
Acknowledgement

I would like to take this opportunity to thank every individual who has directly or indirectly in some way helped me during the course of this assignment.

Initially I would like to give warm thanks to my module lecturer *****, because without whom the aim of successful completion was really impossible. He is very cooperative, supportive by nature. I want to give lot of thanks to him for his suggestion and guidance.

To begin with I wish to express my deepest gratitude towards *****, for providing us a platform to achieve my goals and ambitions.

I also like to express our deepest gratitude towards ***** for motivating us at every juncture.

Again I would like to give thanks to my friends, seniors for their support in the successful completion of the assignment.

Mohit Kumar

.....

Signature

Table of Contents

1	Introduction:	5
2	UML Diagram:	6
2.1	For Login, bus and employee module:	6
2.2	For passenger and route module:	7
2.3	For Scheduling and reservation module:.....	8
2.4	For default module:	9
3	Class Diagram:	10
3.1	Class Reference:	11
4	Sequence Diagram:.....	14
4.1	Bus:.....	14
4.2	Employee:.....	15
4.3	Passenger:.....	16
4.4	Route:	17
4.5	Scheduling:.....	18
4.6	Reservation:.....	19
5	Activity Diagram:	20
6	Source Code:.....	21
6.1	Encapsulation:	21
6.2	Constructor:	24
6.3	Polymorphism and Inheritance:	26
6.4	Interface:.....	29
6.5	Abstract class:	30
6.6	Array:	32
6.7	Vector:	32
6.8	Packages:.....	33
6.9	Exception Handling:.....	33

6.10	Composition:	34
6.11	Aggregation:	34
6.12	Looping:.....	35
7	Sample Output:.....	36
7.1	Bus Module-1:.....	36
7.2	Employee Module 1:	37
7.3	Passenger 1:.....	38
7.4	Route 1:	39
7.5	Scheduling 1:.....	40
7.6	Reservation:.....	41
7.7	Bus module 2.....	42
7.8	Employee module 2:.....	43
7.9	Route module 2:	45
7.10	Schedule module 2:	46
7.11	Ticket Print:	47
7.12	BusList:.....	47
7.13	Employee List:.....	48
7.14	Passenger List:	49
7.16	Default(Menu):	50
7.17	Password:.....	50
8	Assumptions:	51
9	Additional Features:	52
10	References.....	53
11	Appendex:	54
11.1	Java Doc:	54
11.2	Booking:	54
11.3	Bus:	55

11.4	Employee:	55
11.5	List:	56
11.6	Passenger:	56
11.7	Route:.....	57
11.8	Scheduling:	57

1 Introduction:

This project is based on java programming skills, which will help to create a program or interface for the bus operator to perform tasks related to e-transportation. The main objective of this program is to create a user (bus operator) friendly interface through which he can easily work on different modules like Bus management, Route management, Employee management, Passenger management, Scheduling and Reservation.

The e-transportation system is focus on adding, searching, updating, deleting of the details of buses, employees, passengers and route to the database.

This system has only one administrator who will control all the works of e-transportation system and one default page through which bus operator can access all the forms.

Bus management, Route management, Employee management, Passenger management, Scheduling module has two interfaces, one for adding the details of the buses, route, employee, passenger and schedule and other is for searching, updating and deleting of the existing details of the buses, route, employee, passenger and schedule.

Through Bus Reservation module operator can reserve the ticket for the passenger. After Booking the ticket a ticket number will automatically generated and display to the operator and ticket print button on the form will enable. Operator can click on the ticket print button another form will open which shows the basic information of the reserved ticket.

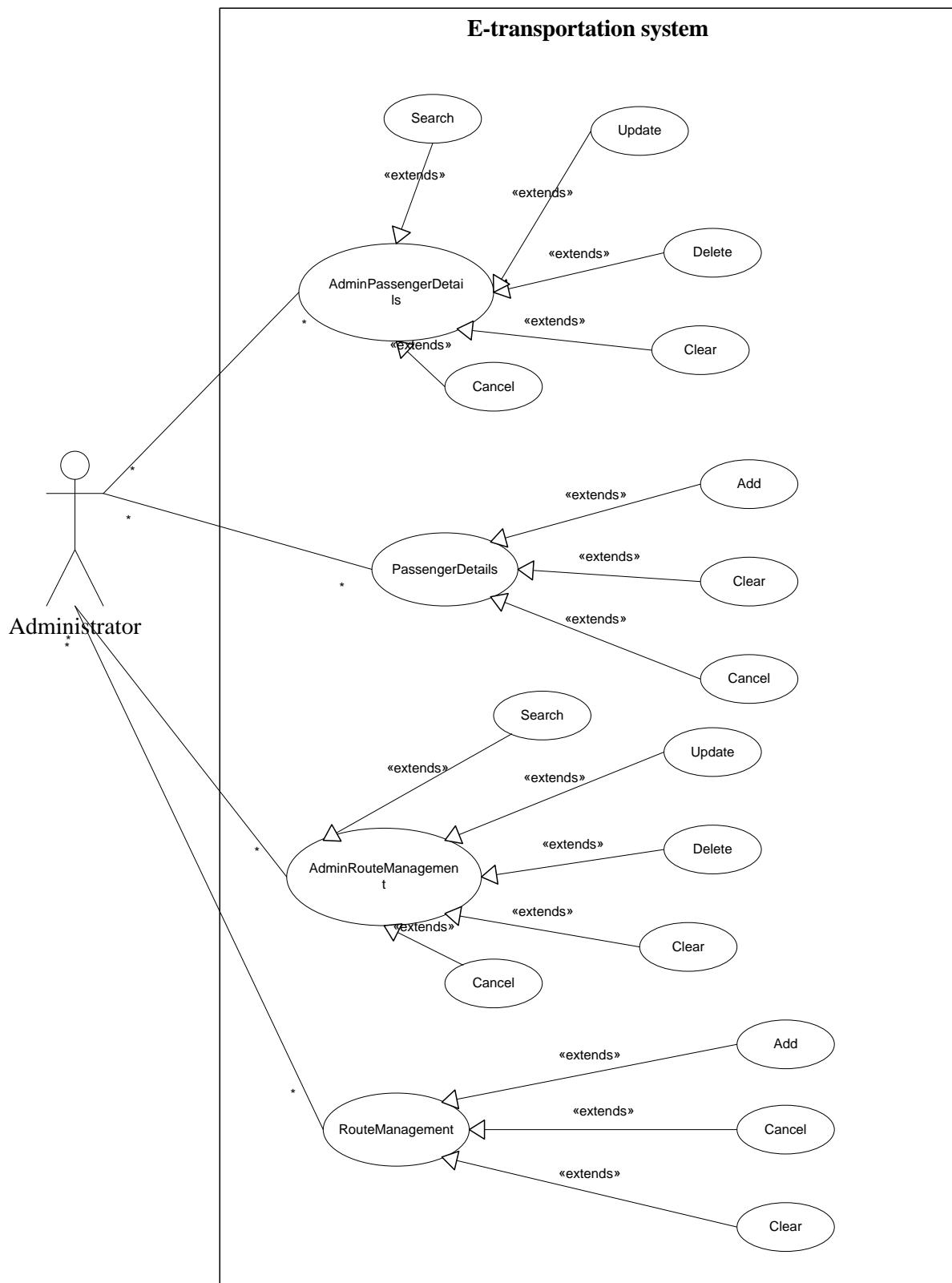
If operator wants to see the details of every buses, passengers, employees or route of buses then he can easily see the reports with the help of list option provided in the default page.

2 UML Diagram:

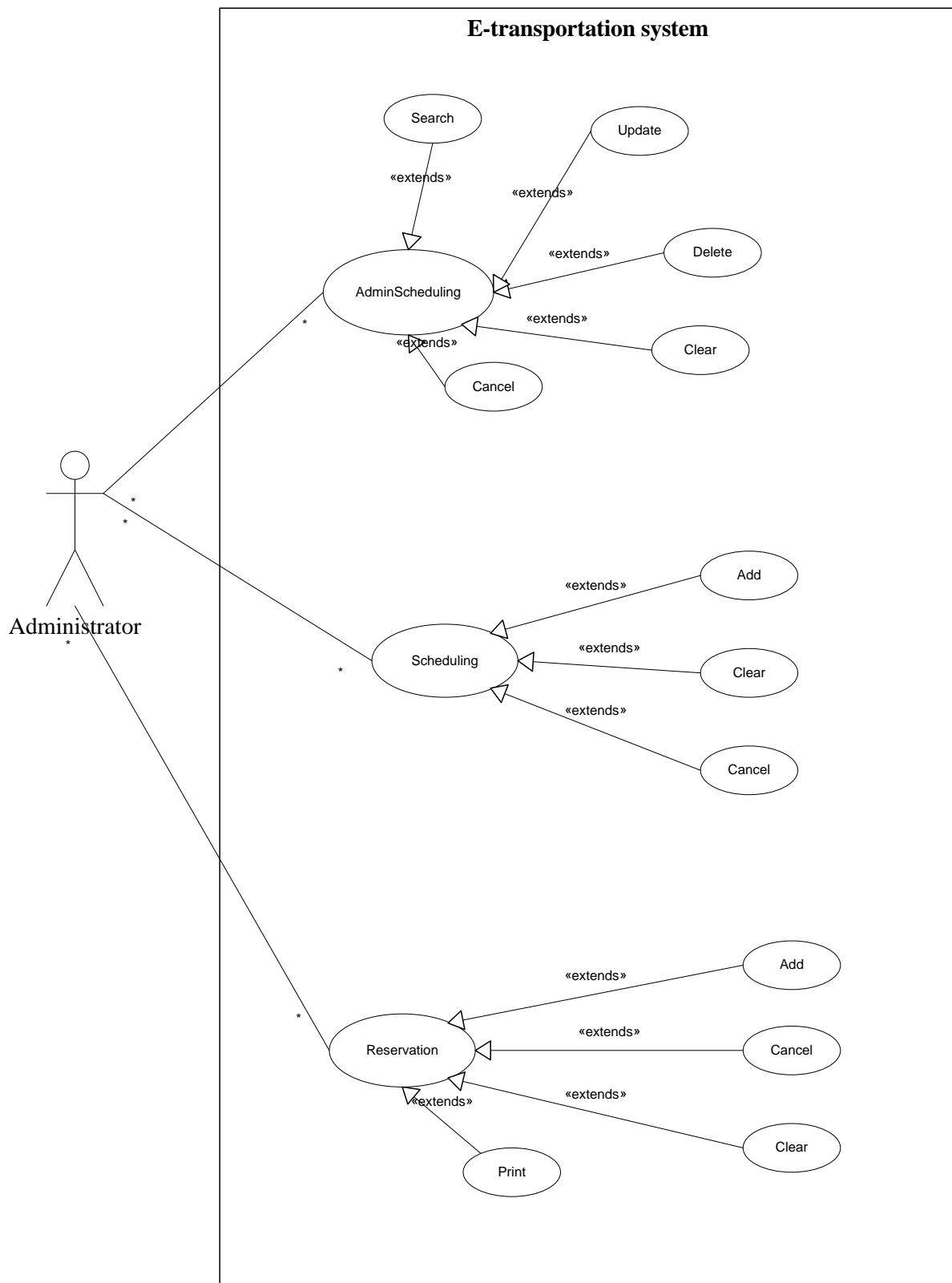
2.1 For Login, bus and employee module:



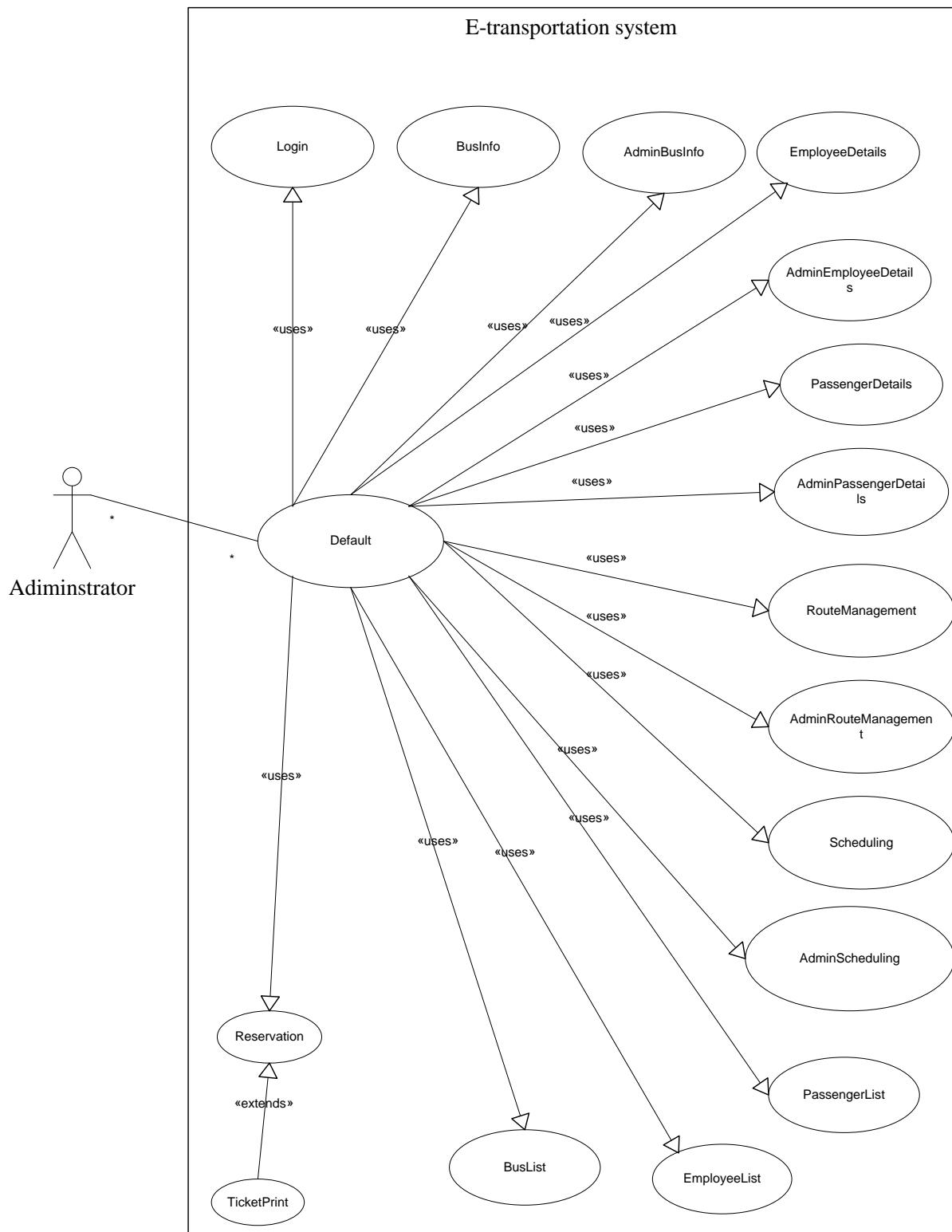
2.2 For passenger and route module:



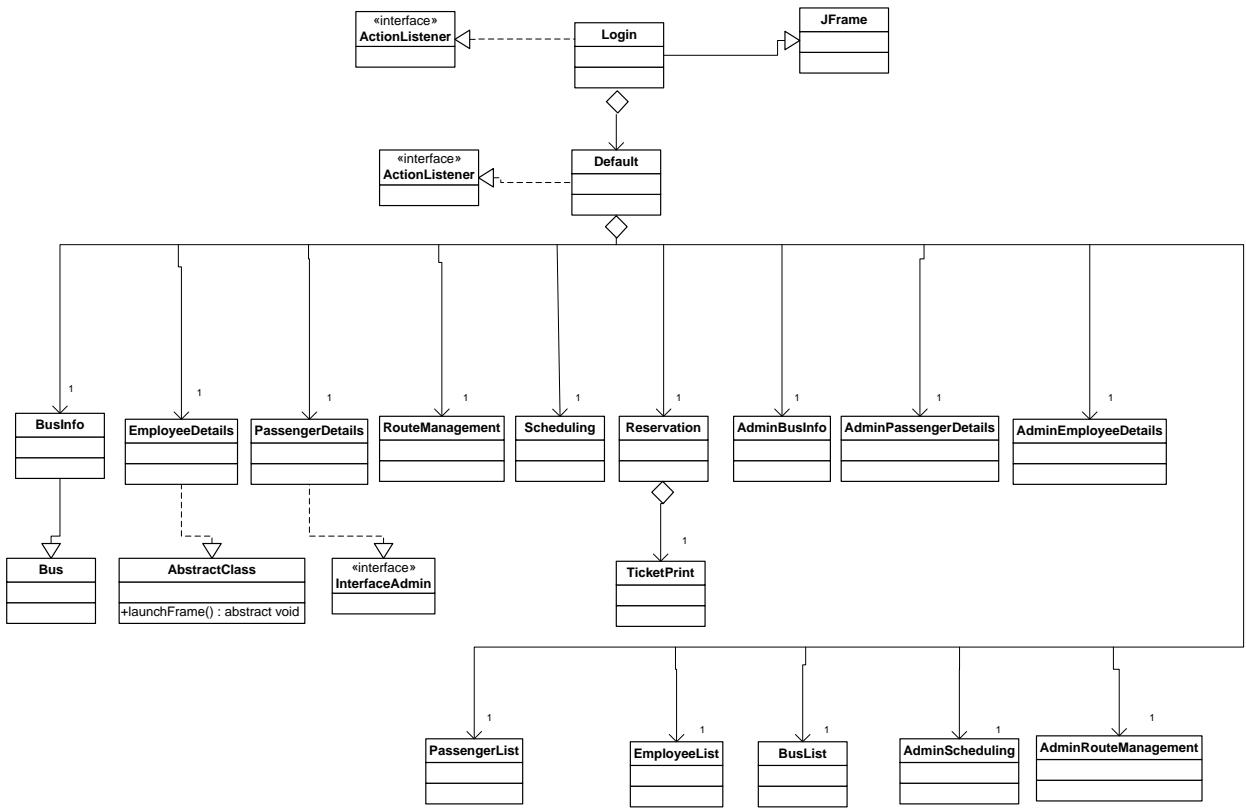
2.3 For Scheduling and reservation module:



2.4 For default module:



3 Class Diagram:



3.1 Class Reference:

Login
-check : Boolean = false
-uname : string
-pass : Char
+Login() : Login
+actionPerformed(in ActionEvent e) : void

Default
+Default() : Default
+actionPerformed(in ActionEvent e) : void

BusInfo
-Busno : string = null
-Model : string = null
-RegNo : string = null
-Sitting : string = null
-date : string = null
-ins : string = null
-cato : string = null
+BusInfo() : BusInfo
+launchFrame_bus() : void
+actionPerformed(in ActionEvent e) : void
+add() : void

EmployeeDetails
-joindate : string = null
-dobdate : string = null
-id : string = null
-fname : string = null
-ph : string = null
-sex : string = null
-mstatus : string = null
+EmployeeDetails() : EmployeeDetails
+launchFrame() : void
+actionPerformed(in ActionEvent e) : void
+add() : void

PassengerDetails
-fname : string = null
-lname : string = null
-father : string = null
-occu : string = null
-date : string = null
-address : string = null
-id : string = null
+PassengerDetails() : PassengerDetails
+launchFrame_passenger() : void
+actionPerformed(in ActionEvent e) : void
+add() : void

RouteManagement
-bus : string = null
-route_id : string = null
-ei : string = null
-from : string = null
-busid : int = 0
+RouteManagement() : RouteManagement
+launchFrame_Root() : void
+actionPerformed(in ActionEvent e) : void
+add() : void
+comboboxvalue() : void

Scheduling
-bus : string = null
-regno : string = null
-to : string = null
-from : string = null
-intregno : int = 0
-year1 : int = 0
-validate : int = 0
-month1 : int = 0
+Scheduling() : Scheduling
+launchFrame_Scheduling_Scheduling() : void
+actionPerformed(in ActionEvent e) : void
+combobox_value() : void
+search(id2 : int, routeid : int) : Boolean
+add() : void
+validation() : int

Reservation
-validate : int = 0
-reservationid : int = 0
-pname : string = null
-root : string = null
-from : string = null
-to : string = null
-date : string = null
-seat : string = null
-distance : string = null
-fare : string = null
+Reservation() : Reservation
+launchFrame_Reservation() : void
+actionPerformed(in ActionEvent e : void) : void
+combobox() : void
+validation() : int
+getvalue() : void
+add() : void
+passenger(p : int) : void
+schedulingfield(route1 : string) : void
+routefill(q : int) : void

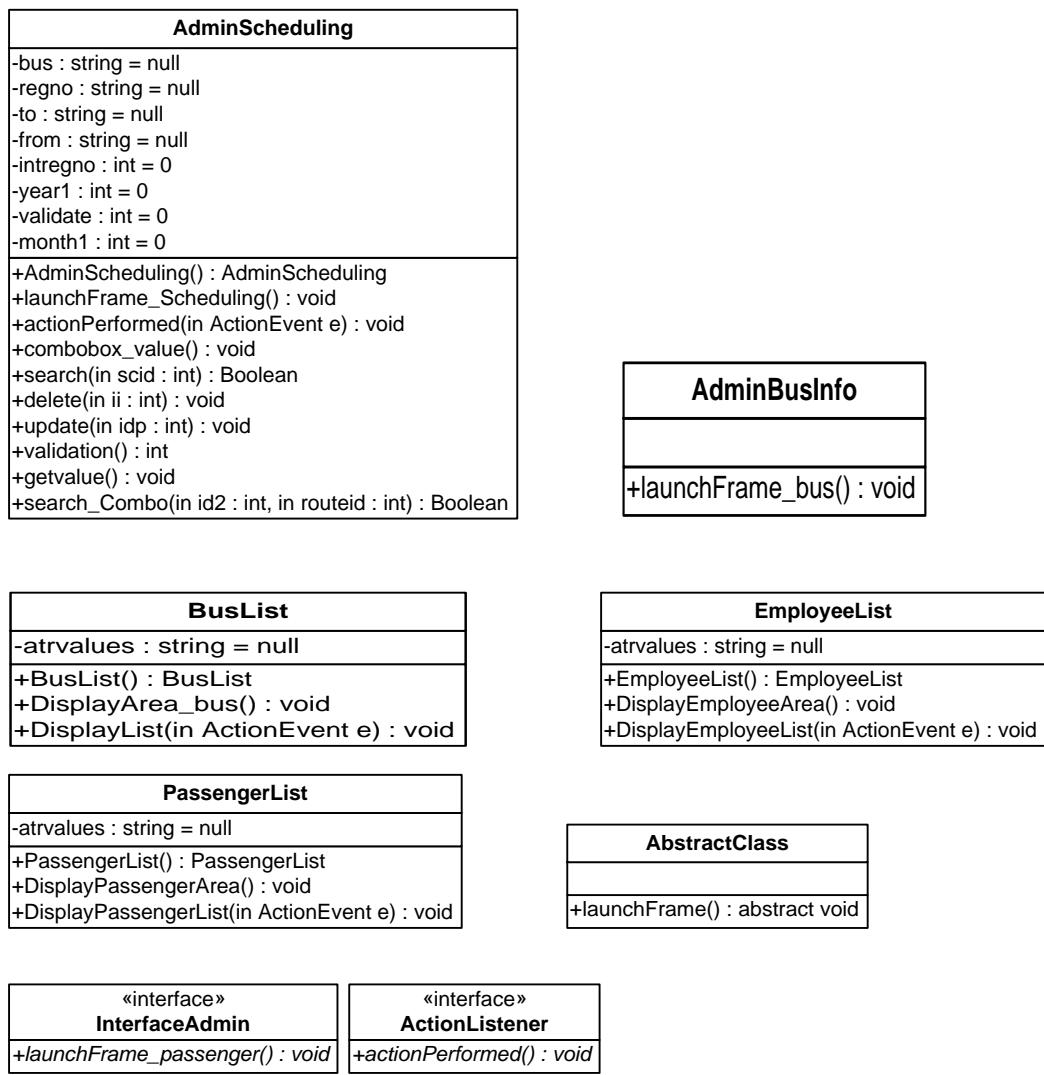
TicketPrint
-atrvalues : string = null -strsecond : string = null -strthird : string = null -strfourth : string = null -strmiddle : string = null
+TicketPrint(in x : int) : TicketPrint +DisplayPassengerArea() : void +DisplayPassengerList(in ActionEvent e) : void

BusInfo
-Busno : string = null -Model : string = null -RegNo : string = null -Sitting : string = null -date : string = null -ins : string = null -cato : string = null
+BusInfo() : BusInfo +launchFrame_bus() : void +actionPerformed(in ActionEvent e) : void +add() : void

AdminEmployeeDetails
-validate : int = 0 -phh : int = 0 -fname : string -dobd : string -dobm : string -sex : string -mstatus : string -joindate : string -dobdate : string -select : string
+AdminEmployeeDetails() : AdminEmployeeDetails +launchFrame() : void +actionPerformed(in ActionEvent e : void) : void +checkbox() : void +search(in iid : int) : Boolean +delete(in i : int) : void +update(in iid : int) : void +validation() : int +getvalue() : void

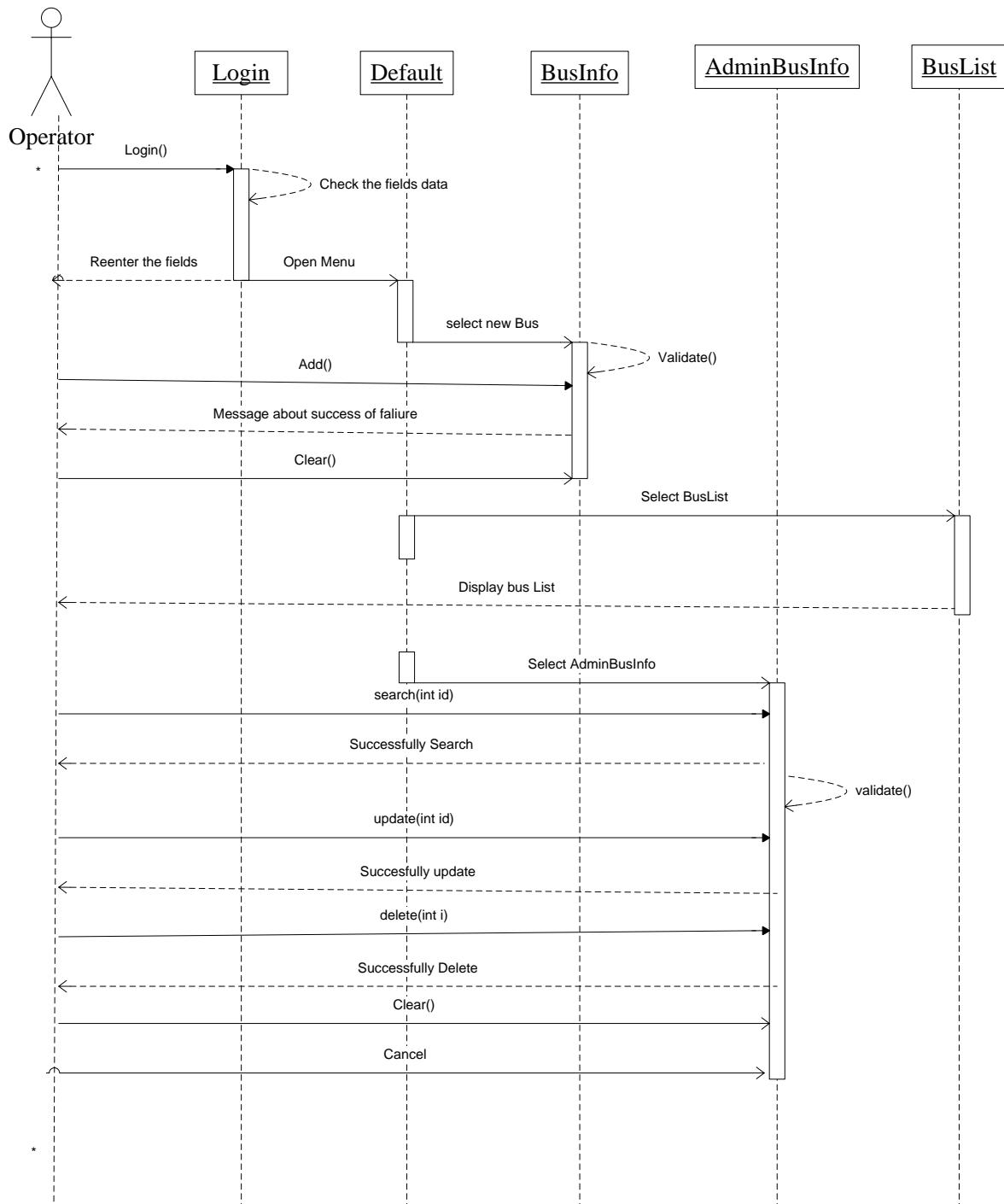
AdminPassengerDetails
-fname : string = null -lname : string = null -father : string = null -occu : string = null -f : int = 0 -bus : int = 0 -validate : int = 0
+AdminPassengerDetails() : AdminPassengerDetails +launchFrame_passenger() : void +actionPerformed(in ActionEvent e) : void +checkbox() : void +search(in id : int) : Boolean +delete(in i : int) : void +update(in id : int) : void +validation() : int +getvalue() : void +clear() : void

AdminRouteManagement
-bus : string = null -route_id : string = null -ei : string = null -from : string = null -f : int = 0 -search_value : int = 0 -validate : int = 0 -busid : int = 0
+AdAdminRouteManagementminPassengerDetails() : AdminRouteManagement +launchFrame_Route() : void +actionPerformed(in ActionEvent e) : void +checkboxvalue() : void +search(in id : int) : Boolean +delete(in i : int) : void +update(in id : int) : void +validation() : int +getvalue() : void +clear() : void

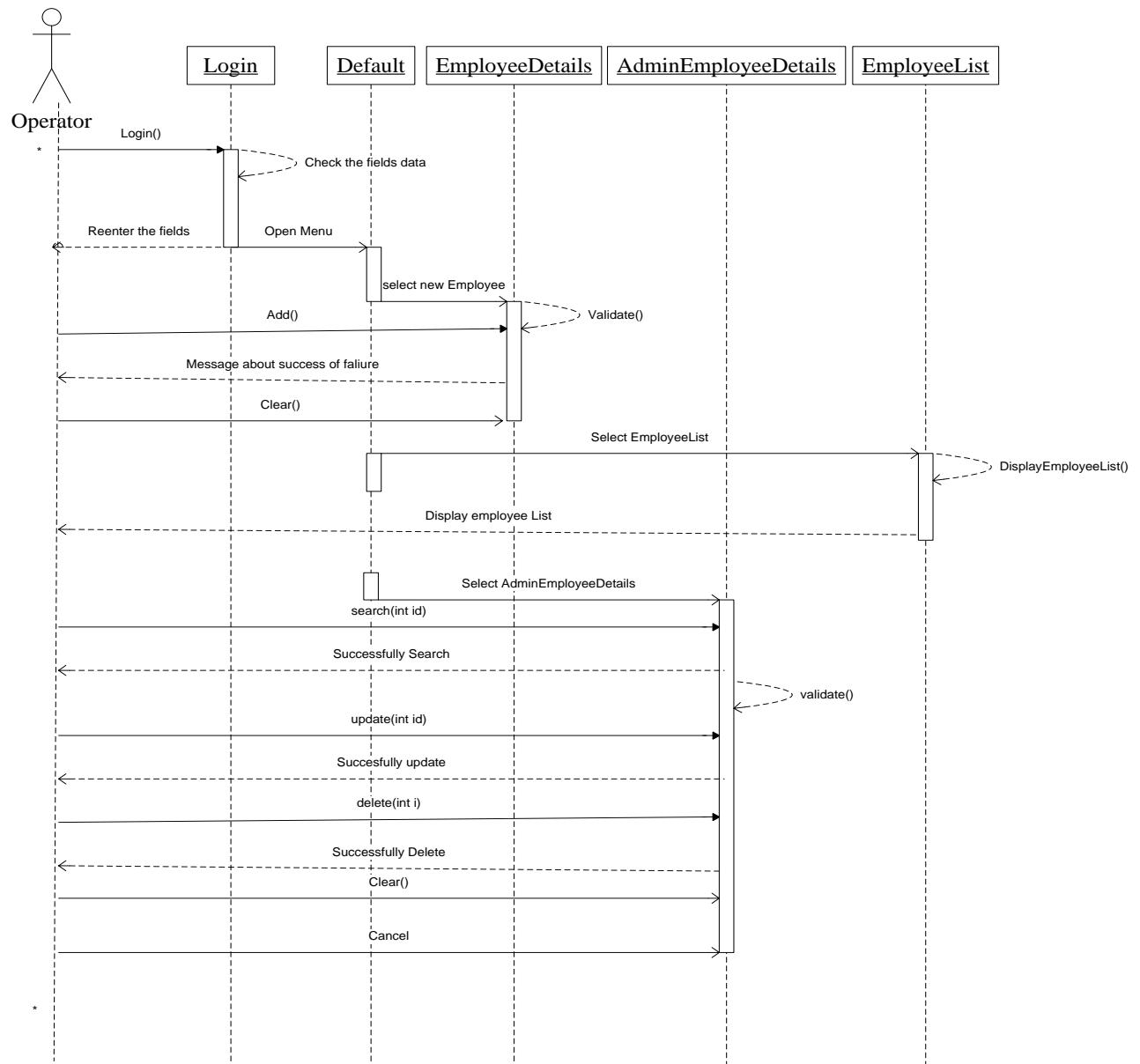


4 Sequence Diagram:

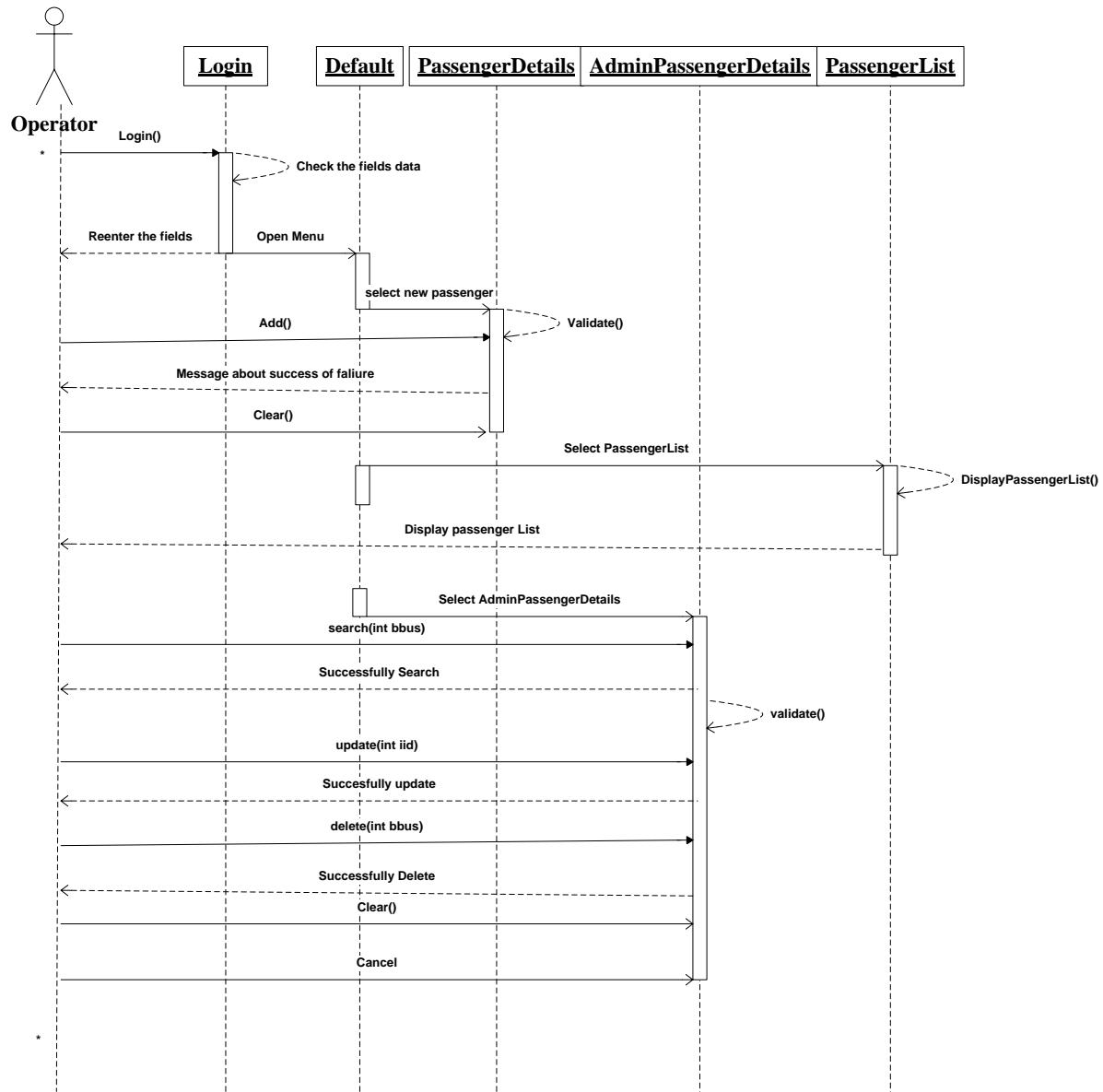
4.1 Bus:



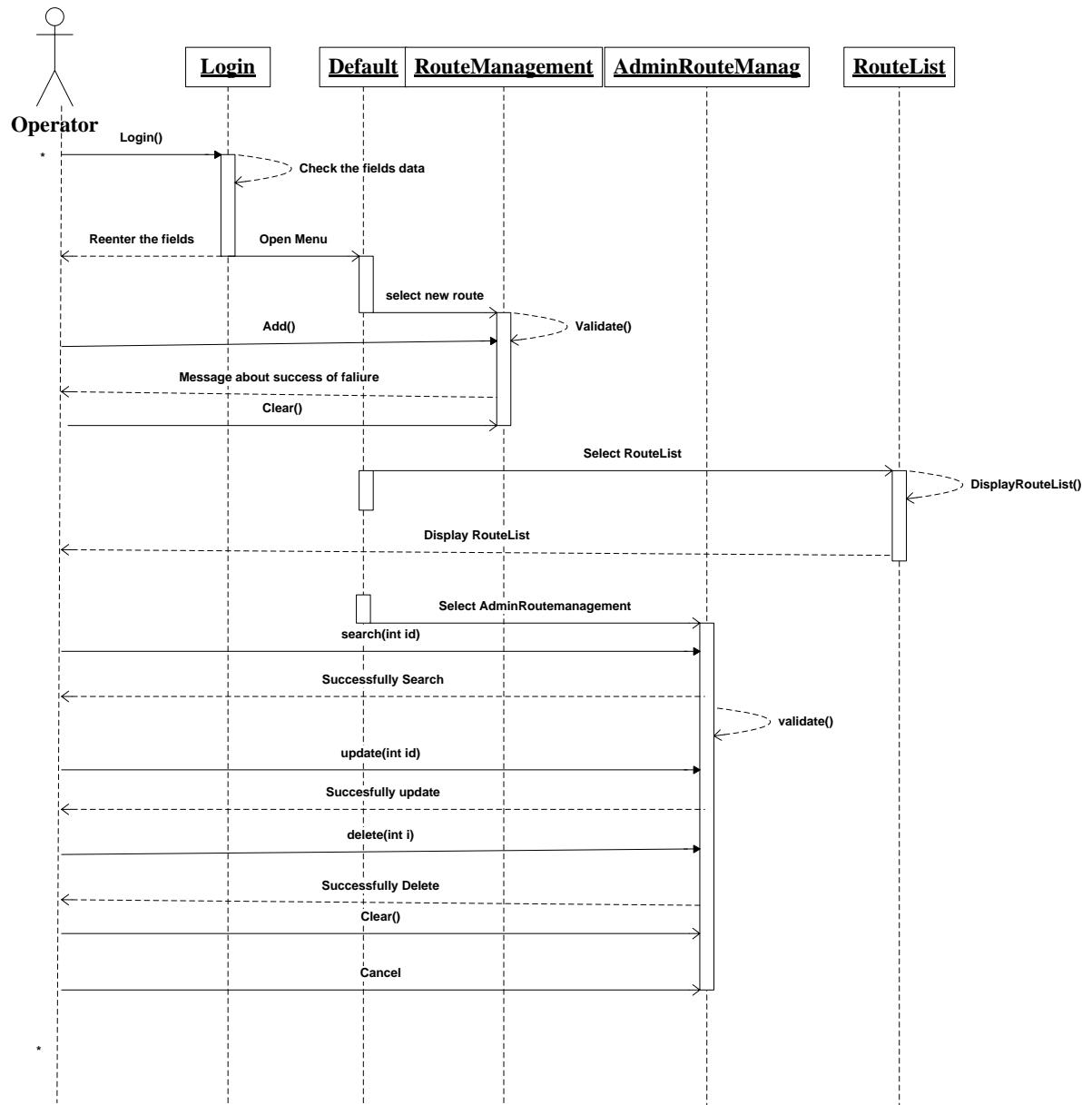
4.2 Employee:



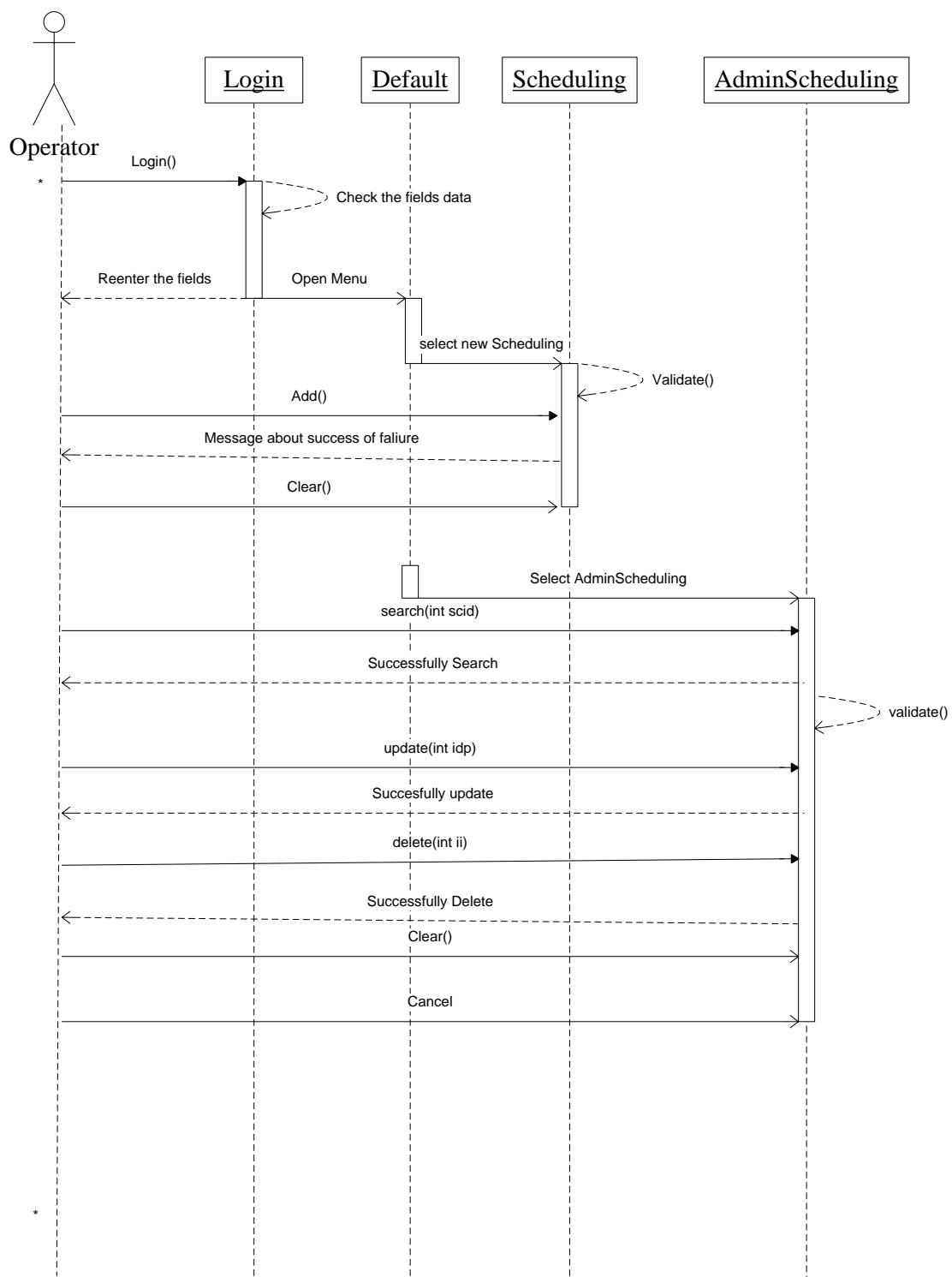
4.3 Passenger:



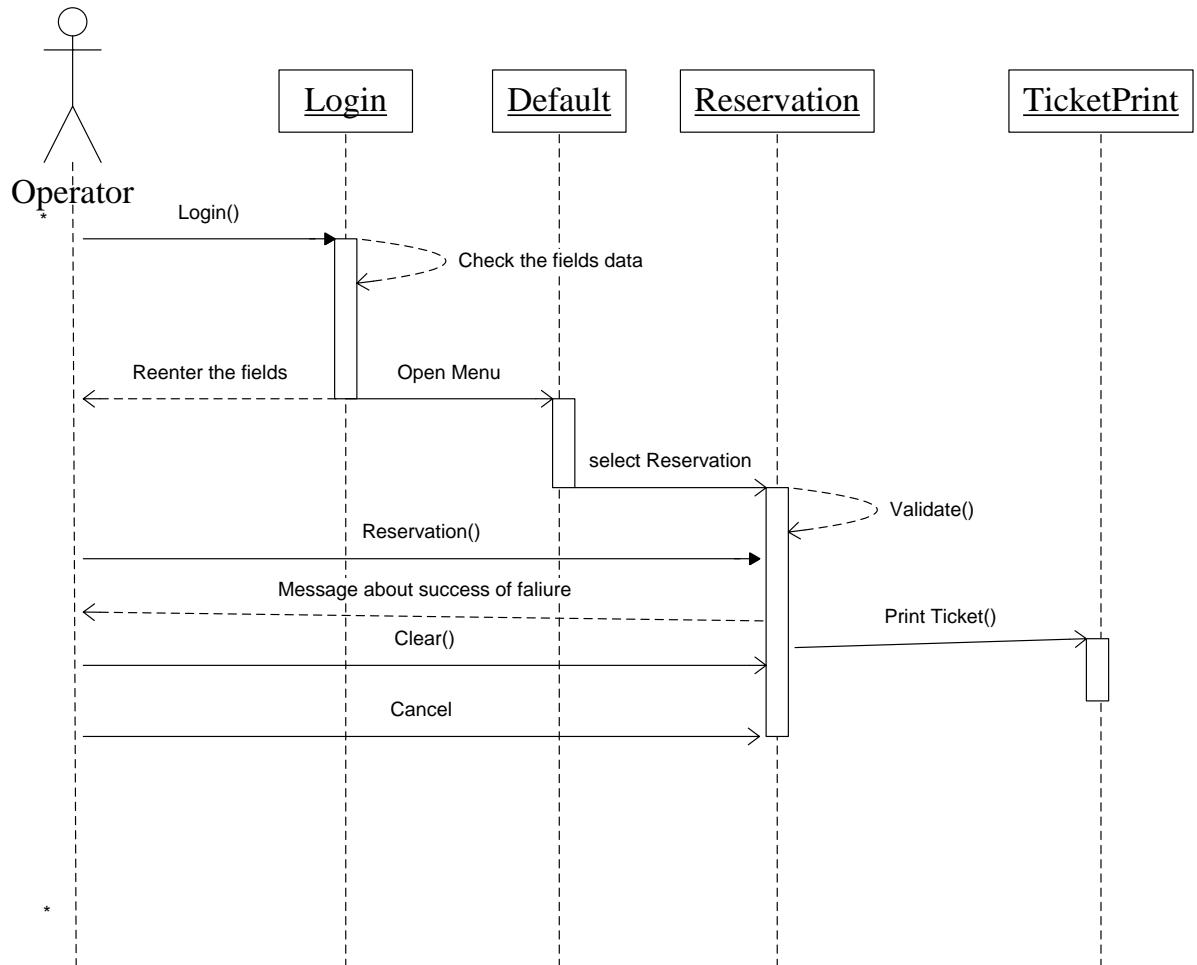
4.4 Route:



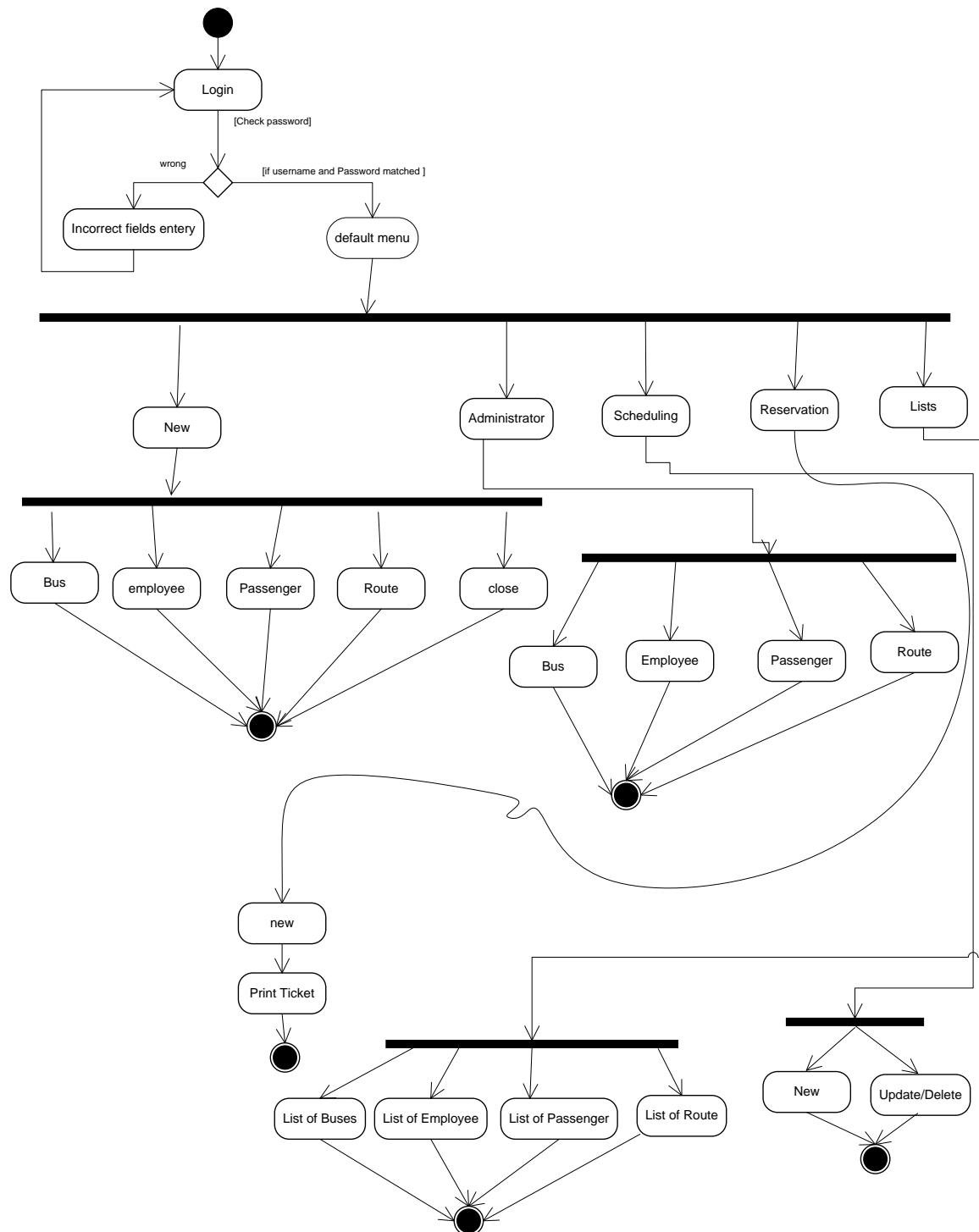
4.5 Scheduling:



4.6 Reservation:



5 Activity Diagram:



6 Source Code:

6.1 Encapsulation:

Through encapsulation developer make the private access of the data to public via public method. Developer used this technique in likely to all the form. Here is the example of AdminRouteManagement module in which he used this technique.

Here is code snip :

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;  
import java.sql.*;  
  
/**Through this class administrator can update,search and delete the records of Route and  
cancel the operation.*/  
  
public class AdminRouteManagement implements ActionListener  
{  
  
    private JFrame frame;  
  
    private JButton btnCancel,btnDelete,btnClear,btnUpdate,btnSearch;  
  
    private JLabel  
        lblHeading,lblBusNo,lblRootNo,lblEmployeeNo,lblFare,lblRunTill,lblSpeed,lblTo,lbl  
        From,lblTravelDuration,lblDistance,lblVBN,lblVRN,lblVTD,lblVEI,lblVT,lblVF,lbl  
        VAT,lblVDT,lblVSA,lblVD;  
  
    private JTextField  
        txtFare,txtRunTill,txtFrom,txtTo,txtTravelDuration,txtDistance,txtspeed;  
  
    private JComboBox cboBusNo,cboEmployeeNo,cboRootNo;  
  
    private JPanel top,center,bottom;
```

```
private String  
bus=null,route_id=null,ei=null,to=null,from=null,runtill=null,travelduration=null,dist  
ance=null,speed=null,fare=null,select=null;  
  
private int busid=0,validate=0,search_value=0,f=0;  
  
private ImageIcon icon;  
  
private Color color1,color2,color3;  
  
/**used to initilise the fields of the form*/  
  
public AdminRouteManagement()  
{  
  
frame=new JFrame("Route Management");  
  
top=new JPanel();  
  
center=new JPanel();  
  
bottom=new JPanel();  
  
lblHeading=new JLabel("Route Management");  
  
lblBusNo=new JLabel("Bus No.:");  
  
lblRootNo=new JLabel("Route No.:");  
  
lblEmployeeNo=new JLabel("Employee ID:");  
  
lblSpeed=new JLabel("Avg. Speed:");  
  
lblFare=new JLabel("Fare:");  
  
lblRunTill=new JLabel("Run Till:");  
  
lblFrom=new JLabel("From:");  
  
lblTo=new JLabel("To:");  
  
lblTravelDuration=new JLabel("Travel Duration:");
```

```
txtTravelDuration=new JTextField();  
  
cboBusNo=new JComboBox();  
  
cboRootNo=new JComboBox();  
  
cboEmployeeNo=new JComboBox();  
  
txtspeed=new JTextField();  
  
txtFare=new JTextField();  
  
txtRunTill=new JTextField();  
  
txtFrom=new JTextField();  
  
txtTo=new JTextField();  
  
lblDistance=new JLabel("Distance");  
  
txtDistance=new JTextField();  
  
  
  
  
icon=new ImageIcon("wrong.jpeg");  
  
lblVBN=new JLabel("Select one of the option",icon,JLabel.CENTER);  
  
lblVRN=new JLabel("enter number only",icon,JLabel.CENTER);  
  
lblVTD=new JLabel("enter number only",icon,JLabel.CENTER);  
  
lblVEI=new JLabel("Select one of the option ",icon,JLabel.CENTER);  
  
lblVT=new JLabel("Enter Text only",icon,JLabel.CENTER);  
  
lblVF=new JLabel("Enter Text only",icon,JLabel.CENTER);  
  
lblVAT=new JLabel("only number ",icon,JLabel.CENTER);  
  
lblVDT=new JLabel("only number ",icon,JLabel.CENTER);  
  
lblVSA=new JLabel("enter number only",icon,JLabel.CENTER);  
  
lblVD=new JLabel("only Enter number",icon,JLabel.CENTER);
```

```

btnDelete=new JButton("Delete",new
ImageIcon(ClassLoader.getSystemResource("add1.gif")));

btnCancel=new JButton("Cancel",new
ImageIcon(ClassLoader.getSystemResource("delete.png")));

btnClear=new JButton("Clear",new
ImageIcon(ClassLoader.getSystemResource("clear.png")));

btnUpdate=new JButton("Update",new
ImageIcon(ClassLoader.getSystemResource("update.jpg")));

btnSearch=new JButton("Search",new
ImageIcon(ClassLoader.getSystemResource("Search.png")));

color1=new Color(25,107,136);

color2=new Color(191,233,247);

color3=new Color(25,107,136);

}

```

6.2 Constructor:

Developer used this concept to initialize the objects.

Code snip of RouteManagement class:

```

public class RouteManagement implements ActionListener
{
    private JFrame f;

    private JButton btnCancel,btnAdd,btnClear;

    private JLabel
    lblHeading,lblBusNo,lblRootNo,lblEmployeeNo,lblFare,lblRunTill,lblSpeed,lblTo,lbl
    From,lblTravelDuration,lblDistance,lblVBN,lblVRN,lblVTD,lblVEI,lblVT,lblVF,lbl
    VAT,lblVDT,lblVSA,lblVD;

```

```
private JTextField  
txtRootNo,txtFare,txtRunTill,txtFrom,txtTo,txtTravelDuration,txtDistance,txtSpeed;  
  
private JComboBox cboBusNo,cboEmployeeNo;  
  
private JPanel top,center,bottom;  
  
public RouteManagement()  
{  
    f=new JFrame("Route Management");  
  
    top=new JPanel();  
  
    center=new JPanel();  
  
    bottom=new JPanel();  
  
    lblHeading=new JLabel("Route Management");  
  
    lblBusNo=new JLabel("Bus No.:");  
  
    lblRootNo=new JLabel("Route No.:");  
  
    lblEmployeeNo=new JLabel("Employee ID:");  
  
    lblSpeed=new JLabel("Avg. Speed:");  
  
    lblFare=new JLabel("Fare:");  
  
    lblRunTill=new JLabel("Run Till:");  
  
    lblFrom=new JLabel("From:");  
  
    lblTo=new JLabel("To:");  
  
    lblTravelDuration=new JLabel("Travel Duration:");  
}
```

6.3 Polymorphism and Inheritance:

Developer used overriding technique in the AdminBusInfo class and extending the class bus .

Code snip:

```
package Bus;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.util.Vector;

/** Overiding the class Bus method */

public class AdminBusInfo extends Bus

{
    public void launchFrame_bus()

    {
        top.setLayout(new FlowLayout(FlowLayout.CENTER,0,15));
        top.setBackground(color1);
        top.add(lblHeading);
        lblHeading.setFont(new Font("Algerian", Font.BOLD, 30));
        lblHeading.setForeground(new Color(234,53,107));

        center.setLayout(null);
        center.setBackground(color2);
        combobox();
```

```
center.add(lblBusNo);

lblBusNo.setBounds(20,10,70,30);

center.add(cboBusNo);

cboBusNo.setBorder(BorderFactory.createLineBorder(Color.green,1));

cboBusNo.setBounds(120,10,150,30);

center.add(lblVBN);

lblVBN.setBounds(260,10,200,30);

lblVBN.setVisible(false);

center.add(lblModel);

lblModel.setBounds(20,60,70,30);

-----
-----
-----



}

}/**class AdminBusInfo used for administrator to change the data. This class inheriting the
JFrame Class and also implements all the method of ActionListener class*/
class Bus implements ActionListener

{

    public JFrame frame;

    public JLabel lblHeading,lblBusNo,lblModel,
    lblRegNo,lblSittingCapacity,lblPurchargeDate,lblInsuranceStatus,lblCato,lblVBN,lblVM,lblV
    RN,lblVSC,lblVPD,lblVIS,lblVC;

    public JTextField txtModel, txtRegNo,txtSittingCapacity,txtPurchargeDate;

    public JButton btnDelete, btnCancel, btnClear,btnUpdate,btnSearch;
```

```
private String sModel=null,sdate=null,sins=null,scato=null;

private int sBusno=0,sRegNo=0,sSitting=0,f=0,validate=0;

public JComboBox cmbInsuaranceStatus,cboCato,cboBusNo;

public ImageIcon icon;

Vector v,c;

public JPanel top,center,bottom;

public Color color1,color2,color3;

private String select;

private String Busno,Model, RegNo,Sitting,date,ins,cato;

private int id;

/**Constructor of class used to Initilize the values*/

public Bus()

{

    frame=new JFrame("Bus Information");

    -----
    -----



}

/**Overridden the method by AdminBusInfo class*/

public void launchFrame_bus()

{

    System.out.println("PT1181120");

}
```

6.4 Interface:

It is pure abstract class in java. Developer implementing interface in AdminPassengerDetails class.

Code snip:

```
interface InterfaceAdmin
```

```
{
```

```
    /**Defing method as abstract*/
```

```
    public abstract void launchFrame_passenger();
```

```
}
```

```
public class AdminPassengerDetails implements ActionListener,InterfaceAdmin
```

```
{
```

```
-----
```

```
-----
```

```
public void launchFrame_passenger()
```

```
{
```

```
    top.setLayout(new FlowLayout(FlowLayout.CENTER,0,15));
```

```
    top.setBackground(color1);
```

```
    top.add(lblHeading);
```

```
    lblHeading.setFont(new Font("Edwardian Script ITC", Font.BOLD, 35));
```

```
    lblHeading.setForeground(new Color(255,255,255));
```

```
    center.setLayout(null);
```

```
center.setBackground(color2);

combobox();

center.add(lblFName);

lblFName.setBounds(100,5,120,20);

center.add(txtFName);

txtFName.setBorder(BorderFactory.createLineBorder(Color.green,1));

txtFName.setBounds(240,5,150,20);

center.add(lblVFN);

lblVFN.setBounds(350,5,200,20);

lblVFN.setVisible(false);

center.add(lblLName);

lblLName.setBounds(100,40,120,20);

center.add(txtLName);

txtLName.setBorder(BorderFactory.createLineBorder(Color.green,1));

txtLName.setBounds(240,40,150,20);

-----
-----
```

}

6.5 Abstract class:

Developer used this technique in AdminEmployeeDetails.

Code snip:

abstract class AbstractClass

{

```
abstract void launchFrame();  
}  
  
/**define the method(launchFrame()) of class AbstractClass. Administrator can  
Search,Update and delete the records of employee, cancel the operation and also clear the  
form fields.*/
```

```
public class AdminEmployeeDetails extends AbstractClass implements ActionListener  
{
```

public void **launchFrame()**

```
{  
    top.setLayout(new FlowLayout(FlowLayout.CENTER,0,15));  
  
    top.setBackground(color1);  
  
    top.add(lblHeading);  
  
    lblHeading.setFont(new Font("Algerian", Font.BOLD, 30));  
  
    lblHeading.setForeground(new Color(244,125,0));  
  
    center.setLayout(null);  
  
    center.setBackground(color2);  
  
    combobox();  
  
    center.add(lblID);  
  
    lblID.setBounds(100,5,120,20);  
  
    center.add(cboID);  
  
    cboID.setBorder(BorderFactory.createLineBorder(Color.green))
```

```
cboID.setBounds(240,5,150,20);

center.add(lblIEmpID);

lblIEmpID.setBounds(300,5,400,20);

lblIEmpID.setVisible(false);

center.add(lblName);

-----
-----
}

}
```

6.6 Array:

Developer used this to declear gender ans status in AdminEmployeeDetails class. Object is the parent of all the classes in java.

Code snip:

```
Object [] gender={"Gender","Male","Female"};

Object [] status={"Status","Married","Unmarried"};
```

6.7 Vector:

Developer used vector to add the elements to the combobox in AdminBusInfo Class.

```
lblCato=new JLabel("Category:");

c=new Vector();

c.addElement("Category");

c.addElement("2*2 AC");

c.addElement("3*2 AC");

c.addElement("Sleeper AC");
```

```
c.addElement("2*2 Non-AC");

c.addElement("3*2 Non-AC");

c.addElement("Sleeper Non-AC");

c.addElement("Cabin");

cboCato=new JComboBox(c);
```

6.8 Packages:

Developer provides the grouping of variety of classes with the help of package.

Code snip:

```
package Schedule;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

/**Through this class administrator can update,search and delete the records of Scheduling
and cancel the operation. */

public class AdminScheduling implements ActionListener

{
```


```
}
```

6.9 Exception Handling:

Developer used this concept to handle run time (uncheck) exception. He used this concept in many places in the code.

Code snip(AdminEmployeeDetails class):

```
try
{
    iid=Integer.parseInt(id);
}

catch (Exception d)
{
    f=1;
    JOptionPane.showMessageDialog(null,"      must      fill
Employee id");
}

}
```

6.10 Composition:

Association is of two type one is composition and other is Aggregation . composition is instance variables that are references to the objects. Developer creates object of the class.

Code snip(Default class):

```
AdminScheduling guiScheduling = new AdminScheduling();
guiScheduling.launchFrame_Scheduling();
```

6.11 Aggregation:

It is a has a relationship.

Code Snip(AdminPassengerDetails class):

```
Object []
month={"Month","January","February","March","April","May","June","July","August","September","October","December"};
```

6.12 Looping:

Developer used loop in validation part of his code.

Code sinp(AdminPassengerDetails):

```
for(i=0;i<id.length();i++)  
{  
    if((id.charAt(i)>='0' && id.charAt(i)<='9') && id.length()==6)  
        p++;  
}
```

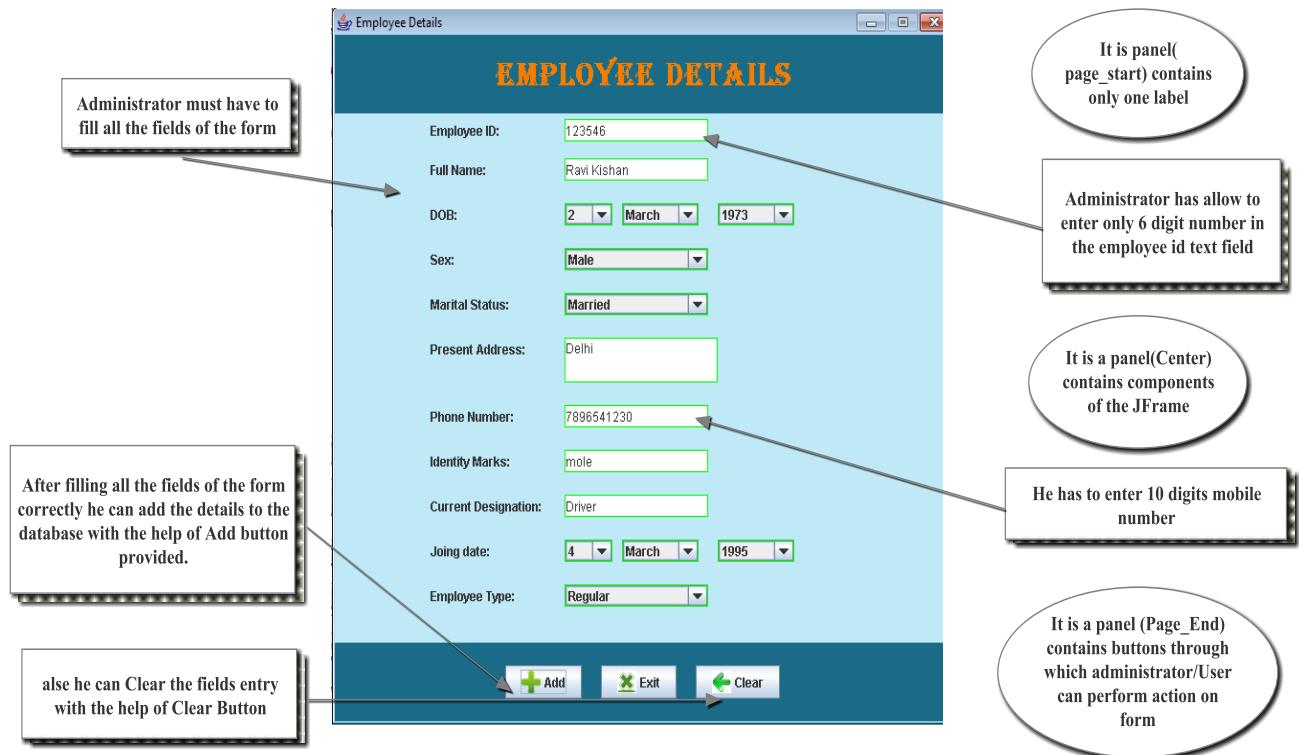
7 Sample Output:

7.1 Bus Module-1:

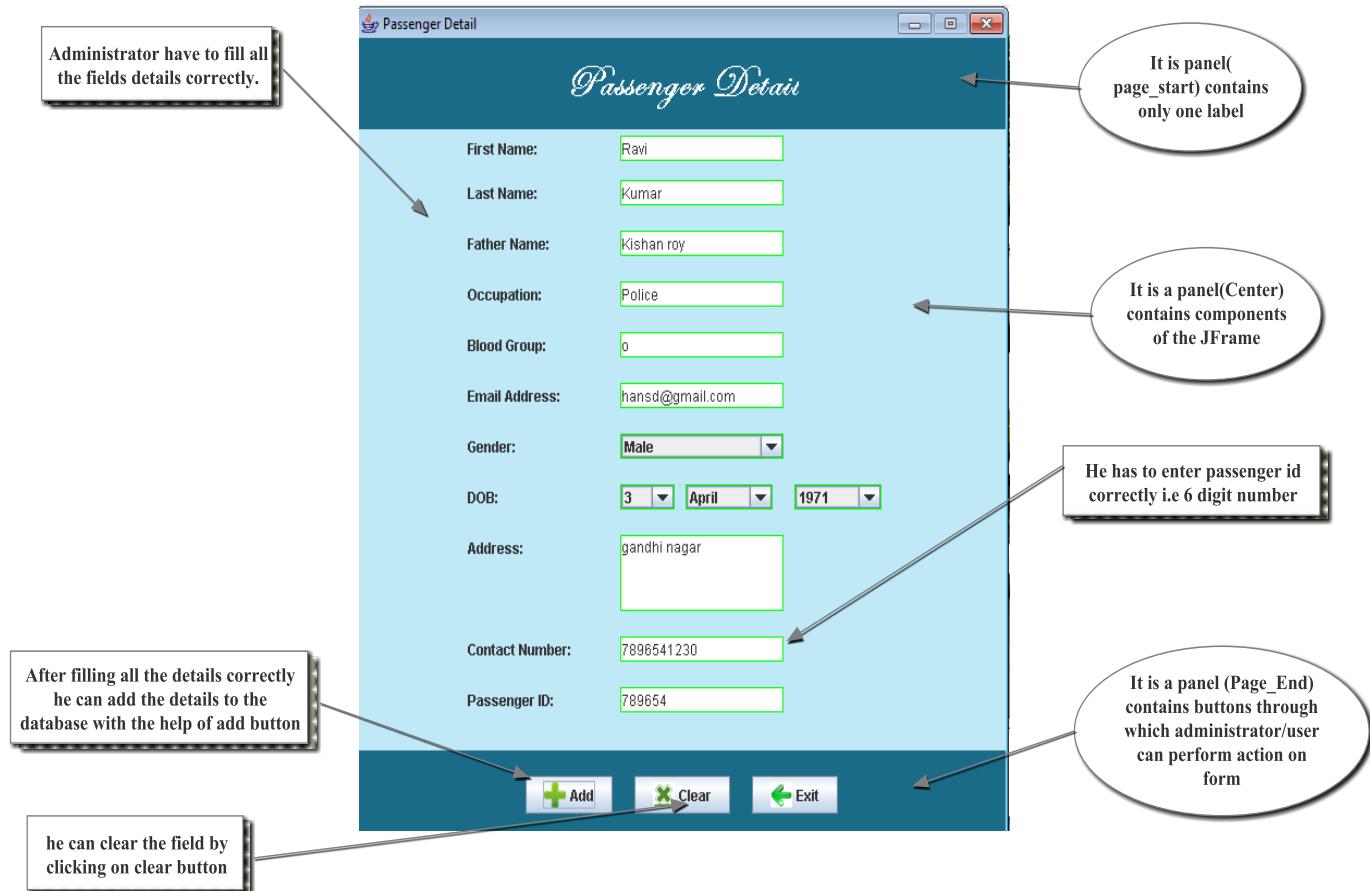
The screenshot shows a Windows application window titled "Bus Information" with a dark blue header bar. The main title "BUS INFORMATION" is centered in red capital letters. Below the title are several input fields and dropdown menus, each with a label and a value. At the bottom right are three buttons: "Add" (with a plus sign), "Cancel" (with a cross), and "Clear" (with a left arrow). The form is annotated with several callouts and arrows pointing to specific elements:

- An oval at the top right states: "It is panel(page_start) contains only one label". An arrow points from this text to the title bar.
- A rectangular box on the right side says: "Administrator should enter 4 digit bus number". An arrow points from this text to the "Bus No." field, which contains "789655".
- A rectangular box below it says: "Administrator should enter bus model". An arrow points from this text to the "Model" field, which contains "Volvo".
- A rectangular box on the left side says: "He has to enter 6 digit reg number of the bus". An arrow points from this text to the "Reg. No." field, which contains "789654".
- A rectangular box below it says: "He has to enter the capacity of bus (2 digit number)". An arrow points from this text to the "Sitting capacity" field, which contains "12".
- A rectangular box on the left side says: "He has to select Insurance status of the bus". An arrow points from this text to the "InsuranceStatus" dropdown menu, which is set to "Expired".
- A rectangular box below it says: "He has to select category of the bus". An arrow points from this text to the "Category" dropdown menu, which is set to "3*2 AC".
- A large rectangular box on the left side says: "After filling all the fields of the form he can add the details to the database with the help of add button". An arrow points from this text to the "Add" button.
- A rectangular box on the right side says: "It is a panel(Center) contains components of the JFrame". An arrow points from this text to the central area of the form containing the input fields.
- A rectangular box below it says: "He has to enter purchase date of the bus from 2000 to 2013". An arrow points from this text to the "Purchase Date" field, which contains "12/12/2002".
- A large oval at the bottom right states: "It is a panel (Page_End) contains buttons through which administrator can perform action on form". An arrow points from this text to the "Add", "Cancel", and "Clear" buttons.

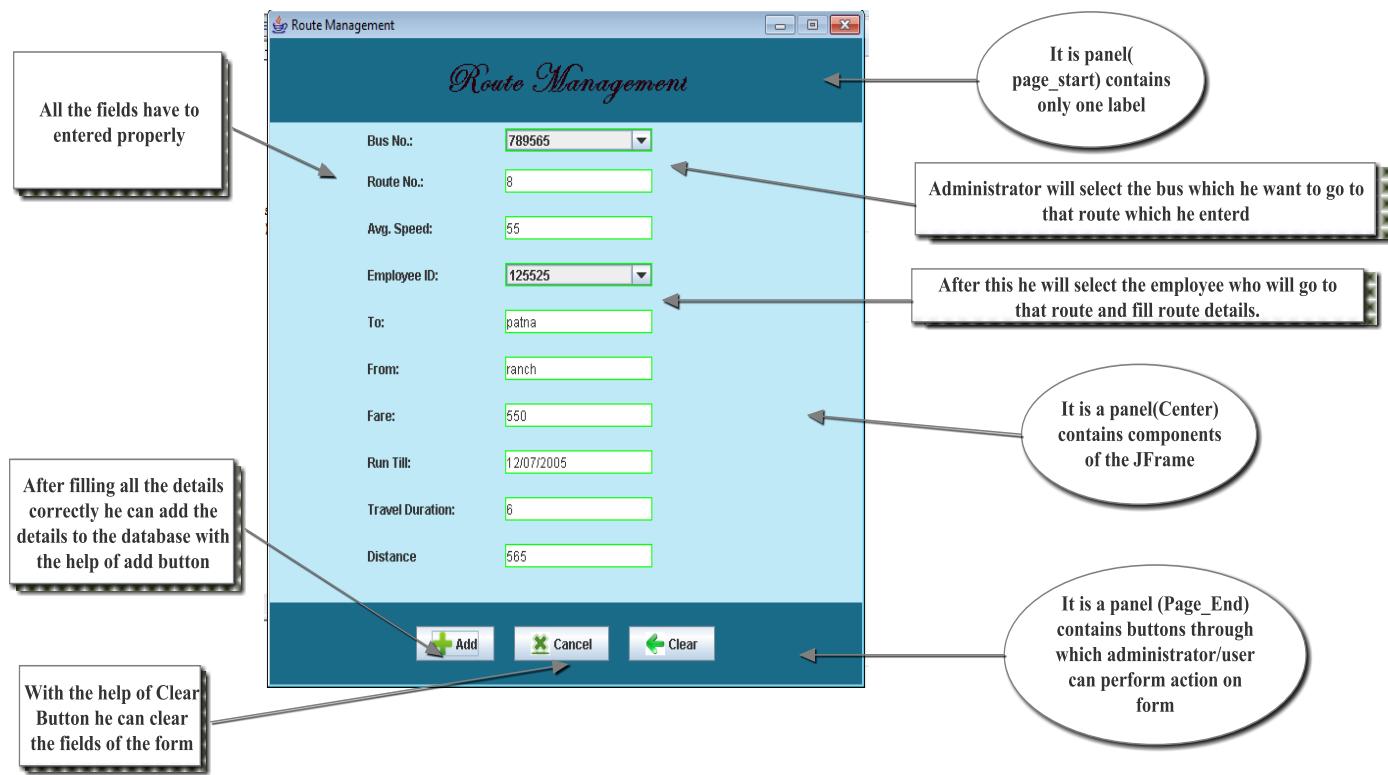
7.2 Employee Module 1:



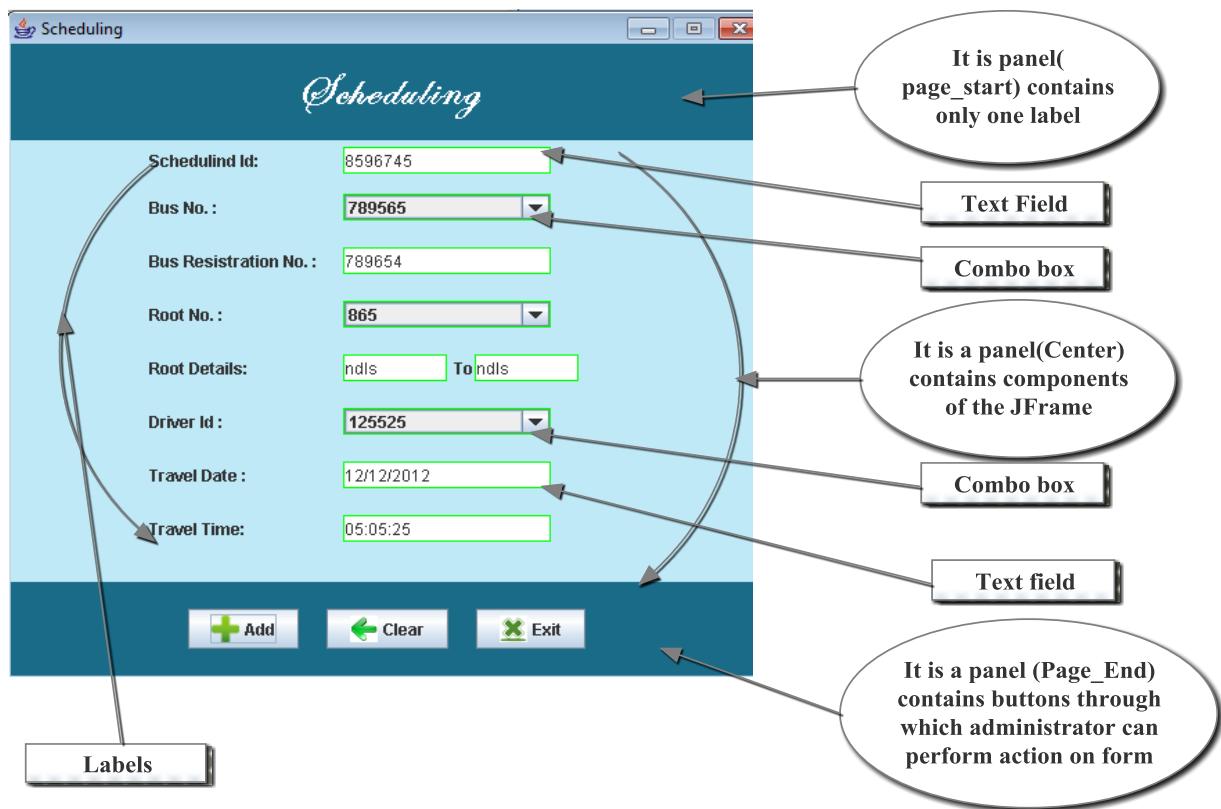
7.3 Passenger 1:



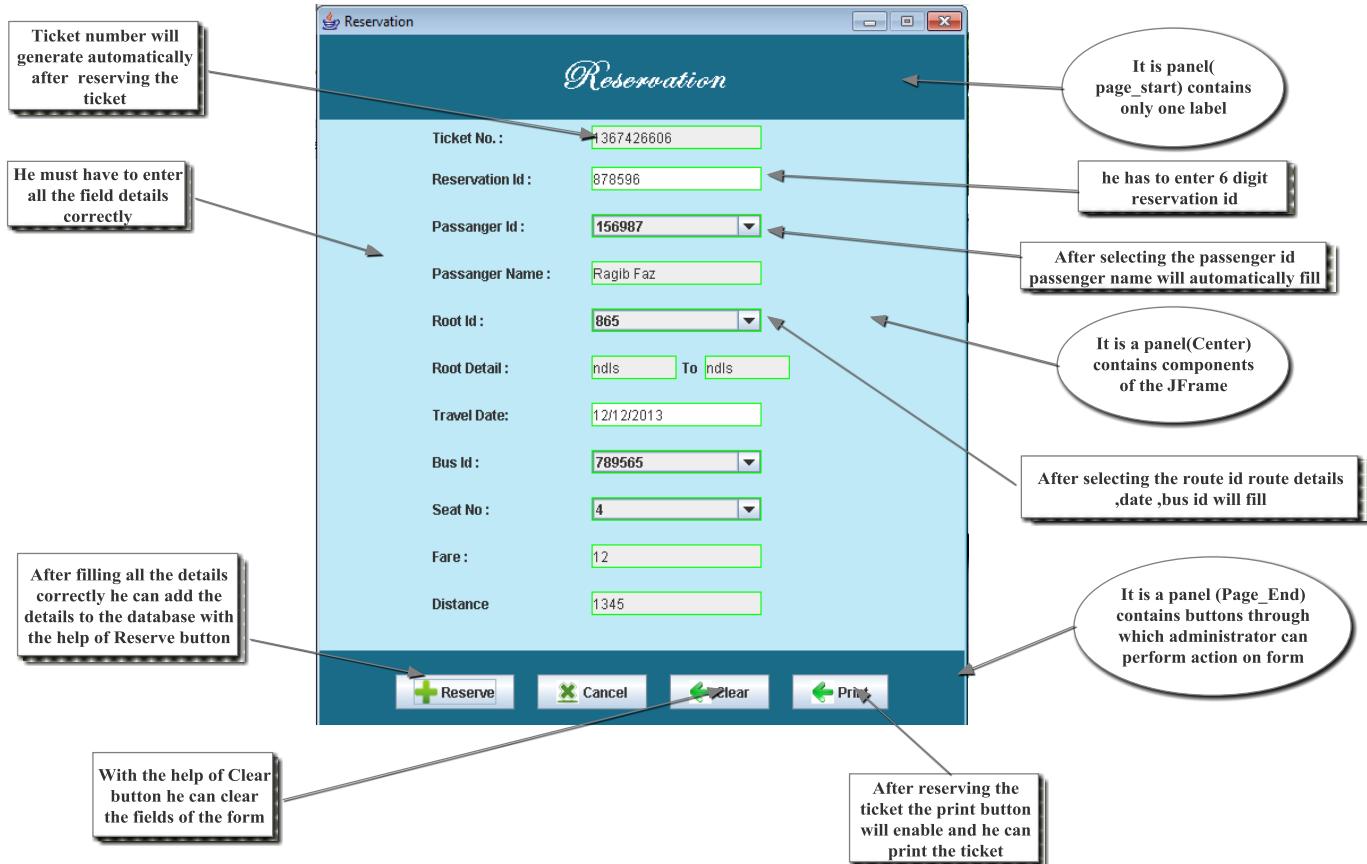
7.4 Route 1:



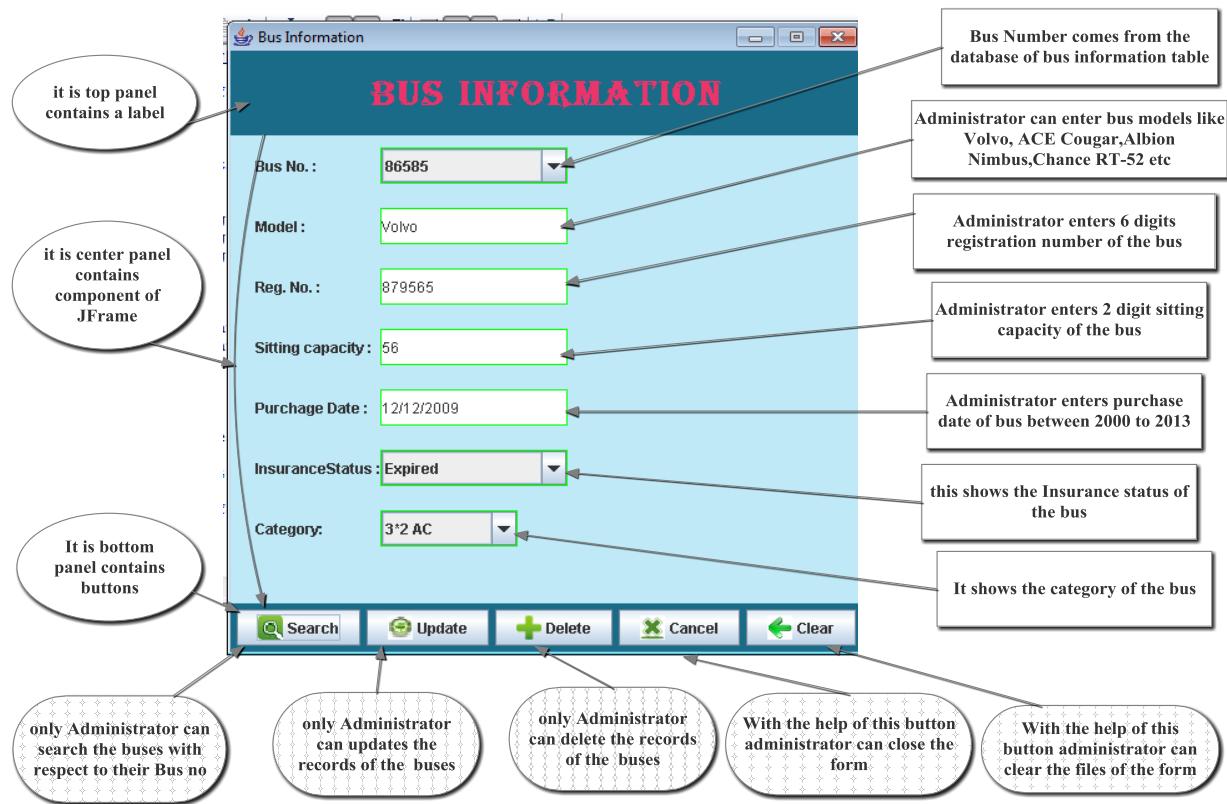
7.5 Scheduling 1:



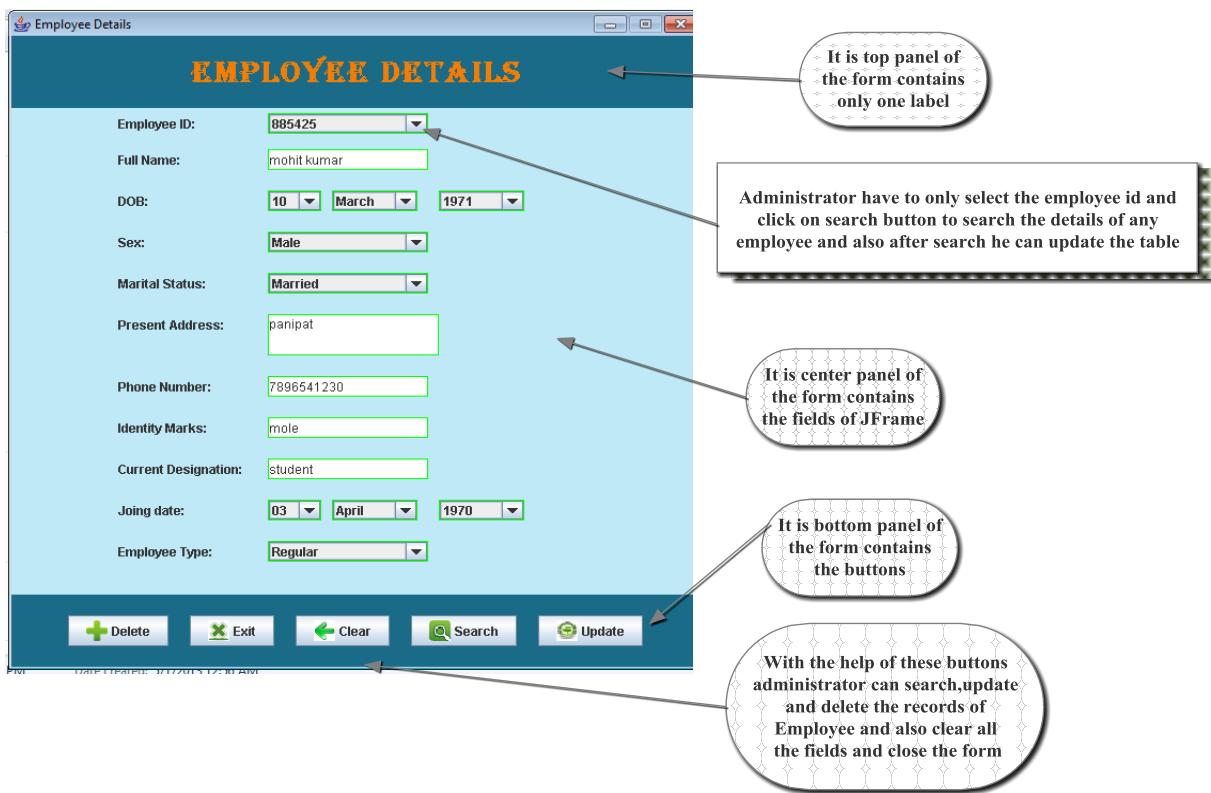
7.6 Reservation:



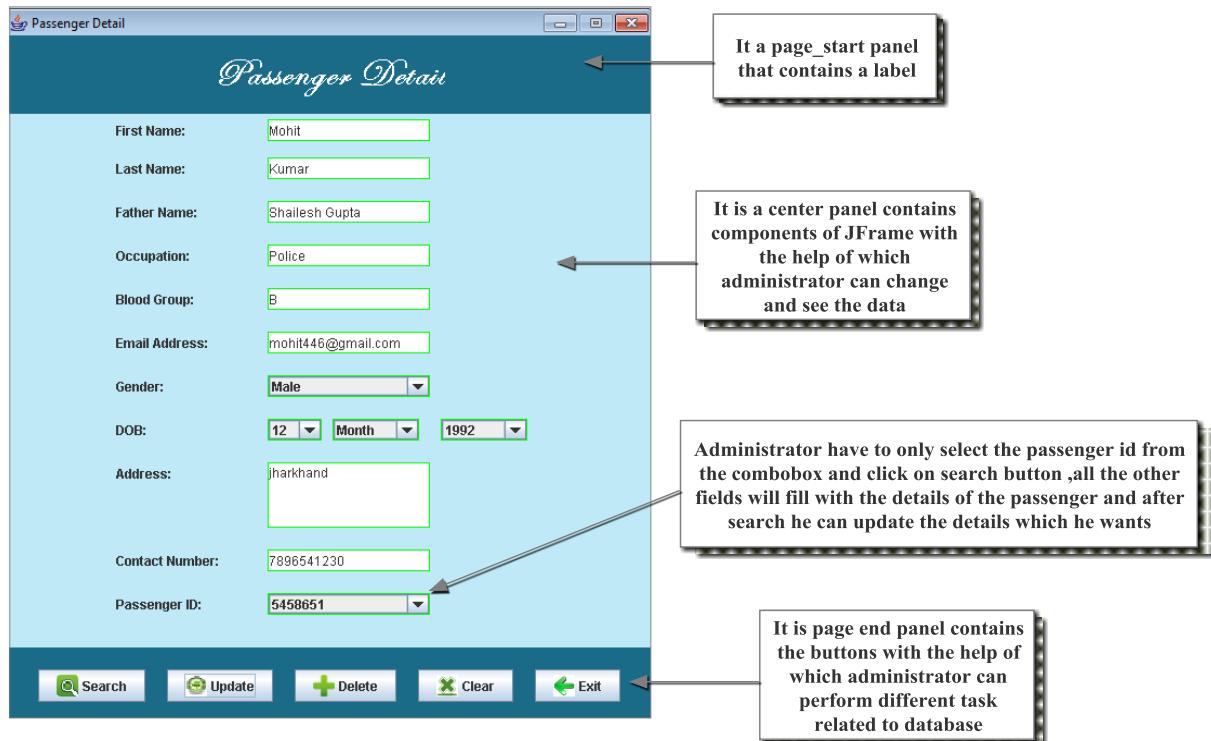
7.7 Bus module 2



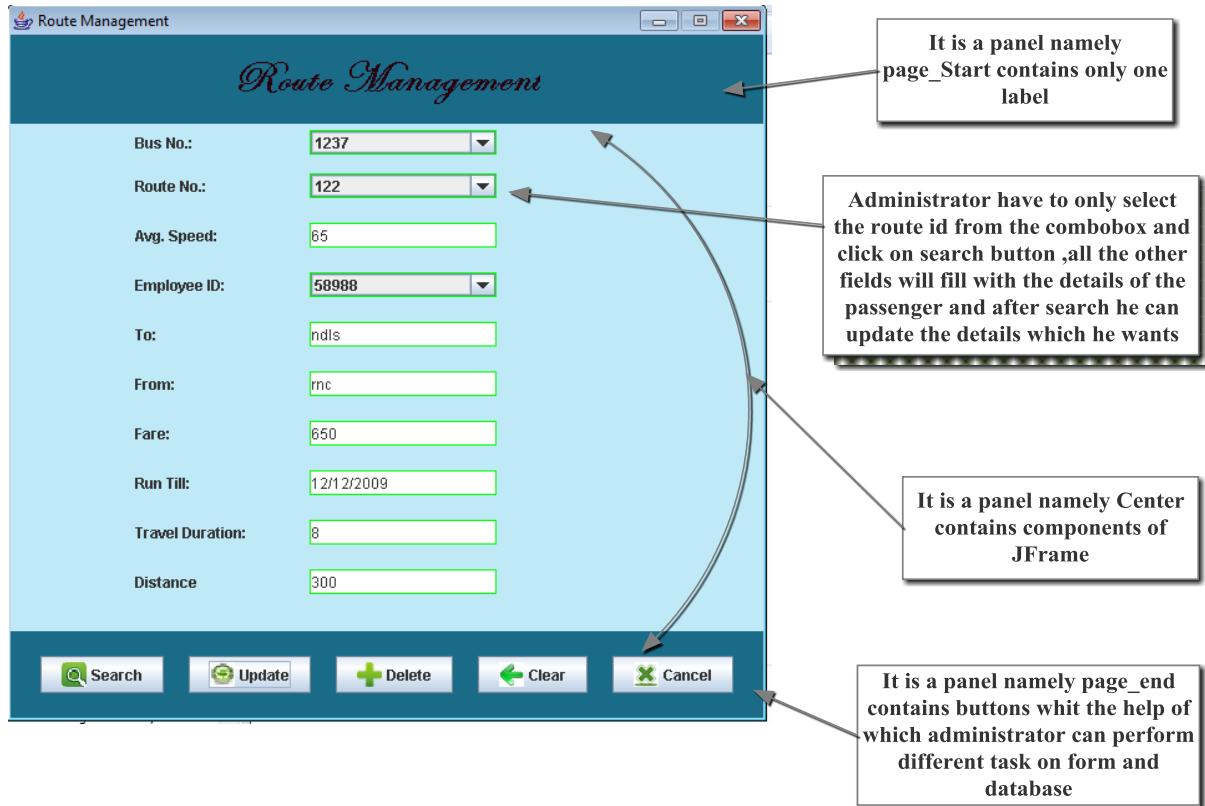
7.8 Employee module 2:



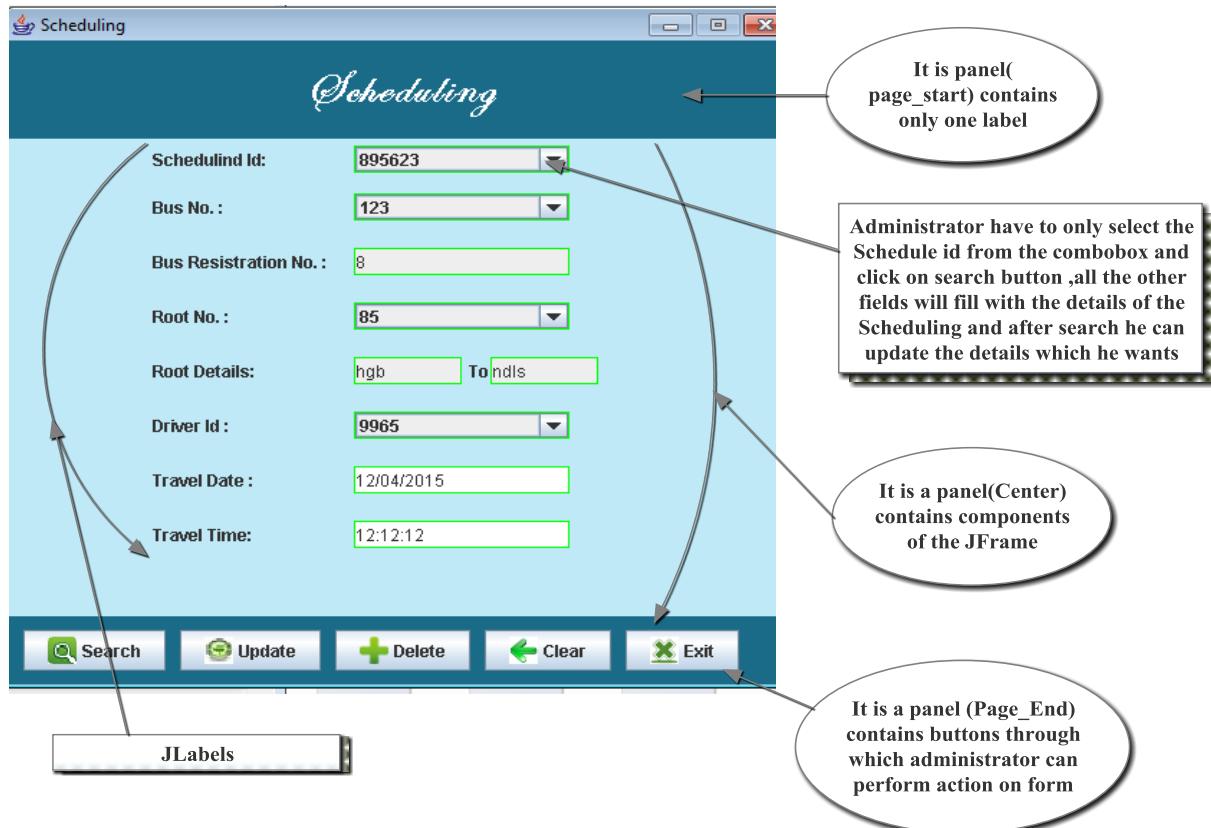
7.9. Passenger module 2:



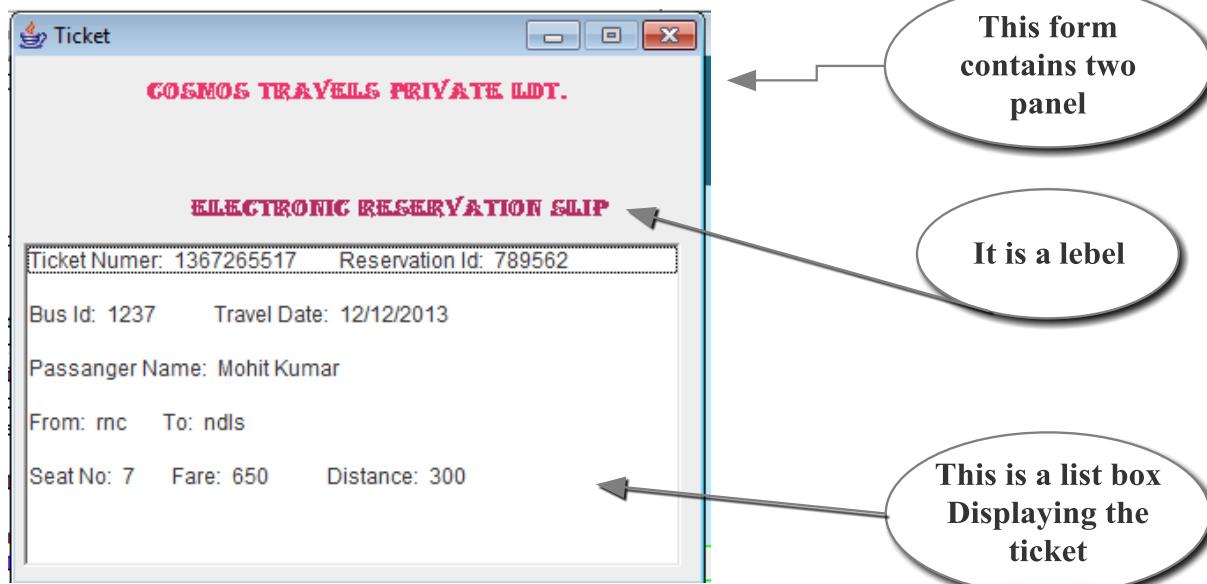
7.9 Route module 2:



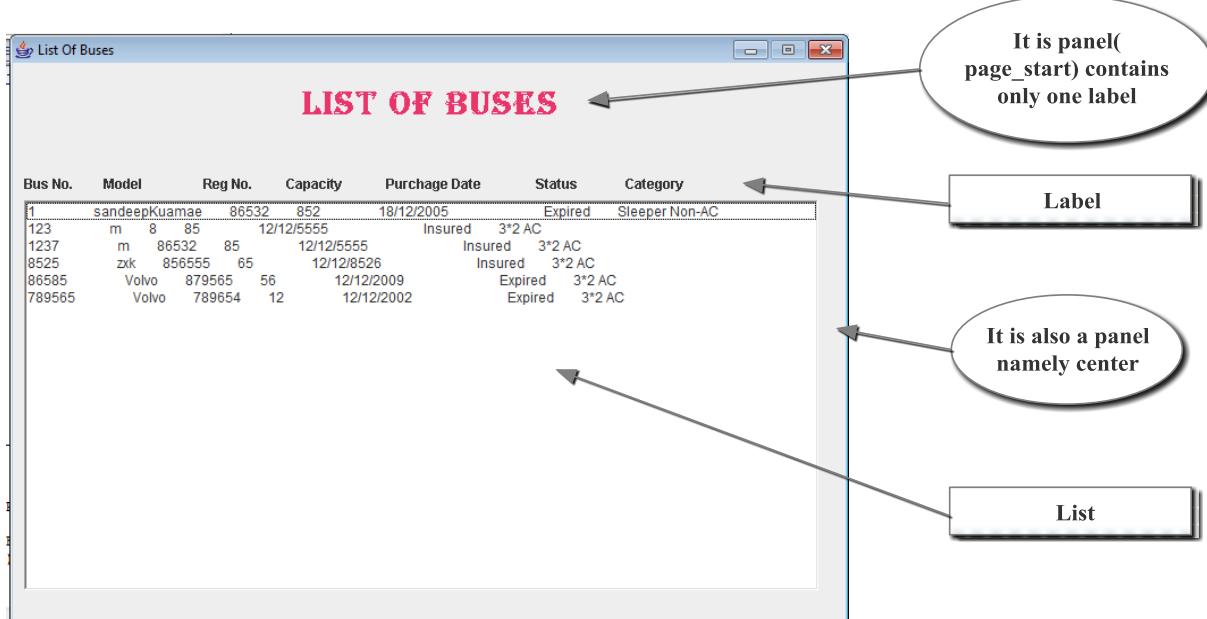
7.10 Schedule module 2:



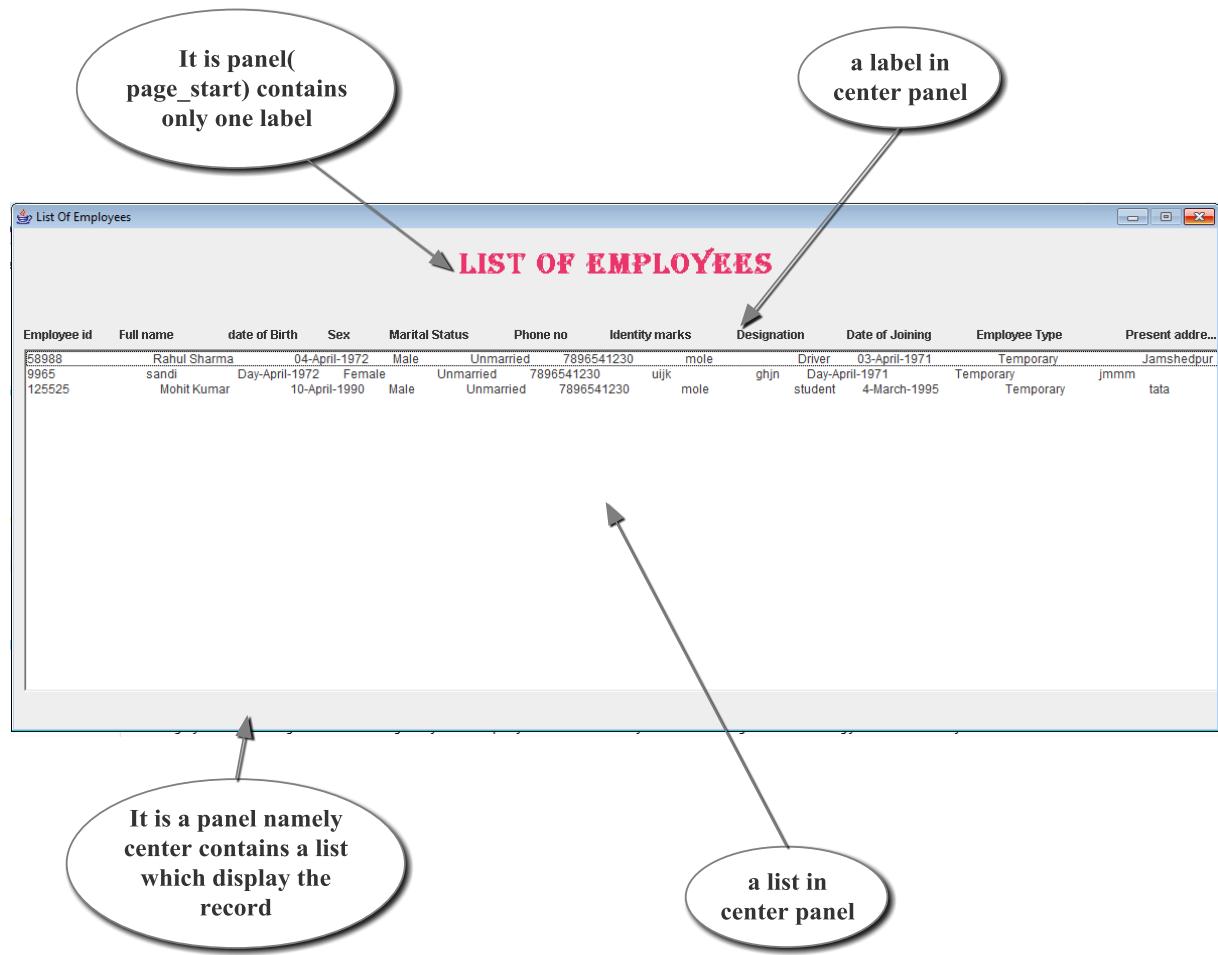
7.11 Ticket Print:



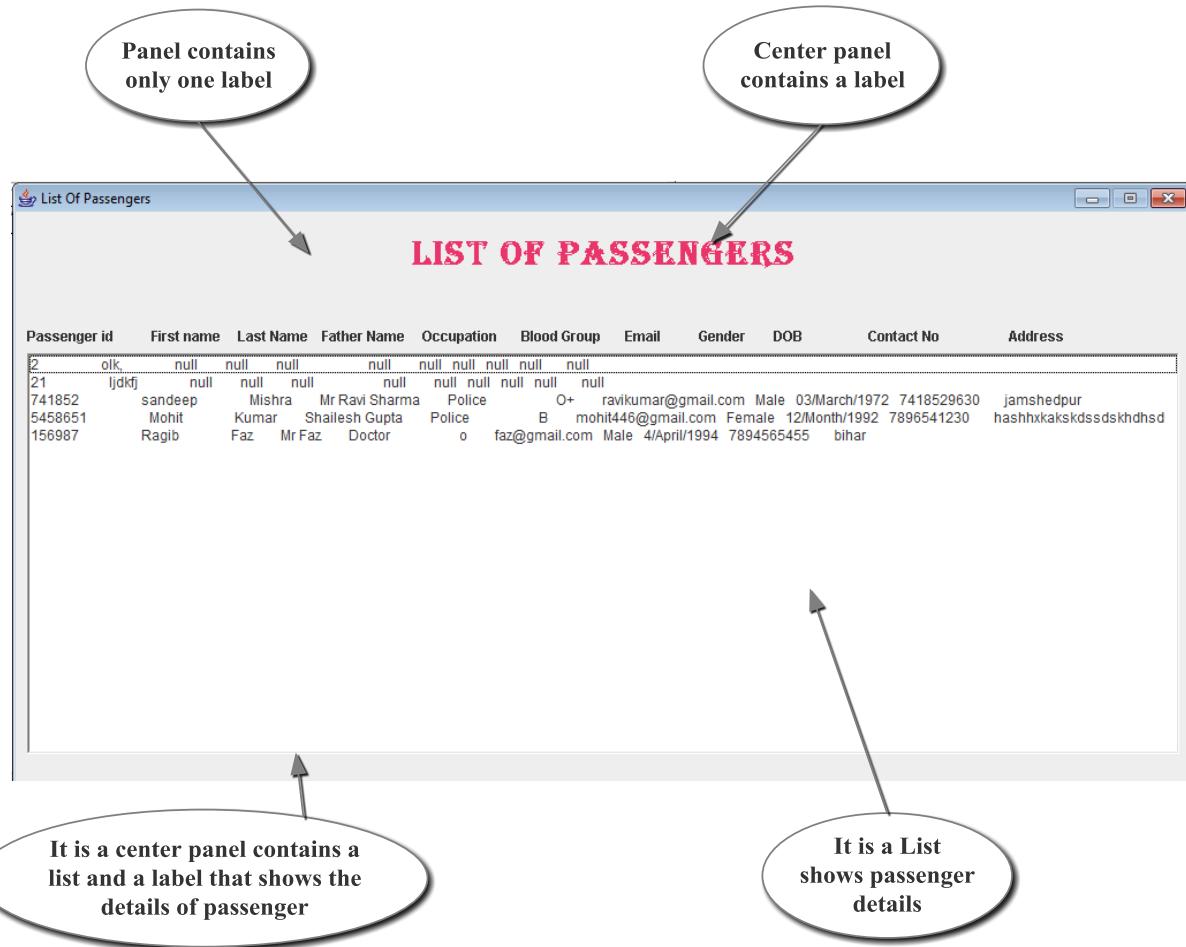
7.12 BusList:



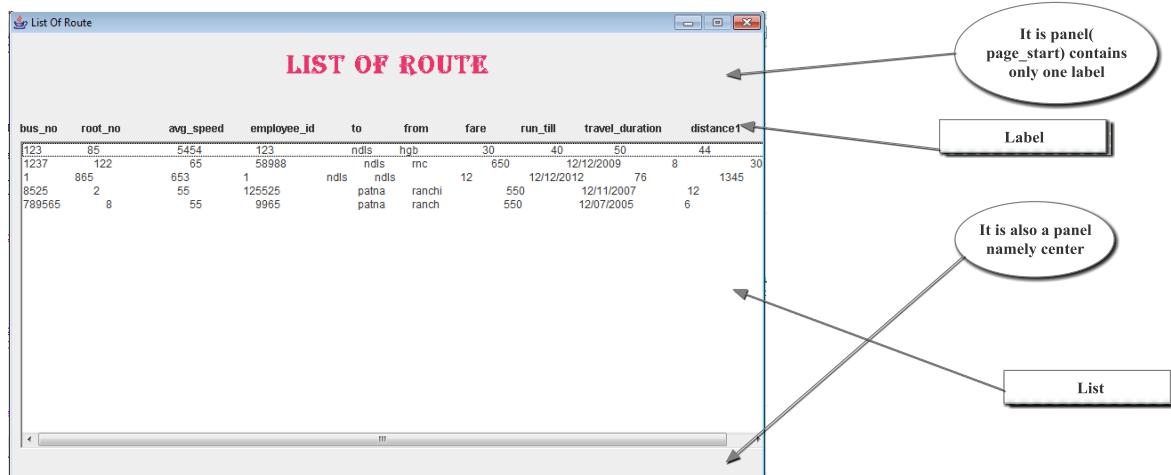
7.13 Employee List:



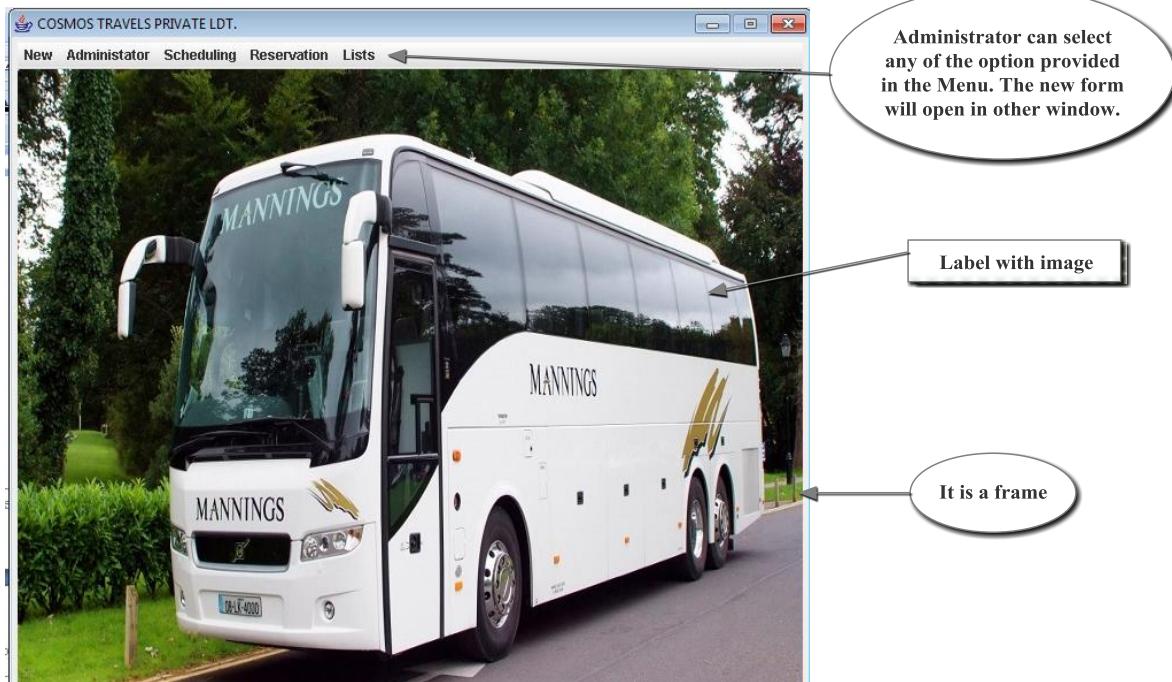
7.14 Passenger List:



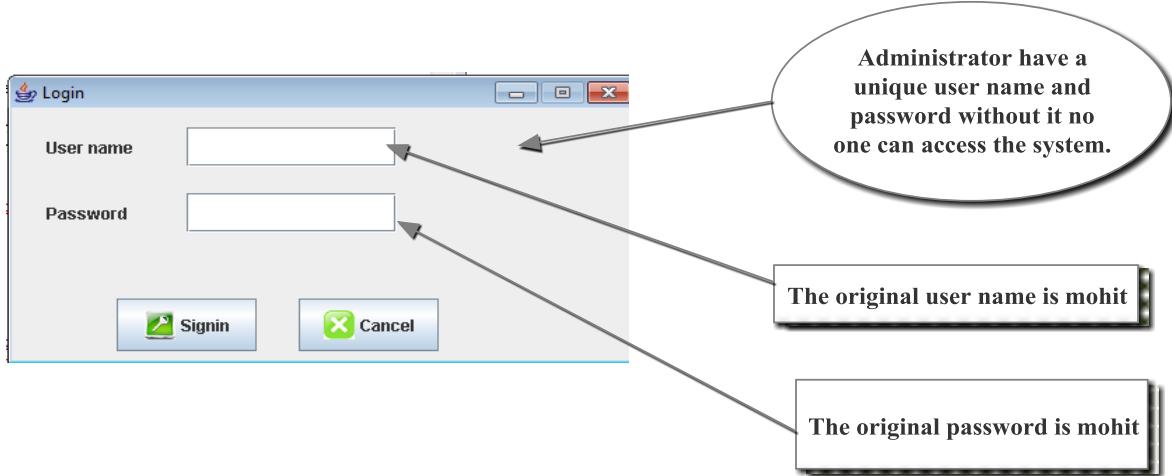
7.15. Route List:



7.16 Default(Menu):



7.17 Password:



8 Assumptions:

- ❖ It is assumed that operator should add the bus details firstly after that he should add the employee details. After this he can able to work with route detail interface because without bus number and employee operator cannot set the route of the bus.
- ❖ For scheduling also operator has to details of bus, route. Without it he can not schedule the bus of specific route.
- ❖ After adding all the above details, the operator has to only fill or add the details of passenger to the passenger module and after this he will able to reserve a ticket for a the specific passenger.
- ❖ It is assumed that all the ids of passenger, route, and employee are of 6 digit number.
- ❖ Bus number should have only 4 digits.
- ❖ The system has only one administrator.
- ❖ The mobile number must be of 10 digits.

9 Additional Features:

- ❖ Through this project developer able to learn about core java concepts.
- ❖ This project has certainly helped the developer to learn a lot of things about java programming language
- ❖ During the project developer were busy to study about the four basic concept of java and other basic concepts namely encapsulation, polymorphism, inheritance and data handling.
- ❖ He can learn how to handle the run time exception.
- ❖ He learnt how to validate different fields data with the help of loop and string methods.
- ❖ Through this project developer came to know about how to create a form with the help of Jframe in java.
- ❖ Developer came to know about different types of classes in java like abstract class and interface.
- ❖ He also learns how to parse the string value to integer value.
- ❖ He is able to learn database management system and how to connect java programme to database through Jdbcodbc driver.
- ❖ He came to know the different query statements about database.

10 References

- ❖ Balagurusamy, E. (2010) Programming with Java. New Delhi: Tata McGraw-Hill Education.
 - ❖ Burd, B. (2007) Java for dummies. Hoboken, NJ: Wiley Publishing Co..
 - ❖ DaniWeb (2006) Check a string is numeric or not. [online] Available at: <http://www.daniweb.com/software-development/java/threads/131256/check-a-string-is-numeric-or-not> [Accessed: 30 march 2013].
 - ❖ Docs.oracle.com (1995) Listeners Supported by Swing Components (The Java™ Tutorials > Creating a GUI With JFC/Swing > Writing Event Listeners). [online] Available at: <http://docs.oracle.com/javase/tutorial/uiswing/events/eventsandcomponents.html> [Accessed: 10 March 2013].
 - ❖ Docs.oracle.com (1995) How to Write an Action Listener (The Java™ Tutorials > Creating a GUI With JFC/Swing > Writing Event Listeners). [online] Available at: <http://docs.oracle.com/javase/tutorial/uiswing/events/actionlistener.html> [Accessed: 16 march 2013].
 - ❖ Farrell, J. (2003) Java programming. London: Course Technology.
- Holzner, S. (2001) Java 2 black book. Scottsdale, Ariz.: Coriolis Group Books.
- ❖ Javatutoronline.com (2011) Java Exception Handling Tutorial With Examples and Code. [online] Available at: <http://www.javatutoronline.com/Java-Exception-Handling-Tutorial.jsp> [Accessed: 15 April 2013].
 - ❖ Russell, J. (2001) Java programming for the absolute beginner. Roseville, Calif.: Prima Tech.
 - ❖ Tutorialspoint.com (2013) Java - Encapsulation. [online] Available at: http://www.tutorialspoint.com/java/java_encapsulation.htm [Accessed: 8 March 2013].
 - ❖ Tutorialspoint.com (2013) Java - Polymorphism. [online] Available at: http://www.tutorialspoint.com/java/java_polymorphism.htm [Accessed: 25 march 2013].

11 Appendix:

11.1 Java Doc:

The screenshot shows a Java documentation interface in a Firefox browser. The title bar says "Cite This For Me - Cite a Website in H... Overview". The main content area is titled "Packages" and lists several packages: Booking, Bus, Employee, Lists, Passenger, and Route. On the left sidebar, there are sections for "All Classes" and "Packages", with "Booking" selected. The bottom of the page has a navigation bar with links for Overview, Package, Class, Tree, Deprecated, Index, and Help, along with "FRAMES" and "NO FRAMES" options.

11.2 Booking:

The screenshot shows a Java documentation interface in a Firefox browser. The title bar says "Reservation Overview". The main content area is titled "Class Reservation" and shows the class hierarchy: java.lang.Object <--> Booking.Reservation. It lists implemented interfaces: java.awt.event.ActionListener, java.util.EventListener. The class definition is provided:

```
public class Reservation  
extends java.lang.Object  
implements java.awt.event.ActionListener
```

. A note states: "Through this class Administrator/user can Reserve the ticket.This class is implementing the ActionListener class". Below this are sections for "Field Summary" (with a single field: reserid), "Constructor Summary" (with a constructor: Reservation() described as "used to initilise the fields of the form"), and "Method Summary" (with a method: actionPerformed(java.awt.event.ActionEvent e) described as "implmenting the abstract method of ActionListener class"). The left sidebar shows "All Classes" and "Packages", with "Reservation" selected. The bottom of the page has a navigation bar with links for Overview, Package, Class, Tree, Deprecated, Index, and Help, along with "FRAMES" and "NO FRAMES" options.

11.3 Bus:

The screenshot shows a Java class documentation page for `AdminBusInfo`. The browser title is "AdminBusInfo". The left sidebar lists packages and classes under the "Bus" package, with `AdminBusInfo` selected. The main content area shows the class hierarchy: `java.lang.Object` → `Bus.AdminBusInfo`. It lists implemented interfaces: `java.awt.event.ActionListener, java.util.EventListener`. The code snippet shows the class definition extending `java.lang.Object`. A note indicates overriding the `Bus` method. The "Field Summary" section contains six fields: `bottom`, `btnCancel`, `btnClear`, `btnDelete`, `btnSearch`, and `btnUpdate`, all of type `javax.swing.JButton`.

11.4 Employee:

The screenshot shows a Java class documentation page for `AdminEmployeeDetails`. The browser title is "AdminEmployeeDetails". The left sidebar lists packages and classes under the "Employee" package, with `AdminEmployeeDetails` selected. The main content area shows the class hierarchy: `java.lang.Object` → `Employee.AdminEmployeeDetails`. It lists implemented interfaces: `java.awt.event.ActionListener, java.util.EventListener`. The code snippet shows the class definition extending `java.lang.Object` and implementing `java.awt.event.ActionListener`. A note states that the `method(launchFrame())` of class `AbstractClass` allows administrators to search, update, delete records, cancel operations, and clear form fields. The "Constructor Summary" section shows the constructor `AdminEmployeeDetails()` with a note: "use to initilise the components". The "Method Summary" section lists three methods: `actionPerformed(java.awt.event.ActionEvent e)` (implements `ActionListener`), `comboBox()` (used to get employee id from database), and `delete(int i)` (Delete Records with respect to given employee id).

11.5 List:

The screenshot shows a Java class documentation page for `PassengerList`. The browser title is "PassengerList". The left sidebar lists packages and classes under "Lists": `All Classes`, `Packages <unnamed package>`, `Booking`, `Bus`, `Employee`, `Lists`, `Passenger`, `Route`, `Lists`, `Classes`, `BusList`, `EmployeeList`, `PassengerList`, `RouteList`, and `class in Lists`. The main content area has tabs for Overview, Package, Class (selected), Tree, Deprecated, Index, and Help. Under the Class tab, it shows the inheritance path: `java.lang.Object` → `Lists.PassengerList`. The class definition is:

```
public class PassengerList extends java.lang.Object
```

 A note below states: "Fetch all the data from the passenger database and display it to screen". The Constructor Summary section contains one constructor: `PassengerList()` which initializes the components of the frame. The Method Summary section contains one method: `DisplayPassengerArea()` which adds the components to the frame. The Methods inherited from class `java.lang.Object` section lists `clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, and `wait`. The Constructor Detail section is empty.

11.6 Passenger:

The screenshot shows a Java class documentation page for `AdminPassengerDetails`. The browser title is "AdminPassengerDetails". The left sidebar lists packages and classes under "Passenger": `All Classes`, `Packages <unnamed package>`, `Booking`, `Bus`, `Employee`, `Lists`, `Passenger`, `Route`, `Passenger`, `Classes`, `AdminPassengerDetails`, and `PassengerDetails`. The main content area has tabs for Overview, Package, Class (selected), Tree, Deprecated, Index, and Help. Under the Class tab, it shows the inheritance path: `java.lang.Object` → `Passenger.AdminPassengerDetails`. The class definition is:

```
public class AdminPassengerDetails extends java.lang.Object
```

 It implements `java.awt.event.ActionListener` and `java.util.EventListener`. A note below states: "Through this class administrator can update, search and delete the records of passengers and cancel the operation. this class is implementing the interface namely Admin method". The Constructor Summary section contains one constructor: `AdminPassengerDetails()` which is used to initialize the fields of the form. The Method Summary section contains three methods: `actionPerformed(java.awt.event.ActionEvent e)` which implements the abstract method of `ActionListener` class, `clear()` which is used to clear the fields of the form, and `comboBox()` which gets the value from the passenger details table and insert it to the combobox. The status bar at the bottom says "Documentation.docx - Microsoft Word".

11.7 Route:

The screenshot shows a Java class documentation page for `AdminRouteManagement`. The browser title is "AdminRouteManagement". The left sidebar lists packages: `<unnamed package>`, `Booking`, `Bus`, `Employee`, `Lists`, `Passenger`, and `Route`. Below these are sections for `Route`, `Classes`, `AdminRouteManagement`, and `RouteManagement`.

Class AdminRouteManagement

All Implemented Interfaces:

- `java.awt.event.ActionListener, java.util.EventListener`

Code Snippet:

```
public class AdminRouteManagement
extends java.lang.Object
implements java.awt.event.ActionListener
```

Through this class administrator can update,search and delete the records of Route and cancel the operation.

Constructor Summary

`AdminRouteManagement()`
used to initialise the fields of the form

Method Summary

void	<code>actionPerformed(java.awt.event.ActionEvent e)</code> implimenting the abstract method of ActionListener class with the help of which admin can perform his task
void	<code>clear()</code> Used to clear the fields of the form
void	<code>checkboxvalue()</code> get value of combobox from the Businformation,EmployeeDetails,routemanagement tables with respect to their id
void	<code>delete(int i)</code> Delete record of the Route from the table

11.8 Scheduling:

The screenshot shows a Java class documentation page for `AdminScheduling`. The browser title is "AdminScheduling". The left sidebar lists packages: `<unnamed package>`, `Booking`, `Bus`, `Employee`, `Lists`, `Passenger`, `Route`, and `Schedule`. Below these are sections for `Schedule`, `Classes`, `AdminScheduling`, and `Scheduling`.

Class AdminScheduling

All Implemented Interfaces:

- `java.awt.event.ActionListener, java.util.EventListener`

Code Snippet:

```
public class AdminScheduling
extends java.lang.Object
implements java.awt.event.ActionListener
```

Through this class administrator can update,search and delete the records of Scheduling and cancel the operation.

Constructor Summary

`AdminScheduling()`
used to initialise the fields of the form

Method Summary

void	<code>actionPerformed(java.awt.event.ActionEvent e)</code> implimenting the abstract method of ActionListener class
void	<code>checkbox_value()</code> getting values from scheduling,Businformation,routemanagement tables and set it to their respective combo box
void	<code>delete(int i)</code> Delete record of the Scheduling from the table