## horizontal line **JAVA SWINGS BASED- CARPOOLING**

**- 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

V Shwetha<1602-19-737-106>

Under the guidance of Ms B. Leelavathy



Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

BONAFIDE CERTIFICATE

This is to certify that this project report titled ‘**CARPOOLING**’ is a project work of Ms. V Shwetha bearing roll no.1602-19-737-106 who carried out the project under my supervision in the IV

semester for the academic year 2020- 2021.

Signature Signature Internal Examiner External Examiner

## 

## 

## 

## 

## DBMS ASSIGNMENT

CAR POOLING



NAME: V Shwetha

ROLL NUMBER: 1602-19-737-106

CLASS: IT-B

**ABSTRACT**

***“CAR POOLING”*** *is a system that helps users who own a car to arrange a carpool and the other users to find a car to travel. It is a system used to find cars to travel in which the owners are traveling alone and are ready to share the journey by charging the customers some money. This can help users save their money and time both by sharing their riders. The database has the data of both customers and owners along with the location where the carpool is arranged. This project has a total of 10 tables. It shows which owner is ready to share their ride along with the provisions of the ride which helps customers to choose the car easily by their requirements.*

**Requirement Analysis:**

*List Of Tables:*

1. Owner
2. Customer
3. Car
4. Pool
5. Provisions
6. Owns
7. Arranges
8. Registered
9. Provides

*List of Attributes with their Domains:*

Owner:

* Owner\_ID
* Name
* Age
* Mobile Number
* Gender
* Address

Customer:

* Customer\_ID
* Name
* Age
* Mobile Number
* Gender
* Address

Car:

* Car No
* Model
* Color
* Licence No

Pool:

* Pool ID
* Start
* End
* Date
* Time

Provisions:

* Owner\_Id
* Luggage Size
* Occupancy

In Registered Table:

* Location

**AIM AND PRIORITY OF THE PROJECT**

To create a Java GUI based information retrieval system of *“Car Pooling System”* which takes the values like: id, Name, Age, etc. from the person who owns the car and the person who wants to access the pool(customer) arranged by customer. These values are to be updated in the database using JDBC connectivity .

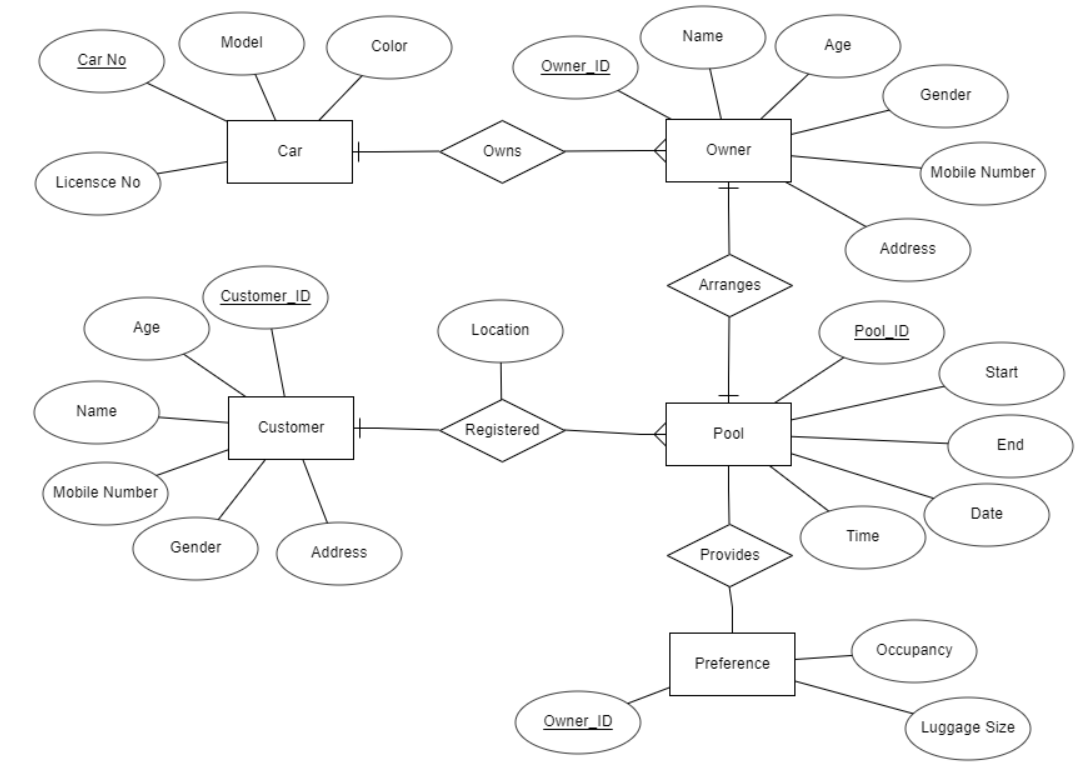
**ARCHITECTURE AND TECHNOLOGY**

**Software used**: Java Eclipse, Oracle 11g Database, Java SE version 13, SQL\*Plus.

**Java SWINGS**: Java SWINGS is an API to develop GUI or window-based applications in java. Java SWING components are platform-independent. It is lightweight. The javax.swing package provides classes for SWING API such as JTextField, JLabel, JTextArea, JRadioButton, JCheckBox, JChoice, JList etc.

**SQL**: Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS

**Entity Relationship Model:**

****

**Mapping Constraints:**

*Owner is connected to Car:* A single Owner can own multiple cars so it is one to many participation

*Owner is connected to Pool:* A single Owner can arrange many car pools at different times, so it is one to many.

*Customer is connected to Pool:* A Pool can be registered by more than one customer, so it is one to many.

*Pool is connected to Provisions:* A pool provides provisions. It is one to one.

DDL Commands:

* SQL>create table car(

2 car\_no varchar2(20) primary key,

3 model varchar2(20),

4 color varchar2(20),

5 licence\_no varchar2(15));

* SQL> create table owner(

2 owner\_id varchar2(20) primary key,

3 name varchar2(30),

4 age number(2),

5 mobile\_number number(10),

6 gender char(5),

7 dress varchar2(100));

* SQL>create table customer(

2 customer\_id varchar2(20) primary key,

3 name varchar2(30),

4 age number(2),

5 mobile\_number number(10),

6 gender char(5),

7 address varchar2(100));

* SQL> create table pool(

2 pool\_id varchar2(20) primary key,

3 start\_p varchar2(25),

4 end varchar2(25),

5 day varchar(20),

6 timing varchar(20));

* SQL> create table provisions(

2 owner\_id varchar2(20) primary key,

3 luggage\_size number(5),

4 occupancy number(2));

* SQL> create table owns(

2 owner\_id varchar2(20),

3 car\_no varchar2(20),

4 foreign key(owner\_id) references owner(owner\_id),

5 foreign key(car\_no) references car(car\_no),

6 primary key(owner\_id, car\_no));

* SQL> create table arranges(

2 owner\_id varchar2(20),

3 pool\_id varchar2(20),

4 foreign key(owner\_id) references owner(owner\_id),

5 foreign key(pool\_id) references pool(pool\_id),

6 primary key(owner\_id, pool\_id));

* SQL> create table registered(

2 customer\_id varchar2(20),

3 pool\_id varchar2(20),

4 location varchar2(100),

5 foreign key(customer\_id) references customer(customer\_id),

6 foreign key(pool\_id) references pool(pool\_id),

7 primary key(customer\_id, pool\_id));

* SQL> create table provides(

2 pool\_id varchar2(20),

3 owner\_id varchar2(20),

4 foreign key(owner\_id) references owner(owner\_id),

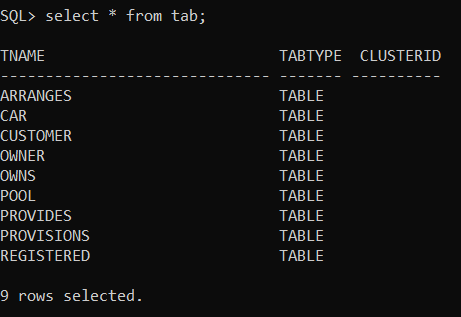
5 foreign key(pool\_id) references pool(pool\_id),

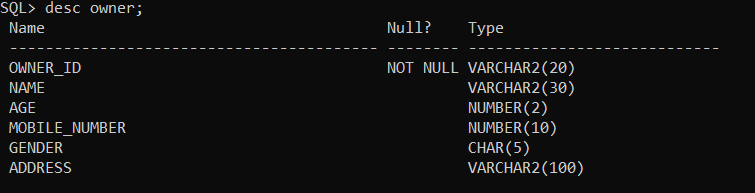
6 primary key(owner\_id, pool\_id));

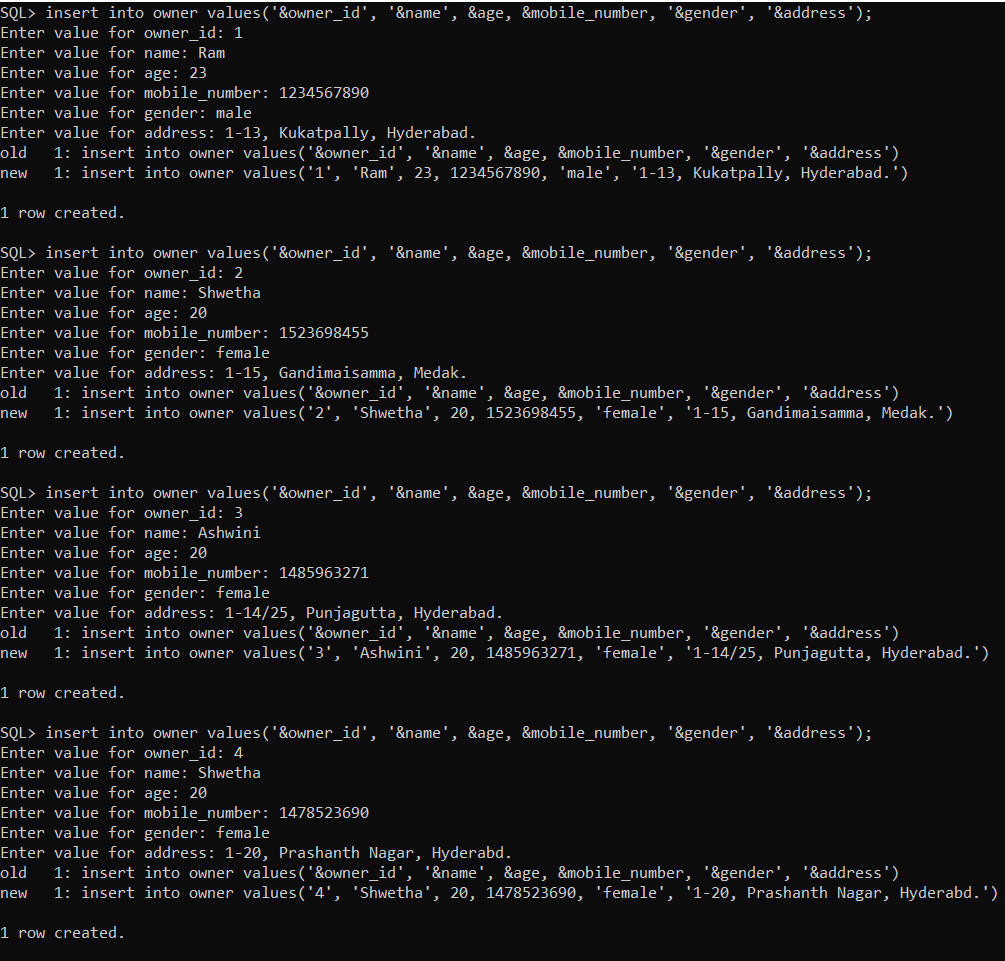
DML Commands:

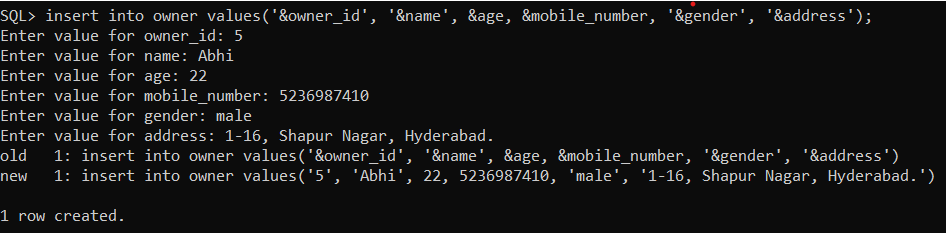
* SQL> insert into owner values('&owner\_id', '&name', &age, &mobile\_number, '&gender', '&address');
* SQL> insert into customer values('&customer\_id', '&name', &age, &mobile\_number, '&gender', '&address');
* SQL> insert into customer('&customer\_id', '&name', &age, &mobile\_number, '&gender', '&address');
* SQL> insert into provisions values('&owner\_id', &luggage\_size, &occupancy);
* SQL> insert into pool values('&pool\_id', '&start\_p', '&end', '&day', '&timing');
* SQL> insert into owns values('&owner\_id', '&car\_no');
* SQL> insert into provides values('&pool\_id', '&owner\_id');
* SQL> insert into arranges values('&owner\_id', '&pool\_id');
* SQL> insert into registered values('&customer\_id', '&pool\_id', '&location');

**OUTPUTS**

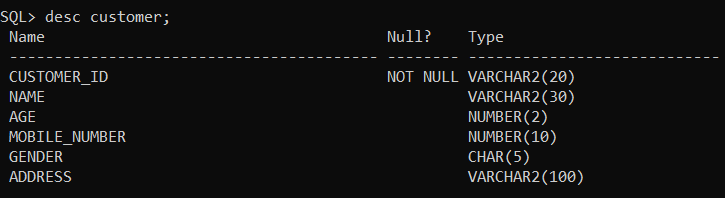
****

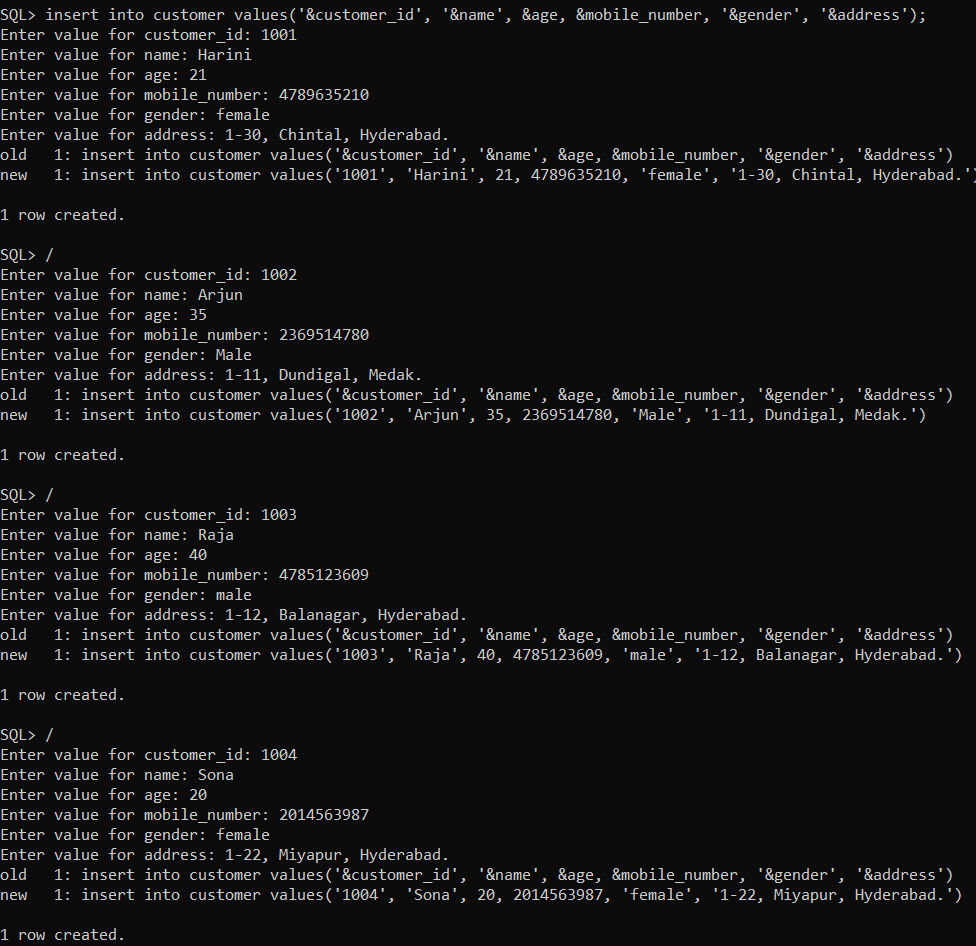
****

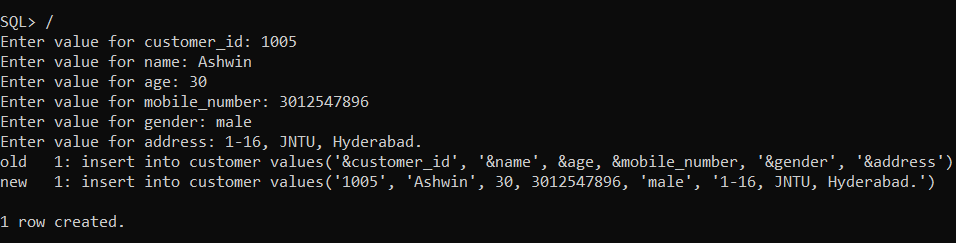
****

****

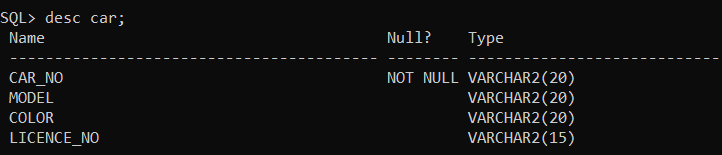
****

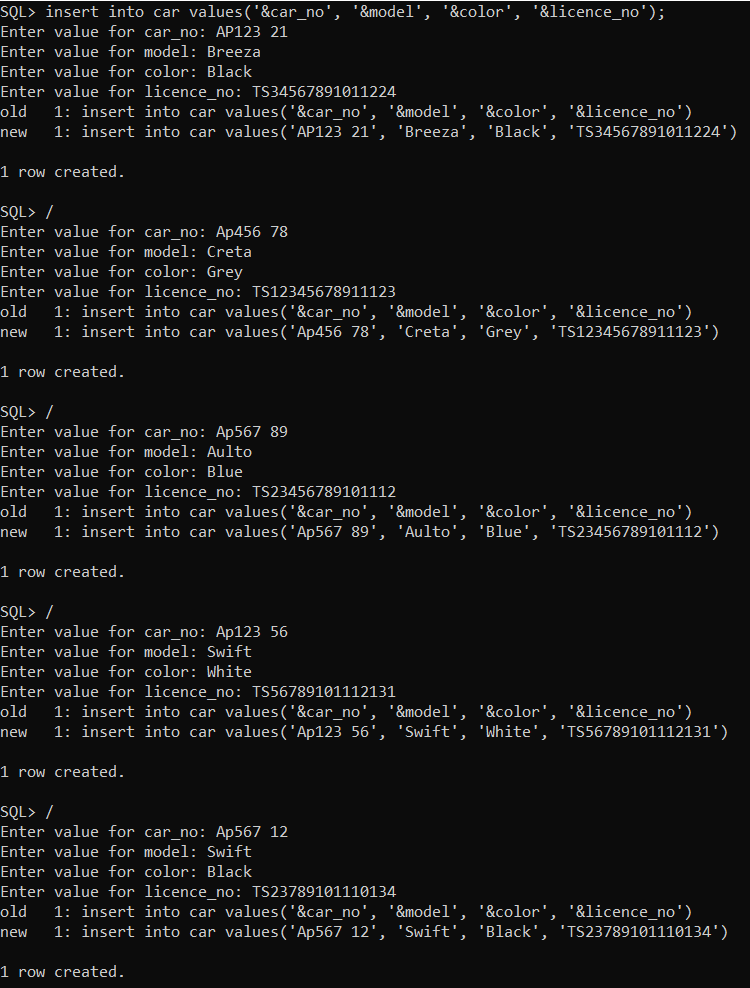
****

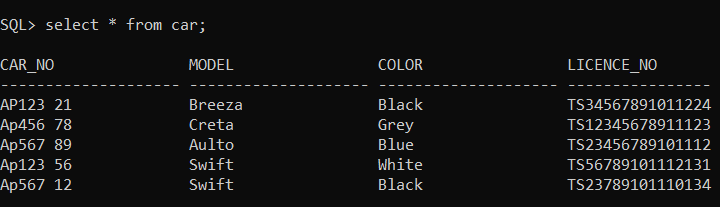
****

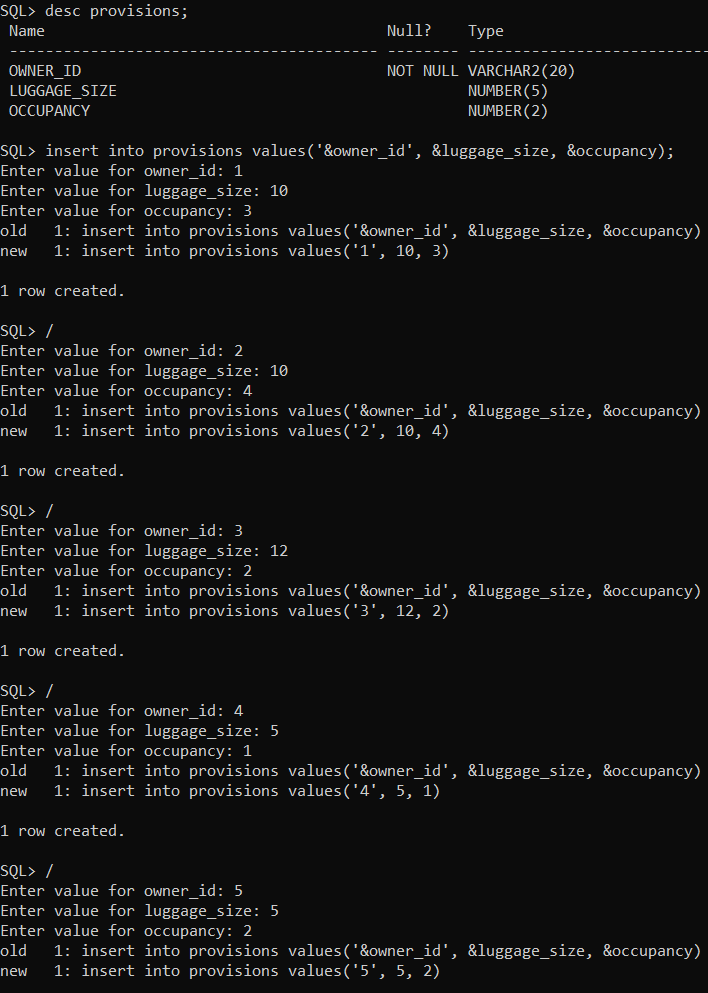
****

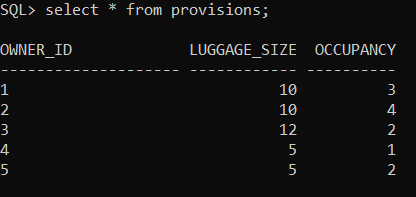
****

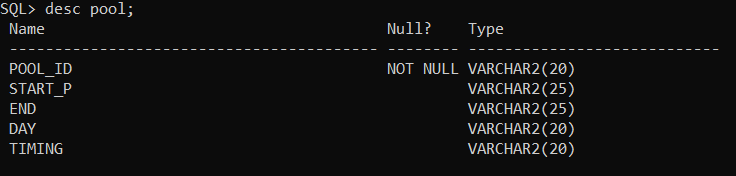
****

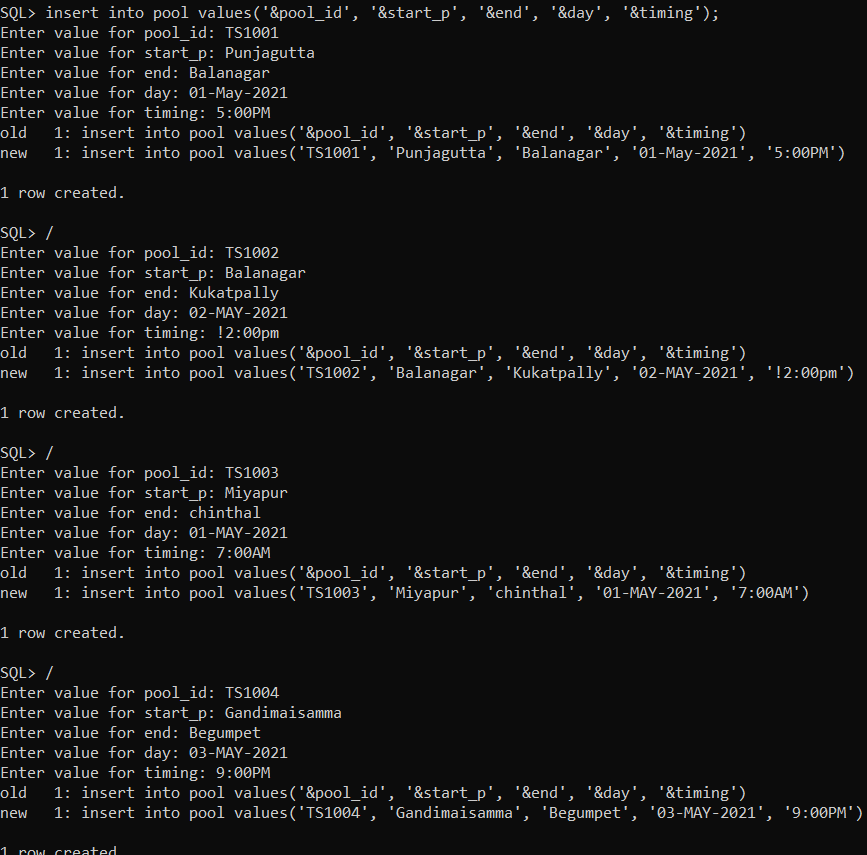
****

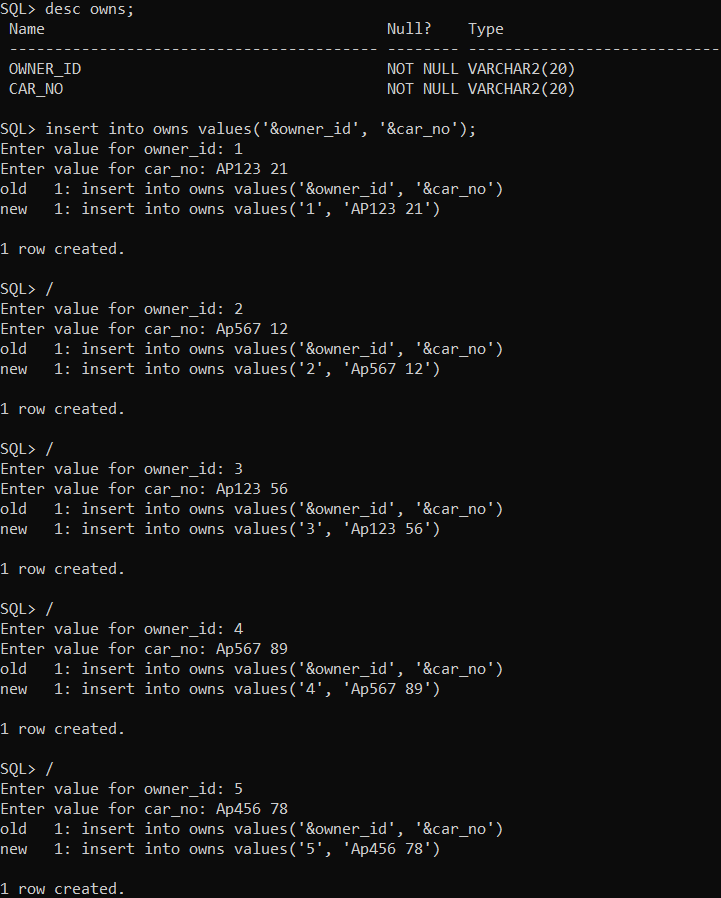
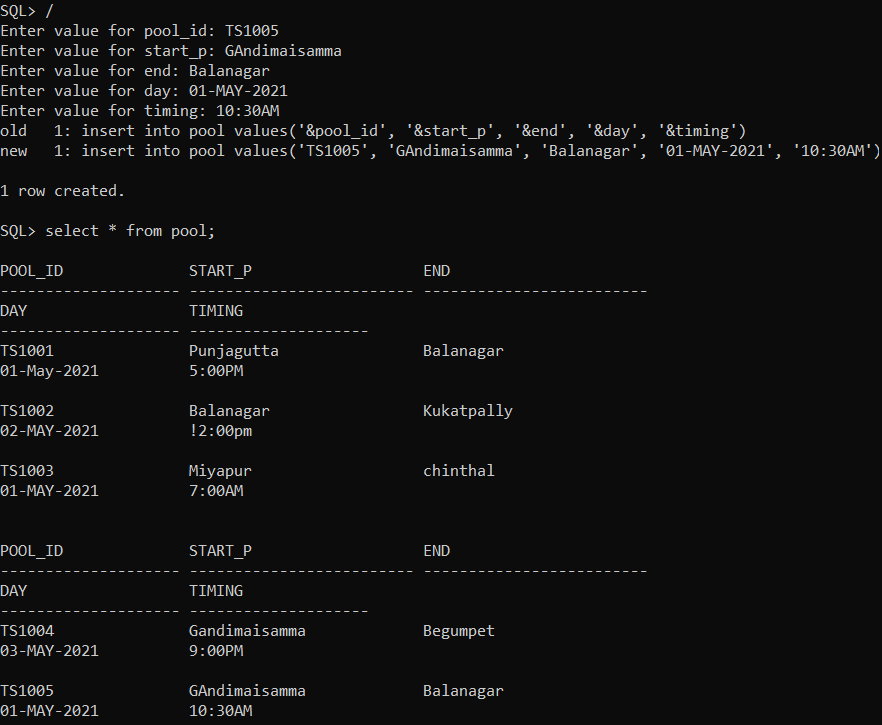
****

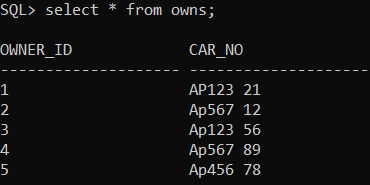
****

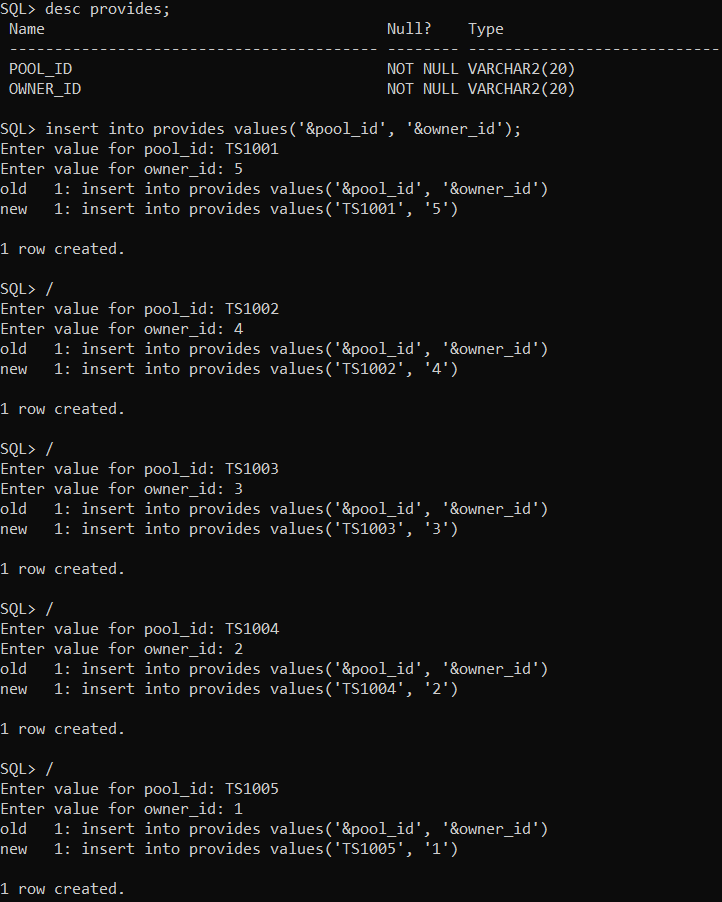
****

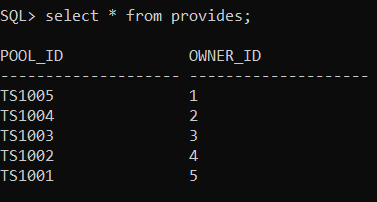
****

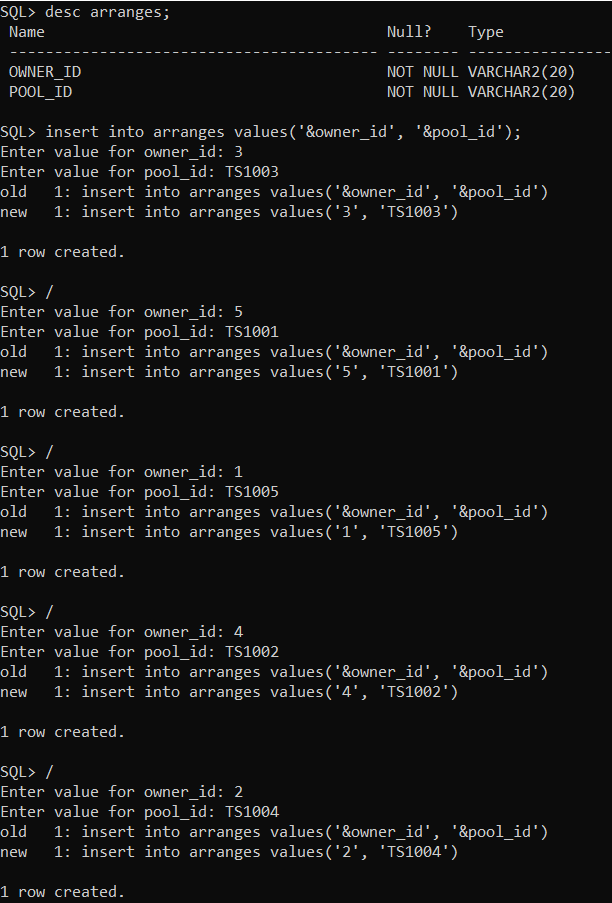
****

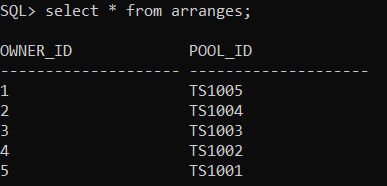
****

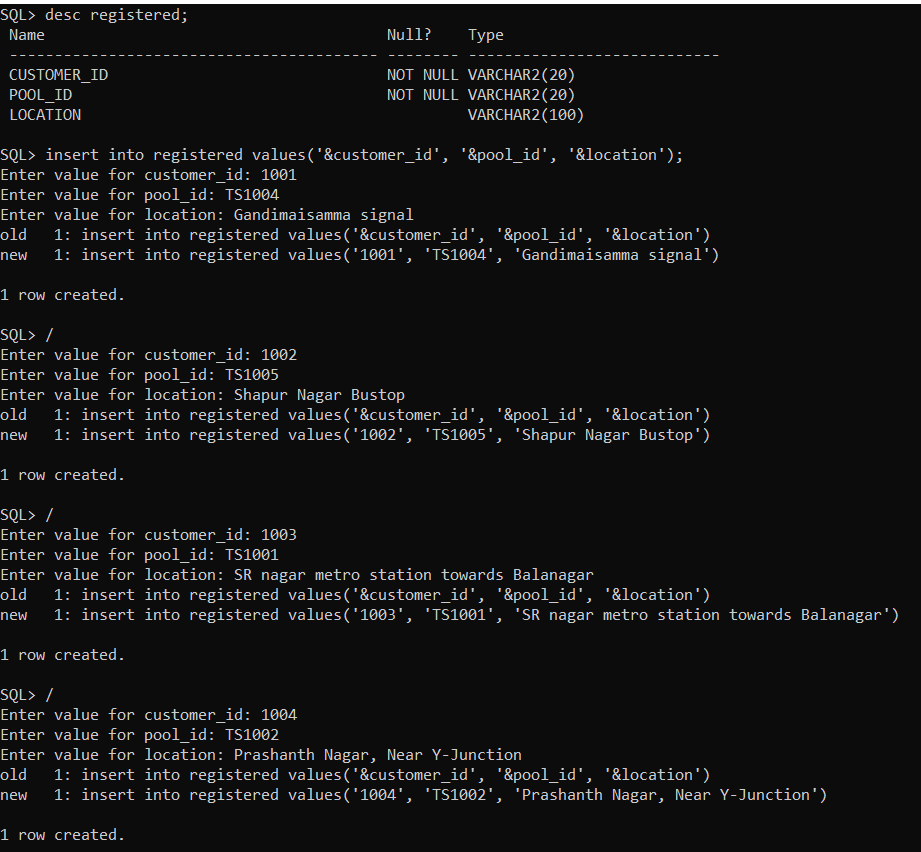
****

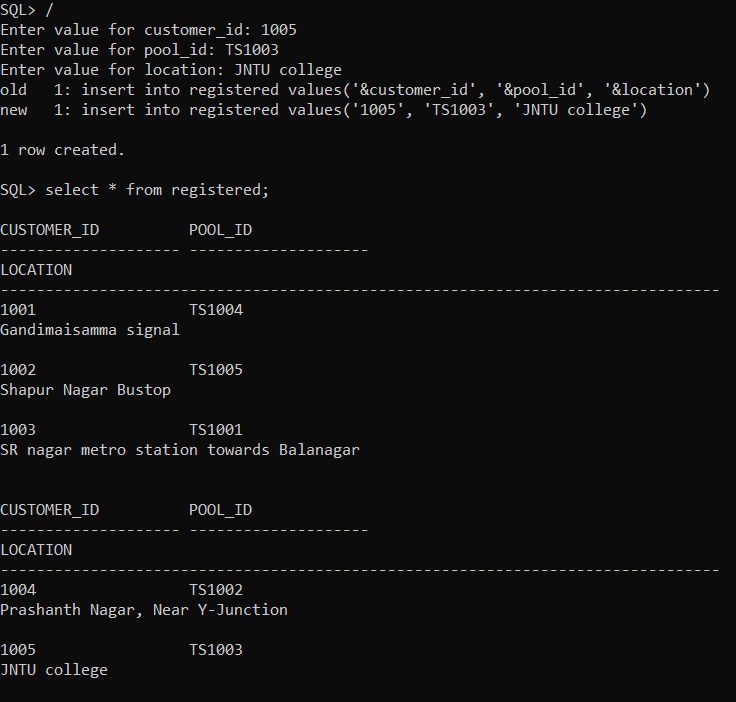
****

****

****

****

****

****

**IMPLEMENTATION**

**Front end programs and its connectivity:**

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:

public void connectToDB()

{

try

{

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","Shwetha","vasavi");

statement=con.createStatement(); statement.executeUpdate("commit");

}

catch (SQLException connectException)

{

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

System.exit(1);

}

}

Thus, the connection from Java to Oracle database is performed and therefore, can be used for updating tables in the database directly.

**Java Code:**

package DBMS;

import java.awt.\*;

import java.awt.event.\*;

import java.sql.\*;

@SuppressWarnings("serial")

public class InsertTables extends Frame implements ActionListener

{

MenuBar mb;

MenuItem m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,m17,m18,m19;

Menu owner,customer,pool,arranges,registered;

Button insertButton,submit;

TextField owner\_idText, nameText, ageText, mobile\_numberText, genderText, addressText;

TextField customer\_idText, NameText,AgeText,Mobile\_NumberText, GenderText,AddressText;

TextField pool\_idText, start\_pText, endText, dayText, timingText;

TextField Owner\_idText,Pool\_idText;

TextField Customer\_IdText,Pool\_IdText,locationText;

TextArea errorText;

Connection connection;

Statement statement;

Button modify;

List ownerList,customerList,poolList,arrangesList,registeredList;

ResultSet rs;

Button deleteRowButton;

public InsertTables()

{

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","Shwetha","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 buildFrame()

{

//Basic Frame Properties

setTitle("Car Pooling");

setSize(500, 600);

setVisible(true);

//menubar

mb = new MenuBar();

setMenuBar(mb);

setSize(550,500);

setLayout(null);

setVisible(true);

owner=new Menu("Owner");

m1=new MenuItem("Insert Owner");

m2=new MenuItem("Update Owner");

m3=new MenuItem("Delete Owner");

m4=new MenuItem("View Owner");

owner.add(m1);

owner.add(m2);

owner.add(m3);

owner.add(m4);

mb.add(owner);

customer=new Menu("Customer");

m5=new MenuItem("Insert Customer");

m6=new MenuItem("Update Customer");

m7=new MenuItem("Delete Customer");

m8=new MenuItem("View Customer");

customer.add(m5);

customer.add(m6);

customer.add(m7);

customer.add(m8);

mb.add(customer);

pool=new Menu("Pool");

m9=new MenuItem("Insert Pool");

m10=new MenuItem("Update Pool");

m11=new MenuItem("Delete Pool");

m12=new MenuItem("View Pool");

pool.add(m9);

pool.add(m10);

pool.add(m11);

pool.add(m12);

mb.add(pool);

arranges=new Menu("Arranges");

m13=new MenuItem("Insert Arranges");

m14=new MenuItem("Delete Arranges");

m15=new MenuItem("View Arranges");

arranges.add(m13);

arranges.add(m14);

arranges.add(m15);

mb.add(arranges);

registered=new Menu("Registered");

m16=new MenuItem("Insert Registerd");

m17=new MenuItem("Update Registered");

m18=new MenuItem("Delete Registered");

m19=new MenuItem("View Registered");

registered.add(m16);

registered.add(m17);

registered.add(m18);

registered.add(m19);

mb.add(registered);

m1.addActionListener(this);

m2.addActionListener(this);

m3.addActionListener(this);

m4.addActionListener(this);

m5.addActionListener(this);

m6.addActionListener(this);

m7.addActionListener(this);

m8.addActionListener(this);

m9.addActionListener(this);

m10.addActionListener(this);

m11.addActionListener(this);

m12.addActionListener(this);

m13.addActionListener(this);

m14.addActionListener(this);

m15.addActionListener(this);

m16.addActionListener(this);

m17.addActionListener(this);

m18.addActionListener(this);

m19.addActionListener(this);

}

public void actionPerformed(ActionEvent ae)

{

String arg = ae.getActionCommand();

if(arg.equals("Insert Owner"))

this.buildGUIOwner();

if(arg.equals("Update Owner"))

this.updateGUIOwner();

if(arg.equals("Delete Owner"))

this.deleteGUIOwner();

if(arg.equals("View Owner"))

this.viewGUIOwner();

if(arg.equals("Insert Customer"))

this.buildGUICustomer();

if(arg.equals("Update Customer "))

this.updateGUICustomer();

if(arg.equals("Delete Customer"))

this.deleteGUICustomer();

if(arg.equals("View Customer"))

this.viewGUICustomer();

if(arg.equals("Insert Pool"))

this.buildGUIPool();

if(arg.equals("Update Pool"))

this.updateGUIPool();

if(arg.equals("Delete Pool"))

this.deleteGUIPool();

if(arg.equals("View Pool"))

this.viewGUIPool();

if(arg.equals("Insert Arranges"))

this.buildGUIArranges();

if(arg.equals("Delete Arranges"))

this.deleteGUIArranges();

if(arg.equals("View Arranges"))

this.viewGUIArranges();

if(arg.equals("Insert Registered"))

this.buildGUIRegistered();

if(arg.equals("Delete Registered"))

this.deleteGUIRegistered();

if(arg.equals("Update Registered"))

this.updateGUIRegistered();

if(arg.equals("View Registered"))

this.viewGUIRegistered();

}

public void buildGUIOwner()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO owner VALUES('"+ owner\_idText.getText() + "', " + "'" + nameText.getText() + "'," + "'"+ ageText.getText() + "'," + "'"+ mobile\_numberText.getText() + "',"+ "'" + genderText.getText() + "',"+ "'" + addressText.getText() + "'"+")";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

owner\_idText = new TextField(15);

nameText = new TextField(15);

ageText = new TextField(15);

mobile\_numberText= new TextField(15);

genderText = new TextField(15);

addressText = 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("Owner ID:"));

first.add(owner\_idText);

first.add(new Label("Owner Name:"));

first.add(nameText);

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

first.add(ageText);

first.add(new Label("Mobile\_Number"));

first.add(mobile\_numberText);

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

first.add(genderText);

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

first.add(addressText);

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

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

second.add(insertButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

add(first);

add(second);

add(third);setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUICustomer()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO customer VALUES('"+ customer\_idText.getText() + "', " + "'" + NameText.getText() + "'," + "'"+ AgeText.getText() + "'," + "'"+ Mobile\_NumberText.getText() + "',"+ "'" + GenderText.getText() + "',"+ "'" + AddressText.getText() + "'"+")";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_idText = new TextField(15);

NameText = new TextField(15);

AgeText = new TextField(15);

Mobile\_NumberText= new TextField(15);

GenderText = new TextField(15);

AddressText = 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("Customer ID:"));

first.add(customer\_idText);

first.add(new Label("Customer Name:"));

first.add(NameText);

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

first.add(AgeText);

first.add(new Label("Mobile\_Number"));

first.add(Mobile\_NumberText);

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

first.add(GenderText);

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

first.add(AddressText);

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

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

second.add(insertButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

add(first);

add(second);

add(third);setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUIPool()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO pool VALUES('"+ pool\_idText.getText() + "', " + "'" + start\_pText.getText() + "'," + "'"+ endText.getText() + "'," + "'"+ dayText.getText() + "',"+ "'" + timingText.getText() +"')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

pool\_idText = new TextField(15);

start\_pText = new TextField(15);

endText = new TextField(15);

dayText= new TextField(15);

timingText = 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("Pool ID:"));

first.add(pool\_idText);

first.add(new Label("Start Point:"));

first.add(start\_pText);

first.add(new Label("End Point:"));

first.add(endText);

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

first.add(dayText);

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

first.add(timingText);

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

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

second.add(insertButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

add(first);

add(second);

add(third);setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUIArranges()

{

removeAll();

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO arranges VALUES('"+ Owner\_idText.getText() + "', '"+Pool\_idText.getText() +"')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

Owner\_idText = new TextField(15);

Pool\_idText=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("Owner ID:"));

first.add(Owner\_idText);

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

first.add(Pool\_idText);

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

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

second.add(insertButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

add(first);

add(second);

add(third);setLayout(new FlowLayout());

setVisible(true);

}

public void buildGUIRegistered()

{

removeAll();

registeredList = new List(6);

loadRegistered();

add(registeredList);

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

registeredList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("customer\_id").equals(registeredList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Customer\_IdText.setText(rs.getString("customer\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Insert Account Button

insertButton = new Button("Submit");

insertButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

String query= "INSERT INTO registered VALUES('"+ Customer\_IdText.getText() + "', '"+Pool\_IdText.getText() +"', '"+locationText.getText() +"')";

int i = statement.executeUpdate(query);

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

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

Customer\_IdText = new TextField(15);

Customer\_IdText.setEditable(false);

Pool\_IdText=new TextField(15);

locationText=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("Customer ID:"));

first.add(Customer\_IdText);

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

first.add(Pool\_IdText);

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

first.add(locationText);

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

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

second.add(insertButton);

second.setBounds(125,220,150,100);

Panel third = new Panel();

third.add(errorText);

third.setBounds(125,320,300,200);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadMenu()

{

try

{

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

while (rs.next())

{

ownerList.add(rs.getString("owner\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUIOwner()

{

removeAll();

ownerList = new List(6);

loadMenu();

add(ownerList);

ownerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("owner\_id").equals(ownerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

owner\_idText.setText(rs.getString("owner\_id"));

nameText.setText(rs.getString("name"));

ageText.setText(rs.getString("age"));

mobile\_numberText.setText(rs.getString("mobile\_number"));

genderText.setText(rs.getString("gender"));

addressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

modify = new Button("Modify");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE owner "

+ "SET name=" +"'"+ nameText.getText() +"',"+"gender='"+genderText.getText()+"',"+"mobile\_number='"+mobile\_numberText.getText()+"',"+"age='"+ageText.getText()+"',"+"address='"+addressText.getText()+"'"

+ " WHERE owner\_id = '" + ownerList.getSelectedItem() + "'");

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

ownerList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

owner\_idText = new TextField(15);

owner\_idText.setEditable(false);

nameText = new TextField(15);

ageText = new TextField(15);

//ageText.setEditable(false);

mobile\_numberText = new TextField(15);

//mobile\_numberText.setEditable(false);

genderText = new TextField(15);

//genderText.setEditable(false);

addressText = new TextField(15);

//addressText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(owner\_idText);

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

first.add(nameText);

first.add(new Label("Age"));

first.add(ageText);

first.add(new Label("Mobile Number:"));

first.add(mobile\_numberText);

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

first.add(genderText);

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

first.add(addressText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIOwner()

{

removeAll();

ownerList = new List(10);

loadMenu();

add(ownerList);

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

ownerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("owner\_id").equals(ownerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

owner\_idText.setText(rs.getString("owner\_id"));

nameText.setText(rs.getString("name"));

ageText.setText(rs.getString("age"));

mobile\_numberText.setText(rs.getString("mobile\_number"));

genderText.setText(rs.getString("gender"));

addressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM owner WHERE owner\_id = '" + ownerList.getSelectedItem()+"'");

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

owner\_idText.setText(null);

nameText.setText(null);

ageText.setText(null);

mobile\_numberText.setText(null);

genderText.setText(null);

addressText.setText(null);

ownerList.removeAll();

loadMenu();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

owner\_idText = new TextField(15);

nameText = new TextField(15);

ageText = new TextField(15);

mobile\_numberText = new TextField(15);

genderText = new TextField(15);

addressText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

owner\_idText.setEditable(false);

nameText.setEditable(false);

ageText.setEditable(false);

mobile\_numberText.setEditable(false);

genderText.setEditable(false);

addressText.setEditable(false);

Panel first = new Panel();

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

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

first.add(owner\_idText);

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

first.add(nameText);

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

first.add(ageText);

first.add(new Label("Mobile Number:"));

first.add(mobile\_numberText);

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

first.add(genderText);

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

first.add(addressText);

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

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewGUIOwner()

{

removeAll();

ownerList = new List(6);

loadMenu();

add(ownerList);

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

ownerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("owner\_id").equals(ownerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

owner\_idText.setText(rs.getString("owner\_id"));

nameText.setText(rs.getString("name"));

ageText.setText(rs.getString("age"));

mobile\_numberText.setText(rs.getString("mobile\_number"));

genderText.setText(rs.getString("gender"));

addressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Update Owner");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE owner "

+ "SET name=" + nameText.getText()

+ " WHERE owner\_id = '" + ownerList.getSelectedItem() + "'");

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

ownerList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

owner\_idText = new TextField(15);

owner\_idText.setEditable(false);

nameText = new TextField(15);

nameText.setEditable(false);

ageText = new TextField(15);

ageText.setEditable(false);

mobile\_numberText = new TextField(15);

mobile\_numberText.setEditable(false);

genderText = new TextField(15);

genderText.setEditable(false);

addressText = new TextField(15);

addressText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(owner\_idText);

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

first.add(nameText);

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

first.add(ageText);

first.add(new Label("Mobile Number:"));

first.add(mobile\_numberText);

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

first.add(genderText);

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

first.add(addressText);

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

//second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadCustomer()

{

try

{

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

while (rs.next())

{

customerList.add(rs.getString("customer\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUICustomer()

{

removeAll();

customerList = new List(6);

loadCustomer();

add(customerList);

customerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("customer\_id").equals(customerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_idText.setText(rs.getString("customer\_id"));

NameText.setText(rs.getString("name"));

AgeText.setText(rs.getString("age"));

Mobile\_NumberText.setText(rs.getString("mobile\_number"));

GenderText.setText(rs.getString("gender"));

AddressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

modify = new Button("Modify");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE customer "

+ "SET name=" +"'"+ NameText.getText() +"',"+"gender='"+GenderText.getText()+"',"+"mobile\_number='"+Mobile\_NumberText.getText()+"',"+"age='"+AgeText.getText()+"',"+"address='"+AddressText.getText()+"'"

+ " WHERE customer\_id = '" + customerList.getSelectedItem() + "'");

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

customerList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_idText = new TextField(15);

customer\_idText.setEditable(false);

NameText = new TextField(15);

AgeText = new TextField(15);

//ageText.setEditable(false);

Mobile\_NumberText = new TextField(15);

//mobile\_numberText.setEditable(false);

GenderText = new TextField(15);

//genderText.setEditable(false);

AddressText = new TextField(15);

//addressText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(customer\_idText);

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

first.add(NameText);

first.add(new Label("Age"));

first.add(AgeText);

first.add(new Label("Mobile Number:"));

first.add(Mobile\_NumberText);

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

first.add(GenderText);

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

first.add(AddressText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUICustomer()

{

removeAll();

customerList = new List(10);

loadCustomer();

add(customerList);

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

customerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("customer\_id").equals(customerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_idText.setText(rs.getString("customer\_id"));

NameText.setText(rs.getString("name"));

AgeText.setText(rs.getString("age"));

Mobile\_NumberText.setText(rs.getString("mobile\_number"));

GenderText.setText(rs.getString("gender"));

AddressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM customer WHERE customer\_id = '" + customerList.getSelectedItem()+"'");

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

customer\_idText.setText(null);

NameText.setText(null);

AgeText.setText(null);

Mobile\_NumberText.setText(null);

GenderText.setText(null);

AddressText.setText(null);

customerList.removeAll();

loadCustomer();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

customer\_idText = new TextField(15);

NameText = new TextField(15);

AgeText = new TextField(15);

Mobile\_NumberText = new TextField(15);

GenderText = new TextField(15);

AddressText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

customer\_idText.setEditable(false);

NameText.setEditable(false);

AgeText.setEditable(false);

Mobile\_NumberText.setEditable(false);

GenderText.setEditable(false);

AddressText.setEditable(false);

Panel first = new Panel();

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

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

first.add(customer\_idText);

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

first.add(NameText);

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

first.add(AgeText);

first.add(new Label("Mobile Number:"));

first.add(Mobile\_NumberText);

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

first.add(GenderText);

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

first.add(AddressText);

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

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewGUICustomer()

{

removeAll();

customerList = new List(6);

loadCustomer();

add(customerList);

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

customerList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("customer\_id").equals(customerList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

customer\_idText.setText(rs.getString("customer\_id"));

NameText.setText(rs.getString("name"));

AgeText.setText(rs.getString("age"));

Mobile\_NumberText.setText(rs.getString("mobile\_number"));

GenderText.setText(rs.getString("gender"));

AddressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Update Customer");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE customer "

+ "SET name=" + nameText.getText()

+ " WHERE customer\_id = '" + customerList.getSelectedItem() + "'");

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

customerList.removeAll();

loadCustomer();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

customer\_idText = new TextField(15);

customer\_idText.setEditable(false);

NameText = new TextField(15);

NameText.setEditable(false);

AgeText = new TextField(15);

AgeText.setEditable(false);

Mobile\_NumberText = new TextField(15);

Mobile\_NumberText.setEditable(false);

GenderText = new TextField(15);

GenderText.setEditable(false);

AddressText = new TextField(15);

AddressText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(customer\_idText);

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

first.add(NameText);

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

first.add(AgeText);

first.add(new Label("Mobile Number:"));

first.add(Mobile\_NumberText);

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

first.add(GenderText);

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

first.add(AddressText);

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

//second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadPool()

{

try

{

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

while (rs.next())

{

poolList.add(rs.getString("pool\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUIPool()

{

removeAll();

poolList = new List(6);

loadPool();

add(poolList);

poolList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("pool\_id").equals(poolList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

pool\_idText.setText(rs.getString("pool\_id"));

start\_pText.setText(rs.getString("start\_p"));

endText.setText(rs.getString("end"));

dayText.setText(rs.getString("day"));

timingText.setText(rs.getString("timing"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

modify = new Button("Modify");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE pool "

+ "SET start\_p=" +"'"+ start\_pText.getText() +"',"+"end='"+endText.getText()+"',"+"day='"+dayText.getText()+"',"+"timing='"+timingText.getText()+"'"

+ " WHERE pool\_id = '" + poolList.getSelectedItem() + "'");

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

poolList.removeAll();

loadPool();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

pool\_idText = new TextField(15);

pool\_idText.setEditable(false);

start\_pText = new TextField(15);

endText = new TextField(15);

//ageText.setEditable(false);

dayText = new TextField(15);

//mobile\_numberText.setEditable(false);

timingText = new TextField(15);

//genderText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(pool\_idText);

first.add(new Label("Start Point:"));

first.add(start\_pText);

first.add(new Label("End Point"));

first.add(endText);

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

first.add(dayText);

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

first.add(timingText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIPool()

{

removeAll();

poolList = new List(10);

loadPool();

add(poolList);

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

poolList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("pool\_id").equals(poolList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

pool\_idText.setText(rs.getString("pool\_id"));

start\_pText.setText(rs.getString("start\_p"));

endText.setText(rs.getString("end"));

dayText.setText(rs.getString("day"));

timingText.setText(rs.getString("timing"));

//addressText.setText(rs.getString("address"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM pool WHERE pool\_id = '" + poolList.getSelectedItem()+"'");

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

pool\_idText.setText(null);

start\_pText.setText(null);

endText.setText(null);

dayText.setText(null);

timingText.setText(null);

poolList.removeAll();

loadPool();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

pool\_idText = new TextField(15);

start\_pText = new TextField(15);

endText = new TextField(15);

dayText = new TextField(15);

timingText = new TextField(15);

//addressText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

pool\_idText.setEditable(false);

start\_pText.setEditable(false);

endText.setEditable(false);

dayText.setEditable(false);

timingText.setEditable(false);

//addressText.setEditable(false);

Panel first = new Panel();

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

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

first.add(pool\_idText);

first.add(new Label("Starting Point:"));

first.add(start\_pText);

first.add(new Label("End Point:"));

first.add(endText);

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

first.add(dayText);

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

first.add(timingText);

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

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewGUIPool()

{

removeAll();

poolList = new List(6);

loadPool();

add(poolList);

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

poolList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("pool\_id").equals(poolList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

pool\_idText.setText(rs.getString("pool\_id"));

start\_pText.setText(rs.getString("start\_p"));

endText.setText(rs.getString("end"));

dayText.setText(rs.getString("day"));

timingText.setText(rs.getString("timing"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Update Pool");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE pool "

+ "SET start\_p=" + start\_pText.getText()

+ " WHERE pool\_id = '" + poolList.getSelectedItem() + "'");

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

poolList.removeAll();

loadPool();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

pool\_idText = new TextField(15);

pool\_idText.setEditable(false);

start\_pText = new TextField(15);

start\_pText.setEditable(false);

endText = new TextField(15);

endText.setEditable(false);

dayText = new TextField(15);

dayText.setEditable(false);

timingText = new TextField(15);

timingText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(pool\_idText);

first.add(new Label("Starting Point:"));

first.add(start\_pText);

first.add(new Label("End Point:"));

first.add(endText);

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

first.add(dayText);

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

first.add(timingText);

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

//second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadArranges()

{

try

{

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

while (rs.next())

{

arrangesList.add(rs.getString("Owner\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void deleteGUIArranges()

{

removeAll();

arrangesList = new List(10);

loadArranges();

add(arrangesList);

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

arrangesList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("Owner\_id").equals(arrangesList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Owner\_idText.setText(rs.getString("Owner\_id"));

Pool\_idText.setText(rs.getString("Pool\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM arranges WHERE Owner\_id = '" + arrangesList.getSelectedItem()+"'");

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

Owner\_idText.setText(null);

Pool\_idText.setText(null);

arrangesList.removeAll();

loadArranges();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

Owner\_idText = new TextField(15);

Pool\_idText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Owner\_idText.setEditable(false);

Pool\_idText.setEditable(false);

Panel first = new Panel();

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

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

first.add(Owner\_idText);

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

first.add(Pool\_idText);

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

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewGUIArranges()

{

removeAll();

arrangesList = new List(6);

loadArranges();

add(arrangesList);

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

arrangesList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("Owner\_id").equals(arrangesList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Owner\_idText.setText(rs.getString("Owner\_id"));

Pool\_idText.setText(rs.getString("pool\_id"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

/\*modify = new Button("Update Arranges");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE arranges "

+ "SET name=" + nameText.getText()

+ " WHERE owner\_id = '" + ownerList.getSelectedItem() + "'");

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

ownerList.removeAll();

loadMenu();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});\*/

Owner\_idText = new TextField(15);

Owner\_idText.setEditable(false);

Pool\_idText = new TextField(15);

Pool\_idText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(Owner\_idText);

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

first.add(Pool\_idText);

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

//second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

private void loadRegistered()

{

try

{

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

while (rs.next())

{

registeredList.add(rs.getString("customer\_id"));

}

}

catch (SQLException e)

{

displaySQLErrors(e);

}

}

public void updateGUIRegistered()

{

removeAll();

registeredList = new List(6);

loadRegistered();

add(registeredList);

registeredList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("Customer\_Id").equals(registeredList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Customer\_IdText.setText(rs.getString("Customer\_Id"));

Pool\_IdText.setText(rs.getString("Pool\_Id"));

locationText.setText(rs.getString("location"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

modify = new Button("Modify");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE registered "

+ "SET location=" +"'"+ locationText.getText() +"'"

+ " WHERE Customer\_Id = '" + registeredList.getSelectedItem() + "'");

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

registeredList.removeAll();

loadRegistered();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

Customer\_IdText = new TextField(15);

Customer\_IdText.setEditable(false);

Pool\_IdText = new TextField(15);

Pool\_IdText.setEditable(false);

locationText = 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("Customer ID:"));

first.add(Customer\_IdText);

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

first.add(Pool\_IdText);

first.add(new Label("Location"));

first.add(locationText);

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

second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public void deleteGUIRegistered()

{

removeAll();

registeredList = new List(10);

loadRegistered();

add(registeredList);

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

registeredList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("Customer\_Id").equals(registeredList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Customer\_IdText.setText(rs.getString("customer\_id"));

Pool\_IdText.setText(rs.getString("Pool\_id"));

locationText.setText(rs.getString("location"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

deleteRowButton = new Button("Delete Row");

deleteRowButton.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("DELETE FROM registered WHERE Customer\_id = '" + registeredList.getSelectedItem()+"'");

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

Customer\_IdText.setText(null);

Pool\_IdText.setText(null);

locationText.setText(null);

registeredList.removeAll();

loadRegistered();

}

catch (SQLException deleteException)

{

displaySQLErrors(deleteException);

}

}

});

Customer\_IdText = new TextField(15);

Pool\_IdText = new TextField(15);

locationText = new TextField(15);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Customer\_IdText.setEditable(false);

Pool\_IdText.setEditable(false);

locationText.setEditable(false);

Panel first = new Panel();

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

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

first.add(Customer\_IdText);

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

first.add(Pool\_IdText);

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

first.add(locationText);

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

second.add(deleteRowButton);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

setLayout(new FlowLayout());

setVisible(true);

}

public void viewGUIRegistered()

{

removeAll();

registeredList = new List(6);

loadRegistered();

add(registeredList);

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

registeredList.addItemListener(new ItemListener()

{

public void itemStateChanged(ItemEvent e)

{

try

{

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

while (rs.next())

{

if (rs.getString("customer\_id").equals(registeredList.getSelectedItem()))

break;

}

if (!rs.isAfterLast())

{

Customer\_IdText.setText(rs.getString("Customer\_id"));

Pool\_IdText.setText(rs.getString("pool\_id"));

locationText.setText(rs.getString("location"));

}

}

catch (SQLException selectException)

{

displaySQLErrors(selectException);

}

}

});

//Handle Update Menu Button

modify = new Button("Update Registered");

modify.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

try

{

Statement statement = connection.createStatement();

int i = statement.executeUpdate("UPDATE registered "

+ "SET location=" + locationText.getText()

+ " WHERE Customer\_Id = '" + registeredList.getSelectedItem() + "'");

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

registeredList.removeAll();

loadRegistered();

}

catch (SQLException insertException)

{

displaySQLErrors(insertException);

}

}

});

Customer\_IdText = new TextField(15);

Customer\_IdText.setEditable(false);

Pool\_IdText = new TextField(15);

Pool\_IdText.setEditable(false);

locationText = new TextField(15);

locationText.setEditable(false);

errorText = new TextArea(10, 40);

errorText.setEditable(false);

Panel first = new Panel();

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

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

first.add(Customer\_IdText);

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

first.add(Pool\_IdText);

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

first.add(locationText);

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

//second.add(modify);

Panel third = new Panel();

third.add(errorText);

add(first);

add(second);

add(third);

//setTitle("Update ....");

//setSize(500, 600);

setLayout(new FlowLayout());

setVisible(true);

}

public 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)

{

InsertTables it = new InsertTables();

it.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent e)

{

System.exit(0);

}

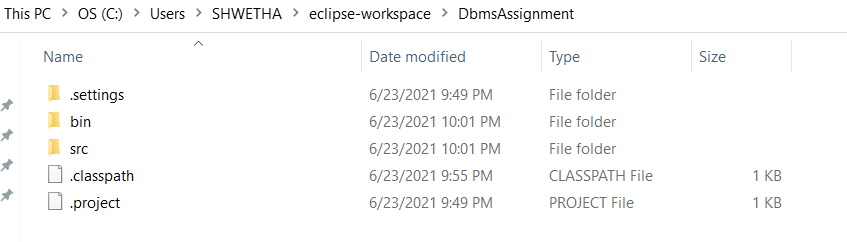
});

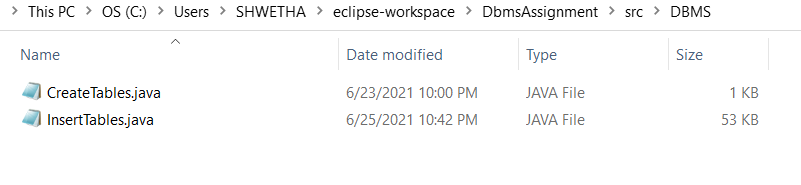
it.buildFrame();

}

}

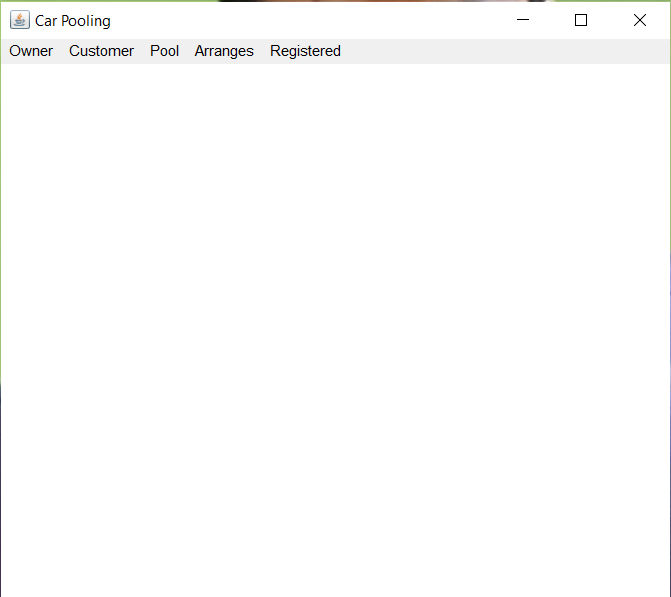
FOLDER STRUCTURE:



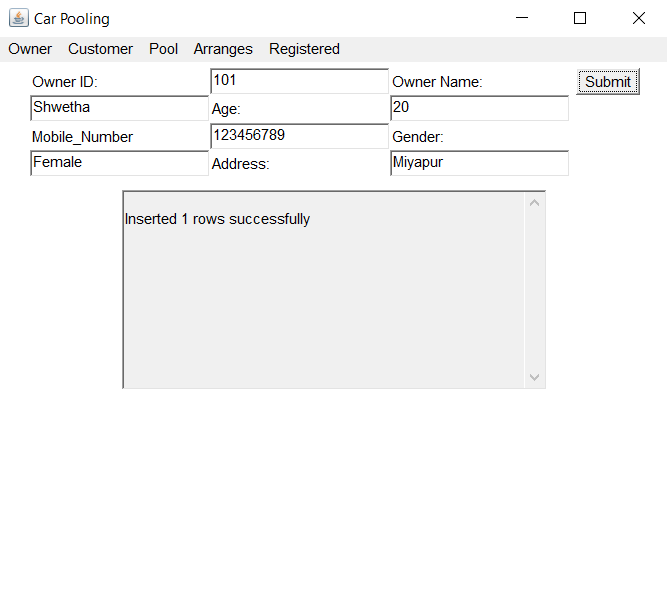


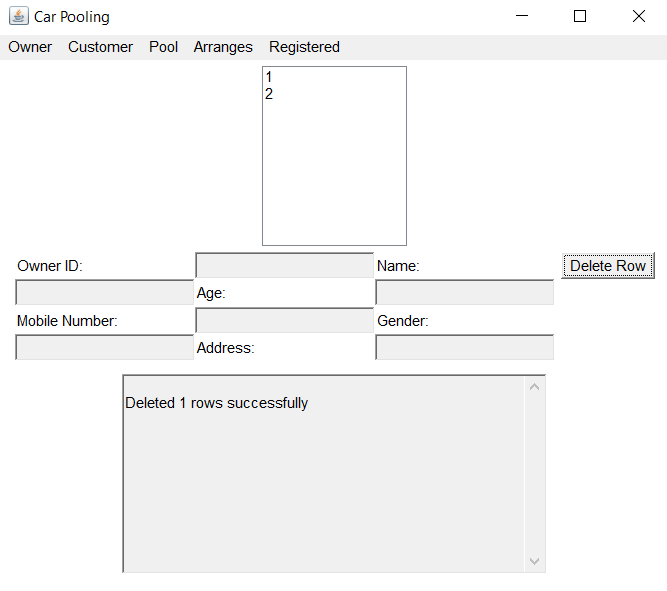
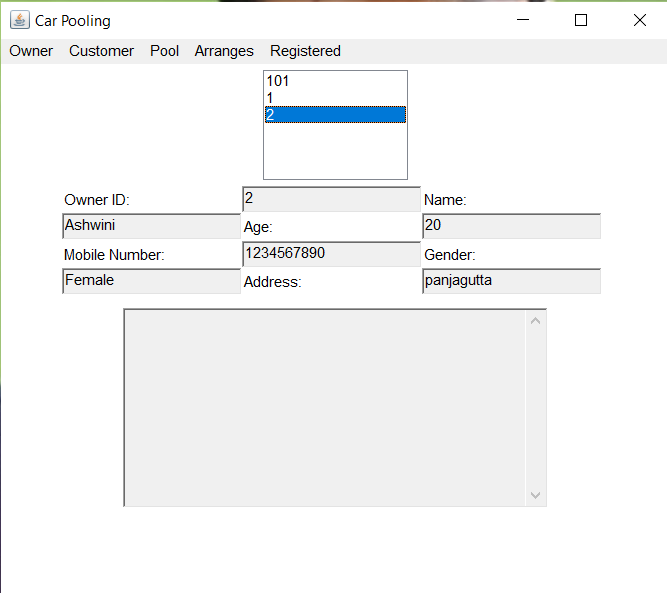
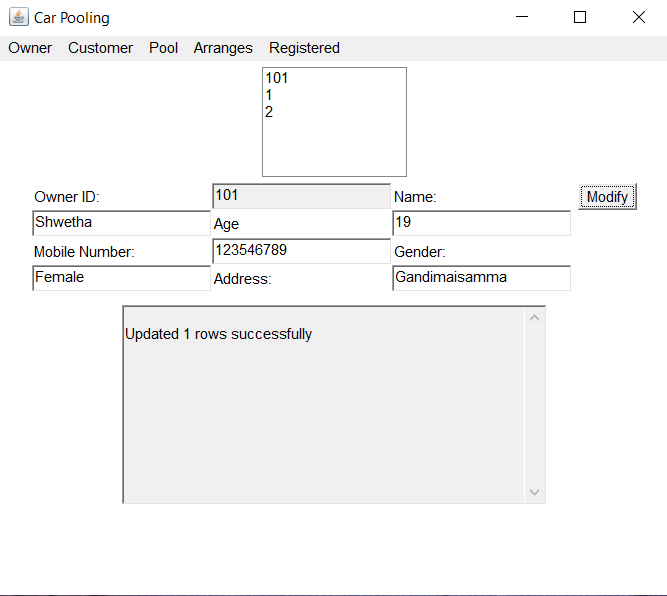
**TESTING**

**OUTPUTS:**

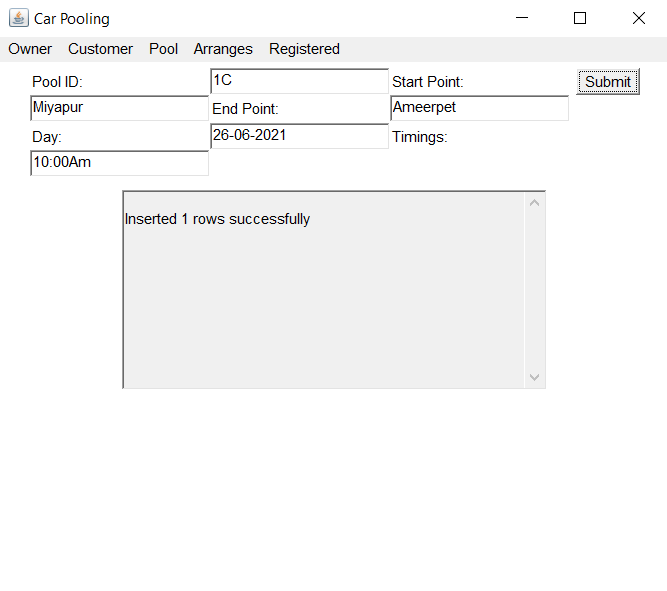
****

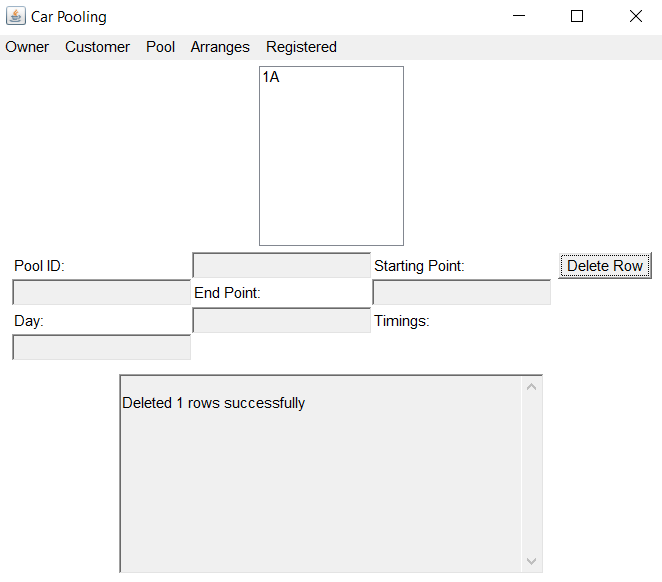
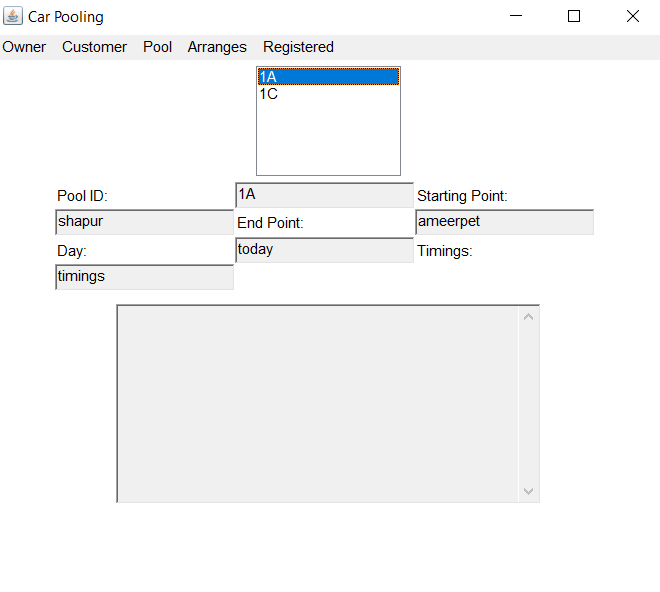
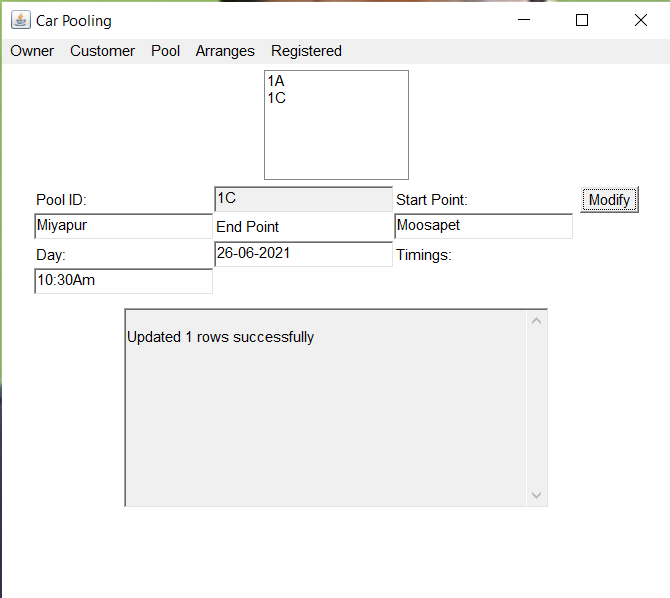
***OWNER TABLE:***

******

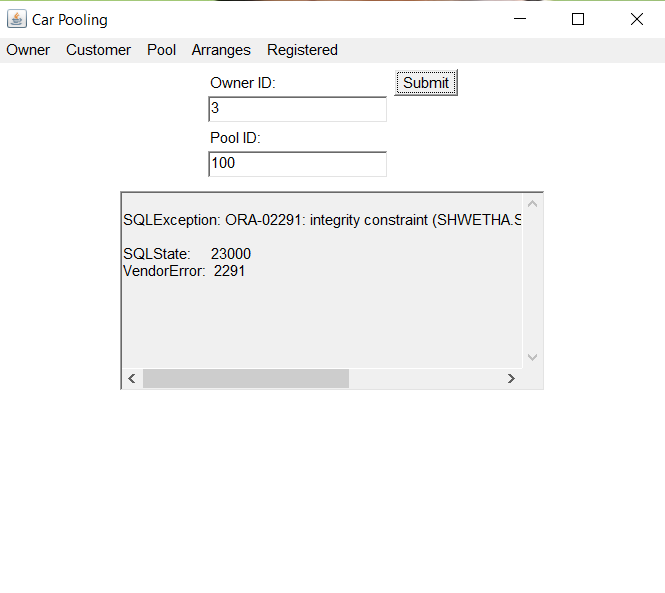
******

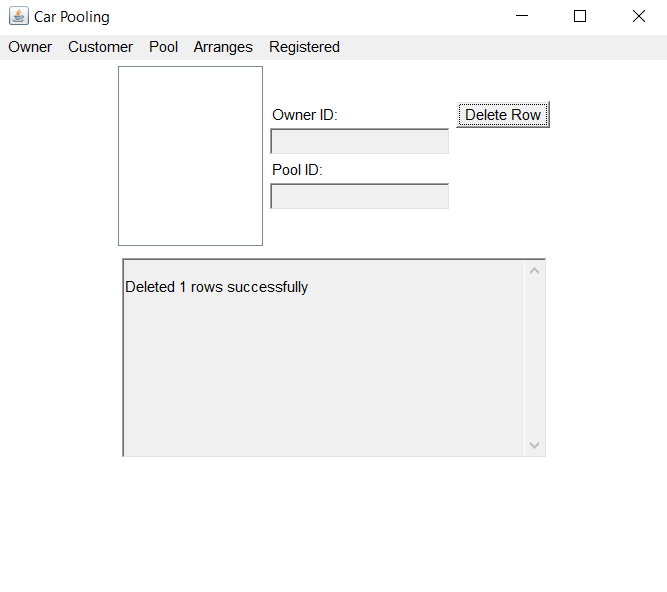
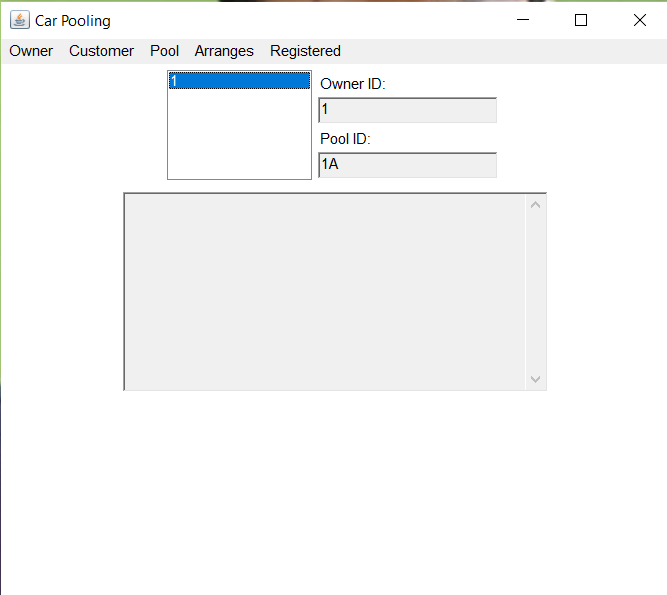
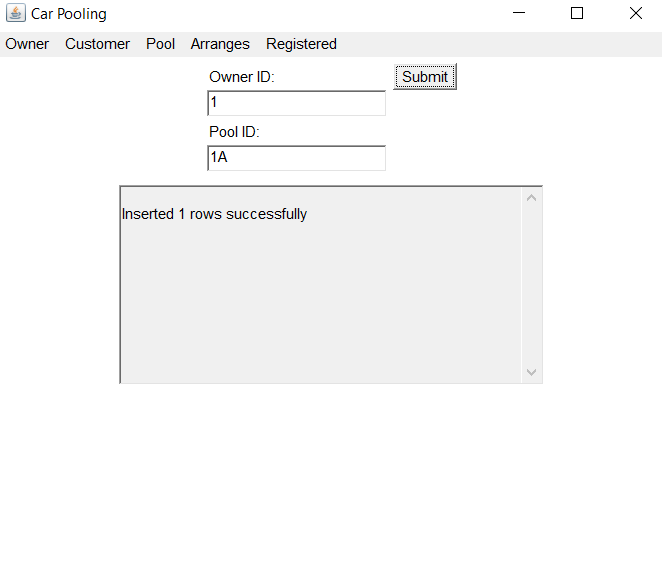
***POOL:***

******

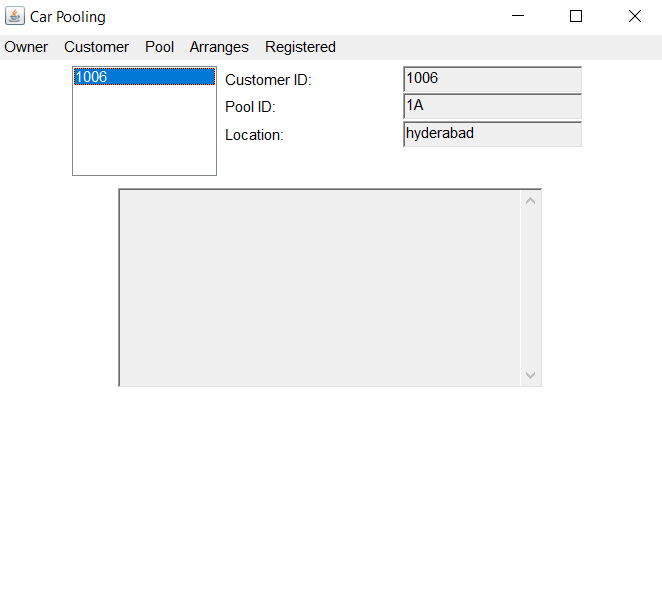


***ARRANGES:***

******



***REGISTERED TABLE:***

******

