31172_SDL_Assignment_4:

Client Side:

```
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.plaf.synth.ColorType;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.net.Socket;
import java.util.*;

public class Client {
    public static void main(String args[])
    {
        String serveraddress = "127.0.0.1";
        int port =8081;
        String user_type = "N";
        Scanner sc=new Scanner(System.in);
        String choice;
        Socket client = null;
        try {
            client = new Socket(serveraddress,port);

            System.out.println("Connected to: "+client.getRemoteSocketAddress());
            DataOutputStream outToServer = new
DataOutputStream(client.getOutputStream());
            DataInputStream inFromServer = new
DataInputStream(client.getInputStream());
            ObjectOutputStream os=new
ObjectOutputStream(client.getOutputStream());
            ObjectInputStream is=new ObjectInputStream(client.getInputStream());
            Boolean authenticated=false;
            String usern="";
            Frame1 l =new Frame1();
            login:
            while(!authenticated){
                while(l.isVisible()){
                    System.out.println("");
                }
                System.out.println(l.opt);
                switch(l.opt){
                    case "Signup":{
                        outToServer.writeUTF("2");
                        Signup l1= new Signup();
                        while(l1.isActive()){}
                        outToServer.writeUTF(l1.name);
                        outToServer.writeUTF(l1.username);
                        outToServer.writeUTF(l1.password);
                        outToServer.writeUTF(l1.contact);
                        outToServer.writeInt(l1.flat_no);
                        outToServer.writeUTF(l1.user_type);
                        String res = inFromServer.readUTF();
                        System.out.println(res);
                        MessageBox m = new MessageBox(res);
                        break;
```

```
                }
                case "Login":{
                    System.out.println("In login");
                    outToServer.writeUTF("1");
                    Login l2=new Login();
                    while(l2.isActive()){
                        System.out.println("");
                    }
                    outToServer.writeUTF(l2.username);
                    outToServer.writeUTF(l2.password);
                    String res = inFromServer.readUTF();
                    System.out.println(res);
                    MessageBox m = new MessageBox(res);
                    usern = inFromServer.readUTF();
                    authenticated = inFromServer.readBoolean();
                    user_type = inFromServer.readUTF();
                    break;
                }
            }

        }
        if(user_type.equals("N")){
            System.out.println("In customer side");
            CustomerOption a = new CustomerOption();
            while(true) {
                while (a.f.isVisible()) {
                    System.out.println("");
                }
                System.out.println(a.choice);
                if(a.choice == 5)
                {
                    System.exit(0);
                    break;
                }
                switch (a.choice) {
                    case 1: {
                        outToServer.writeUTF("Book Tickets");
                        Book b = new Book();
                            while (b.f.isVisible()) {
                                System.out.println("");
                            }
                            outToServer.writeUTF(b.city_name);
                            outToServer.writeByte(b.no_of_passengers);
                            os.writeObject(b.passenger);
                            String res = inFromServer.readUTF();
                            MessageBox m = new MessageBox(res);
                            a.f.setVisible(true);
                            break;
                    }
                    case 2:{
                        outToServer.writeUTF("Cancel Tickets");
                        BookingId c = new BookingId();
                            while(c.f.isVisible()){
                                System.out.println("");
                            }
                            outToServer.writeInt(c.booking_id);
                            //outToServer.writeInt(c.bus_id);
                            String res = inFromServer.readUTF();
                            MessageBox m = new MessageBox(res);
```

```
                                //at.dispose();
                                a.f.setVisible(true);
                                break;
                            }
                            case 3:{
                                outToServer.writeUTF("Check Reservation");
                                BookingId c = new BookingId();
                                while(c.f.isVisible()){
                                    System.out.println("");
                                }
                                outToServer.writeInt(c.booking_id);
                                //outToServer.writeInt(c.bus_id);
                                String res = inFromServer.readUTF();
                                MessageBox m = new MessageBox(res);
                                //at.dispose();
                                a.f.setVisible(true);
                                break;
                            }
                            case 4:{
                                outToServer.writeUTF("View Bookings");
                                String bookings="<html>";
                                while(true){
                                    String s = inFromServer.readUTF();
                                    if(s.equals("dis")){
                                        break;
                                    }
                                    int pass_id = inFromServer.readInt();
                                    int booking_id = inFromServer.readInt();
                                    int bus_id = inFromServer.readInt();
                                    String name = inFromServer.readUTF();
                                    int seat_no = inFromServer.readInt();
                                    bookings+="<br>Passenger Id:
"+pass_id+"<br>"+"Booking Id: "+booking_id+"<br>"+"Bus Id: "+bus_id+"<br>Passenger
Name: "+name+"<br>Seat Number: "+seat_no+"<br>";
                                }
                                ViewBookings p = new ViewBookings(bookings);
                                while(p.f.isVisible()){
                                    System.out.println("");
                                }
                                a.f.setVisible(true);
                                break;
                            }
                        }
                    }
                } else {
                    System.out.println("In customer side");
                    //HashMap<String,Item> stk = new HashMap<String, Item>();
                    AdminOption a = new AdminOption();
                    while(true){
                        while (a.f.isVisible()) {
                            System.out.println("");
                        }
                        System.out.println(a.choice);
                        if(a.choice == 6){
                            System.exit(0);
                            break;
                        }
                        switch (a.choice){
                            case 1:{
```

```
                    outToServer.writeUTF("Add City");
                    City c1 = new City(outToServer, inFromServer);
                    while( c1.f.isVisible()){
                        System.out.println("");

                    }
                    outToServer.writeUTF( c1.city_name);
                    //String re = inFromServer.readUTF();
                    String res= inFromServer.readUTF();
                    MessageBox b = new MessageBox(res);

                    a.f.setVisible(true);
                    break;
                }
                case 2:{
                    outToServer.writeUTF("Add Bus");
                    City c2 = new City(outToServer,inFromServer);
                    while(c2.f.isVisible()){
                        System.out.println("");
                    }
                    outToServer.writeUTF(c2.city_name);
                    //String re = inFromServer.readUTF();
                    String res= inFromServer.readUTF();
                    MessageBox b = new MessageBox(res);

                    a.f.setVisible(true);
                    break;
                }
                case 3:{
                    outToServer.writeUTF("Delete City");
                    DeleteCity c3 = new DeleteCity();
                    while(c3.f.isVisible()){
                        System.out.println("");
                    }
                    outToServer.writeUTF(c3.city_name);
                    String re = inFromServer.readUTF();
                    MessageBox b=  new MessageBox(re);
                    a.f.setVisible(true);
                    break;
                }
                case 4:{
                    outToServer.writeUTF("Cancel Bus");
                    DeleteBus db = new DeleteBus();
                    while(db.f.isVisible()){
                        System.out.println("");
                    }
                    outToServer.writeInt(db.bus_id);
                    String re = inFromServer.readUTF();
                    MessageBox b=  new MessageBox(re);
                    a.f.setVisible(true);
                    break;
                }
                case 5:{
                    outToServer.writeUTF("View Bookings");
                    String bookings="<html>";
                    while(true){
                        String s = inFromServer.readUTF();
                        if(s.equals("dis")){
                            break;
```

```
                                }
                                int pass_id = inFromServer.readInt();
                                int booking_id = inFromServer.readInt();
                                int bus_id = inFromServer.readInt();
                                String name = inFromServer.readUTF();
                                int seat_no = inFromServer.readInt();
                                bookings+="<br>Passenger Id:
"+pass_id+"<br>"+"Booking Id: "+booking_id+"<br>"+"Bus Id: "+bus_id+"<br>Passenger
Name: "+name+"<br>Seat Number: "+seat_no+"<br>";
                            }
                            ViewBookings p = new ViewBookings(bookings);
                            while(p.f.isVisible()){
                                System.out.println("");
                            }
                            a.f.setVisible(true);
                            break;
                        }
                    }
                }
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

class CustomerOption extends JFrame implements ActionListener{
    static JFrame f;
    JMenu menu;
    JMenuItem i1;
    static JButton b, b1, b2, b3,b4;

    public int choice;
    String opt;
    public CustomerOption(){
        f = new JFrame("-------Menu--------");

        JPanel p = new JPanel();

        p.setLayout(new FlowLayout(FlowLayout.LEFT,20,10));

        b = new JButton( new AbstractAction("Book a ticket") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                choice=1;
                System.out.println(choice);
                f.setVisible(false);
                System.out.println("Visiblity");
                f.dispose();
            }
        });

        b1 = new JButton( new AbstractAction("Cancel Tickets!") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                System.out.println(opt);
```

```
            choice=2;
            f.setVisible(false);
            f.dispose();
        }
    });
    b2  = new JButton(new AbstractAction("Check Reservation!") {
        @Override
        public void actionPerformed(ActionEvent e) {
            opt = e.getActionCommand();
            System.out.println(opt);
            choice=3;
            f.setVisible(false);
            f.dispose();
        }
    });
    b3 = new JButton( new AbstractAction("My Bookings") {
        @Override
        public void actionPerformed( ActionEvent e ) {
            opt = e.getActionCommand();
            System.out.println(opt);
            choice=4;
            f.setVisible(false);
            f.dispose();
        }
    });

    b4 = new JButton(new AbstractAction("LogOut") {
        @Override
        public void actionPerformed(ActionEvent e) {
            opt = e.getActionCommand();
            System.out.println(opt);
            choice = 5;
            f.setVisible(false);
            f.dispose();
        }
    });

    b.setAlignmentX(Component.CENTER_ALIGNMENT);
    b.setAlignmentY(Component.CENTER_ALIGNMENT);
    b1.setAlignmentX(Component.CENTER_ALIGNMENT);
    b1.setAlignmentY(Component.CENTER_ALIGNMENT);
    b2.setAlignmentX(Component.CENTER_ALIGNMENT);
    b2.setAlignmentY(Component.CENTER_ALIGNMENT);
    b3.setAlignmentX(Component.CENTER_ALIGNMENT);
    b3.setAlignmentY(Component.CENTER_ALIGNMENT);
    b4.setAlignmentX(Component.CENTER_ALIGNMENT);
    b4.setAlignmentY(Component.CENTER_ALIGNMENT);

    JMenuBar mb = new JMenuBar();
    menu = new JMenu("Menu");
    i1 = new JMenuItem("Exit");

    menu.add(i1);
    mb.add(menu);
    f.setJMenuBar(mb);

    p.add(b);
    p.add(b1);
    p.add(b2);
```

```
        p.add(b3);
        p.add(b4);
        p.setBackground(Color.yellow);

        i1.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ae){
                choice=6;
                f.dispose();
            }
        });
        f.add(p);

        f.setSize(300, 300);

        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }

    public void actionPerformed(ActionEvent e) {

    }
}

class AdminOption extends JFrame implements ActionListener{
    static JFrame f;
    JMenu menu,submenu;
    JMenuItem i1;

    static JButton b, b1, b2, b3,b4,b5;

    static JPanel p;
    public int choice;
    String opt;
    public AdminOption(){
        f = new JFrame("-------Menu--------");

        p = new JPanel();

        p.setLayout(new FlowLayout(FlowLayout.LEFT,20,10));

        b = new JButton( new AbstractAction("Add City!") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                choice=1;
                System.out.println(choice);
                f.setVisible(false);
                System.out.println("Visiblity");
                f.dispose();
            }
        });

        b1 = new JButton( new AbstractAction("Add Bus") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                System.out.println(opt);
                choice=2;
```

```
                f.setVisible(false);
                f.dispose();
            }
        });
        b2 = new JButton( new AbstractAction("Delete City!") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                System.out.println(opt);
                choice=3;
                f.setVisible(false);
                f.dispose();
            }
        });
        b3 = new JButton( new AbstractAction("Delete Bus") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                System.out.println(opt);
                choice=4;
                f.setVisible(false);
                f.dispose();
            }
        });
        b4 = new JButton( new AbstractAction("View Bookings!") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                opt = e.getActionCommand();
                System.out.println(opt);
                choice=5;
                f.setVisible(false);
                f.dispose();
            }
        });

        b5 = new JButton(new AbstractAction("LogOut") {
            @Override
            public void actionPerformed(ActionEvent e) {
                choice = 6;
                f.setVisible(false);
                f.dispose();
            }
        });
        b.setAlignmentX(Component.CENTER_ALIGNMENT);
        b.setAlignmentY(Component.CENTER_ALIGNMENT);
        b1.setAlignmentX(Component.CENTER_ALIGNMENT);
        b1.setAlignmentY(Component.CENTER_ALIGNMENT);
        b2.setAlignmentX(Component.CENTER_ALIGNMENT);
        b2.setAlignmentY(Component.CENTER_ALIGNMENT);
        b3.setAlignmentX(Component.CENTER_ALIGNMENT);
        b3.setAlignmentY(Component.CENTER_ALIGNMENT);
        b4.setAlignmentX(Component.CENTER_ALIGNMENT);
        b4.setAlignmentY(Component.CENTER_ALIGNMENT);
        b5.setAlignmentX(Component.CENTER_ALIGNMENT);
        b5.setAlignmentY(Component.CENTER_ALIGNMENT);

        JMenuBar mb = new JMenuBar();  //Menu Bar for exiting.
        menu = new JMenu("Menu");
        i1 = new JMenuItem("Exit");
```

```
        menu.add(i1);
        mb.add(menu);
        f.setJMenuBar(mb);

        p.add(b);
        p.add(b1);
        p.add(b2);
        p.add(b3);
        p.add(b4);
        p.add(b5);
        p.setBackground(Color.yellow);
        i1.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                choice=6;
                f.dispose();
            }
        });
        f.add(p);

        f.setSize(300, 300);

        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }

    public void actionPerformed(ActionEvent e) {

    }
}

class Book extends JFrame implements ActionListener{
    static  JFrame f;
    static JLabel spacer;
    static JPanel p;
    public String city_name;
    public int no_of_passengers;
    HashMap <String,Integer> passenger = new HashMap<>();
    public Book()
    {
        f = new JFrame();
        p = new JPanel();
        f.setLayout(new FlowLayout());
        JLabel l1 = new JLabel("City Name");
        JTextField t1 = new JTextField(20);

        JLabel l2 = new JLabel("Number of Passengers:");
        JTextField t2 = new JTextField(3);

        JLabel l3 = new JLabel("Name of passengers(use , between them");
        JTextField t3 = new JTextField(20);

        JLabel l4 = new JLabel("Seat Numbers(use , between them)");
        JTextField t4 = new JTextField(10);

        JButton b1 = new JButton(new AbstractAction("Book") {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                city_name = t1.getText();
                no_of_passengers = Integer.parseInt(t2.getText());
                String[] names = t3.getText().split(",");
                String[] seats = t4.getText().split(",");
                for(int i=0;i<no_of_passengers;i++)
                {
                    passenger.put(names[i],Integer.parseInt(seats[i]));
                }
                f.setVisible(false);
                f.dispose();
            }
        });


        f.add(l1);
        //f.add(spacer = new JLabel(" "),"span,grow");
        f.add(t1);
        f.add(l2);
        f.add(t2);
        f.add(l3);
        f.add(t3);
        f.add(l4);
        f.add(t4);
        f.add(b1);
        f.setVisible(true);
        f.setSize(500,500);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae){

    }
}

class BookingId extends JFrame implements ActionListener{
    static JFrame f;
    static JPanel p;
    public int booking_id,bus_id;

    public BookingId()
    {
        f = new JFrame();
        p = new JPanel();
        f.setLayout(new FlowLayout());
        JLabel l1 = new JLabel("Booking Id");
        JTextField t1 = new JTextField(8);
        JLabel l2 = new JLabel("Bus Id:");
        JTextField t2 = new JTextField(5);

        JButton b1 = new JButton(new AbstractAction("Submit") {
            @Override
            public void actionPerformed(ActionEvent e) {
                booking_id = Integer.parseInt(t1.getText());
                bus_id = Integer.parseInt(t2.getText());
                f.setVisible(false);
                f.dispose();
            }
```

```
        });

        f.add(l1);
        f.add(t1);
        f.add(l2);
        f.add(t2);
        f.add(b1);
        f.setSize(200,250);
        f.setVisible(true);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae){}
}

class City extends JFrame implements ActionListener{
    static JFrame f;
    static JPanel p;
    public String city_name;
    public City(DataOutputStream outToServer,DataInputStream inFromServer){
        f = new JFrame();
        p = new JPanel();
        f.setLayout(new FlowLayout());
        JLabel l1 = new JLabel("City Name");
        JTextField t1 = new JTextField(20);

        JButton b1 = new JButton( new AbstractAction("Add") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                // add Action

                //opt = e.getActionCommand();
                //System.out.println(opt);
                city_name = t1.getText();

                f.setVisible(false);
                f.dispose();
            }
        });

        f.add(l1);
        f.add(t1);

        f.add(b1);
        //f.add(p);
        f.setVisible(true);
        f.setSize(250,400);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

    }
}

class DeleteCity extends JFrame implements ActionListener{
```

```
    static JFrame f;
    static JPanel p;
    public String city_name;
    public DeleteCity(){
        f = new JFrame();
        p = new JPanel();

        f.setLayout(new FlowLayout());
        JLabel l1 = new JLabel("City Name");
        JTextField t1 = new JTextField(10);

        JButton b1 = new JButton( new AbstractAction("Delete") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                // add Action

                //opt = e.getActionCommand();
                //System.out.println(opt);
                city_name = t1.getText();

                f.setVisible(false);
                f.dispose();
            }
        });

        f.add(l1);
        f.add(t1);

        f.add(b1);
        //f.add(p);
        f.setVisible(true);
        f.setSize(250,400);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {

    }
}

class DeleteBus extends JFrame implements ActionListener{
    static JFrame f;
    static JPanel p;
    public String city_name;
    public int bus_id;

    public DeleteBus(){
        f = new JFrame();
        p = new JPanel();
        f.setLayout(new FlowLayout());

        JLabel l1=new JLabel("City Name:");
        JTextField t1 = new JTextField(20);

        JLabel l2 = new JLabel("Bus Id:");
        JTextField t2 = new JTextField(5);
```

```
        JButton b1 = new JButton(new AbstractAction("Delete Bus") {
            @Override
            public void actionPerformed(ActionEvent e) {
                city_name = t1.getText();
                String id = t2.getText();
                bus_id = Integer.parseInt(id);
                f.setVisible(true);
                f.dispose();
            }
        });

        //f.add(l1);
        //f.add(t1);
        f.add(l2);
        f.add(t2);
        f.add(b1);
        //f.add(p);
        f.setVisible(true);
        f.setSize(250,400);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae){}
}

class MessageBox {
    JFrame f;
    public MessageBox(String message){
        f = new JFrame();
        JOptionPane.showMessageDialog(f,message);
    }
}

class ViewBookings extends JFrame implements ActionListener{
    JFrame f;

    public ViewBookings(String bookings) {
        f = new JFrame();
        JPanel controlpanel = new JPanel();
        JLabel l1 = new JLabel("");
        JButton b = new JButton("Back to menu");
        l1.setText(bookings);
        //Border border = BorderFactory.createLineBorder(Color.BLACK);
        //l1.setBorder(border);
        b.addActionListener(this);
        controlpanel.add(l1);
        //controlpanel.add(b);
        f.add(new JScrollPane(controlpanel));
        f.setSize(300, 400);
        f.setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        f.dispose();
    }
}
```

```
class Frame1 extends JFrame implements ActionListener{
    JButton b1;
    JButton b2;
    public String opt;

    public Frame1(){
        setLayout(new FlowLayout());
        b1 = new JButton( new AbstractAction("Signup") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                // add Action

                opt = e.getActionCommand();
                System.out.println(opt);
                setVisible(false);
                dispose();
            }
        });
        b2 = new JButton( new AbstractAction("Login") {
            @Override
            public void actionPerformed( ActionEvent e ) {
                // add Action

                opt = e.getActionCommand();
                System.out.println(opt);
                setVisible(false);
                dispose();
            }
        });
        //b.addActionListener(this);
        add(b1);
        add(b2);
        setVisible(true);
        setSize(200,200);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    public void actionPerformed(ActionEvent e) {

    }

}

class Signup extends JFrame implements ActionListener {
    public String username,password,user_type,name,contact;
    public int flat_no;

    JLabel l1;
    JTextField t1;
    JLabel l4;
    JTextField t4;
    JLabel l5;
    JPasswordField t5;
    JLabel l2;
    JTextField t2;
    JLabel l7;
    JTextField t7;
    JButton b;
```

```
    JLabel l3;
    JRadioButton c1,c2;
    JLabel l6;

    public Signup()
    {
        setLayout(new FlowLayout());

        l1=new JLabel("Name:");
        t1=new JTextField(20);

        l7=new JLabel("Contact No:");
        t7=new JTextField(20);

        l4=new JLabel("Username:");
        t4=new JTextField(20);

        l5=new JLabel("Password:");
        t5=new JPasswordField(20);

        l3=new JLabel("UserType");
        c1 = new JRadioButton("Admin");
        c2 = new JRadioButton("Normal");
        ButtonGroup bg=new ButtonGroup();
        bg.add(c1);
        bg.add(c2);
        b=new JButton("Submit");

        l6=new JLabel();
        add(l1);
        add(t1);
        add(l4);
        add(t4);
        add(l7);
        add(t7);
        add(l5);
        add(t5);
        add(l3);
        add(c1);
        add(c2);
        add(b);
        add(l6);


        b.addActionListener(this);

        setVisible(true);
        setSize(250,400);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae)
    {

        name=t1.getText();
        username = t4.getText();
        password = t5.getText();
        contact = t7.getText();
```

```
        String message="Welcome "+name+ " ";

        if(c1.isSelected())
        {
            message=message+c1.getText()+" ";
            user_type="A";
        }
        if(c2.isSelected())
        {
            message=message+c2.getText()+" ";
            user_type="N";
        }

        //l6.setText(message);
        //Login l = new Login();
        dispose();


    }
}

class Login extends JFrame implements ActionListener
{
    public String username,password;
    JLabel l1;
    JTextField t1;
    JLabel l4;
    JPasswordField t4;
    JButton b;
    JLabel l6;

    public Login()
    {
        setLayout(new FlowLayout());

        l1=new JLabel("Username:");
        t1=new JTextField(20);

        l4=new JLabel("Password:");
        t4=new JPasswordField(20);
        b=new JButton("Submit");
        l6=new JLabel();
        add(l1);
        add(t1);
        add(l4);
        add(t4);
        add(b);
        add(l6);
        b.addActionListener(this);
        setVisible(true);
        setSize(250,400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent ae){
        username=t1.getText();
        password=t4.getText();
        String message = "Welcome "+username;
```

```
        //l6.setText(message);
        dispose();
    }
}
```

Server Side:

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.sql.*;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.xml.transform.Result;

class Booking
{
    public synchronized String book(String dest_city, int no_of_pass,
HashMap<String, Integer> passenger_list, Connection conn, int user_id)  {
        try {
            PreparedStatement ps = null;
            ps = conn.prepareStatement("select * from cities where city_name =
?");
            ps.setString(1,dest_city);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "City not available";
            }
            String city_name=rs.getString(2);
            int city_id = rs.getInt(1);
            ps = conn.prepareStatement("select * from buses where city_id = ?");
            ps.setInt(1,city_id);
            rs = ps.executeQuery();
            if(!rs.next())
            {
                return "Bus not available";
            }
            int bus_id = rs.getInt(1);
            int seat_booked = rs.getInt(2);
            if(seat_booked==40) {
                return "Seats Not Available";
            }
            Random rand = new Random();
            int booking_id = rand.nextInt(999999);
            ps = conn.prepareStatement("select seat_no from passengers where
bus_id = ?");
            ps.setInt(1,bus_id);
            rs = ps.executeQuery();
            while(rs.next())
            {
                if(passenger_list.containsValue(rs.getInt(1)))
                {
                    return rs.getInt(1)+" seat is already booked!!\n Please see
seat available option!!";
                }
            }
```

```
            ps = conn.prepareStatement("insert into booking values(?,?,?,?,?)");
            ps.setInt(1, user_id);
            ps.setInt(2, booking_id);
            ps.setInt(3, city_id);
            ps.setInt(4, bus_id);
            ps.setInt(5, no_of_pass);
            ps.executeUpdate();
            ps = conn.prepareStatement("insert into passengers
values(?,?,?,?,?)");
            ps.setInt(2, booking_id);
            ps.setInt(3, bus_id);
            Set<String> key = passenger_list.keySet();
            Iterator<String> i = key.iterator();
            int j=1;
            String book_statement="Booking Id:" + booking_id + "\nBus Id:" +bus_id
+ "\nPassenger List:\n";
            while(i.hasNext())
            {
                ps.setInt(1,j++);
                String name = String.valueOf(i.next());
                int seat = passenger_list.get(name);
                ps.setString(4,name);
                ps.setInt(5, seat);
                ps.executeUpdate();
                book_statement += i + ". " + name + " " + seat + "\n";
            }
            ps = conn.prepareStatement("update buses set seat_booked = seat_booked
+ ? where bus_id = ?");
            ps.setInt(1,no_of_pass);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            return book_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }

    public String check(int booking_id,Connection conn)
    {
        try{

            PreparedStatement ps = conn.prepareStatement(" select
city_name,bus_id,passenger_name,seat_no,username from booking natural join
passengers natural join cities natural join users where booking_id = ?");
            ps.setInt(1,booking_id);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "No Booking!!";
            }
            String check_statement="";
            String city = rs.getString(1) ;
            int bus_id = rs.getInt(2 );
            String name = rs.getString(3);
            int seat = rs.getInt(4);
            String username = rs.getString(5);
            check_statement += "Destination City: " + city + "\nBus id: " + bus_id
```

```
+ "\nPassenger List:\nName: "+name+" Seat Number: " + seat;
            while(rs.next())
            {
                name = rs.getString(3);
                seat = rs.getInt(4);
                check_statement += "\nName: "+name+" Seat Number: "+seat;
            }
            check_statement += "\nBooked by: " + username;
            return check_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
    public String cancel(int booking_id,int bus_id,Connection conn)
    {
        try
        {
            PreparedStatement ps = conn.prepareStatement("select * from booking
where booking_id = ? and bus_id = ?");
            ps.setInt(1,booking_id);
            ps.setInt(2,bus_id);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
                return "No bookings!!";
            int no_of_pass = rs.getInt(5);
            ps = conn.prepareStatement("delete from booking where booking_id = ?
and bus_id = ?");
            ps.setInt(1,booking_id);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            ps = conn.prepareStatement("update buses set seat_booked = seat_booked
- ? where bus_id = ?");
            ps.setInt(1,no_of_pass);
            ps.setInt(2,bus_id);
            ps.executeUpdate();
            return "Ticket Cancelled!!";
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
    public String availability(String dest_city,Connection conn)
    {
        try
        {
            PreparedStatement ps = conn.prepareStatement("select city_id from
cities where city_name = ?");
            ps.setString(1,dest_city);
            ResultSet rs = ps.executeQuery();
            if(!rs.next())
            {
                return "City Not Available!!";
            }
            int city_id = rs.getInt(1);
            ps = conn.prepareStatement("select bus_id,seat_booked from buses where
city_id = ?");
```

```
            ps.setInt(1,city_id);
            rs = ps.executeQuery();
            String available_statement ="";
            while(rs.next()) {
                int bus_id = rs.getInt(1);
                available_statement += "Bus Id:" + bus_id + "\n";
                int seat_booked = rs.getInt(2);
                if (seat_booked == 0) {
                    for (int i = 1; i <= 40;) {
                        available_statement += (i++) + ".A  " + (i++) + ".A\t\t" +
(i++) + ".A  " + (i++) + ".A\n";
                    }
                } else {
                    ps = conn.prepareStatement("select seat_no from passengers
where bus_id = ?");
                    ps.setInt(1,bus_id);
                    ResultSet rs1 = ps.executeQuery();
                    ArrayList<Integer> seats = new ArrayList<>();
                    while(rs1.next())
                    {
                        seats.add(rs1.getInt(1));
                    }
                    for(int i=1;i<=40;i++)
                    {
                        if(seats.contains(i))
                        {
                            if(i%4==0)
                                available_statement += i + ".B\n";
                            else if(i%2==0)
                                available_statement += i+ ".B\t\t";
                            else
                                available_statement += i + ".B  ";
                        }
                        else
                        {
                            if(i%4==0)
                                available_statement += i + ".A\n";
                            else if(i%2==0)
                                available_statement += i+ ".A\t\t";
                            else
                                available_statement += i + ".A  ";

                        }
                    }
                }
            }
            return available_statement;
        }
        catch(Exception e)
        {
            return String.valueOf(e);
        }
    }
}

public class Server {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket(8081);
        ExecutorService executor = Executors.newFixedThreadPool(5);
```

```
        System.out.println("Starting Server...");
        while (true) {
            Socket s = null;
            try {
                s = ss.accept();
                System.out.println("New Client Connected:" + s);
                DataOutputStream outToClient = new
DataOutputStream(s.getOutputStream());
                DataInputStream inFromClient = new
DataInputStream(s.getInputStream());
                ObjectOutputStream os=new ObjectOutputStream(s.getOutputStream());
                ObjectInputStream is=new ObjectInputStream(s.getInputStream());
                System.out.println("Assigning Thread to Client");
                Thread t = new ClientHandler(s, inFromClient, outToClient,os,is);
                executor.execute(t);
            } catch (Exception e) {
                assert s != null;
                s.close();
                e.printStackTrace();
            }
        }
    }
}
class ClientHandler extends Thread {
    final DataInputStream inFromClient;
    final DataOutputStream outToClient;
    final Socket s;
    final ObjectInputStream is;
    final ObjectOutputStream os;

    public ClientHandler(Socket s, DataInputStream inFromClient, DataOutputStream
outToClient,ObjectOutputStream os,ObjectInputStream is) throws Exception {
        this.s = s;
        this.inFromClient = inFromClient;
        this.outToClient = outToClient;
        this.os = os;
        this.is = is;
    }

    public void run() {
        try {
            Scanner sc = new Scanner(System.in);
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/sdl_assignment_3?autoReco
nnect=true&useSSL=false", "root", "varunkarwa");
            Statement stmt = conn.createStatement();
            String choice,user_type="N";
            PreparedStatement ps=null;
            int user_id=0;
            do{
                boolean Authentication=false;
                while(!Authentication){
                    //outToClient.writeUTF("1.Login\n2.Sign Up:");
                    choice = inFromClient.readUTF();
                    switch(choice)
                    {
                        case "1":{
                            //outToClient.writeUTF("Enter Username:");
```

```
                            String username = inFromClient.readUTF();
                            //outToClient.writeUTF("Enter Password:");
                            String password = inFromClient.readUTF();
                            ps = conn.prepareStatement("select
user_id,name,contact_no,user_type from users where username = ? and password =
?");
                            ps.setString(1, username);
                            ps.setString(2, password);
                            try{ResultSet rs = ps.executeQuery();
                                if (rs.next()) {
                                    outToClient.writeUTF("User_id:" + rs.getInt(1)
+ "\nName:" + rs.getString(2) + "\nContact: " + rs.getString(3));
                                    user_type=rs.getString(4);
                                    user_id = rs.getInt(1);
                                    Authentication = true;
                                   // System.out.println("User logged in!\n" +
"User_id:" + rs.getInt(1) + "\nName:" + rs.getString(2) + "\nContact: " +
rs.getString(3));

                                } else {
                                    outToClient.writeUTF("Invalid Credentials!!");
                                }}
                            catch (Exception e)
                            {
                                System.out.println(e);
                                //outToClient.writeUTF("Error!");
                            }
                            outToClient.writeUTF(username);
                            outToClient.writeBoolean(Authentication);
                            outToClient.writeUTF(user_type);
                            break;
                        }
                        case "2": {
                            //outToClient.writeUTF("Enter Name:");
                            String name = inFromClient.readUTF();
                            //outToClient.writeUTF("Enter username:");
                            String username = inFromClient.readUTF();
                            //outToClient.writeUTF("Enter password:");
                            String password = inFromClient.readUTF();
                            int contact = inFromClient.readInt();
                            //outToClient.writeUTF("Enter Contact_no:");
                            //outToClient.writeUTF("User Type(Admin A/Normal
N):");
                            String type = inFromClient.readUTF();
                            ps = conn.prepareStatement("insert into
users(username,password,name,contact_no,user_type) values(?,?,?,?,?)");
                            ps.setString(1,username);
                            ps.setString(2,password);
                            ps.setString(3,name);
                            ps.setInt(4,contact);
                            ps.setString(5,type);
                            try {
                                int rs = ps.executeUpdate();
                                outToClient.writeUTF("Account Created!!\nLogin to
continue!!");
                                //System.out.println("Account Created!!\n Users
Details:\nUsername:" + username + "\nPassword:" + password + "\nName:" + name +
"\nContact No:" + contact);
                            }catch(Exception e)
```

```
                            {
                                outToClient.writeUTF(String.valueOf(e));
                            }
                            break;
                        }
                        default:
                            System.out.println("Incorrect Choice!!");
                            break;
                    }
                }
                Booking b = new Booking();
                if(user_type.equals("N"))
                {
                    do {
                        choice = inFromClient.readUTF();
                        switch (choice) {
                            case "Book Tickets": {
                                String dest_city = inFromClient.readUTF();
                                int no_of_pass = inFromClient.readByte();
                                HashMap<String, Integer> passenger_list =
(HashMap<String, Integer>) is.readObject();
                                String book_statement = b.book(dest_city,
no_of_pass, passenger_list, conn, user_id);
                                outToClient.writeUTF(book_statement);
                                break;
                            }
                            case "Check Reservation":{
                                int booking_id = inFromClient.readInt();
                                String check_statement = b.check(booking_id,conn);
                                outToClient.writeUTF(check_statement);
                                break;
                            }
                            case "Cancel Tickets":{
                                int booking_id = inFromClient.readInt();
                                //int bus_id = inFromClient.readInt();
                                ps = conn.prepareStatement("select bus_id from
passengers where booking_id = ?");
                                ps.setInt(1,booking_id);
                                ResultSet rs = ps.executeQuery();
                                rs.next();
                                int bus_id = rs.getInt(1);
                                String cancel_statement =
b.cancel(booking_id,bus_id,conn);
                                outToClient.writeUTF(cancel_statement);
                                break;
                            }
                            case "View Bookings":
                            {
                                ps = conn.prepareStatement("select * from
passengers where booking_id in(select booking_id from booking where user_id = ?)
order by bus_id,seat_no;");
                                ps.setInt(1,user_id);
                                ResultSet rs =ps.executeQuery();
                                while (rs.next()) {
                                    outToClient.writeUTF("con");
                                    outToClient.writeInt(rs.getInt(1));
                                    outToClient.writeInt(rs.getInt(2));
                                    outToClient.writeInt(rs.getInt(3));
                                    outToClient.writeUTF(rs.getString(4));
```

```
                                outToClient.writeInt(rs.getInt(5));
                            }
                            outToClient.writeUTF("dis");
                            break;
                        }
                    }
                }while(!choice.equals("6"));
            }
            else
            {
                do
                {
                    choice = inFromClient.readUTF();
                    switch(choice)
                    {
                        case "Add City":
                        {
                            String dest_city = inFromClient.readUTF();
                            ps = conn.prepareStatement("select * from cities
where city_name=?");
                            ps.setString(1,dest_city);
                            ResultSet rs = ps.executeQuery();
                            if(rs.next()){
                                outToClient.writeUTF("City Already Added!");
                                break;
                            }
                            ps = conn.prepareStatement("insert into
cities(city_name) values(?)");
                            ps.setString(1,dest_city);
                            try
                            {
                                ps.executeUpdate();
                                outToClient.writeUTF("Added Successfully");
                            }
                            catch(Exception e)
                            {
                                outToClient.writeUTF(String.valueOf(e));
                            }
                            break;
                        }
                        case "Add Bus":{
                            String dest_city = inFromClient.readUTF();
                            ps = conn.prepareStatement("select city_id from
cities where city_name = ?");
                            ps.setString(1,dest_city);
                            ResultSet rs = ps.executeQuery();
                            if(!rs.next())
                            {
                                outToClient.writeUTF("Please first add this
city!!");
                                break;
                            }
                            int city_id = rs.getInt(1);
                            ps = conn.prepareStatement("insert into
buses(seat_booked,city_id) values(0,?)");
                            ps.setInt(1,city_id);
                            ps.executeUpdate();
                            outToClient.writeUTF("Added Successfully!");
                            break;
```

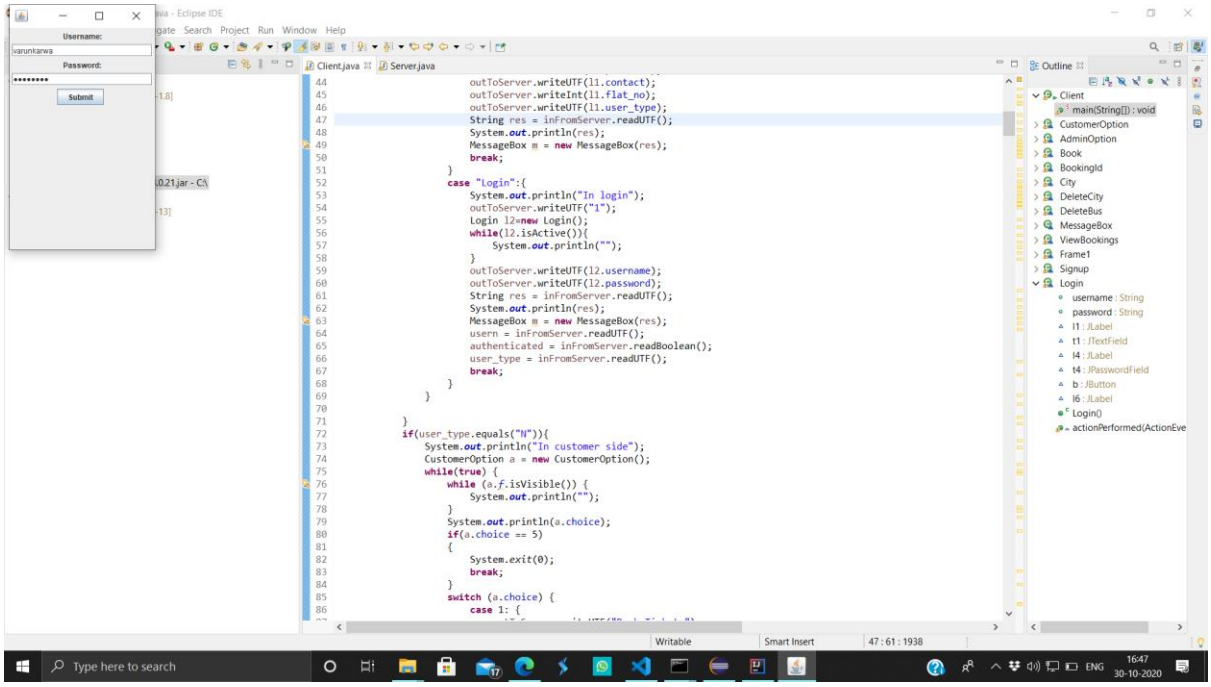
```
                            }
                            case "Cancel Bus":{
                                //outToClient.writeUTF("Enter bus id:");
                                int bus_id = inFromClient.readInt();
                                ps = conn.prepareStatement("delete from buses
where bus_id = ?");
                                ps.setInt(1,bus_id);
                                if(ps.executeUpdate() == 0)
                                    outToClient.writeUTF("Bus already
cancelled!!");
                                else
                                    outToClient.writeUTF("Bus Cancelled!!");
                                break;
                            }
                            case "Delete City":{
                                //outToClient.writeUTF("Enter city_name:");
                                String dest_city = inFromClient.readUTF();
                                ps = conn.prepareStatement("delete from cities
where city_name = ?");
                                ps.setString(1,dest_city);
                                if(ps.executeUpdate() == 0)
                                    outToClient.writeUTF("First add the city
please!!");
                                else
                                    outToClient.writeUTF("City deleted
Successfully!!");
                                break;
                            }
                            case "View Bookings":
                            {
                                String passenger_statement = "Today's
Bookings:\n";
                                ResultSet rs =stmt.executeQuery("select * from
passengers order by bus_id,seat_no");
                                while (rs.next()) {
                                    outToClient.writeUTF("con");
                                    outToClient.writeInt(rs.getInt(1));
                                    outToClient.writeInt(rs.getInt(2));
                                    outToClient.writeInt(rs.getInt(3));
                                    outToClient.writeUTF(rs.getString(4));
                                    outToClient.writeInt(rs.getInt(5));
                                }
                                outToClient.writeUTF("dis");
                                break;
                            }
                        }
                    }while(!choice.equals("6"));
                }

                outToClient.writeUTF("1.Exit\n2.Continue");
                choice = inFromClient.readUTF();
            }while(!choice.equals("1"));
        }
        catch (Exception e) {
            System.out.println(e);
        }
        try{
            this.inFromClient.close();
            this.outToClient.close();
```

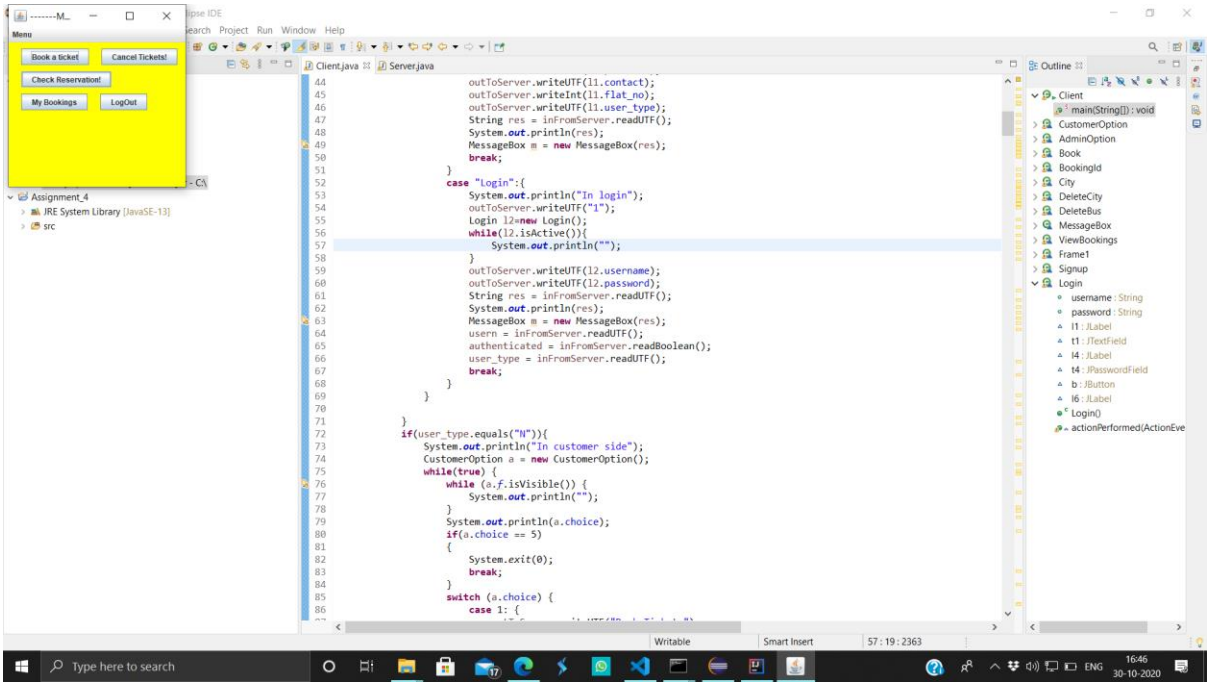
```
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

Output Screenshots:

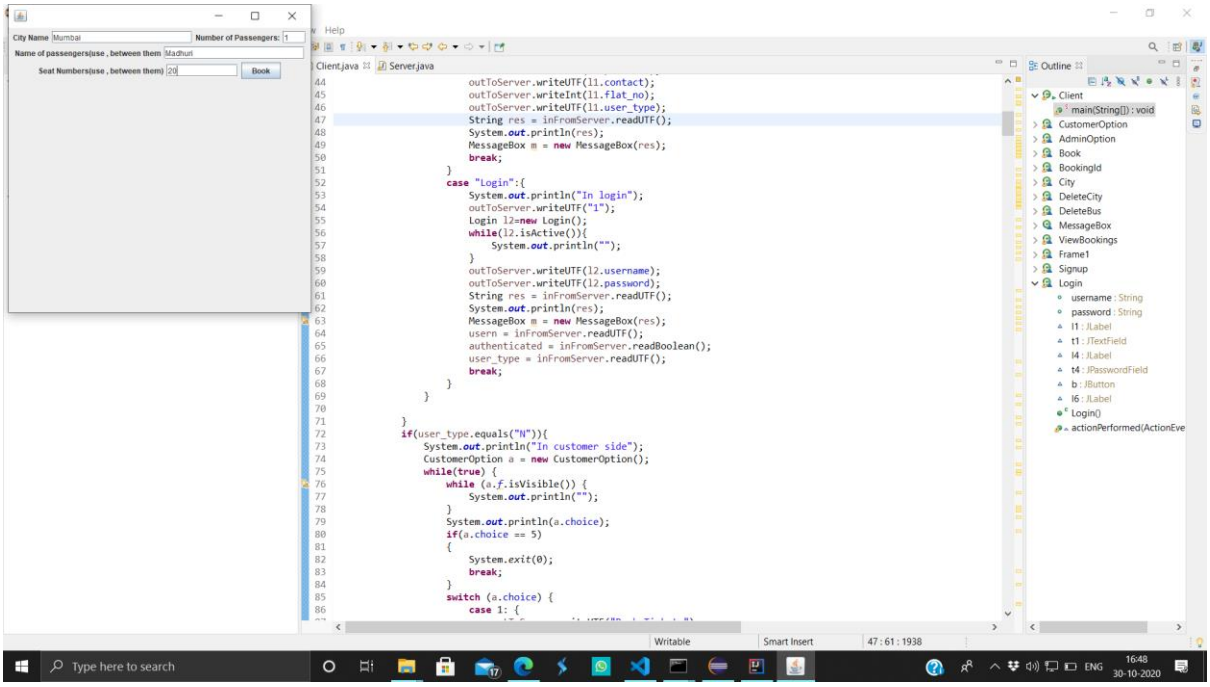
1. Login:



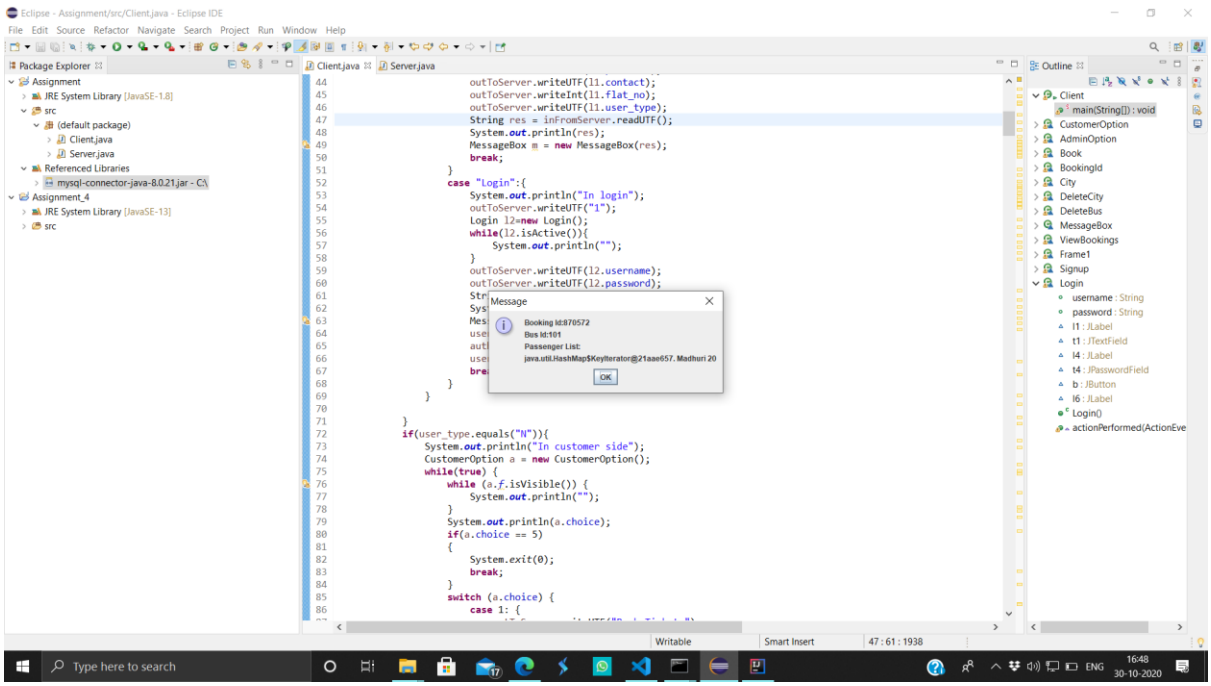
2. User Interface:



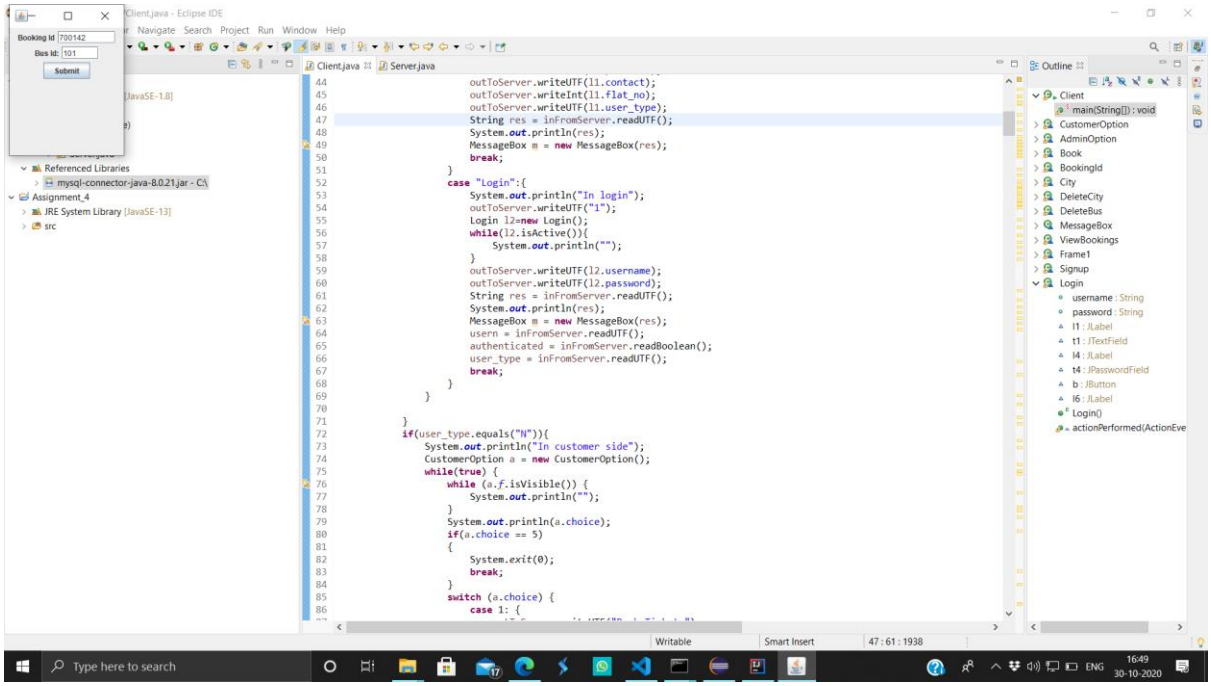
3.Booking Interface:



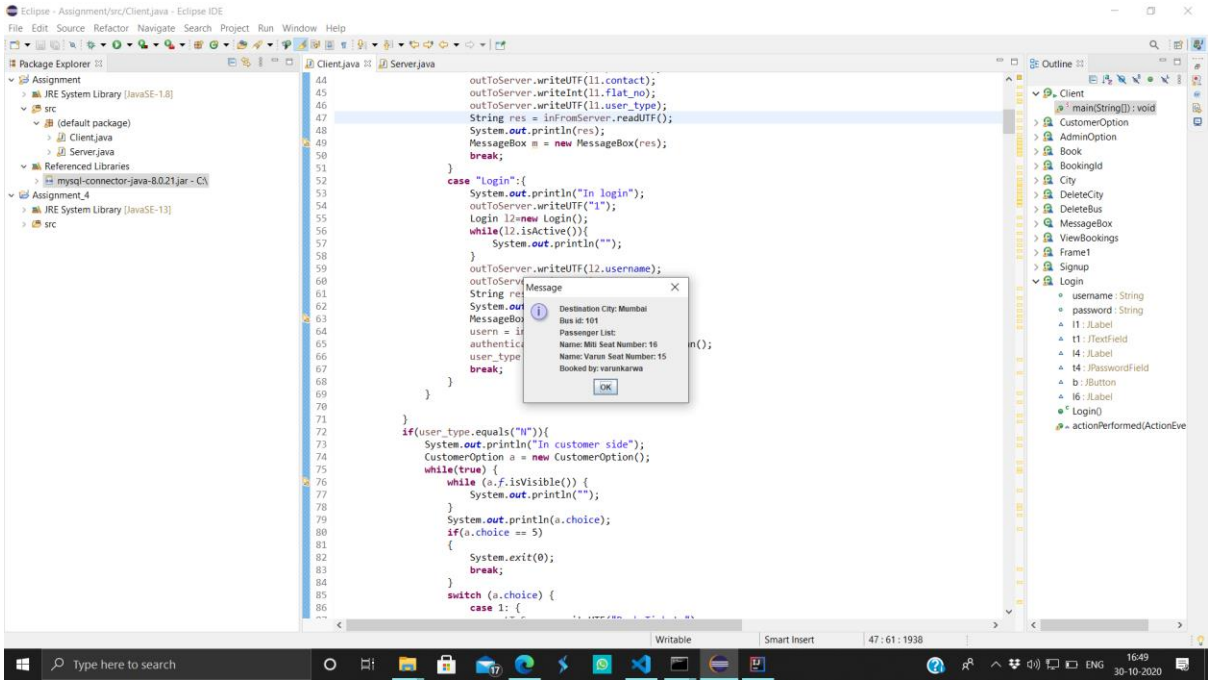
4. Booking Message:



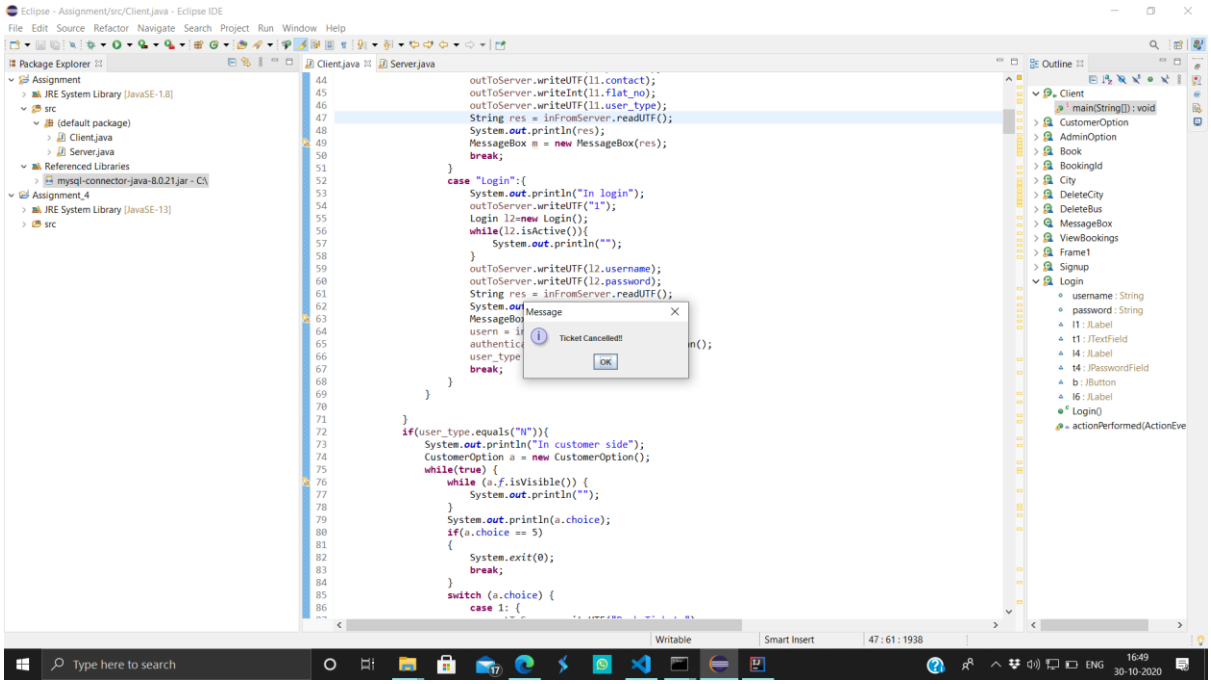
5. Check Reservation interface:



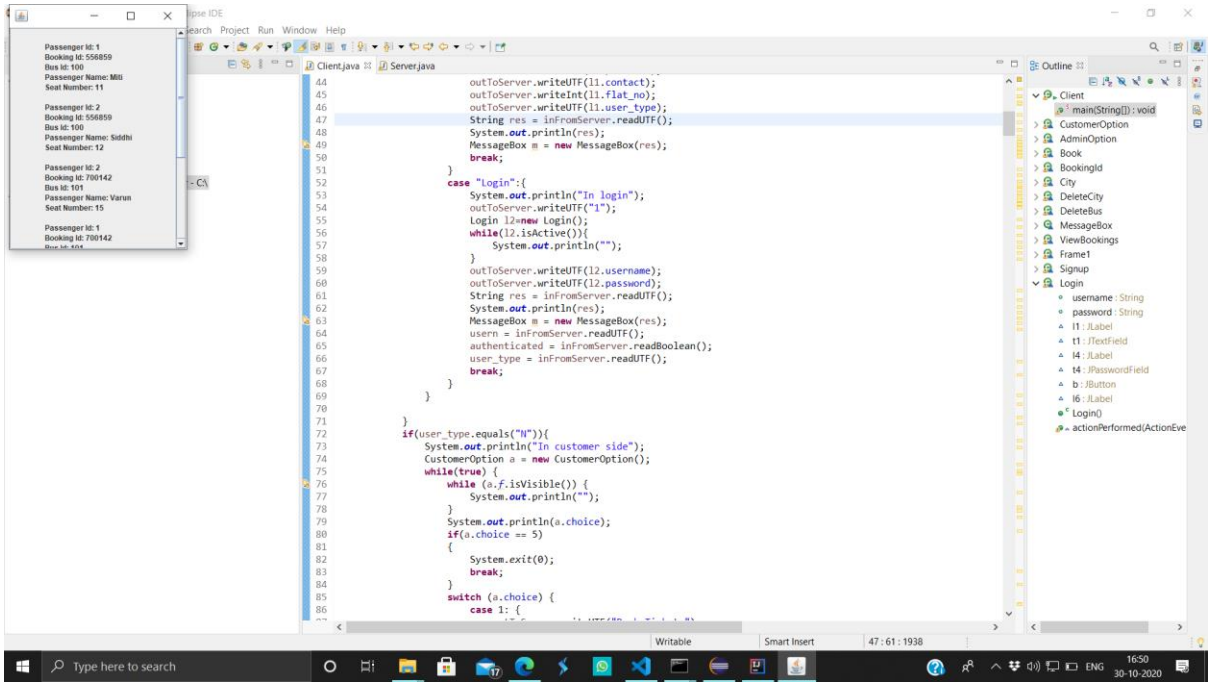
6. Check reservation message:



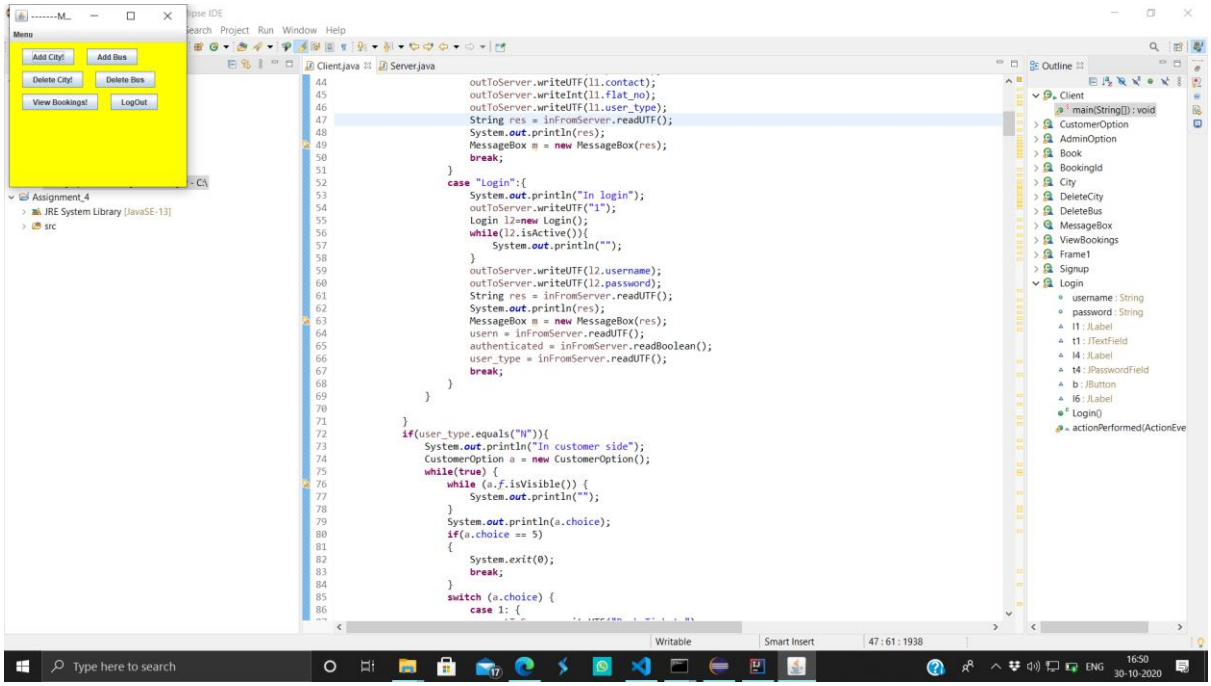
7. Cancel Ticket message:



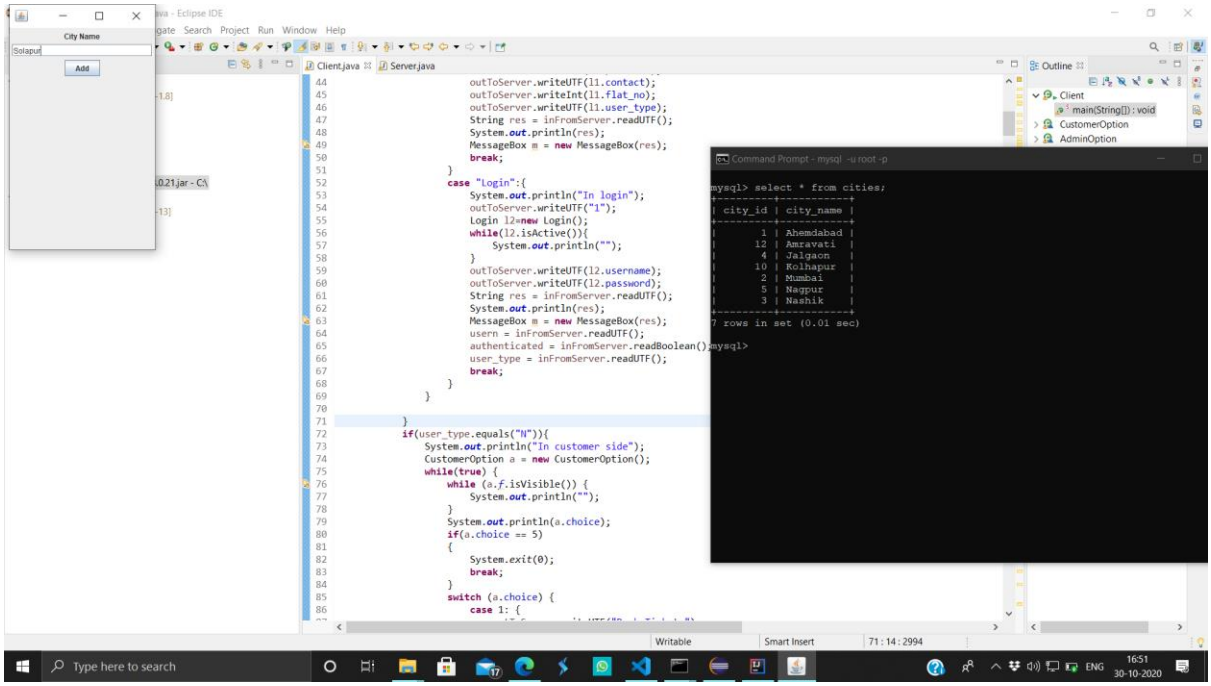
8. My Bookings:



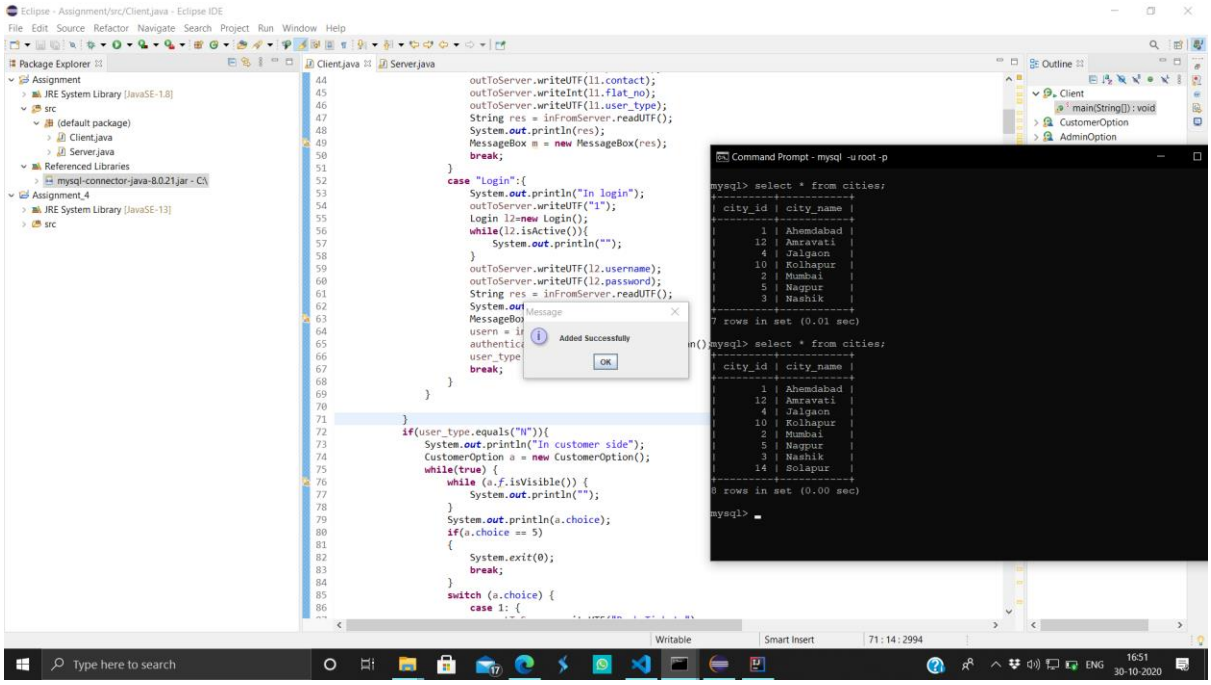
9.Admin Interface:



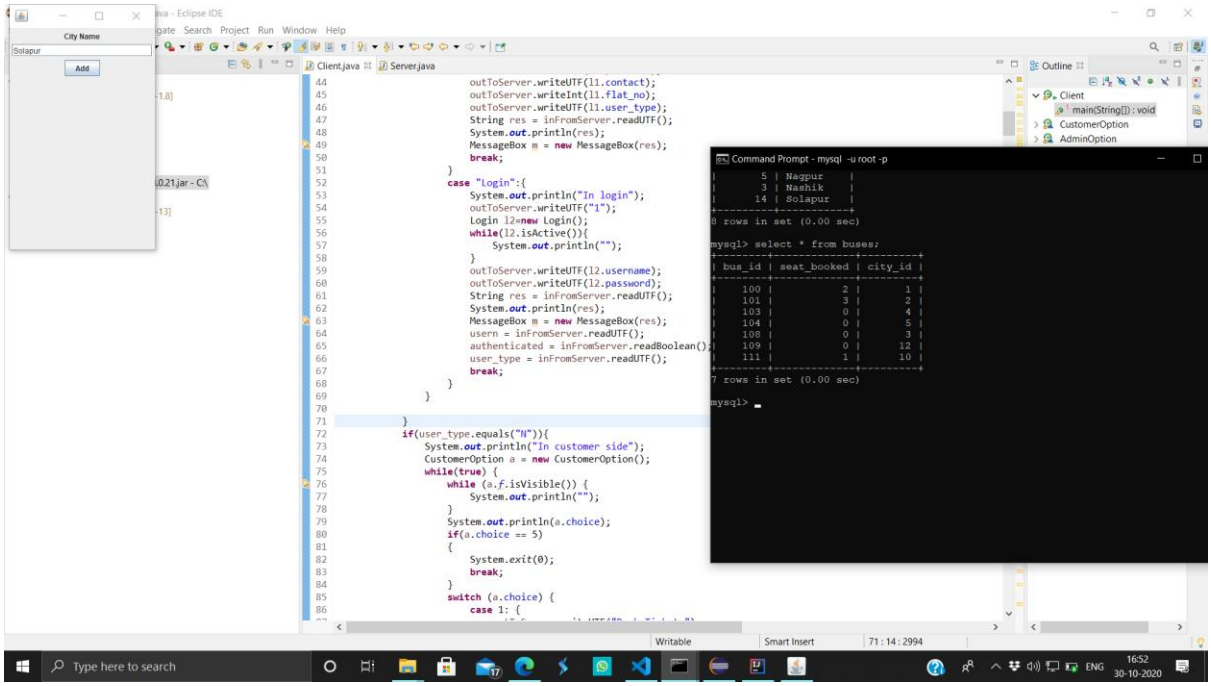
10.Add City interface:



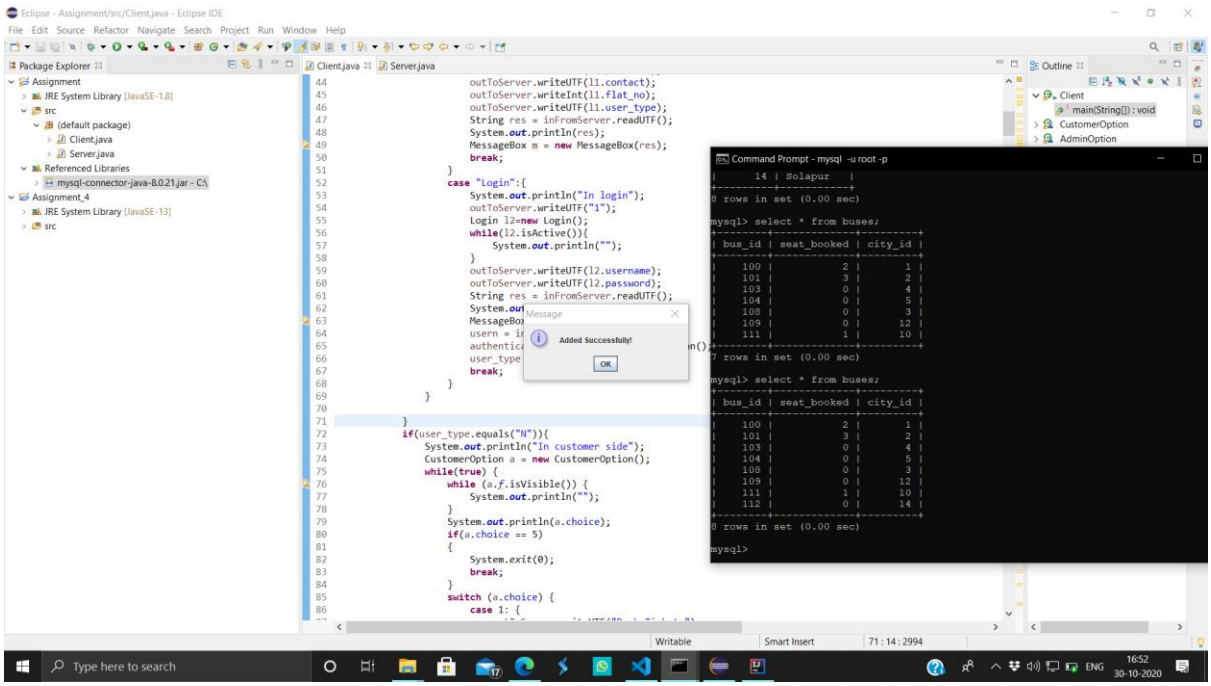
11.Add City message:



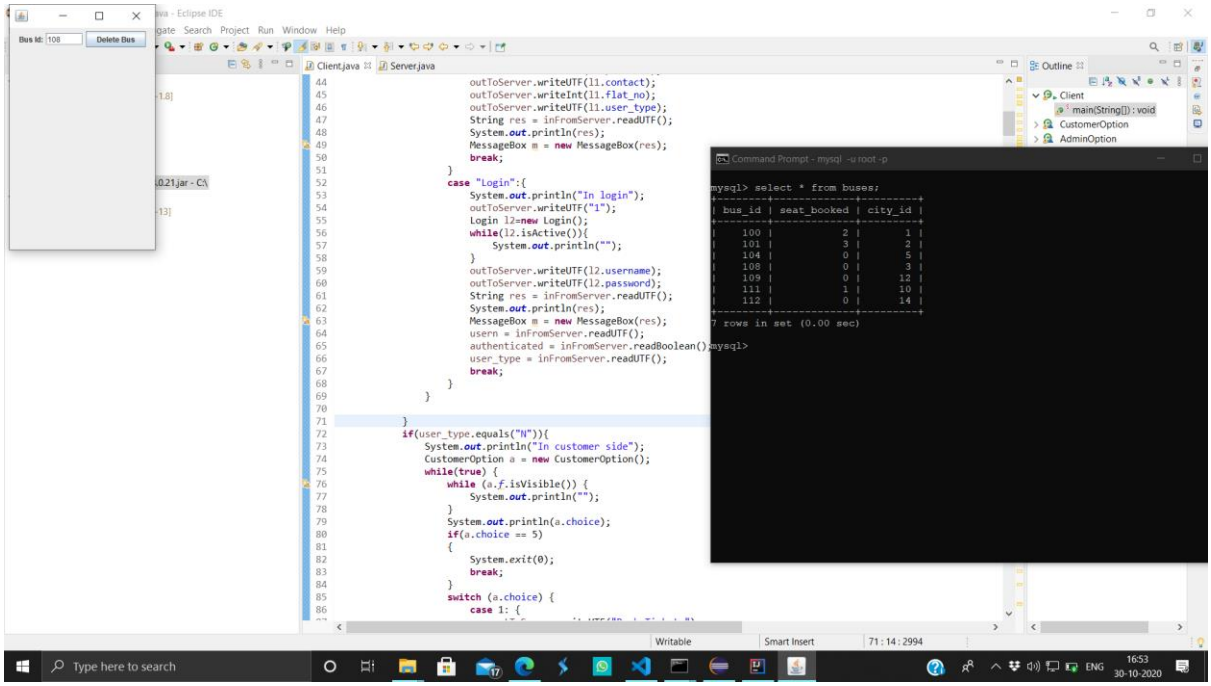
12:Add Bus Interface:



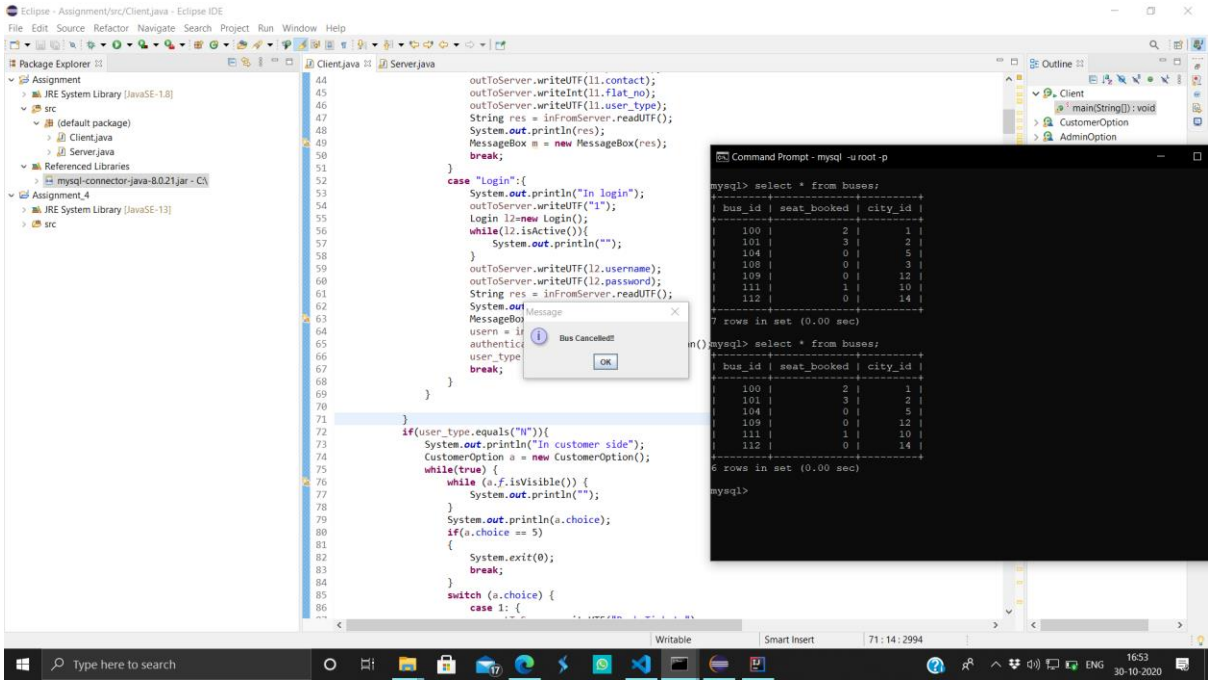
13. Add Bus message:



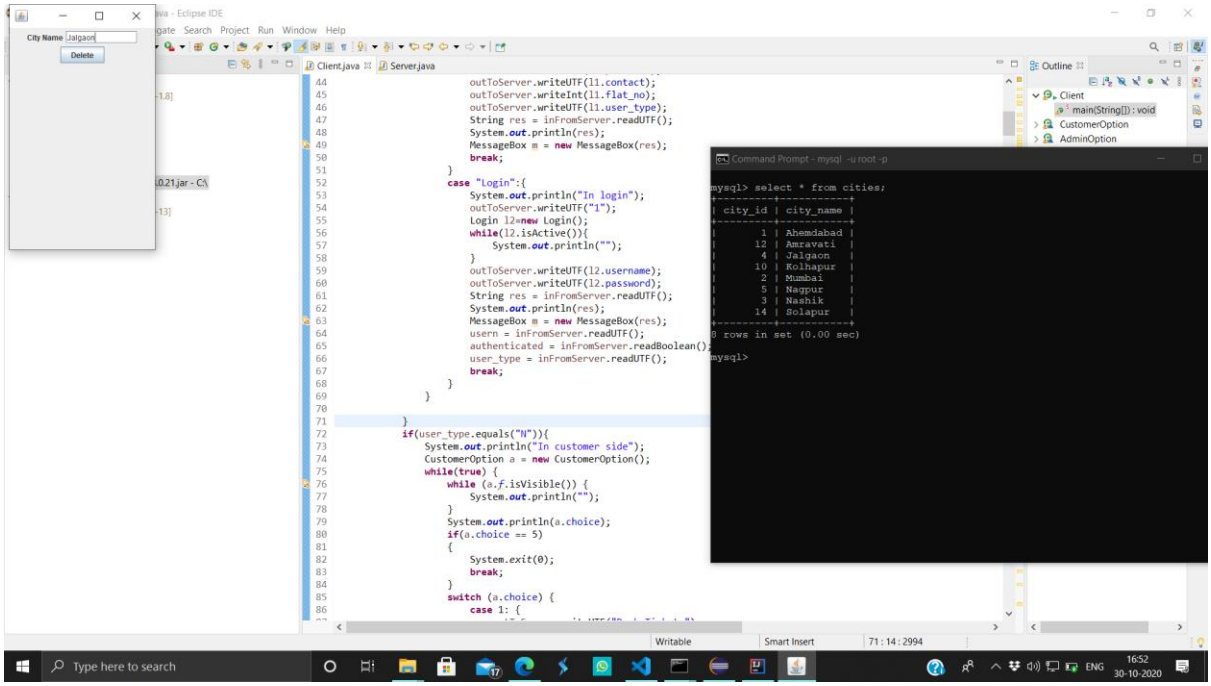
14.Delete Bus:



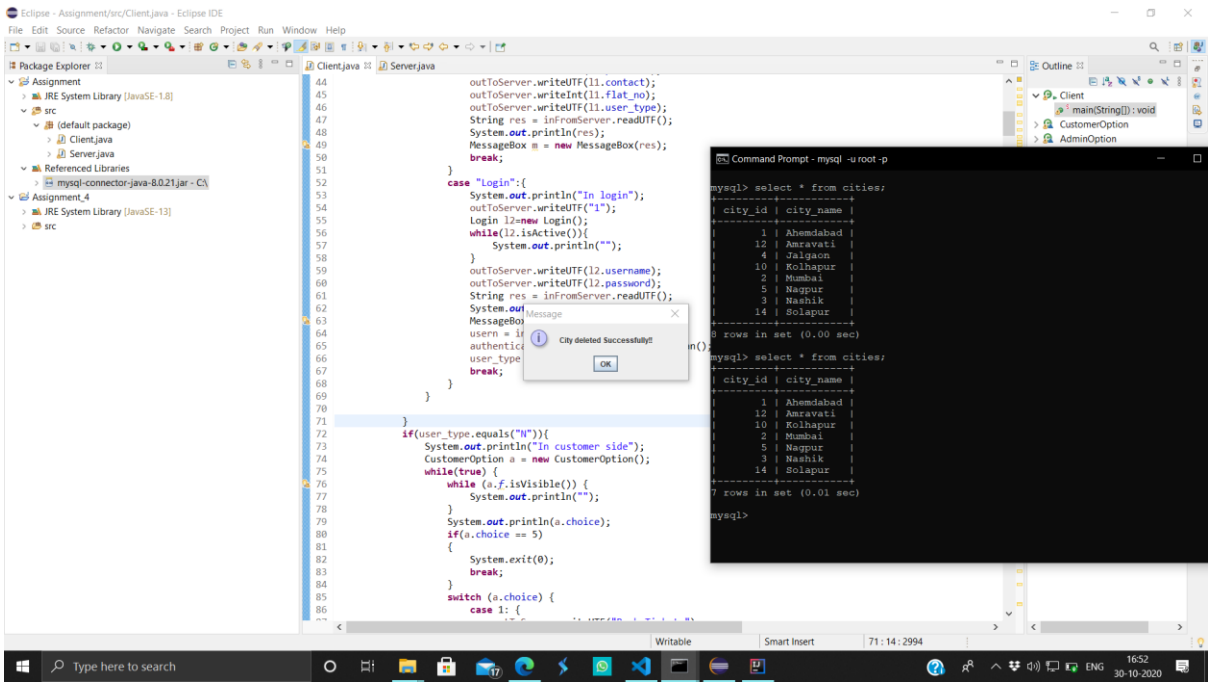
15. Delete Bus Message:



16. Delete City:



17. Delete City message:



18.All Bookings: