ADA UNIVERSITY
SCHOOL OF IT & ENGINEERING

Course: Object-Oriented Analysis & Design

HOMEWORK ASSIGNMENT 3

Project title: ***"Taxi Dispatch Application"***

Students: Farida Aliyeva
Mustafa Aslanov
Aliya Bannayeva
Fidan Ismayilova

Instructor: **Dr. Abzatdin Adamov**

Baku 2020

Table of Contents

Functional Requirements

Core Requirements.

R1. The system is required to handle the transaction in cash and credit card

Depending on the desire of the customer, he/she should be able to select at least one transaction method to proceed with payment.

R2. The driver is required to register and login by providing required information.

The required registration information for driver is driver's name, phone number, password, car model and car number. And after already being registered, the required login information is only username (the driver's name) and password that was mentioned when registering.

R3. The customer is required to register and login by providing required information.

The required registration information for customer is customer's name, phone number and password. And after already being registered, the required login information is only username (the customer's name) and password that was mentioned when registering.

R4. The system is required to display at least in 3 languages

To make sure the local residents, as well as tourists will be able to easily work with the system, it should be displayed at least in Azerbaijani, English and Russian languages.

R5. The system should be able to track user geo-locations.

To easily track both customers and drivers, the system should use geo-fencing and locate both. Therefore, customers will be able to visually see how far the driver is and at the same time the drivers will be able to locate customers in case if they moved.

R6. The system is required to calculate fare prices.

The system should calculate fare prices according to wait-time, distance travelled and the hour of the day. If it is the peak of requests, fares will be slightly increased in prices (to compensate wait-time in traffics).

R7. The system should be able to alert customer when driver arrives.

In the case of issues of visual geo-tracking, the customer should be able to receive a notification or an alert when the driver has arrived to the pick-up destination.

Optional Requirements.

R8. The user should be able to communicate usage of the system on social media.

As an optional requirement, the user should be able to share latest rides on social media of their own choice. Thus, spreading the popularity of the system as well.

R9. The customer should be able to select type of the vehicle.

Depending on the desire of the customer, the system should be able to provide 3 basic choices for cars: Econom – cheaper in price but with less comfort, VIP – more expensive compared to econom type but with more comfort, Comfort – similar to VIP pricing but with more functionalities, namely easier to get in for people with disabilities or if there are more people for a single ride.

R10. The customer should be able to select multiple destinations

The system should allow customer to pick multiple destination points depending on their desire. This may result in cheaper prices for both sides in comparison if customer would have had to order taxi multiple times.

R11. The customer should be able to pre-book taxi ride.

In case of strict schedule for customer, they should be able to pre-order taxi at specific location and specific time.

R12. The user should be able to view opposite side's information.

Drivers should be able to see customer's phone number and name in order to easily contact them in case of need. And vice versa, customers should be able to see driver's information.

R13. The system is expected to calculate shortest or best route.

As an additional requirement, the system should be able to calculate the route, that is which has the shortest travel distance and with least traffic. This will result in less prices for customer, and more time for driver to have another ride.

R14. The users should be able to rate each other.

The customer must be given the option to rate their designated driver and their experience. Likewise, the driver should also be able to rate their experience with the customer.

R15. The system is expected to have a built-in map.

The built-in map is required to help the customer track the driver with respect to their current location, all from one app. Likewise, the driver can see their location with respect to the customer's given location.

R16. The system is intended to have communication means.

The customer is supposed to be able to communicate with their designated driver. Likewise, the driver is required to communicate with the customer to inform them of their arrival.

R17. The system should be able to collect usage analytics.

The system should track customer's usage i.e. number of fares they've had, distance travelled and top five destinations.

Non-Functional Requirements.

R18. The system should not lag or freeze.

The system should run smoothly and not freeze or lag should there be an overflow of users.

R19. The system should be fault resistant.

The system should run until the user chooses to terminate it and never terminate on its own, i.e. crash when faced with an error.

R20. The system should not fail on basic requirements.

The application will perform well on basic requirements like proper location detection, failure on payment through credit card, correct pickup timing, correct destination points, etc. (Reliability – Failure)

R21. The system is expected to provide proper pricing.

The system should correctly determine the prices for rides, information about previous and current rides. The switch from cash to credit card payment must flow smoothly (Usability).

R22. The system is expected to respond quickly.

The system must respond quickly when ordering taxi and be in constant connection with server (Performance).

R23. The system is required to be secure.

Cash and credit card information for payments must be handled securely to ensure no private information is leaked and the company is not held liable (Security).

R24. The user information should be able to stay private.

The system must run while keeping personal information of the customer hidden from the driver and driver's personal information hidden from the customer, to respect their privacies.

R25. The system should be able to operate on various operating systems.

The system should run equally optimally on both Android (5.0+) and iOS (6.0+) systems as well as on Desktop. (Supportability)

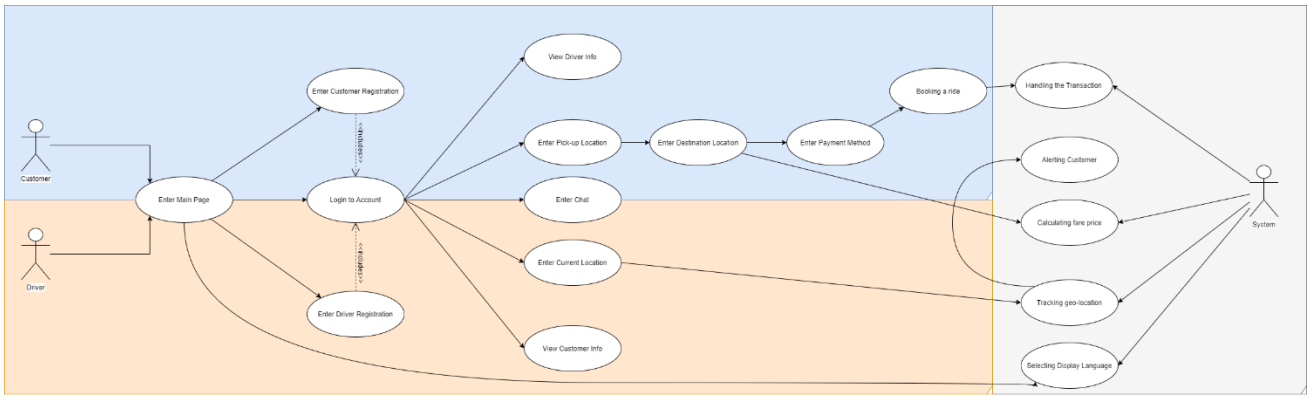
R26. The system is expected to be easily maintained.

The system should be easily maintained, and any new additions or deletions implemented without drastic changes to the existing system. (Maintainability)

R27. The system is required to have a user-friendly interface.

Each page should be accessed with less than 3 clicks and the interface must feel intuitive for the user so they know how to use the system without having to go through any manuals.

Use Case Diagram



Preparation for Design Phase

Internal Consistency

Functional Model V&V

There must be a one-to-one correspondence of Use Cases in the Use Case Diagram and Use Case description.

UC-1 describes how the customer may check their information after logging in. This is also portrayed on the Use Case Diagram, following the sequence Enter Main Page -> Login User -> View Customer Info/View Driver Info.

UC-2 describes how the customer may choose their payment option. This is portrayed in the Use Case Diagram following the sequence Enter Main Page -> Login User -> Enter Pickup Location -> Enter Destination Location -> Enter Payment Method.

UC-4 contains that the customer may input their current location to be picked up from. This is portrayed in the Use Case Diagram following the sequence Enter Main Page -> Login User -> Enter Pickup Location.

UC-5 describes the ability of the customer to communicate with the driver. This is portrayed in the Use Case diagram following the sequence Enter Main Page -> Login User -> Enter Chat

UC-7 describes how a user may pick a language to use the system in. This is describing in the Use Case diagram following the sequence Enter Main Page -> Select Display Language.

All Actors listed in a Use Case description must be portrayed on the Use-Case Diagram.

The Actors listed in the Use Case description are: Customer, System and Driver. This the same as in the Use-Case diagram.

All relationships listed in a Use-Case description are portrayed on a Use-Case Diagram.

Structural Model V&V

As it was not mentioned in the project requirements, we do not have CRC cards to compare them with the Class Diagram.

Behavioral Model V&V

As it was not mentioned in the project requirements, we do not have Communication Diagrams to compare them with the Sequence Diagram.

External Consistency

Subjects depicted in use cases are Customer, Driver and System are respectively depicted as User, Driver and Application in class diagram.

Objects that are depicted in use cases also are depicted in class diagram, such as Registration for Customer and Driver, and Billing (transaction). While our use case has mention of one login page, class diagram includes two for Driver and Customer login pages for database storing purposes, yet visually they do not differ.

By transtitioning from analysis model to design model phase, we have implemented factoring design model. We have improved our system by refining our functionalities and ensuring they are abstracted. New classes were added (database,login) with appropriate generalization and aggregation. Modules that account for similarities and differences between units of interests are coupled together. We did not use partitioning and layers design models yet.

Classes and Methods Design

Java Code

```java
package HW3;

public class Application implements Booking, Billing {
    Taxi taxi;
    User user;
    Driver driver;
    WelcomePage wp;
    Payroll payroll;

    public Application(TaxiDatabase db, User user) {
        this.taxi = db.callTaxi();
        this.user = user;
        this.driver = db.callDriver();
        openWelcomePage();
    }

    private void openWelcomePage() {
        wp = new WelcomePage();
        wp.menuControl(taxi, user, driver);
        generatePayroll();
    }

    @Override
    public void generatePayroll() {
        payroll = new Payroll();
        payroll.menuControl(taxi, user, driver);
        bookTaxi();
    }

    @Override
    public void bookTaxi() {
        System.out.println("Taxi booked");
    }

}
```

```java
package HW3;

public interface Booking {
    public void bookTaxi();
}
```

```java
package HW3;

public interface Billing {

    public void generatePayroll();
}
```

```java
package HW3;

public class Main {

    public static void main(String[] args) {
        Registration re = new Registration();
        re.setVisible(true);

    }

}
```

```java
package HW3;

import java.util.Scanner;

public final class Payroll {
    static Scanner sc = new Scanner(System.in);
    Payroll() {
    }

    void menuControl(Taxi taxi, User user, Driver driver) {
        try {
            String menuControl = JOptionPane.showInputDialog(null,
                    "********* Taxi Dispatch System *********" + "\n\n        1.  User Info \t\t "
                            + "\n        2.  Enter Chat" + "\n        3.  Driver info\t    "
                            + "\n        4.  Vehicle Info" + "\n\nQ.  Close Program" + "\n\n");
            switch (menuControl.toLowerCase()) {
            case "1":
                System.out.println(user);
                menuControl(taxi, user, driver);
                break;
            case "2":
                startChat();
                menuControl(taxi, user, driver);
                break;
            case "3":
                System.out.println(driver);
                menuControl(taxi, user, driver);
                break;
            case "4":
                System.out.println(taxi);
                break;
            case "q":
                int selectedValue = JOptionPane.showConfirmDialog(null, "Do you want to close the program ", "",
                        JOptionPane.YES_NO_OPTION, JOptionPane.ERROR_MESSAGE);
                if (selectedValue == 0) {
                    System.exit(0);
                } else
                    menuControl(taxi, user, driver);
                break;
            default:
                JOptionPane.showMessageDialog(null,
                        "        Invalid character entered ! \nPlease enter valid character from MENU",
                        "Data Input Error", JOptionPane.ERROR_MESSAGE);
                menuControl(taxi, user, driver);
            }
        } catch (Exception e) {
            e.getMessage();
        }
    }

    public void startChat() {
        System.out.println(" Welcome to the Chat. Here you can talk with the driver");

    }

}
```

```java
package HW3;

import java.util.Scanner;

public final class WelcomePage {
    static Scanner sc = new Scanner(System.in);

    WelcomePage() {
    }

    void menuControl(Taxi taxi, User user, Driver driver) {
        try {

            String address = JOptionPane.showInputDialog(null, "Current Location:");
            user.setCurrentLocation(address);
            String addressnew = JOptionPane.showInputDialog(null, "Destination Location:");
            user.setDestination(addressnew);

        } catch (Exception e) {
            e.getMessage();
        }
    }

}
```

```java
package HW3;

public class VIPTaxi extends Taxi {

    public VIPTaxi(String vehicleType, String registrationNumber, String model, int capacity,
            String vehicleAvailability) {
        super(vehicleType, registrationNumber, model, capacity, vehicleAvailability);
        price = 20;
    }

    @Override
    public String toString() {
        return super.toString();
    }

}
```

```java
package HW3;
import java.util.Random;

public abstract class Taxi {
    private String vehicleType, registrationNumber, model, vehicleAvailability;
    double price;
    Random rn = new Random();
    private int capacity;
    int minutes;

    public Taxi(String vehicleType, String registrationNumber, String model, int capacity, String vehicleAvailability) {
        this.capacity = capacity;
        this.model = model;
        this.registrationNumber = registrationNumber;
        this.vehicleType = vehicleType;
        this.vehicleAvailability = vehicleAvailability;
        minutes = rn.nextInt(40);
    }

    public String getVehicleType() {
        return vehicleType;
    }

    public String getRegistrationNumber() {
        return registrationNumber;
    }

    public String getModel() {
        return model;
    }

    public String getVehicleAvailability() {
        return vehicleAvailability;
    }

    public float getCapacity() {
        return capacity;
    }

    @Override
    public String toString() {
        return ("\nVehicle Type\t" + getVehicleType() + "\nRegistration No: " + getRegistrationNumber()
                + "\nModel:\t\t\t " + getModel() + "\nCapacity:\t\t " + getCapacity() + "\nAvailability:\t "
                + vehicleAvailability + "\"\n\tTotal Price:\t  " + price + "\n Your driver will arrive in " + minutes +" minutes"
                + "\n*******************************************************");
    }

}
```

```java
package HW3;

public class Driver {

    private String driverName, driverAvailability, rating, taxiType, taxiClass;
    Taxi taxi;

    public Driver(String driverName, String rating, Taxi taxi) {
        this.driverName = driverName;
        this.rating = rating;
        this.taxiType = taxi.getModel();
        this.taxiClass = taxi.getVehicleType();
        this.driverAvailability = taxi.getVehicleAvailability();
    }

    @Override
    public String toString() {
        return "\nNAME:\t      " + driverName + "\nAvailability: " + driverAvailability + "\nRating: " + rating
                + "\nTaxi Car Model: " + taxiType + "\nTaxi Class: " + taxiClass
                + "\n*********************************";
    }

}
```

```java
package HW3;


public class ComfortTaxi extends Taxi {

    public ComfortTaxi(String vehicleType, String registrationNumber, String model, int capacity,
            String vehicleAvailability) {
        super(vehicleType, registrationNumber, model, capacity, vehicleAvailability);
        price = 6.99;
    }

    @Override
    public String toString() {
        return super.toString();
    }
}
```

```java
package HW3;


public class EconomTaxi extends Taxi {

    public EconomTaxi(String vehicleType, String registrationNumber, String model, int capacity,
            String vehicleAvailability) {
        super(vehicleType, registrationNumber, model, capacity, vehicleAvailability);
        price = 3.5;
    }

    @Override
    public String toString() {
        return super.toString();
    }

}
```

```java
package HW3;

import java.util.ArrayList;

public final class TaxiDatabase {
    static Taxi taxi, taxi2, taxi3, taxi4, taxi5, taxi6;
    static Driver driver1, driver2, driver3, driver4, driver5, driver6;
    ArrayList<Taxi> taxiData = new ArrayList<Taxi>();
    ArrayList<Driver> driverData = new ArrayList<Driver>();
    Random rn = new Random();
    int num;

    public TaxiDatabase() {
        createTaxi();
        createDriver();
        num = rn.nextInt(taxiData.size());
    }

    public Taxi callTaxi() {
        return taxiData.get(num);
    }

    public Driver callDriver() {
        return driverData.get(num);
    }

    private void createDriver() {
        driver1 = new Driver("Pustexanim", "5", taxi);
        driver2 = new Driver("Bagdagul", "5", taxi2);
        driver3 = new Driver("Aligulam", "5", taxi3);
        driver4 = new Driver("Tofig", "5", taxi4);
        driver5 = new Driver("Teodor", "5", taxi5);
        driver6 = new Driver("Ali", "5", taxi6);
        driverData.add(driver1);
        driverData.add(driver2);
        driverData.add(driver3);
        driverData.add(driver4);
        driverData.add(driver5);
        driverData.add(driver6);
    }

    private void createTaxi() {
        taxi = new EconomTaxi("EconomTaxi", "12C4956", "Hyundai", 65172, "Yes");
        taxi2 = new VIPTaxi("VIPTaxi", "14C89365", "Ford", 33892, "Yes");
        taxi3 = new ComfortTaxi("ComfortTaxi", "15C46046", "VW", 23897, "Yes");
        taxi4 = new ComfortTaxi("ComfortTaxi", "14C38492", "Nissan", 29418, "Yes");
        taxi5 = new ComfortTaxi("ComfortTaxi", "10C99393", "Skoda", 89678, "Yes");
        taxi6 = new ComfortTaxi("ComfortTaxi", "15C23796", "Seat", 12812, "Yes");
        taxiData.add(taxi);
        taxiData.add(taxi2);
        taxiData.add(taxi3);
        taxiData.add(taxi4);
        taxiData.add(taxi5);
        taxiData.add(taxi6);
    }

}
```

```java
package HW3;

public class User {
    private String name, surname, mobileNum, currentLocation, destination;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getMobileNum() {
        return mobileNum;
    }

    public void setMobileNum(String mobileNum) {
        this.mobileNum = mobileNum;
    }

    public String getCurrentLocation() {
        return currentLocation;
    }

    public void setCurrentLocation(String currentLocation) {
        this.currentLocation = currentLocation;
    }

    public User() {
        // TODO Auto-generated constructor stub
    }

    public User(String name, String surname, String mobileNum, String currentLocation) {
        this.mobileNum = mobileNum;
        this.currentLocation = currentLocation;
        this.name = name;
        this.surname = surname;
    }

    @Override
    public String toString() {
        return ("\nName: " + name + " " + surname + "\nMobile No: " + mobileNum + "\nCurrentLocation: "
                + currentLocation + "\nDestination: " + destination
                + "\n*****************************************************");
    }

    public String getDestination() {
        return destination;
    }

    public void setDestination(String destination) {
        this.destination = destination;
    }
}
```

```java
package HW3;

import java.awt.EventQueue;

public class RegistrationUser extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField user;
    private JTextField email;
    private JTextField pass;
    private JLabel lblNewLabel;
    private JLabel lblNewLabel_1;
    private JLabel lblNewLabel_2;
    String url = "jdbc:mysql://localhost:3306/test?useTimezone=true&serverTimezone=UTC";
    String usern = "root";
    String password = "";
    static RegistrationUser frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new RegistrationUser();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public RegistrationUser() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 326, 504);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(null);
        setContentPane(contentPane);

        user = new JTextField();
        user.setBounds(73, 149, 174, 27);
        contentPane.add(user);
        user.setColumns(10);
```

```
        email = new JTextField();
        email.setColumns(10);
        email.setBounds(73, 223, 174, 27);
        contentPane.add(email);

        pass = new JPasswordField();
        pass.setColumns(10);
        pass.setBounds(73, 306, 174, 27);
        contentPane.add(pass);

        lblNewLabel = new JLabel("Username");
        lblNewLabel.setBounds(73, 121, 88, 22);
        contentPane.add(lblNewLabel);

        lblNewLabel_1 = new JLabel("Phone");
        lblNewLabel_1.setBounds(73, 187, 88, 22);
        contentPane.add(lblNewLabel_1);

        lblNewLabel_2 = new JLabel("Password");
        lblNewLabel_2.setBounds(73, 273, 88, 22);
        contentPane.add(lblNewLabel_2);

        JButton btnNewButton = new JButton("Register");
        btnNewButton.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    Connection con = DriverManager.getConnection(url, usern, password);
                    PreparedStatement ps = con
                            .prepareStatement("insert into user(user_name, phone,password) values(?,?,?);");
                    ps.setString(1, user.getText());
                    ps.setString(2, email.getText());
                    ps.setString(3, pass.getText());
                    int x = ps.executeUpdate();
                    if (x > 0) {
                        System.out.println("Registered Successfully");
                    } else {
                        System.out.println("Failed");
                    }
                } catch (Exception e1) {
                    System.out.println(e1);
                }

            }
        });
        btnNewButton.setBounds(111, 363, 89, 23);
        contentPane.add(btnNewButton);

        JLabel login = new JLabel("Log in");
        login.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {

                LoginUser l = new LoginUser();
                l.setVisible(true);
                l.toFront();
            }
        });
        login.setBounds(139, 397, 45, 45);
        contentPane.add(login);
    }
}
```

```java
package HW3;

import java.awt.EventQueue;

public class RegistrationDriver extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    static RegistrationDriver frame;
    private JTextField user;
    private JTextField email;
    private JTextField pass;
    private JLabel lblNewLabel;
    private JLabel lblNewLabel_1;
    private JLabel lblNewLabel_2;
    private JTextField textField;
    private JLabel lblNewLabel_2_2;
    private JTextField textField_1;
    private JButton register;
    String url = "jdbc:mysql://localhost:3306/test?useTimezone=true&serverTimezone=UTC";
    String usern = "root";
    String password = "";

    /**
     * Launch the application.
     */

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new RegistrationDriver();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */

    public RegistrationDriver() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 326, 504);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(null);
        setContentPane(contentPane);
        user = new JTextField();
        user.setBounds(10, 104, 174, 27);
        contentPane.add(user);
        user.setColumns(10);

        email = new JTextField();
        email.setColumns(10);
        email.setBounds(10, 173, 174, 27);
        contentPane.add(email);

        pass = new JPasswordField();
        pass.setColumns(10);
        pass.setBounds(10, 238, 174, 27);
        contentPane.add(pass);

        lblNewLabel = new JLabel("Username");
        lblNewLabel.setBounds(10, 71, 88, 22);
        contentPane.add(lblNewLabel);

        lblNewLabel_1 = new JLabel("Phone");
        lblNewLabel_1.setBounds(10, 140, 88, 22);
        contentPane.add(lblNewLabel_1);
```

```
        lblNewLabel_2 = new JLabel("Password");
        lblNewLabel_2.setBounds(10, 211, 88, 22);
        contentPane.add(lblNewLabel_2);

        textField = new JTextField();
        textField.setColumns(10);
        textField.setBounds(10, 309, 174, 27);
        contentPane.add(textField);

        JLabel lblNewLabel_2_1 = new JLabel("Car Model");
        lblNewLabel_2_1.setBounds(10, 276, 88, 22);
        contentPane.add(lblNewLabel_2_1);

        lblNewLabel_2_2 = new JLabel("Car Number");
        lblNewLabel_2_2.setBounds(10, 347, 88, 22);
        contentPane.add(lblNewLabel_2_2);

        textField_1 = new JTextField();
        textField_1.setColumns(10);
        textField_1.setBounds(10, 382, 174, 27);
        contentPane.add(textField_1);

        register = new JButton("Register");
        register.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    Connection con = DriverManager.getConnection(url, usern, password);
                    PreparedStatement ps = con.prepareStatement(
                            "insert into driver(driver_name, phone,password, car_model,car_num) values(?,?,?,?,?);");
                    ps.setString(1, user.getText());
                    ps.setString(2, email.getText());
                    ps.setString(3, pass.getText());
                    ps.setString(4, textField.getText());
                    ps.setString(5, textField_1.getText());
                    int x = ps.executeUpdate();
                    if (x > 0) {
                        System.out.println("Registered Successfully");
                    } else {
                        System.out.println("Failed");
                    }
                } catch (Exception e1) {
                    System.out.println(e1);
                }

            }
        });
        register.setBounds(10, 433, 89, 23);
        contentPane.add(register);

        JLabel login = new JLabel("Log in");
        login.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                LoginDriver l = new LoginDriver();
```

```java
package HW3;

import java.awt.Color;

public class Registration extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    static Registration frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new Registration();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public Registration() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 326, 504);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(null);

        setContentPane(contentPane);

        JButton usid = new JButton("User");
        usid.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                RegistrationUser us = new RegistrationUser();
                us.setVisible(true);
                driv.toFront();

            }

        });
        drivid.setBounds(96, 297, 115, 31);
        contentPane.add(drivid);

        JLabel image = new JLabel("");
        image.setBackground(Color.LIGHT_GRAY);
        image.setIcon(new ImageIcon("/OOAD/src/icon.jpg"));
        image.setBounds(78, 34, 148, 162);
        contentPane.add(image);
    }
}
```

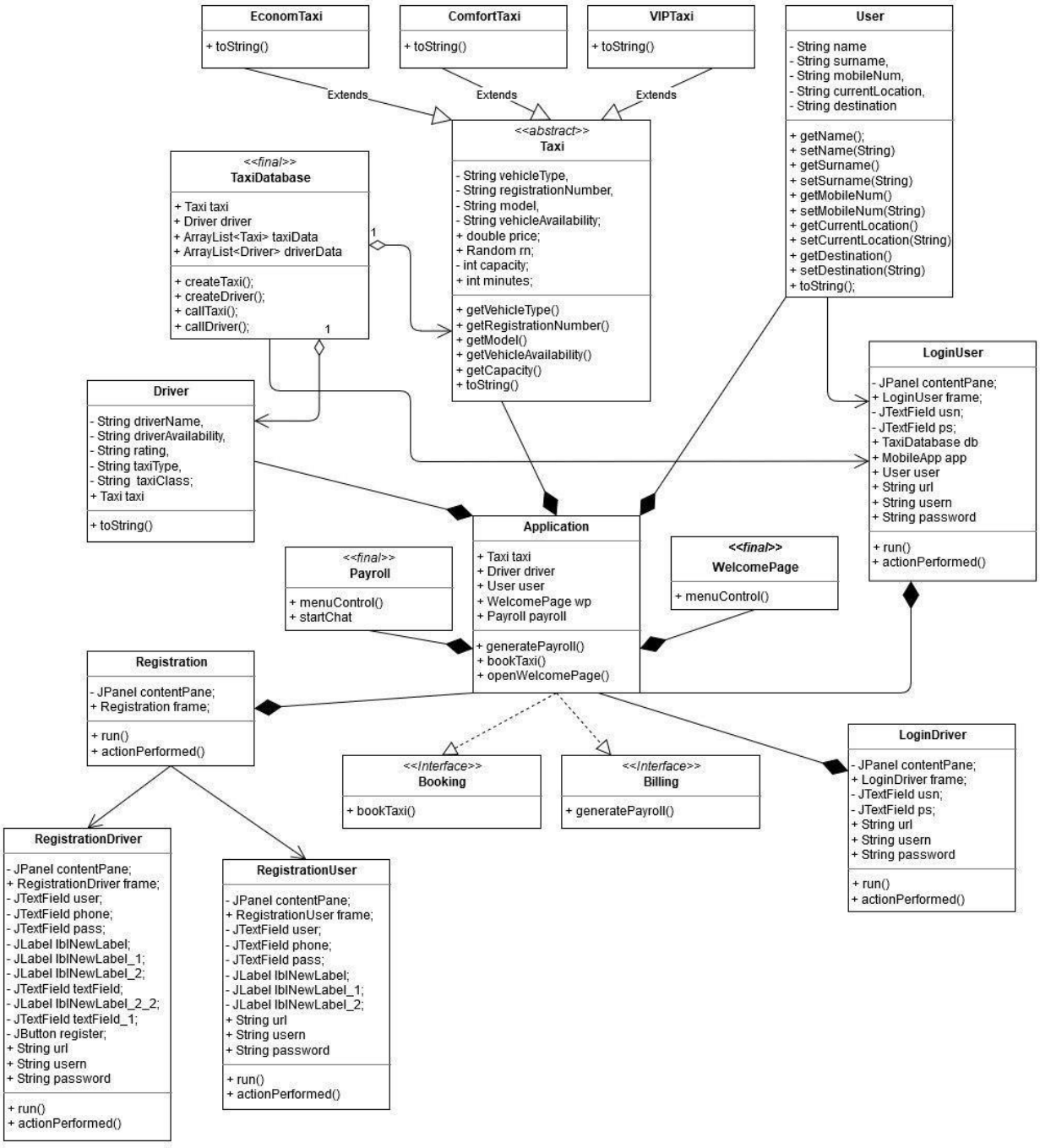
```java
package HW3;

import java.awt.Color;

public class LoginUser extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField usn;
    private JTextField ps;
    static TaxiDatabase db = new TaxiDatabase();
    static Application app;
    static User user = new User();
    String url = "jdbc:mysql://localhost:3306/test?useTimezone=true&serverTimezone=UTC";
    String usern = "root";
    String password = "";
    static LoginUser frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new LoginUser();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public LoginUser() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 326, 504);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(null);
        setContentPane(contentPane);

        usn = new JTextField();
        usn.setBounds(72, 175, 173, 28);
        contentPane.add(usn);
        usn.setColumns(10);

        ps = new JPasswordField();
        ps.setColumns(10);
        ps.setBounds(72, 257, 173, 28);
        contentPane.add(ps);

        JLabel lblNewLabel = new JLabel("Username");
        lblNewLabel.setBounds(72, 132, 73, 21);
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Password");
        lblNewLabel_1.setBounds(72, 225, 73, 21);
        contentPane.add(lblNewLabel_1);

        JButton login = new JButton("Login");
        login.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    Connection con = DriverManager.getConnection(url, usern, password);
                    Statement s = con.createStatement();
                    String sql = "Select * from user where user_name ='" + usn.getText() + "' and password='"
                            + ps.getText() + "'";
                    ResultSet rs = s.executeQuery(sql);
                    if (rs.next()) {
                        JOptionPane.showMessageDialog(null, "Login done");
                        app = new Application(db, user);
                    } else {
                        JOptionPane.showMessageDialog(null, "Failed");
                    }

                } catch (Exception e1) {
                    System.out.println(e1);
                }

            }
```

```
        });
        login.setBounds(103, 321, 89, 23);
        contentPane.add(login);
        JLabel registration = new JLabel("Registration");
        registration.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                RegistrationUser re = new RegistrationUser();
                re.setVisible(true);
                re.toFront();
            }
        });
        registration.setBounds(116, 383, 89, 45);
        contentPane.add(registration);

        JLabel image1 = new JLabel("");
        image1.setBackground(Color.LIGHT_GRAY);
        image1.setIcon(new ImageIcon("/OOAD/src/user.png"));
        image1.setBounds(103, 2, 156, 119);
        contentPane.add(image1);
    }
}
```

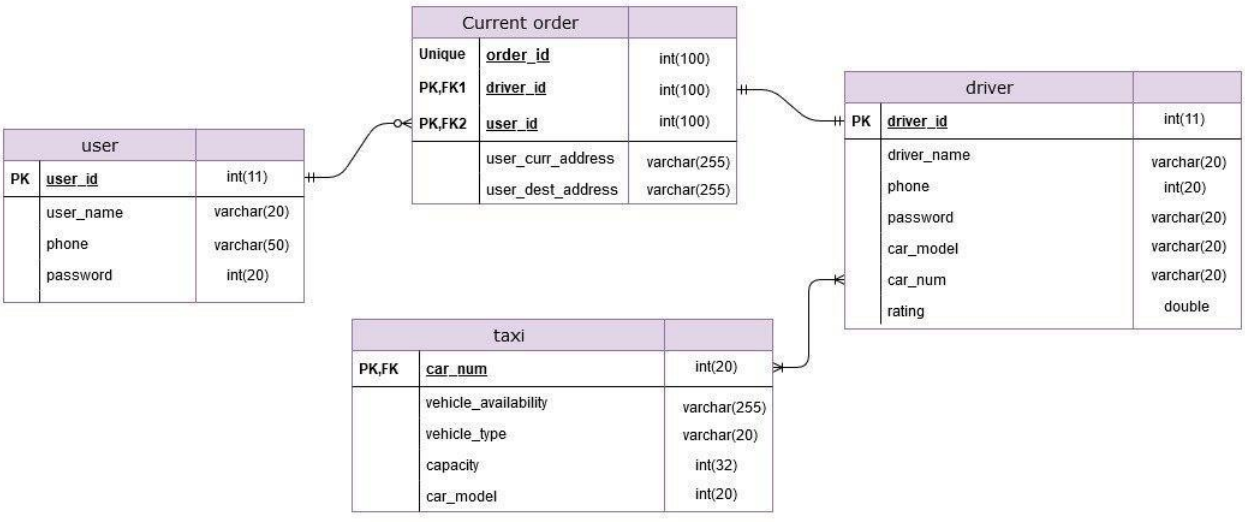
```
package HW3;

import java.awt.Color;

public class LoginDriver extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField usn;
    private JTextField ps;
    String url = "jdbc:mysql://localhost:3306/test?useTimezone=true&serverTimezone=UTC";
    String usern = "root";
    String password = "";
    static LoginDriver frame;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    frame = new LoginDriver();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public LoginDriver() {

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 326, 504);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        contentPane.setLayout(null);
        setContentPane(contentPane);
```

```
        usn = new JTextField();
        usn.setBounds(72, 175, 173, 28);
        contentPane.add(usn);
        usn.setColumns(10);

        ps = new JPasswordField();
        ps.setColumns(10);
        ps.setBounds(72, 257, 173, 28);
        contentPane.add(ps);

        JLabel lblNewLabel = new JLabel("Username");
        lblNewLabel.setBounds(72, 132, 73, 21);
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Password");
        lblNewLabel_1.setBounds(72, 225, 73, 21);
        contentPane.add(lblNewLabel_1);

        JButton login = new JButton("Login");
        login.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    Class.forName("com.mysql.jdbc.Driver");
                    Connection con = DriverManager.getConnection(url, usern, password);
                    Statement s = con.createStatement();
                    String sql = "Select * from driver where driver_name ='" + usn.getText() + "' and password='"
                            + ps.getText() + "'";
                    ResultSet rs = s.executeQuery(sql);
                    if (rs.next()) {
                        JOptionPane.showMessageDialog(null, "Login done");
                    } else {
                        JOptionPane.showMessageDialog(null, "Failed");
                    }

                } catch (Exception e1) {
                    System.out.println(e1);
                }

            }
        });
        login.setBounds(103, 321, 89, 23);
        contentPane.add(login);
        JLabel registration = new JLabel("Registration");
        registration.addMouseListener(new MouseAdapter() {

            @Override
            public void mouseClicked(MouseEvent e) {
                RegistrationDriver reg = new RegistrationDriver();
                reg.setVisible(true);
                reg.toFront();
            }
        });
        registration.setBounds(116, 383, 89, 45);
        contentPane.add(registration);

        JLabel image1 = new JLabel("");
        image1.setBackground(Color.LIGHT_GRAY);
        image1.setIcon(new ImageIcon("/OOAD/src/driver.png"));
        image1.setBounds(103, 2, 105, 119);
        contentPane.add(image1);
    }

}
```

Class Diagram



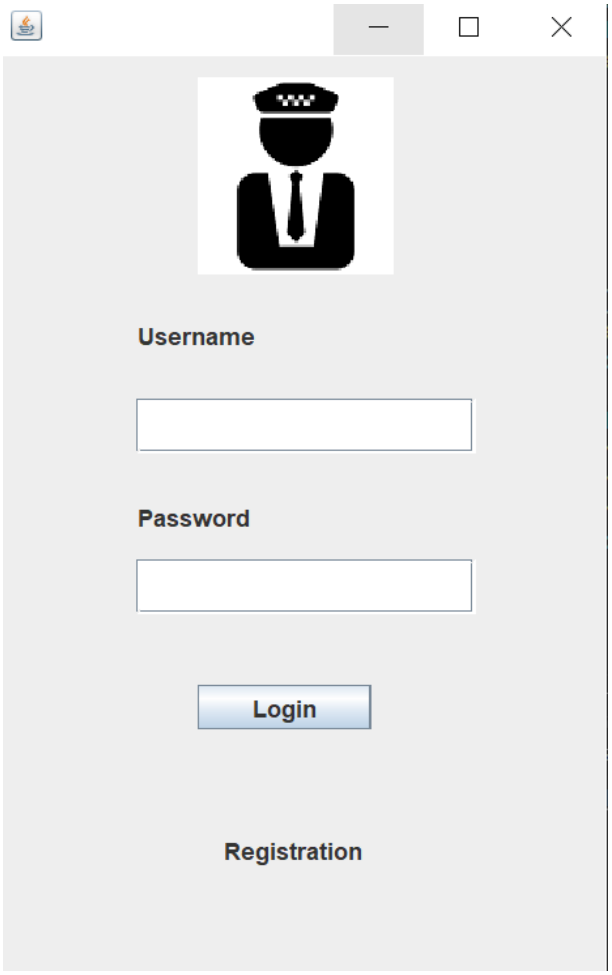
Data Management Layer

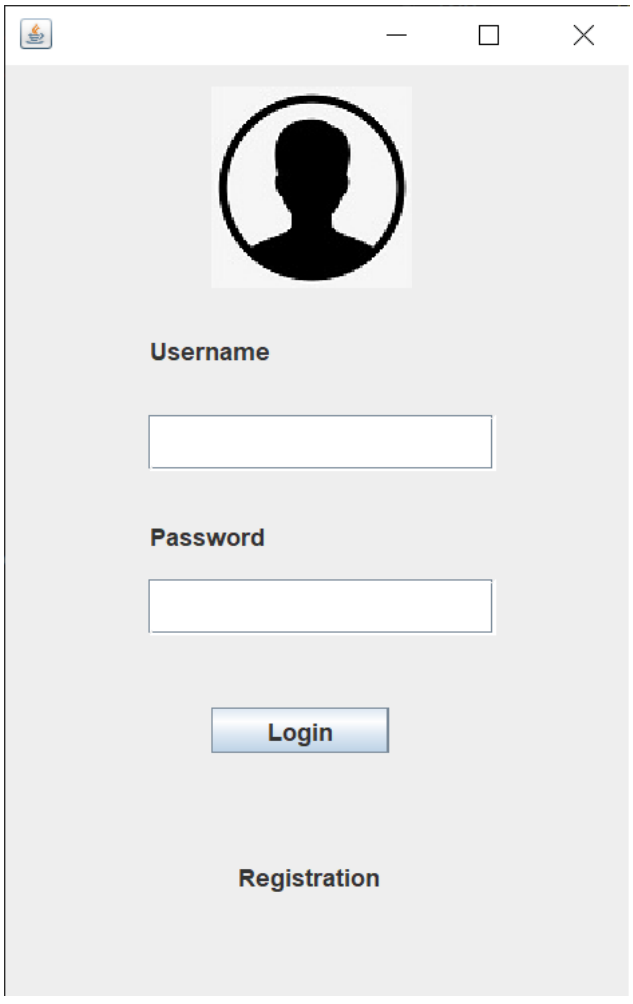


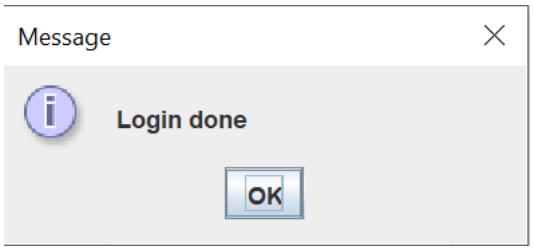
User Interface Layer

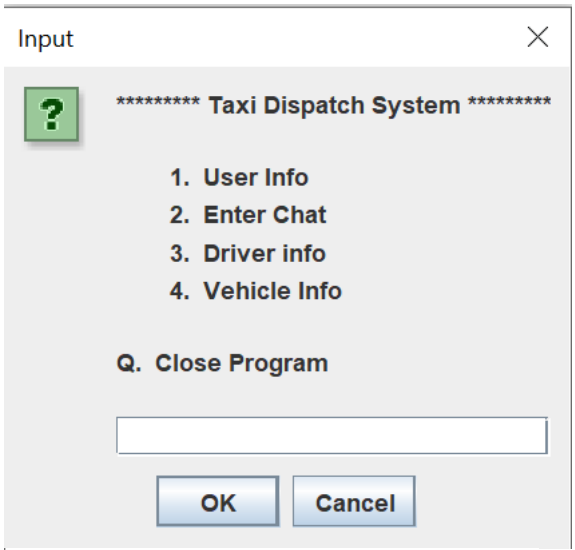
At the very start of our application the user enters the Startup page where he chooses between Customer and Driver buttons. Then depending on the button chosen the user enters Customer or Driver's Registration page. At this page the customer can see the input fields for Username, Phone number, and Password. Additionally, the customer can switch to Login page if he has already registered. In case of driver, the Registration page for the driver looks different from the customer's in terms of input fields. In addition to Username, Phone number and Password driver should enter the Car Model and Card Number for the successful registration. Right after successful registration or login the user enters the Home page of the application where he indicates his current location and destination. Afterwards the user can choose between several options such as: 1) see Customer information, 2) see Driver information, 3) See vehicle information 4) Enter the chat and 5) book a taxi. For the first option the system produces a payroll with the user information which is retrieved from the database. Consequently, for the second option the system produces a payroll of driver information for the customer. Then, as for a third option the vehicle information is generated: mainly the car model, the car ID, color and the status (VIP, Comfort, Economy). Moreover, we would have a chat option which will enable driver and a customer to have the ability to communicate with the use of our system. And finally, a book taxi option. The user can reach any desired page of the application just within 3-4 clicks. Additionally, the update functionality will be added to some of the input texts meaning if the user would like to change some information about him after registration he will be able to change it. The change will be assosiated with the MySQL database where the data will be updated accordingly. For the simplicity of use the Return buttons were added so that a user can switch back to previous pages of application as well as a Quit option. The data in our application is retrieved from the MySQL database.

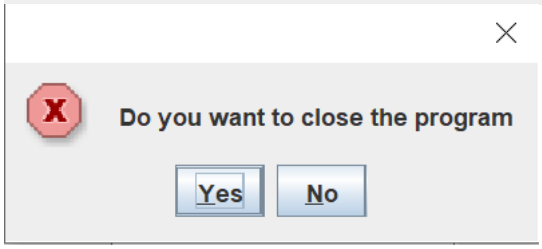

TAXI
User
Driver


Username
Password
Login
Registration


Username
Password
Login
Registration


Message
Login done
OK


Input
********* Taxi Dispatch System *********
1. User Info
2. Enter Chat
3. Driver info
4. Vehicle Info
Q. Close Program
OK
Cancel


Do you want to close the program
Yes
No

Physical Architecture Layer

Our system is a Client-Server based system with a 2-tier architecture. It mainly consists of Database tier and Client-Application tier. In our system the clients are drivers and the users. A Database server handles the data management layer which consists of data storage in our case it is database and methods for storing and retrieving data from the data storage. For our system we used MySQL which is an open-source relational database management system. For the connection between Application and Database we installed and made use of JDBC which stands for Java Database Connectivity, It is a Java API to connect and execute the query with the database. The system is programmed on Java programming language for the purpose of using Object Oriented Programming style. We used Eclipse which is an integrated development environment for developing using Java. Our system will mainly work as a Desktop application. Considering the programming language to be Java, each and every device that has Java installed will be able to execute and run our application. Considering the limited time, we were not able to design and develop a mobile application, however it is planned to be the future work for our system.