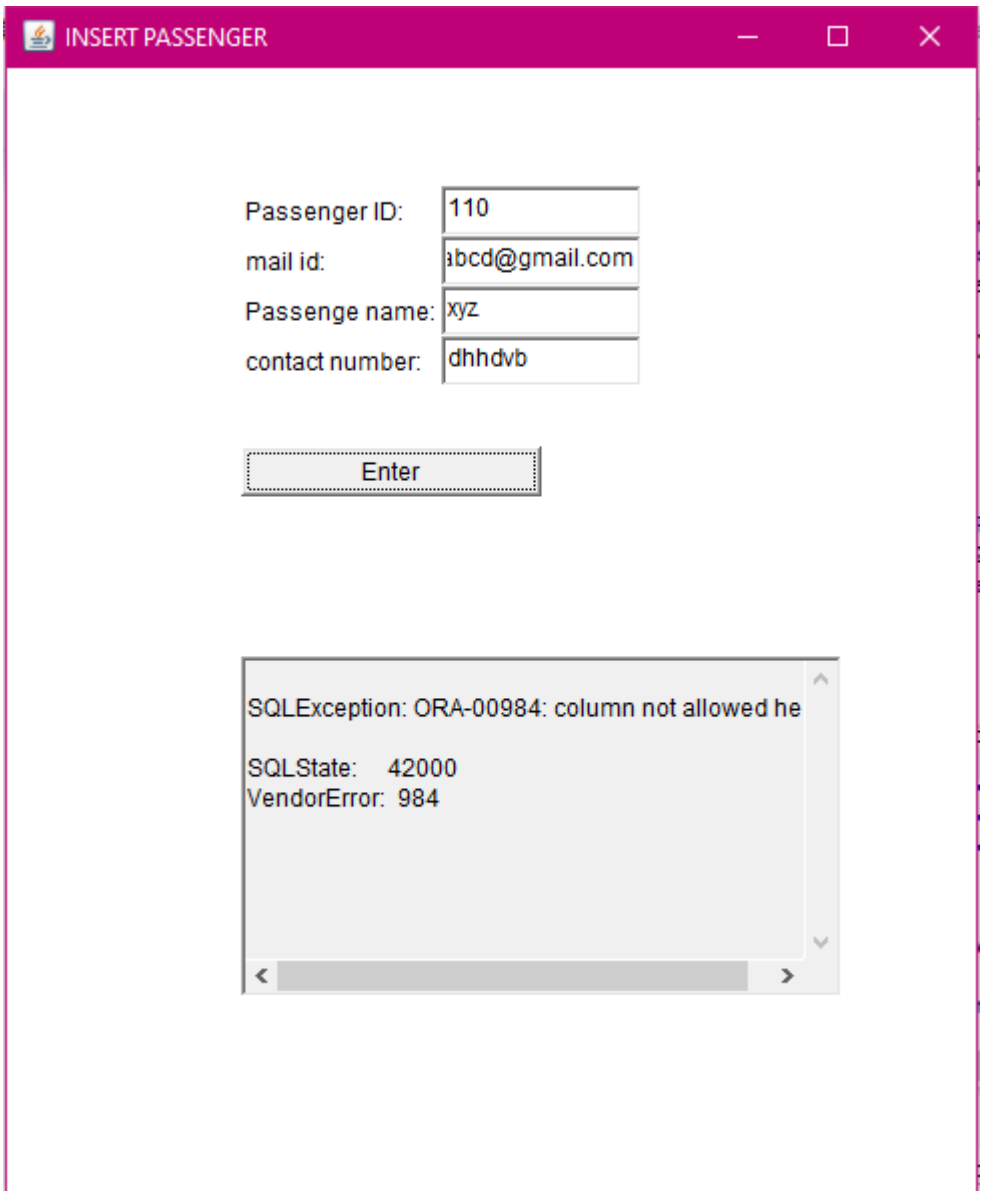


OUTPUT OF DELETE PASSENGER:



TESTING:

If a user enters a invalid values then an error message is displayed in pop up box.



RESULT:

The process of entering information into the frame created by java code so that the data is reflected in the database using **JDBC connectivity** is done successfully.

DISCUSSION AND FUTURE WORK!

The application till now done is a basic interface in which a passenger can travel just by placing his finger on the scanner without even carrying purse or wallet. It ensures a faster and safer journey for the passengers instead of making them wait in long queues.

REFERENCES:

https://docs.oracle.com/javase/8/docs/api/

https://www.geeksforgeeks.org/establishing-jdbc-connection-in-java/

https://www.javatpoint.com/java-awt