AUTOMATED DANGER ZONE LIVE DETECTION DURING COMMUTE

A

Report

Submitted in partial fulfilment of the

BE IV SEMESTER DATABASE MANAGEMENT SYSTEMS LAB

INFORMATION TECHNOLOGY

By

B. SWETHA <1602-18-737-109>

Under the Guidance of

B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2019-2020

BONAFIDE CERTIFICATE

This to certify that the project report titled "**Automated Danger Zone Live Detection During Commute**" project work of Ms. **B.Swetha** bearing Roll.no:**1602-18-737-109** who carried out this project under my supervision in the IV semester for the academic year 2019-2020.

Signature of the examiner
B.LEELAVATHY
Associate Professor
Department of Information Technology

ABSTRACT:

This project is based on danger zone live detection during commute. A large number of precious lives are lost due to road traffic accidents. There is a need to have an effective road accident detection and information communication system in place to save injured persons. A system that sends information messages to nearby emergency services about the accident location for timely response is absolutely in need. In research literature,a number of automatic accident detection systems are proposed by numerous researches. These include accident detection using smart phones,GSM and GPS technologies,vehicular networks. The implementation of an automatic road accident detection and information communication system in every vehicle is very crucial.

AIM:

To create a Java GUI based Automated danger Zone live list which takes the values like:vehicle_no,route_id,duration,area,routes by the users. These values are to updated in the database using JDBC connectivity.

INTRODUCTION:

List of tables:

1.Users
2.COMMUTE
3.SUGGESTED ROUTES
4.POSTING DETAILS
5.ADMIN

LIST OF ATTRIBUTES THEIR DOMAIN TYPES:

1.Users:

Users name char(10)
Users vehicle_no varchar2(20)
Users email_id varchar2(20)

2.Suggested Routes:

Route_id number(10)
Area char(10)
Safe route char(10)
Danger route char(10)

3.Admin:

Admin id number(10)
Admin name char(10)
Admin age number(5)
Route_id number(10)

4. Commute:

Users Vehicle_no varchar2(20)
Route_id number(10)

Area char(10)
Duration varchar2(20)

5. Posting Details:

Route_id number(10)
Users vehicle_no varchar2(20)
Area char(10)

ARCHITECTURE AND TECHNOLOGY :

SOFTWARE USED:

Java Eclipse, Oracle 11g Database, Java SE version 7, SQL*Plus.

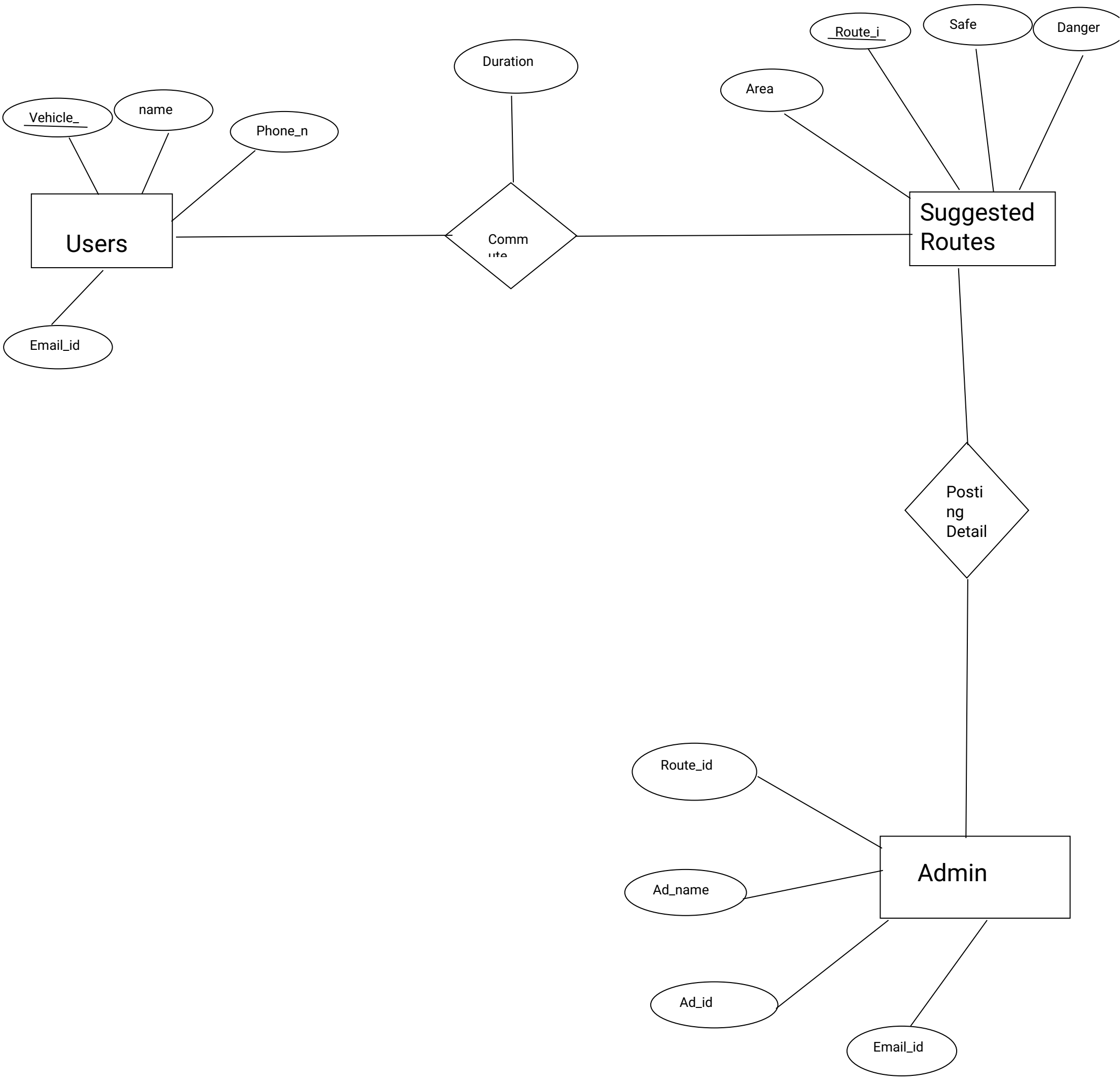
JAVA AWT:

- 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 operating system. AWT is heavyweight i.e. its components are using the resources of OS.
- The java.awt package provides classes for AWT API such as Text Field, Label, Text Area,Radio Button, Check Box, Choice, List 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.

E-R DIAGRAM:


Route_i
Safe
Danger
Duration
Area
Vehicle_
name
Phone_n
Users
Comm
ute
Suggested
Routes
Email_id
Posti
ng
Detail
Route_id
Admin
Ad_name
Ad_id
Email_id

DDL COMMANDS:

- Create table users(uname char(10), uvehicle_no varchar2(20), uphone_no number(10),uemail_id varchar2(20), primarykey(uvehicle_no));

- Create table commute(uvehicle_no varchar2(20),route_id number(5),area char(10), foreign key(vehicle_no) references users ,foreign key(route_id)references suggested routes, primary key(uvehicle_no,route_id));

- Create table suggested_routes(route_id number(5), Safe route char(5),danger route char(5),areas char(5),primary key(route_id));

- Create table posting_details(route_id number(5),uvehicle_no varchar2(20),area char(5), foreign key (uvehicle_no) references users , foreign key(route_id)references suggested routes, primary key(vehicle,route_id));

- Create table admin(ad_name char(5), adage number(5),ad_id number(10),primary key(route_id)); Table created.

```
SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 UNAME                                              CHAR(5)
 UVEHICLE_NO                               NOT NULL VARCHAR2(20)
 UPHONE_NO                                          NUMBER(20)
 UEMAIL_ID                                          VARCHAR2(20)
```

```
SQL> desc commute;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
 UVEHICLE_NO                                        VARCHAR2(20)
 ROUTE_ID                                           NUMBER(5)
 AREA                                               CHAR(20)
 DURATION                                           VARCHAR2(20)
```

```
SQL> desc suggested_routes;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
 SAFE_ROUTE                                         VARCHAR2(20)
 DANGER_ROUTE                                       VARCHAR2(20)
 AREA                                               VARCHAR2(20)
 ROUTE_ID                                  NOT NULL NUMBER(5)
```

```
SQL> desc posting_details;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ROUTE_ID                                  NOT NULL NUMBER(5)
 UVEHICLE_NO                               NOT NULL VARCHAR2(20)
 AREA                                               VARCHAR2(20)

SQL>
```

```
SQL> desc admin;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ADNAME                                             CHAR(5)
 ADAGE                                              NUMBER(5)
 AD_ID                                              VARCHAR2(20)
 ROUTE_ID                                  NOT NULL NUMBER(5)
```

```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
ADMIN                          TABLE
COMMUTE                        TABLE
POSTING_DETAILS                TABLE
SUGGESTED_ROUTES               TABLE
USERS                          TABLE

SQL> _
```

```
UNAME UVEHICLE_NO            UPHONE_NO UEMAIL_ID
----- -------------------- ---------- ------------------
andy  ts pa01 9987         8738392256 ramesh@gmail.com
vijay ts wq98 7844         9456779987 vijay@gmail.com
ashu  ap gd02 1234         9888279083 ashu@gmail.com
jessy ap cx05 7777         9927369917 jessy@gmail.com
riya  ts wa32 6658         9976545331 riya@gmail.com
mihir ap tr54              8798766554 mihir@gmail.com
amit  ap yu76 1111         9977554437 amit@gmail.com
andy  ts wq29              9987302176 andy@gmail.com
riya  ts wq                  99752102 riya@gmail.com

9 rows selected.
```

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:

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

Table Created in SQL for above mentioned purpose is as:

Create table commute(uvehicle_no varchar2(20),route_id number(5),area char(10), foreign key(vehicle_no) references users ,foreign key(route_id)references suggested routes, primary key(uvehicle_no,route_id));

PROGRAM:

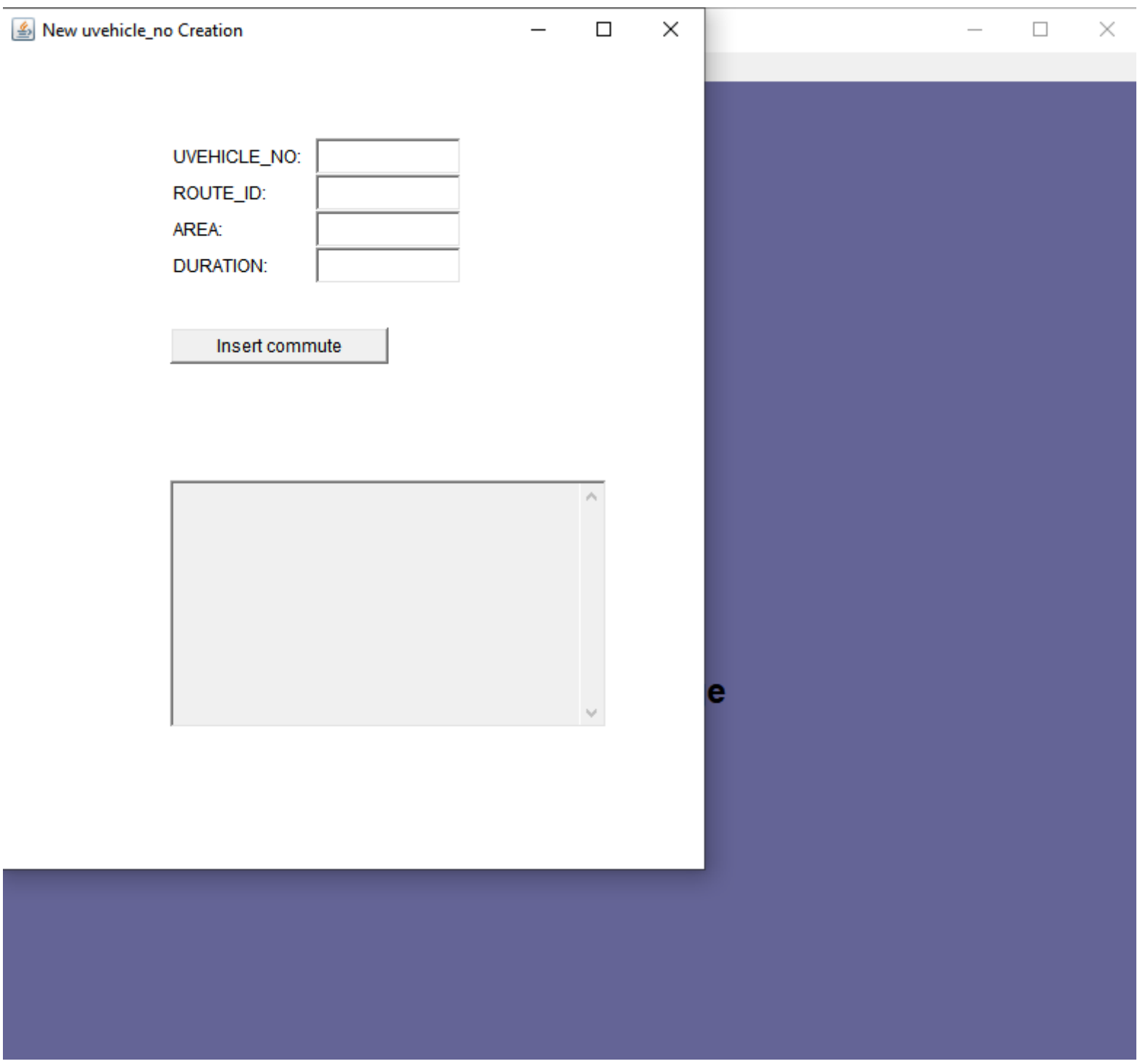
```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class InsertCommute extends Frame
{
    Button InsertCommuteButton;
```

```
TextField uvehicle_noText,route_idText, areaText, durationText;
TextArea errorText;
Connection connection;
Statement statement;
public InsertCommute()
{
    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","swetha","manager");
            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()
    {
            InsertCommuteButton = new Button("Insert commute");
        InsertCommuteButton.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                try
                {

                    String query= "INSERT INTO commute VALUES(" + uvehicle_noText.getText() +
", " + "'" + route_idText.getText() + "'," + areaText.getText() + "," + durationText.getText() + ")";
                    int i = statement.executeUpdate(query);
                    errorText.append("\nInserted " + i + " rows successfully");
                }
                catch (SQLException insertException)
                {
                    displaySQLErrors(insertException);
```

```
            }
        }
    });


    uvehicle_noText = new TextField(15);
    route_idText = new TextField(15);
    areaText = new TextField(15);
    durationText = 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("UVEHICLE_NO:"));
    first.add(uvehicle_noText);
    first.add(new Label("ROUTE_ID:"));
    first.add(route_idText);
    first.add(new Label("AREA:"));
    first.add(areaText);
    first.add(new Label("DURATION:"));
    first.add(durationText);
    first.setBounds(125,90,200,100);

    Panel second = new Panel(new GridLayout(4, 1));
    second.add(InsertCommuteButton);
        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("New uvehicle_no Creation");
        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)
    {
        InsertCommute icomm = new InsertCommute();

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

        icomm.buildGUI();
    }
}
```

OUTPUT:



PROGRAM:

DELETE COMMUTE:

```
 import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class DeleteCommute extends Frame
{
    Button DeleteCommuteButton;
    List commuteList;
    TextField uvehicle_noText, route_idText, areaText, durationText;
    TextArea errorText;
    Connection connection;
    Statement statement;
    ResultSet rs;

    public DeleteCommute()
    {
        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","
swetha","manager");
            statement = connection.createStatement();

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

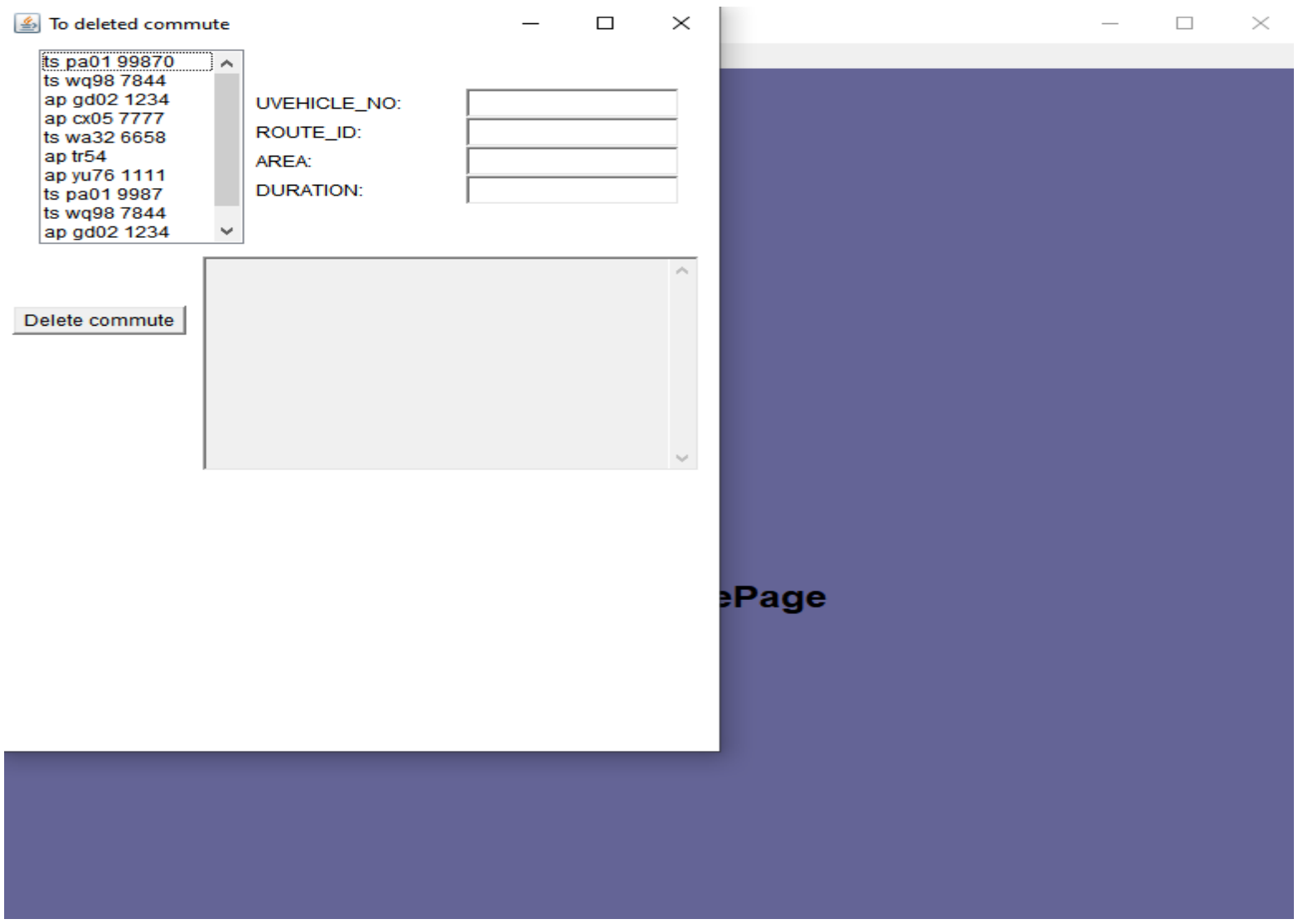
```
        }
       }

    private void loadcommute()
    {
        try
        {
            rs = statement.executeQuery("SELECT * FROM commute");
            while (rs.next())
            {
             commuteList.add(rs.getString("UVEHICLE_NO"));
            }
        }
        catch (SQLException e)
        {
            displaySQLErrors(e);
        }
    }

    public void buildGUI()
    {
           commuteList = new List(10);
        loadcommute();
        add(commuteList);

        //When a list item is selected populate the text fields
        commuteList.addItemListener(new ItemListener()
        {
            public void itemStateChanged(ItemEvent e)
            {
                try
                {
                    rs = statement.executeQuery("SELECT * FROM commute");
                    while (rs.next())
                    {
                        if
(rs.getString("uvehicle_no").equals(commuteList.getSelectedItem()))
                        break;
                    }
                    if (!rs.isAfterLast())
                    {
                        uvehicle_noText.setText(rs.getString("UVEHICLE_NO"));
                        route_idText.setText(rs.getString("Route_id"));
                        areaText.setText(rs.getString("AREA"));
                        durationText.setText(rs.getString("DURATION"));
                    }
                }
```

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


    DeleteCommuteButton = new Button("Delete commute");
    DeleteCommuteButton.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            try
            {
                Statement statement = connection.createStatement();
                int i = statement.executeUpdate("DELETE FROM commute WHERE
uvehicle_no = "
                        + commuteList.getSelectedItem());
                errorText.append("\nDeleted " + i + " rows successfully");
                uvehicle_noText.setText(null);
                route_idText.setText(null);
                areaText.setText(null);
                durationText.setText(null);
                commuteList.removeAll();
                loadcommute();
            }
            catch (SQLException insertException)
            {
                displaySQLErrors(insertException);
            }
        }
    });

    uvehicle_noText = new TextField(15);
    route_idText = new TextField(15);
    areaText = new TextField(15);
    durationText = 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("UVEHICLE_NO:"));
    first.add(uvehicle_noText);
    first.add(new Label("Route_id:"));
```

```
        first.add(route_idText);
        first.add(new Label("Area:"));
        first.add(areaText);
        first.add(new Label("Duration:"));
        first.add(durationText);

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

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

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

        setTitle("To deleted commute");
        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)
    {
        DeleteCommute delc= new DeleteCommute();

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

        delc.buildGUI();
    }
```

}

OUTPUT:



UPDATE COMMUTE:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
public class UpdateCommute extends Frame
{
    Button UpdateCommuteButton;
    List commuteList;
    TextField uvehicle_noText, route_idText, areaText, durationText;
    TextArea errorText;
    Connection connection;
    Statement statement;
    ResultSet rs;

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

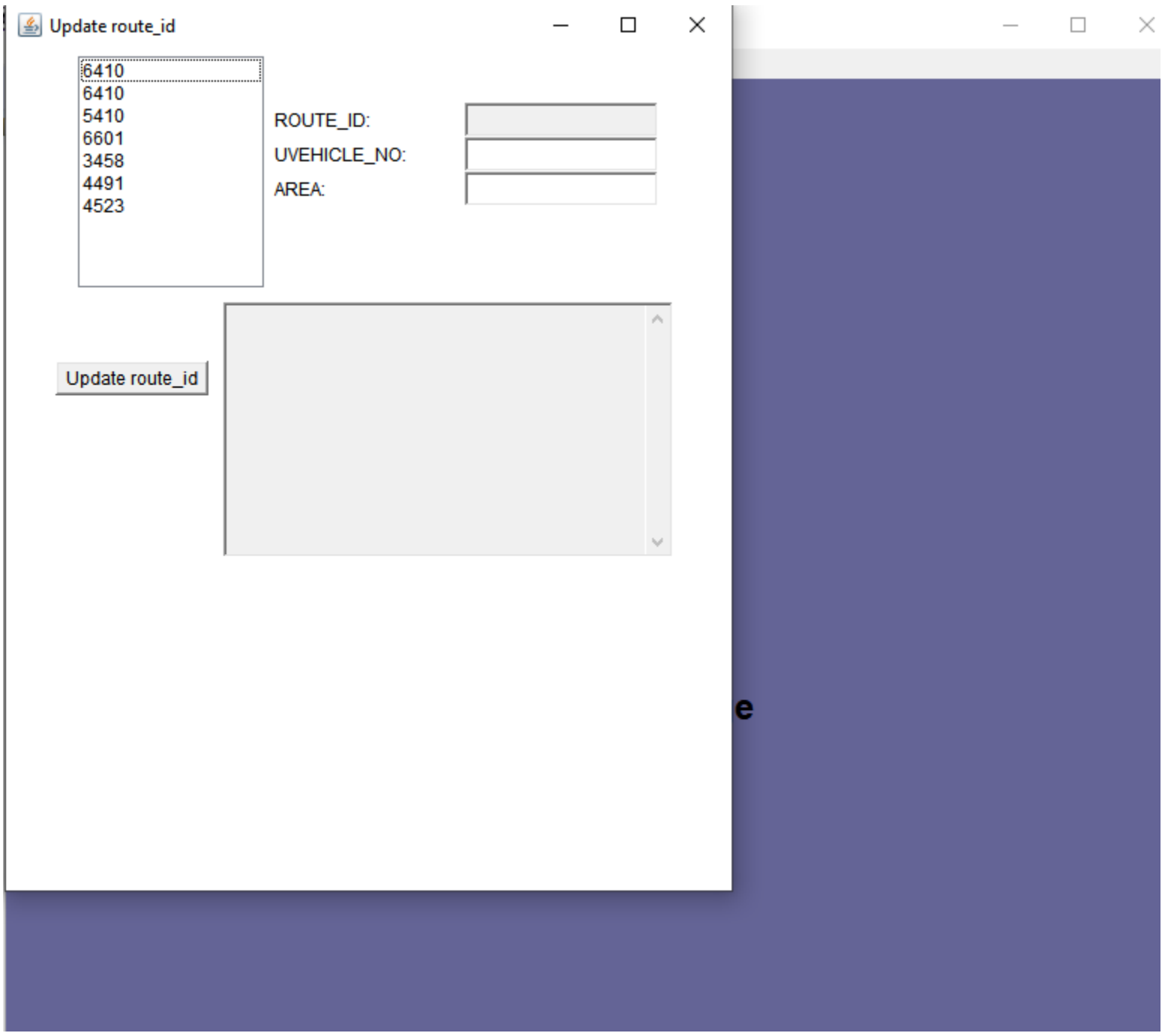
```
        {
            System.err.println("Unable to find and load driver");
            System.exit(1);
        }
        connectToDB();
    }

    public void connectToDB()
        {
        try
        {
            connection =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","swet
ha","manager");
            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 loadcommute()
    {
        try
        {
            rs = statement.executeQuery("SELECT route_id FROM commute");
            while (rs.next())
            {
            commuteList.add(rs.getString("ROUTE_ID"));
            }
        }
        catch (SQLException e)
        {
            displaySQLErrors(e);
        }
    }
```

```
public void buildGUI()
{
      commuteList = new List(10);
   loadcommute();
   add(commuteList);

        commuteList.addItemListener(new ItemListener()
   {
      public void itemStateChanged(ItemEvent e)
      {
         try
         {
            rs = statement.executeQuery("SELECT * FROM commute
where route_id ="+commuteList.getSelectedItem());
               rs.next();
               uvehicle_noText.setText(rs.getString("UVEHICLE_NO"));
               route_idText.setText(rs.getString("ROUTE_ID"));
               areaText.setText(rs.getString("AREA"));
               durationText.setText(rs.getString("DURATION"));

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

   UpdateCommuteButton = new Button("Update uvehicle_no");
UpdateCommuteButton.addActionListener(new ActionListener()
{
   public void actionPerformed(ActionEvent e)
   {
      try
      {
         Statement statement = connection.createStatement();
         int i = statement.executeUpdate("UPDATE commute "
         + "SET uvehicle_no='" + uvehicle_noText.getText() + "', "
         + "duration=" + durationText.getText() + ", "
         + "area ="+ areaText.getText() + " WHERE route_no = "
         + commuteList.getSelectedItem());
```

```
                errorText.append("\nUpdated " + i + " rows successfully");
                commuteList.removeAll();
                loadcommute();
            }
            catch (SQLException insertException)
            {
                displaySQLErrors(insertException);
            }
        }
    });

    route_idText = new TextField(15);
    route_idText.setEditable(false);
    uvehicle_noText = new TextField(15);
    areaText = new TextField(15);
    durationText = 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("UVEHICLE_ID:"));
    first.add(uvehicle_noText);
    first.add(new Label("ROUTE_ID:"));
    first.add(route_idText);
    first.add(new Label("AREA:"));
    first.add(areaText);
    first.add(new Label("DURATION:"));
    first.add(durationText);

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

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

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

    setTitle("Update route_id");
```

```
        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)
    {
        UpdateCommute ucomm = new UpdateCommute();

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

        ucomm.buildGUI();
    }
}
```

OUTPUT:



DML COMMANDS:

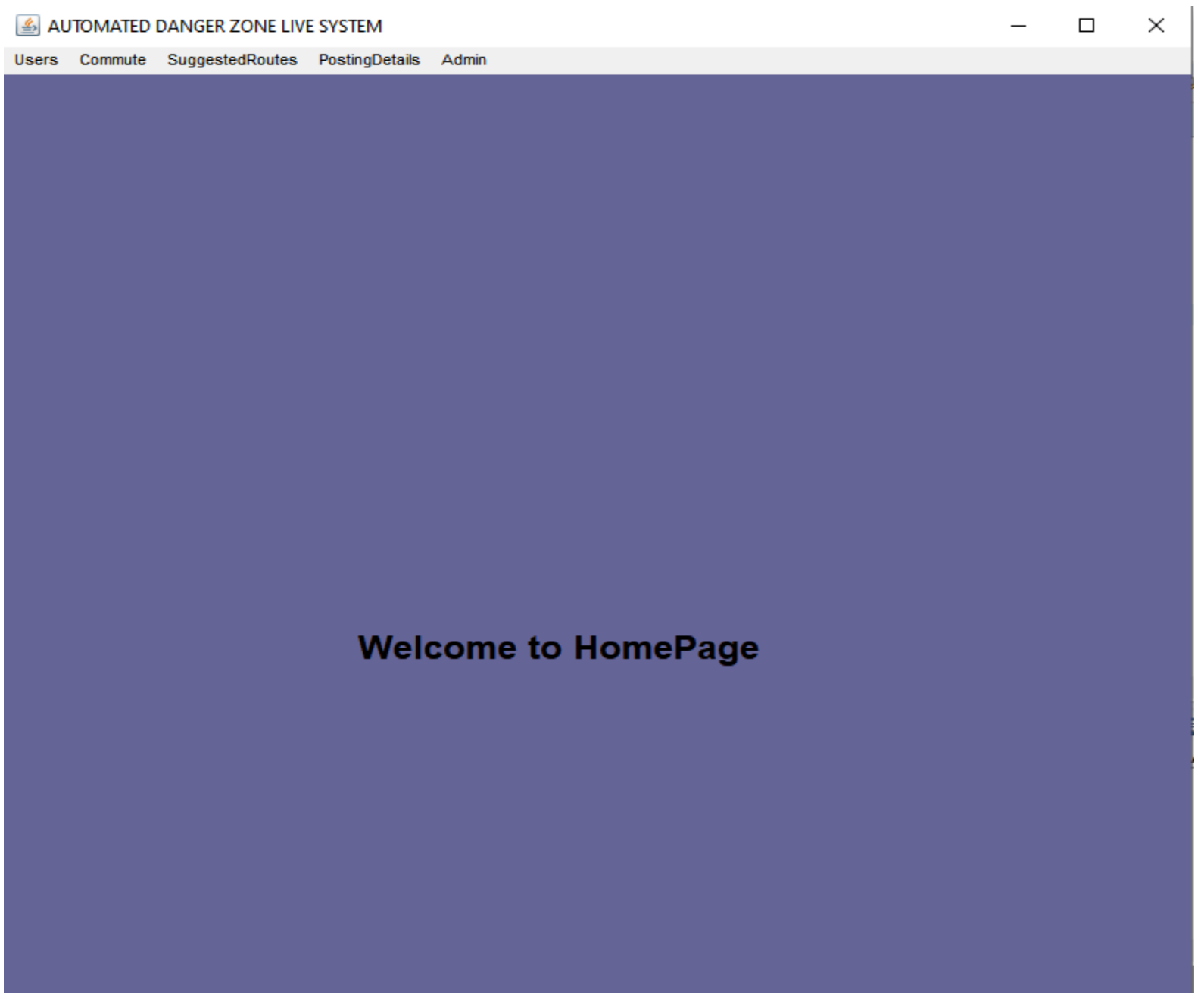
- Insert into users values(‘ &uname’, ‘&uvehicle_ no’, ‘ &uphone_ no’, ‘&uemail_id’)
- Insert into posting_details values(‘&route_id’, ‘&uvehicle_no’, ‘&area’)
- Insert into commute values(‘&uvehicle_no’, ‘&route_id’, ‘&area', ‘&duration’)
- Insert into suggested_routes values(‘&route_id’, ‘&safe route’, ‘&danger route’, ‘&area’)

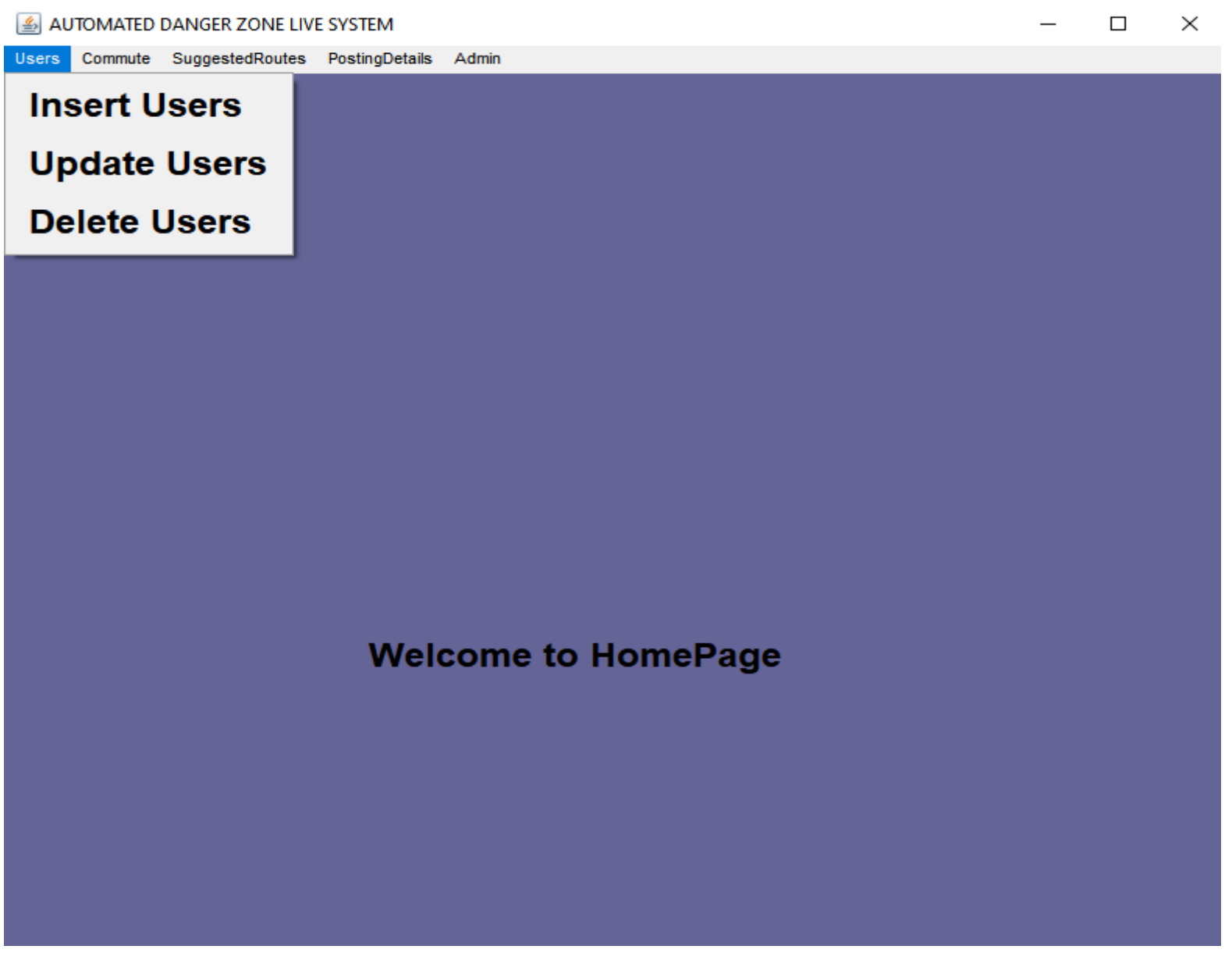
- Insert into admin values(‘&adname’, ‘&adage’, ‘&ad_ id’, ‘ &rout e_id’) ;

GITHUB LINK:

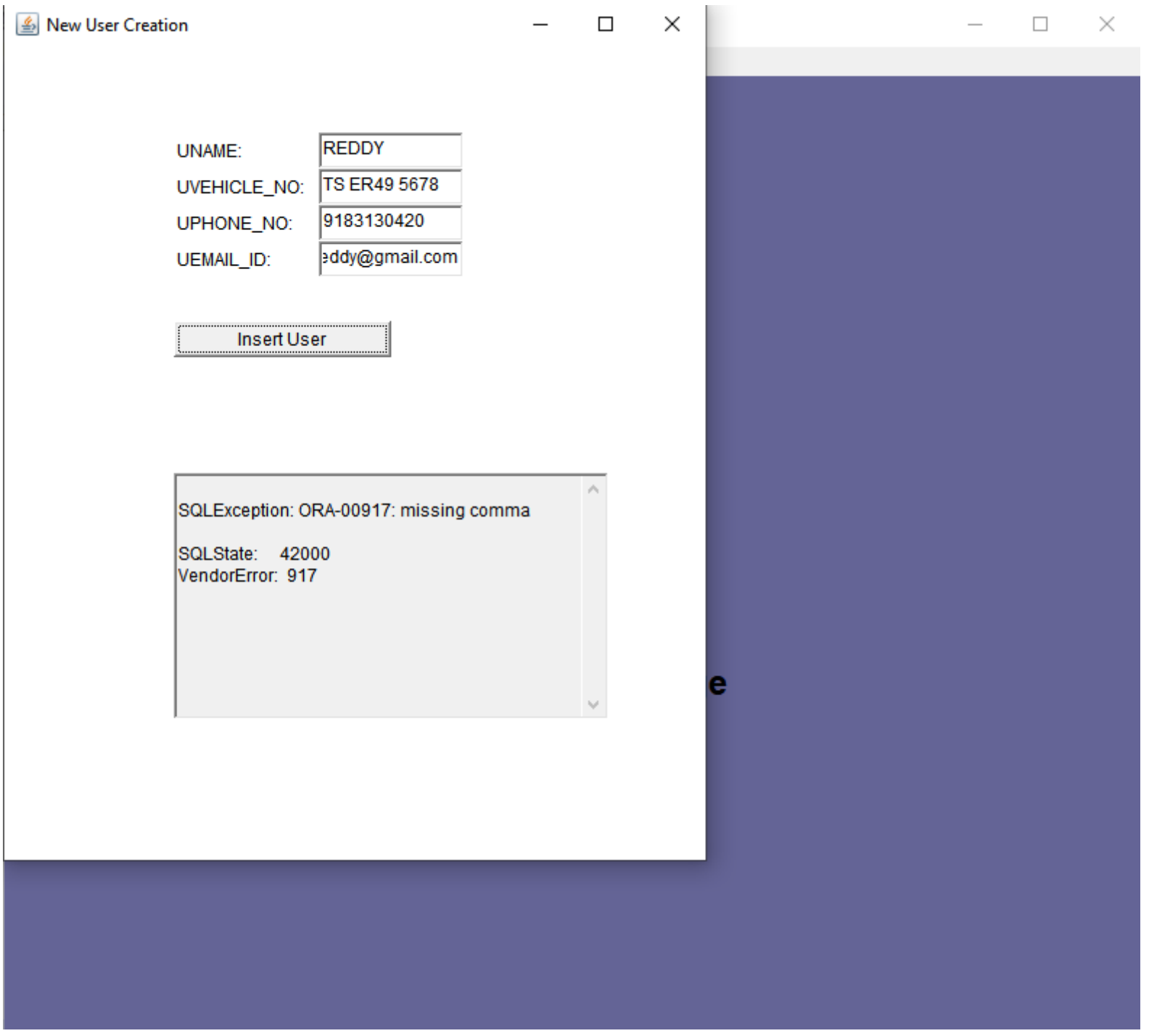
https://github.com/swetha71/B-Swetha

MAIN GUI:


AUTOMATED DANGER ZONE LIVE SYSTEM
Users
Commute
SuggestedRoutes
PostingDetails
Admin
Welcome to HomePage


AUTOMATED DANGER ZONE LIVE SYSTEM
Users
Commute
SuggestedRoutes
PostingDetails
Admin
Insert Users
Update Users
Delete Users
Welcome to HomePage

TESTING:



DISCUSSION & FUTURE WORK:

The application done till now is to store all the information related to the Automated danger zone live system . Furthermore, other programming languages can also be used along with database by connecting SQL with it. This application can be extended further more to store information automated danger zone live system.

CONCLUSION:

Thus, a Java AWT based Automated danger zone List is created which is connected to the Oracle 11g database. Therefore, all the entries in the form are directly updated on the table created in the database.

REFERENCES:

◆ Abraham Silberschatz,Henry F. Korth and S. Sudarshan, Database System concepts, McGraw-Hill Education(Asia), Fifth Edition 2006.

◆ Raghu Ramakrishna DataBase Management System, Third Edition.