Pimpri Chinchwad College Of Engineering

Even Semester 22-23

Course: PBL-2

End term Project Documentation

Vehicle Warehouse Management System

SY A1 batch:

1	*Aditya Agre*	SYCOA006
2	*Yash Athawale*	SYCOA013
3	*Atharv Bardapurkar*	SYCOA019

Contents

Vehicle Warehouse Management System

Abstract

The Vehicle Warehouse Management System is a software solution aimed at automating and streamlining the warehouse management processes for second-hand vehicle resellers. With the growing prominence of the vehicle resale industry, there is a need for efficient management of warehouses to handle tasks such as sorting, storage, servicing, maintenance, quality assessment, and transport of vehicles. The existing operations of many vehicle resellers are disorganized, leading to challenges in managing inventory, tracking vehicles, assigning tasks to workers, and maintaining accurate records.

To address these challenges, our software system aims to automate the logistics processes involved in vehicle warehousing. By leveraging technology, we seek to reduce human effort, enhance efficiency, minimize errors, and provide a smoother workflow for warehouse administrators and workers. The system will offer functionalities such as inventory management, vehicle tracking, maintenance scheduling, quality assessment, and data management for vehicles stored in warehouses.

Software Requirement Specification

1. **Introduction:**

The Vehicle Warehouse Management System is designed to revolutionize the way second-hand vehicle resellers manage their warehouses. The vehicle resale industry has witnessed rapid growth and increasing organization in recent years. However, the internal operations of many resellers are still largely disorganized, leading to inefficiencies and coordination challenges. Our software system aims to address these issues by providing a comprehensive solution for warehouse management.

The system will offer a wide range of features to automate key warehouse processes. It will enable warehouse administrators to efficiently assign parking stalls to vehicles, track the location and condition of vehicles within the warehouse, and maintain detailed inventory records. The system will also facilitate the assignment of drivers for each incoming vehicle, ensuring smooth coordination and optimized workflow. Additionally, the software will support the management of warehouse layouts, accommodating the unique requirements of different storage complexes.

One of the crucial aspects of warehouse management is maintaining accurate and up-to-date data related to each vehicle. Our system will enable administrators to record essential information such as car model, number, quality check records, estimated price, and mileage for each vehicle.

Furthermore, it will leverage quality assessment reports to calculate estimated prices and provide characteristic-specific search options. The system will notify administrators about pending maintenance jobs and facilitate the assignment of maintenance and quality assurance workers based on availability.

In compliance with government regulations, the system will ensure adherence to the rules and regulations formulated by authorities such as the Ministry of Road Transport and Highways (MoRTH). It will notify administrators about vehicles that do not meet the minimum standards set by MoRTH and calculate the remaining years of use for each vehicle. The software will also promote sustainability by storing data related to the carbon emissions of vehicles and suggesting electric vehicles (EVs) within the same price range as combustion vehicles.

To ensure a high-quality user experience, the system will prioritize high response speed and feature a simple user interface. Warehouse administrators and workers will be able to adapt to the system with ease, reducing the learning curve and maximizing efficiency. The software will prioritize security and data privacy, implementing robust measures to protect sensitive information such as vehicle inventory, customer data, and financial records. It will be reliable, consistently performing its intended functions without failure, and scalable to accommodate a growing user base. Provisions for regular maintenance will be incorporated to ensure the system's long-term maintainability.

In summary, the Vehicle Warehouse Management System aims to provide a comprehensive and efficient solution for automating warehouse management processes in the second-hand vehicle resale industry. By leveraging technology, the system will streamline operations, reduce human effort, minimize errors, and enhance the overall efficiency of vehicle resellers' warehouses.

End User:
Warehouse Administrator.

Stake-Holders:

1. **Direct Stakeholders:**
   - Warehouse administrators.
   - Businesses such as Cars24, CarDekho, OlxAutos,etc
   - Warehouse workers, on-ground staff.
   - Warehouse owners indulging in large scale leasing.

2. **Indirect Stakeholders:**
   - Secondary workers such as technicians, quality control specialists. (Outsourced)
   - Spare-part suppliers
   - Ministry of Road Transport and Highways (MoRTH).
   - Car buyers and resellers.

2. Scope

Riding the digital wave, India's **used** car market is set to grow at a compounded annual growth of **11%** and likely to touch sales of up to **8.3 million units** by financial year 26 as more people have been opting for pre-owned cars for personal mobility. This growth is driven by increased sales and a rise in online sales platforms.

So, developing a software to reduce human efforts and tackling the above described issues will bring greater organization to the used-car industry. As these organizations are looking to expand their businesses to underdeveloped areas, efficient management systems will be needed. And hence, we believe that the target market for this software will only increase.

- **Deliverables:** Inventory Management, Quality Assessment, Adaptability.

- **Resources:** Web services for backend servers, Project developer team, Funds

- **Risks:** Changing product requirements, etc.

3. Functional Requirements:

- Assigning parking stalls to particular cars, etc.
- Assigning drivers for each incoming vehicle.
- Storing and accommodating to warehouse layouts.
- Maintaining inventory data of all the vehicles present at a given warehouse.
- Maintaining data associated with each vehicle such as car model, number, quality check records, estimated price, mileage,etc.
- Calculating estimated price of vehicles based on reports of quality assessments.
- Facilitating characteristic specific search options. For instance, if the manager wants to know how many SUVs are present at a given storage complex.
- Notifying the manager about pending maintenance jobs, etc.
- Maintaining data of all workers at the warehouse, if they are available and if not, which task they have been assigned.
- Assigning maintenance, quality assurance workers to particular vehicles based on availability.
- System should comply all the rules and regulation formulated by Government authorities.
- Give notifications about the vehicles which are not satisfying minimum standards set by MoRTH
- Must be able to calculate the number of years remaining for the use of vehicle.
- Should be able to suggest EVs in the same price range over combustion vehicles.
- Must store the data related to carbon emission of that vehicle.

4. Non-Functional Requirements:

- High response **speed** for decent user experience.
- Should have **simple UI** so that users can adapt with ease.
- **Security** & Data **privacy**.
- **Reliability:** consistently perform its intended functions without failure
- **Scalability**: Cater to a growing user-base.
- **Maintainability**: Provisions for regular maintenance.

5. User Interface:

1. The user interface shall provide an organised and easy to navigate layout.
2. It should include features such as functionality, filters and sortable columns to facilitate data exploration and management.
3. Maintain pages for various functionalities.
4. Act as an interface for data flow, taking inputs and displaying outputs.

6. System Architecture:

This software follows a tree-tier architecture.

a. Presentation: Java.
b. Application Logic: Java
c. Database: MySQL

7. Assumptions and Constraints:

a. The WMS assumes that the warehouse has a reliable network infrastructure and hardware devices (barcode scanners, RFID readers) to support the software's functionalities.
b. The system should be compatible with the existing ERP system used by the organisation.

8. Dependencies:

1. The WMS shall integrate with the organization's ERP system to synchronize inventory data and order information.
2. The software may rely on third-party shipping providers' APIs to generate shipping labels and track shipment status.

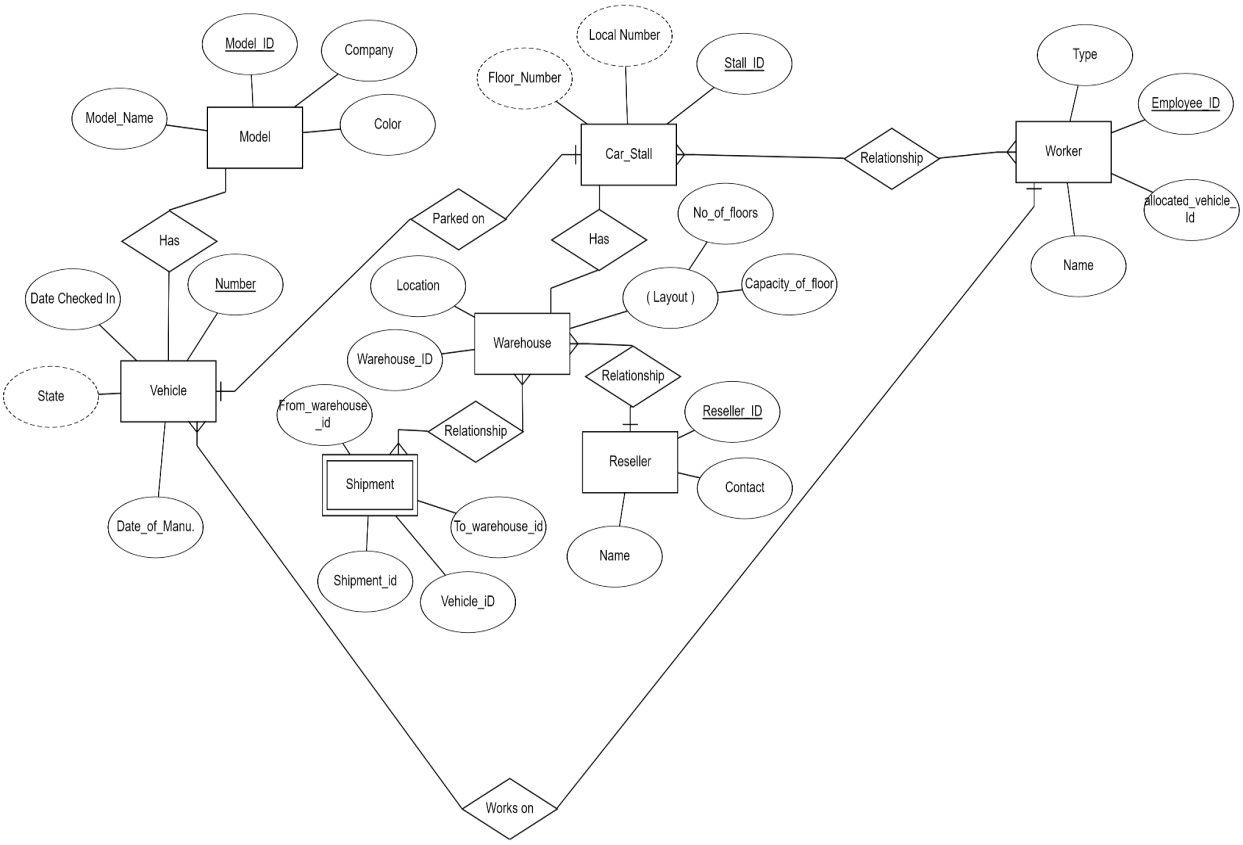
9. Verification and Validation:

1. The software will undergo comprehensive testing, including unit testing, integration testing, and user acceptance testing.
2. Test cases and acceptance criteria will be defined to ensure the software meets the specified requirements.

10. Appendices:

1. Sample wireframes/mockups of the user interface.
2. Entity-relationship diagrams or database schema for data structure representation.

Conceptual Design using ER feature



Relational Model in appropriate Normalized Form

Entities:

1. Vehicle:
- Strong Entity: Vehicle is a strong entity as it has its own unique identifier (Vehicle ID) and contains
essential attributes.

2. Model:
- Strong Entity: Model is a strong entity as it has its own unique identifier (Model ID) and contains essential attributes.

3. Warehouse:
- Strong Entity: Warehouse is a strong entity as it has its own unique identifier (Warehouse ID) and contains essential attributes.

4. Reseller:
- Strong Entity: Reseller is a strong entity as it has its own unique identifier (Reseller ID) and contains
essential attributes.

5. Shipment:
- Weak Entity: Shipment is a weak entity as it does not have its own unique identifier. It relies on the
Vehicle ID and Shipment ID together to form a unique identifier.

Relationships:

1. Vehicle - Model:
Cardinality: One-to-Many (1:M) relationship. One vehicle can be associated with one model, but one model can be associated with many vehicles.
Participation: Total participation of Vehicle (mandatory) and partial participation of Model (optional). Each vehicle must be associated with a model, but a model may or may not have any associated vehicles.

2. Vehicle - Warehouse:
Cardinality: Many-to-One (M:1) relationship. Many vehicles can be located in one warehouse, but one vehicle is located in only one warehouse at a time.
Participation: Total participation of Vehicle (mandatory) and partial participation of Warehouse (optional). Each vehicle must be located in a warehouse, but a warehouse may or may not have any associated vehicles.

3. Reseller - Warehouse:

Cardinality: One-to-Many (1:M) relationship. One reseller can have many warehouses, but one warehouse is associated with only one reseller.

Participation: Total participation of Reseller (mandatory) and partial participation of Warehouse (optional). Each reseller must have at least one warehouse, but a warehouse may or may not be associated with any reseller.

4. Shipment - Vehicle - Warehouse:

Cardinality: Many-to-Many (M:M) relationship. Many shipments can involve the same vehicle and many shipments can involve the same warehouse.

5. Worker - Vehicle

(One-to-One or Optional One-to-Many):

One Worker is allocated to one Vehicle for maintenance or quality checks.

One Vehicle can be allocated to one Worker at a time.

It's possible to have a Worker allocated to multiple Vehicles (optional).

----------**Reduced & 1NF**—----------

1. Vehicle (Table Name)
 - Vehicle(Vehicle_Identification_no, Date_of_Manu, Model, Year, State, Check_In_Date)
2. Model (Table Name)
 - Model(Model ID, Model Name, Company, Color)
3. Warehouse (Table Name)
 - Warehouse(Warehouse ID, Location, Layout, No_of_floors, capacity_of_floor)
4. Reseller (Table Name)
 - Reseller(Reseller ID, Name, Contact Details)
5. Shipment (Table Name)
 - Shipment(Shipment ID, Vehicle ID, From Warehouse ID, To Warehouse ID)
6. Worker (Table Name)
 - Worker(Worker ID, Name, Vehicle Allocated ID, Worker Type)
7. Car_Stall (Table Name)
 - Car_Stall(Stall ID, Floor Number, Local Number)
8.Shipment_Vehicle_Warehouse (Table Name)
 - Shipment_Vehicle_Warehouse (Shipment_ID, Vehicle_ID, From_Warehouse_ID, To_Warehouse_ID)

—----------**2NF**----------

1. Vehicle (Table Name)
 - Vehicle(Vehicle_Identification_no, Date_of_Manu, Model, Year, State, Check_In_Date)

Normalized Vehicle Table:
- Vehicle(Vehicle_Identification_no [PK], Date_of_Manu, Model [FK], State, Check_In_Date)
2. Model (Table Name)
- Model(Model ID [PK], Model Name, Company, Color)
The Model table is already in 2NF.
3. Warehouse (Table Name)
- Warehouse(Warehouse ID [PK], Location, Layout, No_of_floors, capacity_of_floor)
The Warehouse table is already in 2NF.
4. Reseller (Table Name)
- Reseller(Reseller ID [PK], Name, Contact Details)
The Reseller table is already in 2NF.
5. Shipment (Table Name)
- Shipment(Shipment ID [PK], Vehicle ID [FK], From Warehouse ID [FK], To Warehouse ID [FK])
The Shipment table is already in 2NF.
6. Worker (Table Name)
- Worker(Worker ID [PK], Name, Vehicle Allocated ID [FK], Worker Type)
The Worker table is already in 2NF.
7. Car_Stall (Table Name)
- Car_Stall(Stall ID [PK], Floor Number, Local Number)
The Car_Stall table is already in 2NF.
8. Shipment_Vehicle_Warehouse (Table Name)
- Shipment_Vehicle_Warehouse (Shipment_ID [FK], Vehicle_ID [FK], From_Warehouse_ID [FK], To_Warehouse_ID [FK])

—----------**3NF**—----------

To normalize the tables into 3NF, we need to eliminate any transitive dependencies and ensure that each non-key attribute depends only on the primary key.
1. Vehicle (Table Name)
- Vehicle(Vehicle_Identification_no [PK], Date_of_Manu, Model [FK], State, Check_In_Date)
The Vehicle table is already in 3NF.
2. Model (Table Name)
- Model(Model ID [PK], Model Name, Company, Color)
The Model table is already in 3NF.
3. Warehouse (Table Name)
- Warehouse(Warehouse ID [PK], Location, Layout, No_of_floors, capacity_of_floor)
The Warehouse table is already in 3NF.
4. Reseller (Table Name)
- Reseller(Reseller ID [PK], Name, Contact Details)
The Reseller table is already in 3NF.
5. Shipment (Table Name)
- Shipment(Shipment ID [PK], Vehicle ID [FK], From Warehouse ID [FK], To Warehouse ID [FK])

The Shipment table is already in 3NF.

6. Worker (Table Name)

- Worker(Worker ID [PK], Name, Vehicle Allocated ID [FK], Worker Type)

The Worker table is already in 3NF.

7. Car_Stall (Table Name)

- Car_Stall(Stall ID [PK], Floor Number, Local Number)

The Car_Stall table is already in 3NF.

8. Shipment_Vehicle_Warehouse (Table Name)

- Shipment_Vehicle_Warehouse (Shipment_ID [FK], Vehicle_ID [FK], From_Warehouse_ID [FK], To_Warehouse_ID [FK])

The Shipment_Vehicle_Warehouse table is already in 3NF

UML Design

Requirement Elaboration:

1. Assigning drivers for each incoming vehicle

a. Understand the requirement: System should be able to identify available drivers, schedule of arriving vehicles including the number of vehicles and then assign the drivers to the subsequent vehicles.

b. Decomposition:

i. Incoming vehicles: Vehicles arriving at the warehouse.
ii. Drivers: Available drivers who can be assigned to the vehicles.
iii. Parking slots: Spaces designated for parking the vehicles in the warehouse.
iv. Assignment process: The mechanism by which drivers are allocated to incoming vehicles and parking slots.

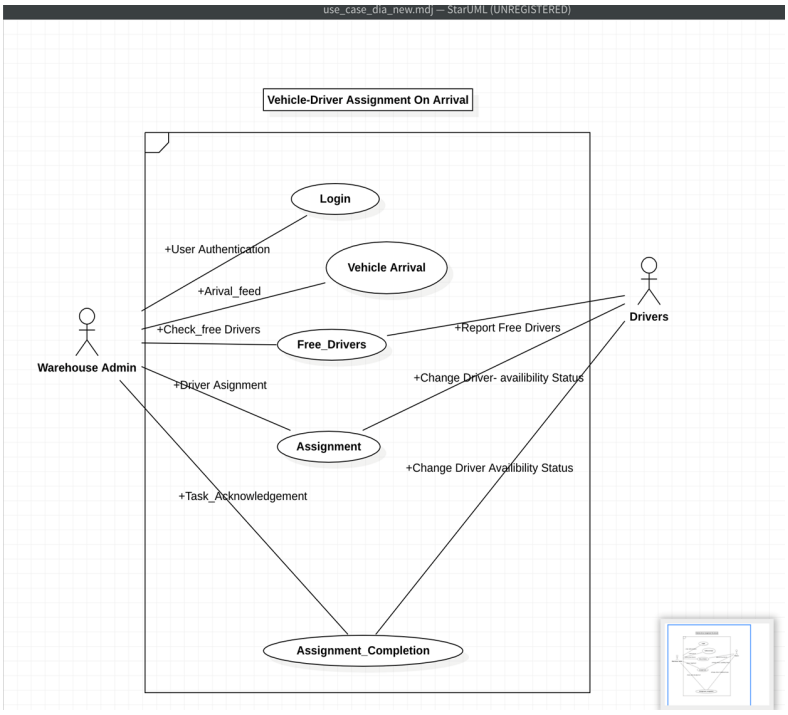
c. Actors:

i. Primary: Warehouse administrator.
ii. Secondary: Drivers.

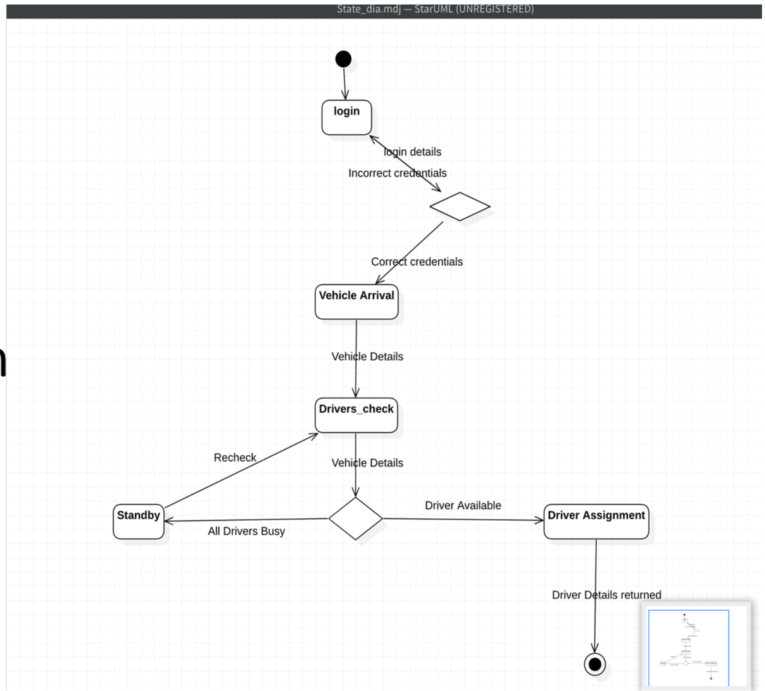
d. Precondition:

i. User logged in.
ii. Vehicle details fed to database.

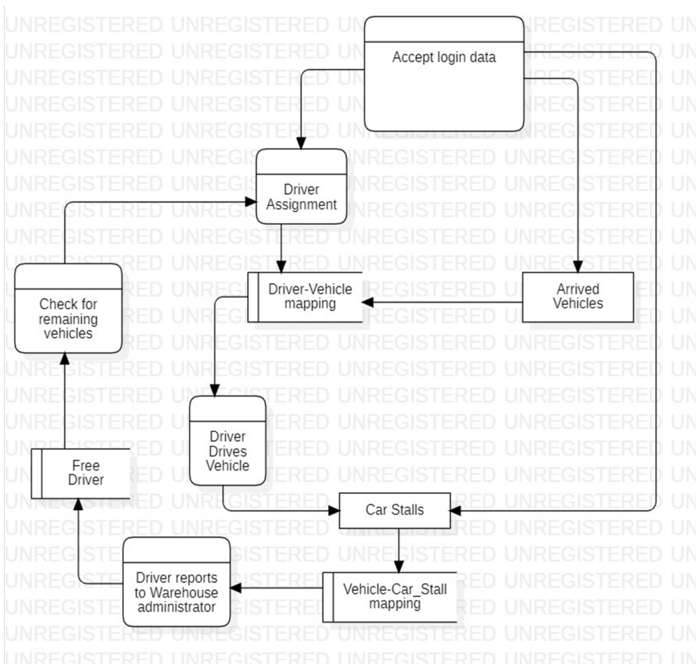
Use Case Diagram



State Chart Diagram



Data Flow Diagram



Requirement Elaboration

2: Facilitating characteristic specific search options.

For instance, if the manager wants to know how many SUVs are present at a given storage complex:

a. Understand the requirement: System should be able to search for the most similar vehicle as per the description given by the customers and his requirements.

b. Decomposition:

i. Vehicle Description: Description or requirement of customer.

ii. Vehicles: Available vehicles in the warehouse.

Iii Most searched vehicles: System should identify most searched vehicles.

c. Actors:

i. Primary: Warehouse administrator.

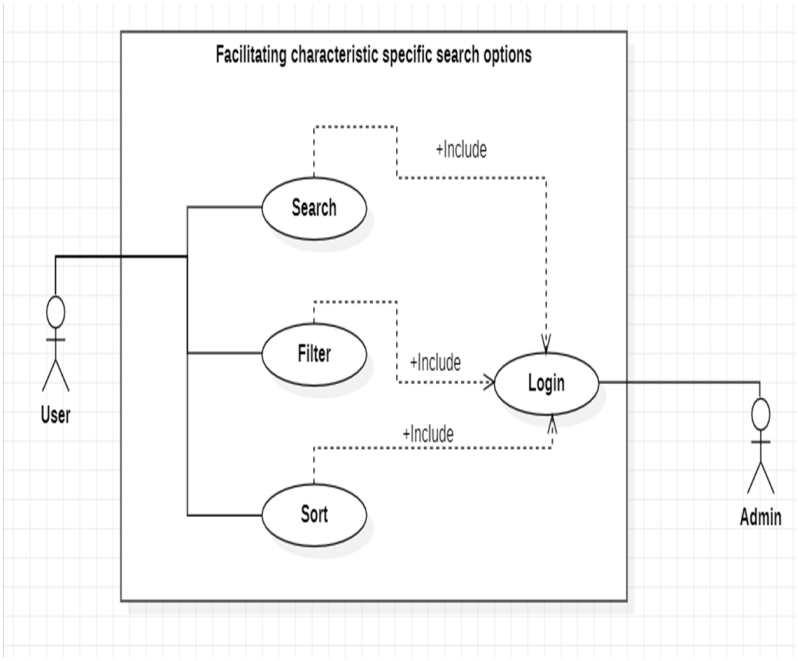
ii. Secondary: Customers.

d. Precondition:

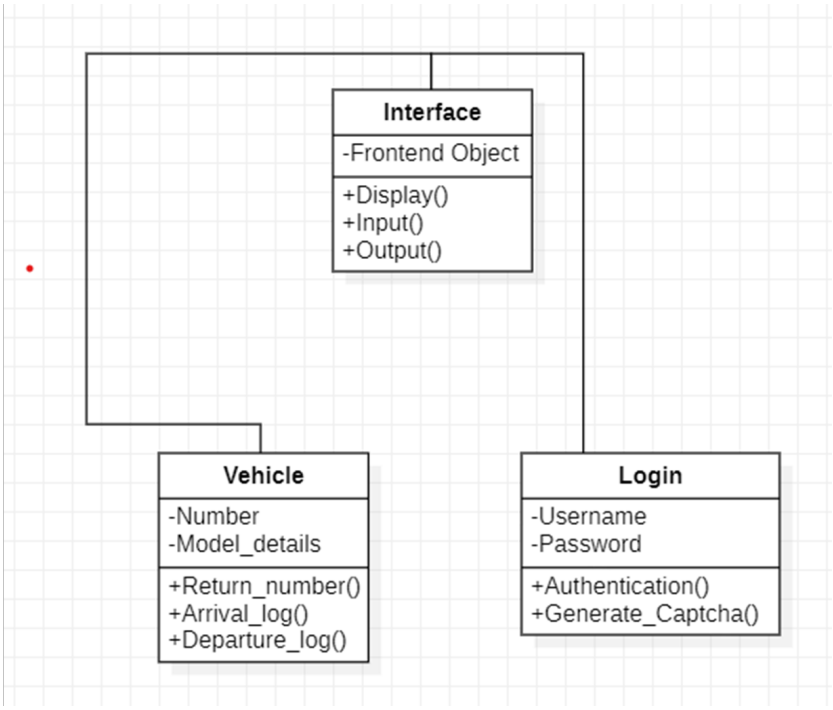
i. User logged in.

ii. Vehicle details fed to database.

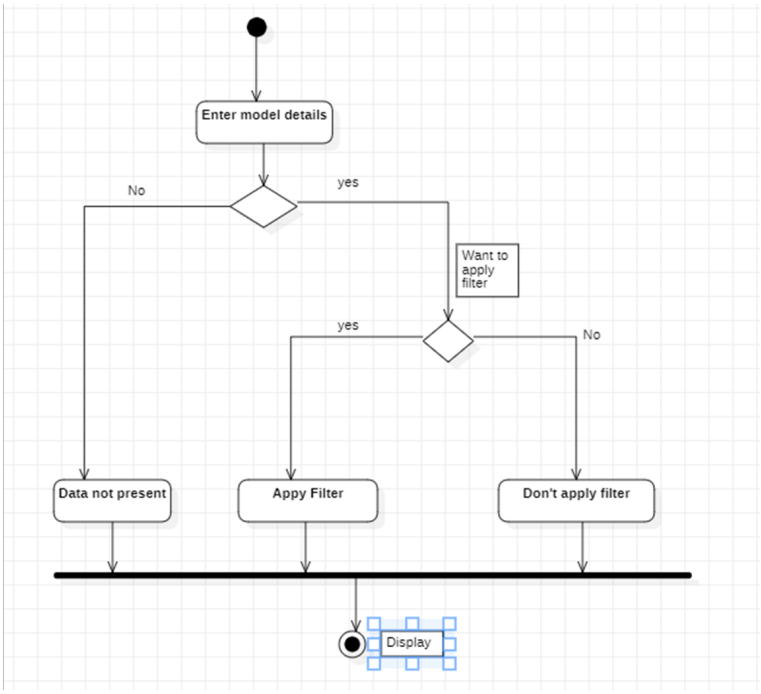
Use Case Diagram



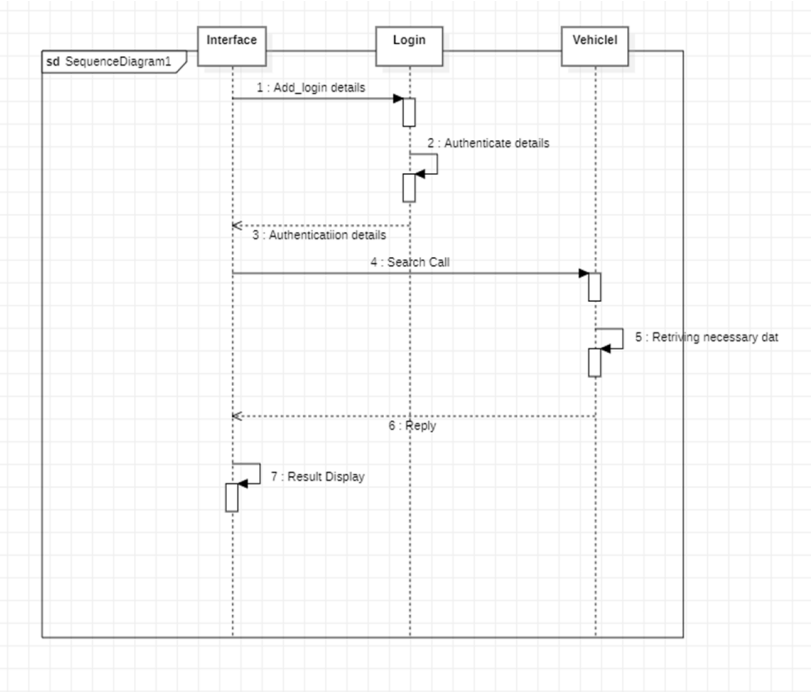
Class Diagram



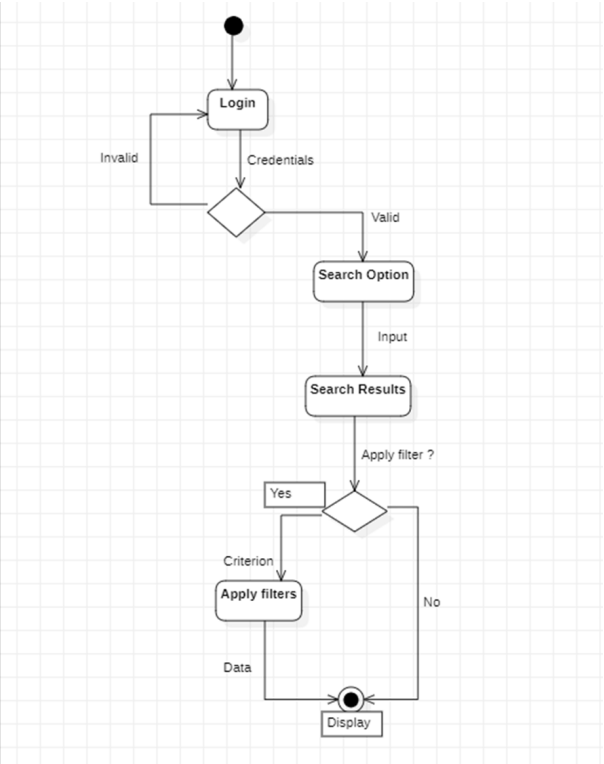
Activity Diagram



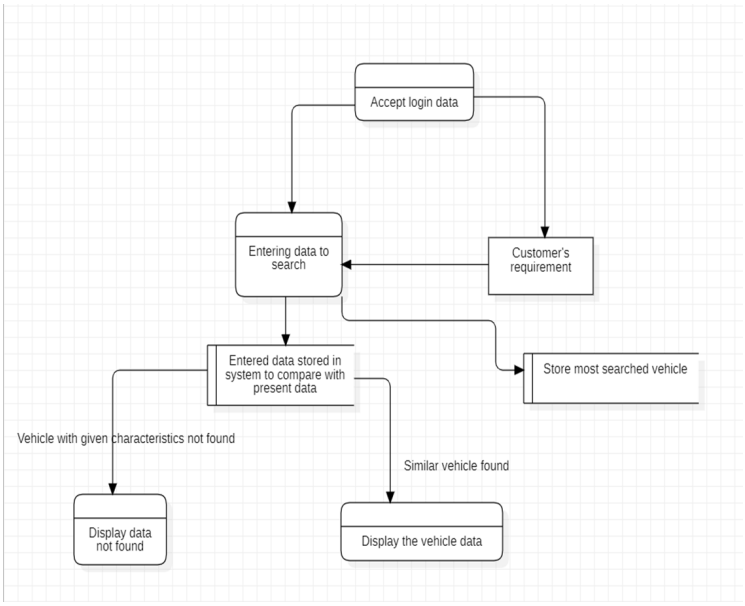
Sequence Diagram



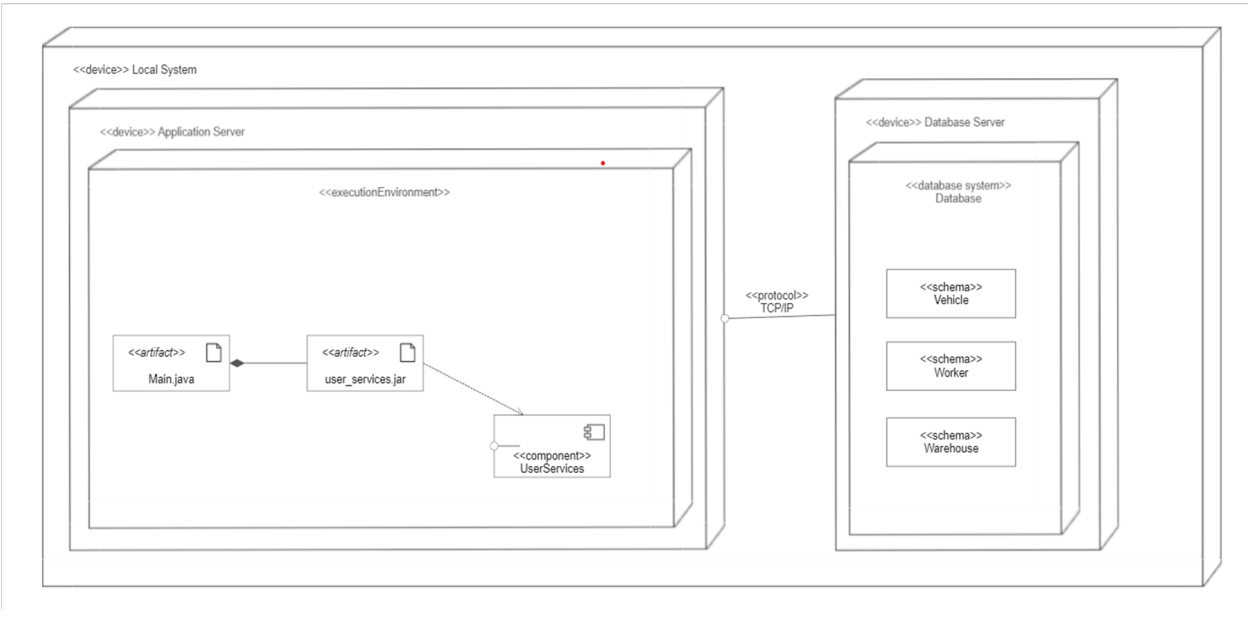
State Chart Diagram



Data Flow Diagram

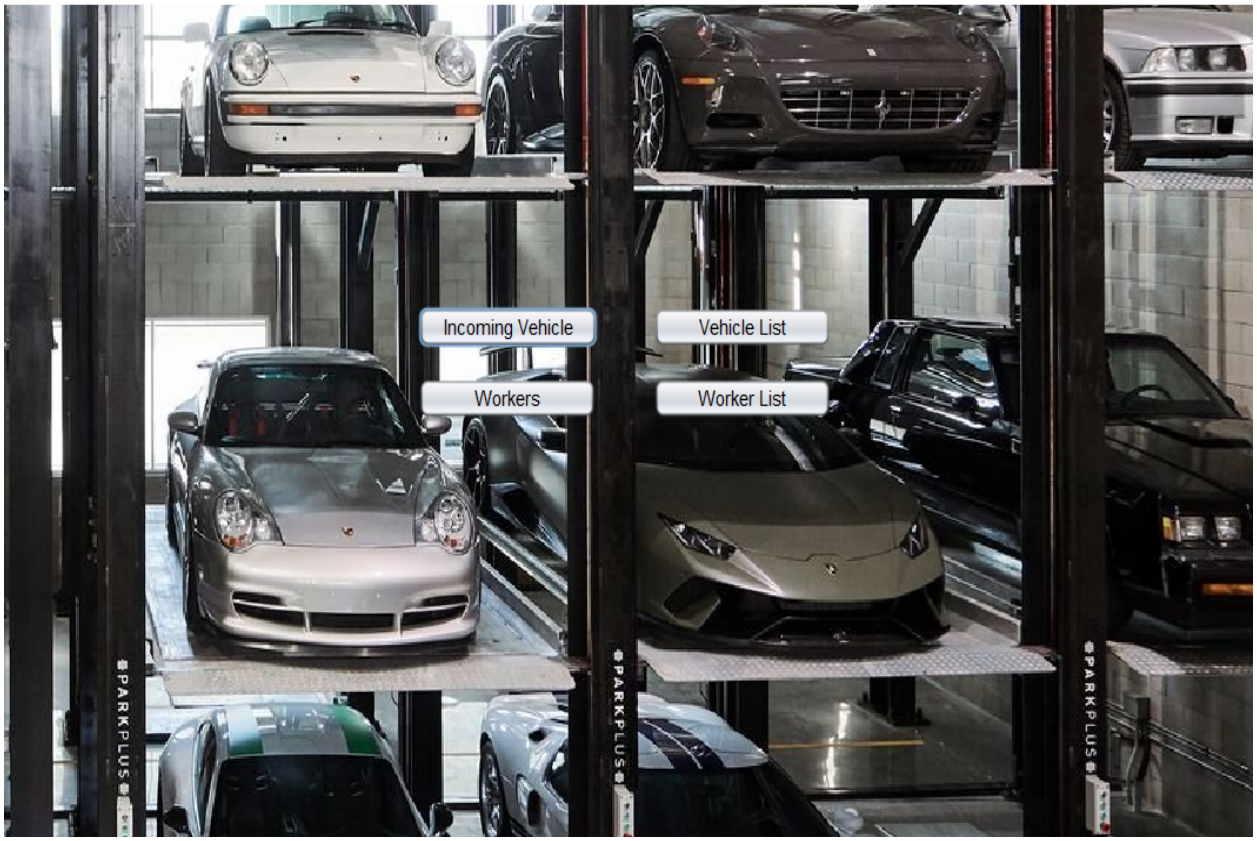


Deployment Diagram



Graphical User Interface

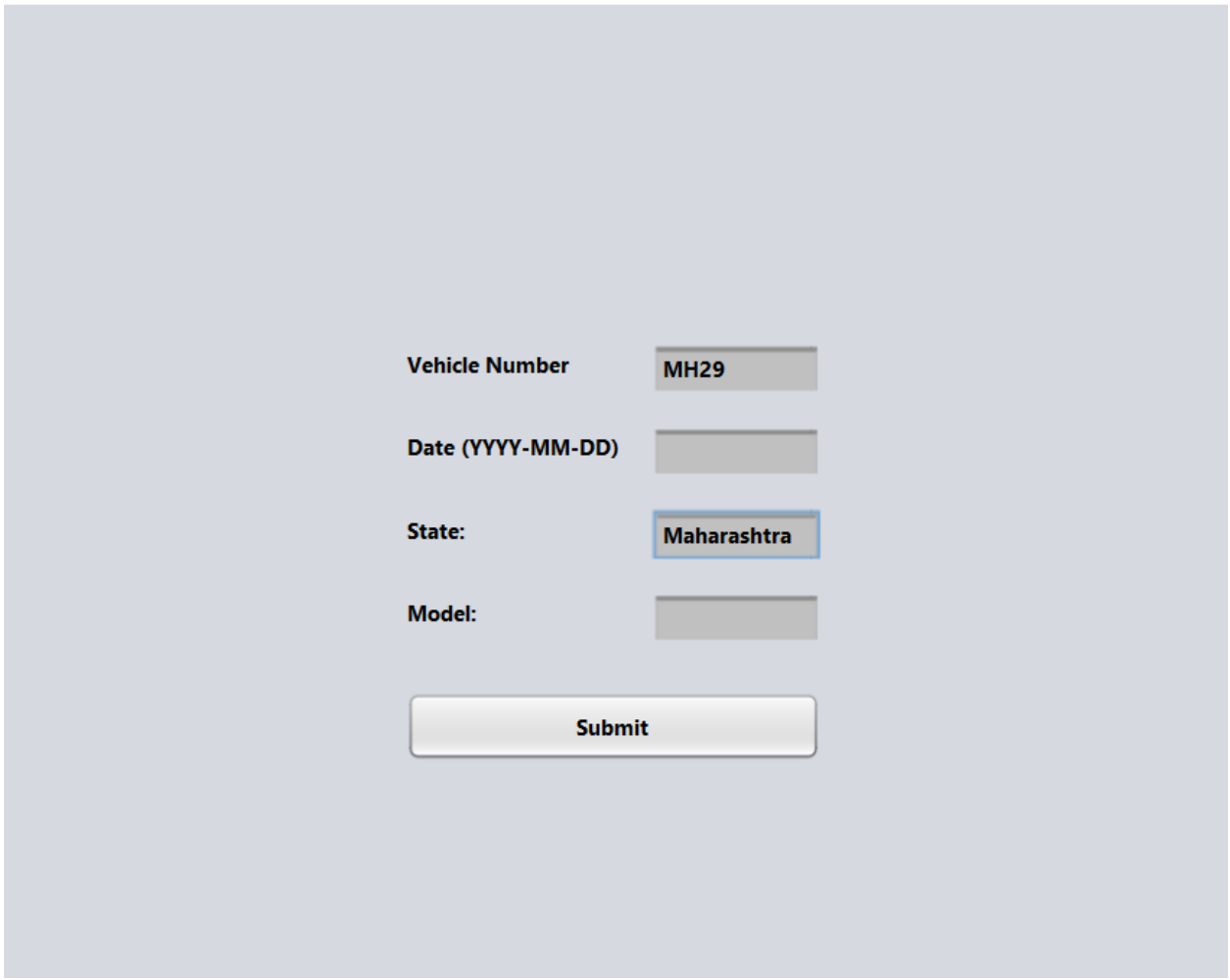
Frame1: Home



This frame represents the home page of the software. The graphical user interface has been designed using Java. This home page is meant to be used by the warehouse administrator.
The user will find various buttons here, linking to various functionalities. These buttons have on click commands which will redirect the user to subsequent frames.

1. Incoming vehicle.
2. Vehicle List
3. Workers
4. Worker List

Frame2: To accept vehicle details for each incoming vehicle.

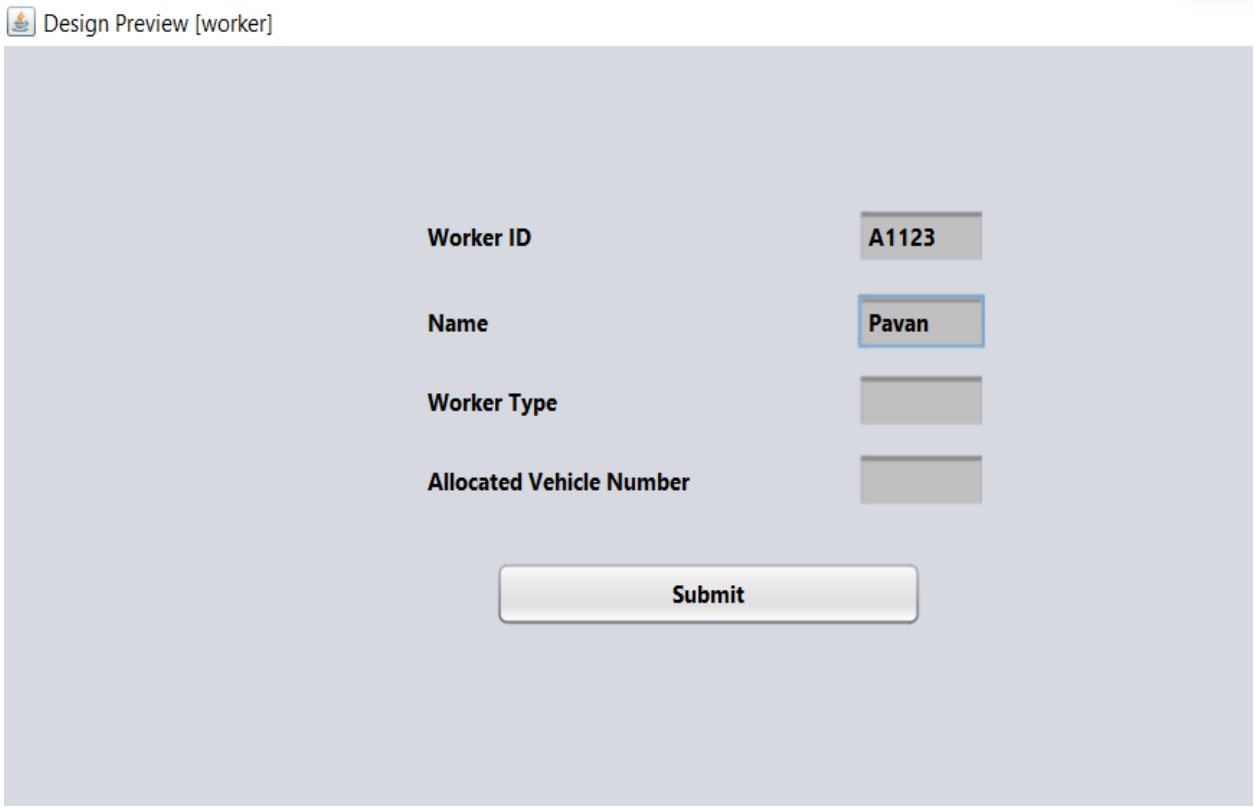


Here, the details of a new vehicle are accepted from the warehouse administrator. The input data is passed from the user interface to the application logic. This data is then passed through the application logic to the backend tables using MySQL queries.

Backend Query through Java:

```
String query = "INSERT INTO vehicle(vehicle_id, date_om, model, checkin_date)
                VALUES (?, ?, ?, sysdate())";

    PreparedStatement statement = con.prepareStatement(query);
    statement.setString(1, V_ID);
    statement.setString(2, Date);
    statement.setString(3, Model);
```

Frame 3: To accept Worker details

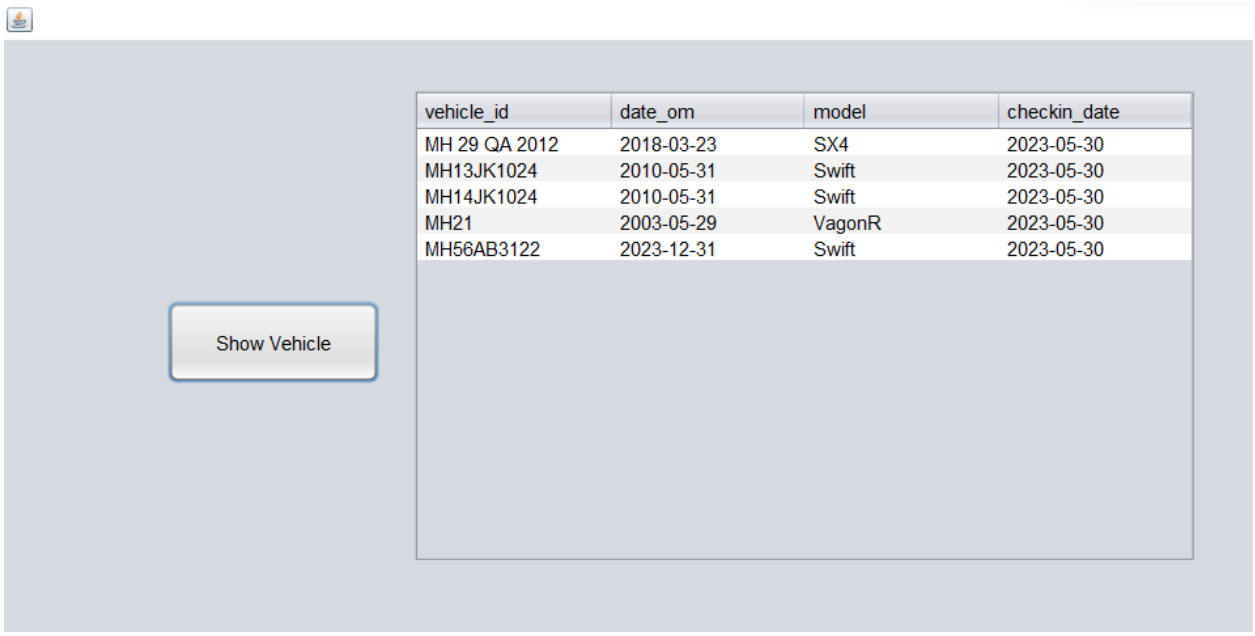


Here, the details of a new worker are accepted from the warehouse administrator. The input data is passed from the user interface to the application logic. This data is then passed through the application logic to the backend tables using MySQL queries.

Backend Query through Java:

```
String query = "INSERT INTO worker(worker_id, name, type, vehicle_number)
                VALUES (?, ?, ?, ?)";

    PreparedStatement statement = con.prepareStatement(query);
    statement.setString(1, W_ID);
    statement.setString(2, name);
    statement.setString(3, Type);
    statement.setString(4, Allotted_vehicle_id);
```

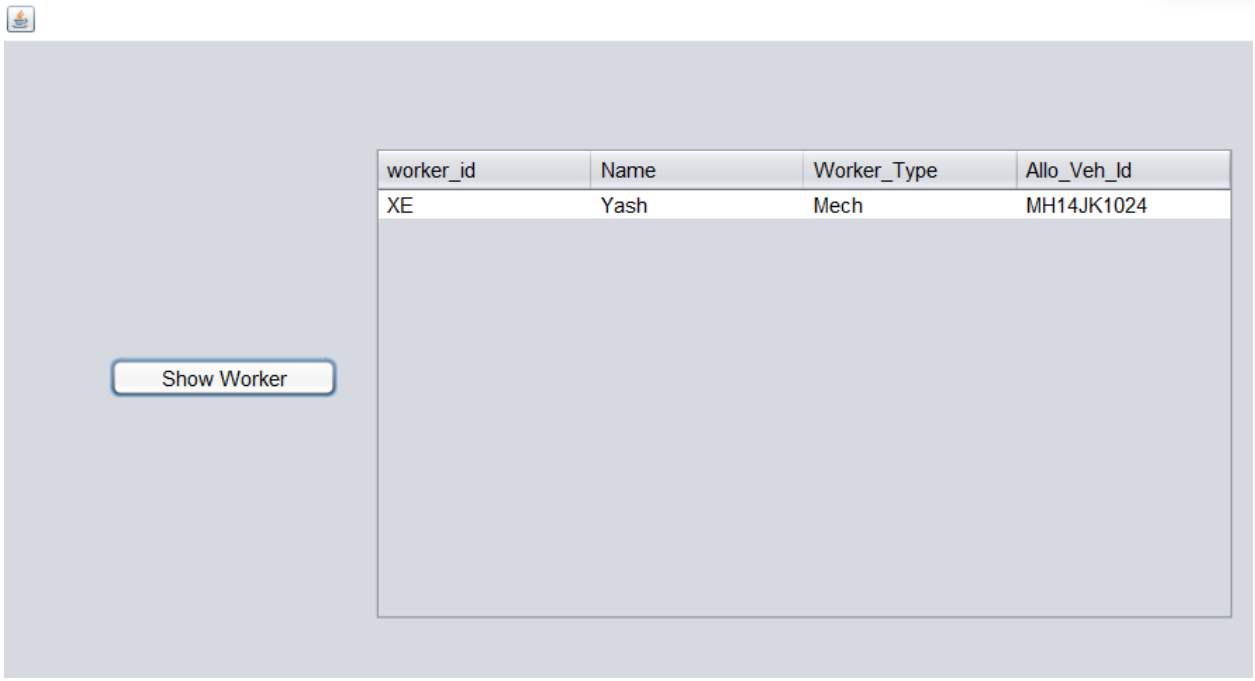
Frame 4: Displaying Vehicle details from Vehicle table.



The functionality served by this frame is to display vehicles' details to the user. The appropriate data is taken from the backend, brought to the application logic and then presented to the user.
The algorithm is as follows:

```
Statement st= con.createStatement();
        String query="select * from vehicle;";
        ResultSet rs=st.executeQuery(query);

        ResultSetMetaData rsmd=rs.getMetaData();

        DefaultTableModel model= (DefaultTableModel) VehicleData.getModel();

        int cols=rsmd.getColumnCount();
        String[] colName=new String[cols];
        for(int i=0;i<cols;i++)
           colName[i]=rsmd.getColumnName(i+1);
        model.setColumnIdentifiers(colName);
        String no,date1,mod,date2;
        while(rs.next()){
           no=rs.getString(1);
           date1=rs.getString(2);
           mod=rs.getString(3);
           date2=rs.getString(4);
           String[] row={no,date1,mod,date2};

           model.addRow(row);
```

Frame 5: Displaying worker details from Worker table.



Similar to the previous frame, the functionality served by this frame is to display workers' details to the user. The appropriate data is taken from the backend, brought to the application logic and then presented to the user.
The algorithm is as follows:

```
Statement st= con.createStatement();
        String query="select * from vehicle;";
        ResultSet rs=st.executeQuery(query);
        ResultSetMetaData rsmd=rs.getMetaData();

        DefaultTableModel model= (DefaultTableModel) VehicleData.getModel();

        int cols=rsmd.getColumnCount();
        String[] colName=new String[cols];
        for(int i=0;i<cols;i++)
           colName[i]=rsmd.getColumnName(i+1);
        model.setColumnIdentifiers(colName);
        String no,date1,mod,date2;
        while(rs.next()){
           no=rs.getString(1);
           date1=rs.getString(2);
           mod=rs.getString(3);
           date2=rs.getString(4);
           String[] row={no,date1,mod,date2};

           model.addRow(row);
```

Source Code

1. **Home Page:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package pbl_new;


/**
 *
 * @author ASUS
 */
public class DragnDrop extends javax.swing.JFrame {
   static void f1(){
System.out.println("");
}

    /**
     * Creates new form DragnDrop
     */
    public DragnDrop() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jButton1 = new javax.swing.JButton();
        jButton2 = new javax.swing.JButton();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
getContentPane().setLayout(null);

jButton1.setText("Incoming Vehicle");
jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jButton1MouseClicked(evt);
   }
});
jButton1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton1ActionPerformed(evt);
   }
});
getContentPane().add(jButton1);
jButton1.setBounds(310, 170, 122, 23);

jButton2.setText("Vehicle List");
jButton2.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jButton2MouseClicked(evt);
   }
});
jButton2.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton2ActionPerformed(evt);
   }
});
getContentPane().add(jButton2);
jButton2.setBounds(470, 170, 121, 23);

jButton3.setText("Workers");
jButton3.addMouseListener(new java.awt.event.MouseAdapter() {
   public void mouseClicked(java.awt.event.MouseEvent evt) {
      jButton3MouseClicked(evt);
   }
});
jButton3.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton3ActionPerformed(evt);
   }
```

```
        });
        getContentPane().add(jButton3);
        jButton3.setBounds(310, 210, 121, 23);

        jButton4.setText("Worker List");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });
        getContentPane().add(jButton4);
        jButton4.setBounds(470, 210, 121, 23);

        jLabel1.setIcon(new javax.swing.ImageIcon("C:\\Users\\ASUS\\OneDrive\\Desktop\
\CjZwzGN0.jpeg")); // NOI18N
        getContentPane().add(jLabel1);
        jLabel1.setBounds(-930, -610, 2710, 1350);

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Incoming_Vehicle Input_Vehicle_frame = new Incoming_Vehicle();
        Input_Vehicle_frame.setVisible(true);
    }

    private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:



    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Vehicle1 Show_Vehicle_frame = new Vehicle1();
        Show_Vehicle_frame.setVisible(true);
    }

    private void jButton2MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }
```

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        worker INPUT_Vehicle_frame = new worker();
        INPUT_Vehicle_frame.setVisible(true);



    }

    private void jButton3MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Display_Worker Display_Worker_Frame = new Display_Worker();
        Display_Worker_Frame.setVisible(true);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(DragnDrop.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (InstantiationException ex) {
```

```
java.util.logging.Logger.getLogger(DragnDrop.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(DragnDrop.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(DragnDrop.class.getName()).log(java.util.logging.Level.SEVE
RE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new DragnDrop().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}
```

2. **Entering Vehicle Information Window:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package pbl_new;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.*;

/**
 *
 * @author ASUS
 */
public class Incoming_Vehicle extends javax.swing.JFrame {

    /**
     * Creates new form Incoming_Vehicle
     */
    static void B_Insert_Vehicle(String V_ID,String Date,String State,String Model)
        {
            System.out.println(V_ID+Date+State+Model);
            try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
pbl","root","pbl@123");
           // String s = " insert into vehicle(vehicle_id, date_om, model, checkin_date) values(,
'2010-05-31', 'Swift', sysdate());";
            Statement st = con.createStatement();


           String query = "INSERT INTO vehicle(vehicle_id, date_om, model, checkin_date) VALUES
(?, ?, ?, sysdate())";

            PreparedStatement statement = con.prepareStatement(query);
            statement.setString(1, V_ID);
            statement.setString(2, Date);
```

```
        statement.setString(3, Model);



        int rowsInserted = statement.executeUpdate();



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

    }
public Incoming_Vehicle() {
    initComponents();
      }

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    Vehicle_ID_input = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    Date_input = new javax.swing.JTextField();
    State_input = new javax.swing.JTextField();
    Model_input = new javax.swing.JTextField();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setBackground(new java.awt.Color(30, 74, 104));

    Vehicle_ID_input.setBackground(java.awt.Color.lightGray);
```

```
Vehicle_ID_input.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
Vehicle_ID_input.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        Vehicle_ID_inputActionPerformed(evt);
    }
});

jButton1.setBackground(java.awt.Color.lightGray);
jButton1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton1.setText("Submit");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

Date_input.setBackground(java.awt.Color.lightGray);
Date_input.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
Date_input.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        Date_inputActionPerformed(evt);
    }
});

State_input.setBackground(java.awt.Color.lightGray);
State_input.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
State_input.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        State_inputActionPerformed(evt);
    }
});

Model_input.setBackground(java.awt.Color.lightGray);
Model_input.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

jLabel1.setBackground(java.awt.Color.lightGray);
jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel1.setText("Vehicle Number");

jLabel2.setBackground(java.awt.Color.lightGray);
jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel2.setText("Date (YYYY-MM-DD)");
```

```
        jLabel3.setBackground(java.awt.Color.lightGray);
        jLabel3.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        jLabel3.setText("State:");

        jLabel4.setBackground(java.awt.Color.lightGray);
        jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
        jLabel4.setText("Model:");

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(223, 223, 223)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(6, 6, 6)
                        .addComponent(Vehicle_ID_input, javax.swing.GroupLayout.PREFERRED_SIZE,
93, javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel2)
                        .addGap(18, 18, 18)
                        .addComponent(Date_input, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(6, 6, 6)
                        .addComponent(State_input, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(6, 6, 6)
                        .addComponent(Model_input, javax.swing.GroupLayout.PREFERRED_SIZE, 93,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 229,
javax.swing.GroupLayout.PREFERRED_SIZE)))
        );
```

```
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(188, 188, 188)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)
                        .addComponent(jLabel1))
                    .addComponent(Vehicle_ID_input, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)
                        .addComponent(jLabel2))
                    .addComponent(Date_input, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)
                        .addComponent(jLabel3))
                    .addComponent(State_input, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(3, 3, 3)
                        .addComponent(jLabel4))
                    .addComponent(Model_input, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(27, 27, 27)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );

        pack();
    }// </editor-fold>

    private void Vehicle_ID_inputActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
          String V_ID = Vehicle_ID_input.getText();
          String Date = Date_input.getText();
          String State = State_input.getText();
          String Model = Model_input.getText();

          B_Insert_Vehicle(V_ID,Date,State,Model);

          setVisible(false);



    }

    private void Date_inputActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void State_inputActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

```
java.util.logging.Logger.getLogger(Incoming_Vehicle.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Incoming_Vehicle.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Incoming_Vehicle.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Incoming_Vehicle.class.getName()).log(java.util.logging.Level.
SEVERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Incoming_Vehicle().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField Date_input;
    private javax.swing.JTextField Model_input;
    private javax.swing.JTextField State_input;
    private javax.swing.JTextField Vehicle_ID_input;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    // End of variables declaration
}
```

3. **Displaying Vehicles in Warehouse:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package pbl_new;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import javax.swing.table.DefaultTableModel;



/**
 *
 * @author ASUS
 */
public class Vehicle1 extends javax.swing.JFrame {

    /**
     * Creates new form Vehicle1
     */
  public void show_vehicle()
    {
      try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
pbl","root","pbl@123");

            Statement st= con.createStatement();
            String query="select * from vehicle;";
            ResultSet rs=st.executeQuery(query);

            ResultSetMetaData rsmd=rs.getMetaData();

            DefaultTableModel model= (DefaultTableModel) VehicleData.getModel();
```

```
        int cols=rsmd.getColumnCount();
        String[] colName=new String[cols];
        for(int i=0;i<cols;i++)
            colName[i]=rsmd.getColumnName(i+1);
        model.setColumnIdentifiers(colName);
        String no,date1,mod,date2;
        while(rs.next()){
            no=rs.getString(1);
            date1=rs.getString(2);
            mod=rs.getString(3);
            date2=rs.getString(4);
            String[] row={no,date1,mod,date2};

            model.addRow(row);


        }

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

    }


public Vehicle1() {
    initComponents();

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {
```

```
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    VehicleData = new javax.swing.JTable();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(null);

    jButton1.setBackground(java.awt.Color.lightGray);
    jButton1.setText("Show Vehicle");
    jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton1MouseClicked(evt);
        }
    });
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton1);
    jButton1.setBounds(100, 160, 130, 50);

    jScrollPane1.setBackground(java.awt.Color.lightGray);

    VehicleData.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {

        },
        new String [] {

        }
    ));
    jScrollPane1.setViewportView(VehicleData);

    getContentPane().add(jScrollPane1);
    jScrollPane1.setBounds(250, 30, 480, 290);

    pack();
}// </editor-fold>

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
```

```
        show_vehicle();

    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Vehicle1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTable VehicleData;
    private javax.swing.JButton jButton1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration
}
```

4. **Entering Worker information Window:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package pbl_new;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import javax.swing.table.DefaultTableModel;



/**
 *
 * @author ASUS
 */
public class Vehicle1 extends javax.swing.JFrame {

    /**
     * Creates new form Vehicle1
     */
  public void show_vehicle()
    {
      try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
pbl","root","pbl@123");

            Statement st= con.createStatement();
            String query="select * from vehicle;";
            ResultSet rs=st.executeQuery(query);

            ResultSetMetaData rsmd=rs.getMetaData();

            DefaultTableModel model= (DefaultTableModel) VehicleData.getModel();
```

```
        int cols=rsmd.getColumnCount();
        String[] colName=new String[cols];
        for(int i=0;i<cols;i++)
            colName[i]=rsmd.getColumnName(i+1);
        model.setColumnIdentifiers(colName);
        String no,date1,mod,date2;
        while(rs.next()){
            no=rs.getString(1);
            date1=rs.getString(2);
            mod=rs.getString(3);
            date2=rs.getString(4);
            String[] row={no,date1,mod,date2};

            model.addRow(row);


        }

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

    }


public Vehicle1() {
    initComponents();

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {
```

```
    jButton1 = new javax.swing.JButton();
    jScrollPane1 = new javax.swing.JScrollPane();
    VehicleData = new javax.swing.JTable();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    getContentPane().setLayout(null);

    jButton1.setBackground(java.awt.Color.lightGray);
    jButton1.setText("Show Vehicle");
    jButton1.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jButton1MouseClicked(evt);
        }
    });
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });
    getContentPane().add(jButton1);
    jButton1.setBounds(100, 160, 130, 50);

    jScrollPane1.setBackground(java.awt.Color.lightGray);

    VehicleData.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {

        },
        new String [] {

        }
    ));
    jScrollPane1.setViewportView(VehicleData);

    getContentPane().add(jScrollPane1);
    jScrollPane1.setBounds(250, 30, 480, 290);

    pack();
}// </editor-fold>

private void jButton1MouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
```

```
        show_vehicle();

    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Vehicle1.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
```

```
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            @Override
            public void run() {
                new Vehicle1().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTable VehicleData;
    private javax.swing.JButton jButton1;
    private javax.swing.JScrollPane jScrollPane1;
    // End of variables declaration
}
```

5. **Displaying Worker Information Window:**

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 * Click nbfs://nbhost/SystemFileSystem/Templates/GUIForms/JFrame.java to edit this template
 */
package pbl_new;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Statement;
import static pbl_new.Incoming_Vehicle.B_Insert_Vehicle;

/**
 *
 * @author ASUS
 */
public class worker extends javax.swing.JFrame {

    /**
     * Creates new form worker
     */

    static void B_Insert_Worker(String W_ID,String Name,String W_Type,String Allocated_Veh_Id)
        {
            try
        {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/
pbl","root","pbl@123");
          // String s = " insert into vehicle(vehicle_id, date_om, model, checkin_date) values(,
'2010-05-31', 'Swift', sysdate());";
            Statement st = con.createStatement();


          String query = "INSERT INTO worker(worker_id, Name, Worker_Type,Allo_Veh_Id)
VALUES (?, ?, ?, ?)";

            PreparedStatement statement = con.prepareStatement(query);
            statement.setString(1, W_ID);
```

```
        statement.setString(2, Name);
        statement.setString(3, W_Type);
        statement.setString(4, Allocated_Veh_Id);



        int rowsInserted = statement.executeUpdate();



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


  }


public worker() {
    initComponents();
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    Input_W_ID = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    Input_Name = new javax.swing.JTextField();
    Input_W_Type = new javax.swing.JTextField();
    Input_V_Allo_No = new javax.swing.JTextField();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setBackground(java.awt.Color.lightGray);
jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel1.setText("Worker ID");

jLabel2.setBackground(java.awt.Color.lightGray);
jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel2.setText("Name");

jLabel3.setBackground(java.awt.Color.lightGray);
jLabel3.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel3.setText("Worker Type");

jLabel4.setBackground(java.awt.Color.lightGray);
jLabel4.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jLabel4.setText("Allocated Vehicle Number");

Input_W_ID.setBackground(java.awt.Color.lightGray);
Input_W_ID.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
Input_W_ID.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        Input_W_IDActionPerformed(evt);
    }
});

jButton1.setBackground(java.awt.Color.lightGray);
jButton1.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
jButton1.setText("Submit");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

Input_Name.setBackground(java.awt.Color.lightGray);
Input_Name.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

Input_W_Type.setBackground(java.awt.Color.lightGray);
Input_W_Type.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N
```

```
        Input_V_Allo_No.setBackground(java.awt.Color.lightGray);
        Input_V_Allo_No.setFont(new java.awt.Font("Segoe UI", 1, 12)); // NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(235, 235, 235)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 148,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRA
ILING)
                                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 139,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 139,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 139,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(91, 91, 91)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
                            .addComponent(Input_W_ID, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(Input_V_Allo_No,
javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(Input_Name, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(Input_W_Type, javax.swing.GroupLayout.PREFERRED_SIZE,
71, javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(273, 273, 273)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 237,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(270, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
```

```
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(82, 82, 82)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Input_W_ID, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1))
            .addGap(15, 15, 15)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(Input_Name, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(12, 12, 12)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Input_W_Type, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel4)
                .addComponent(Input_V_Allo_No, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(27, 27, 27)
            .addComponent(jButton1)
            .addContainerGap(105, Short.MAX_VALUE))
      );

      pack();
   }// </editor-fold>

   private void Input_W_IDActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
   }

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:

        String W_ID = Input_W_ID.getText();
        String Name = Input_Name.getText();
        String W_Type = Input_W_Type.getText();
        String Allocated_Vehicle = Input_V_Allo_No.getText();
```

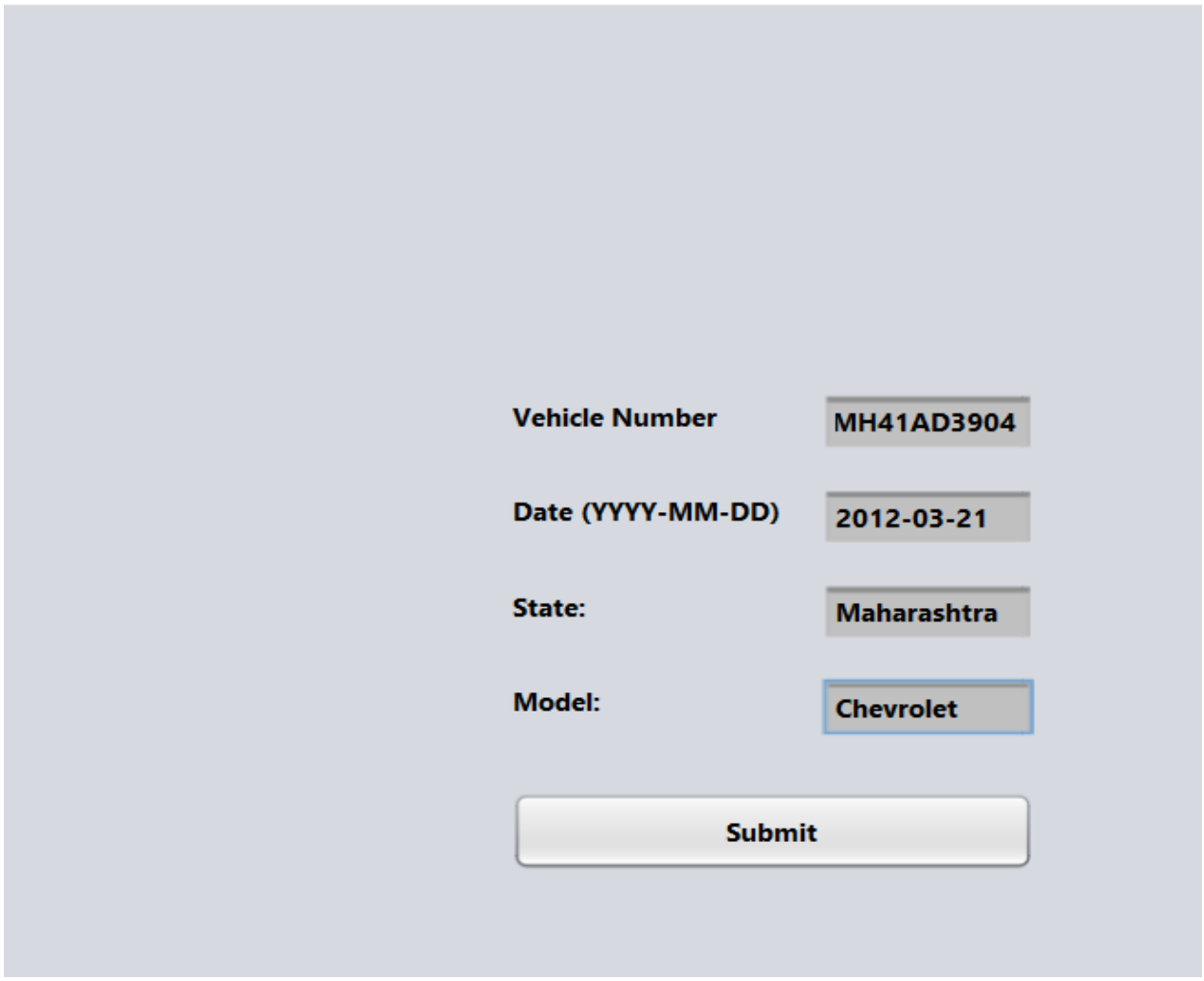
```
        B_Insert_Worker(W_ID,Name,W_Type,Allocated_Vehicle);

        setVisible(false);

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(worker.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(worker.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(worker.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(worker.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>
```

```
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new worker().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField Input_Name;
    private javax.swing.JTextField Input_V_Allo_No;
    private javax.swing.JTextField Input_W_ID;
    private javax.swing.JTextField Input_W_Type;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    // End of variables declaration
}
```

Test Cases

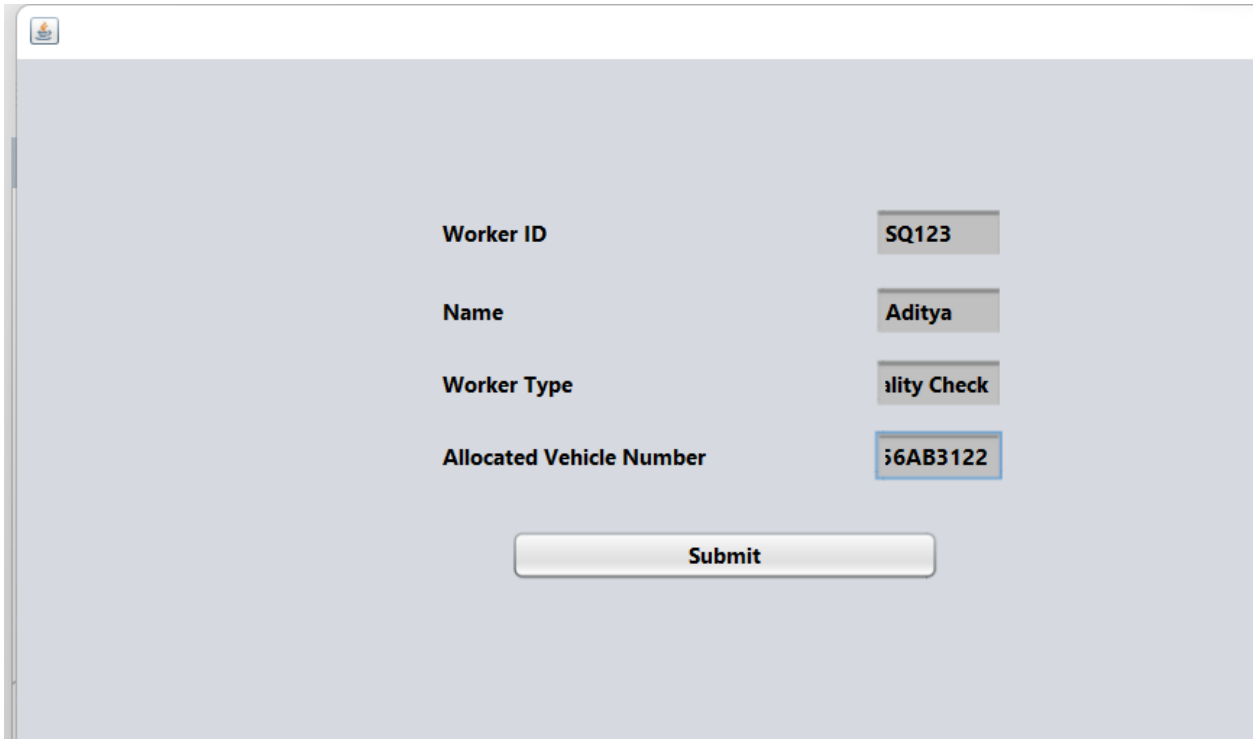
1: To accept the information about any vehicle that has arrived in the ware house:

The system accepts the complete information about any vehicle such as vehicle no, date of manufacturing, state, etc and stores it into A table named ad VEHICLE.



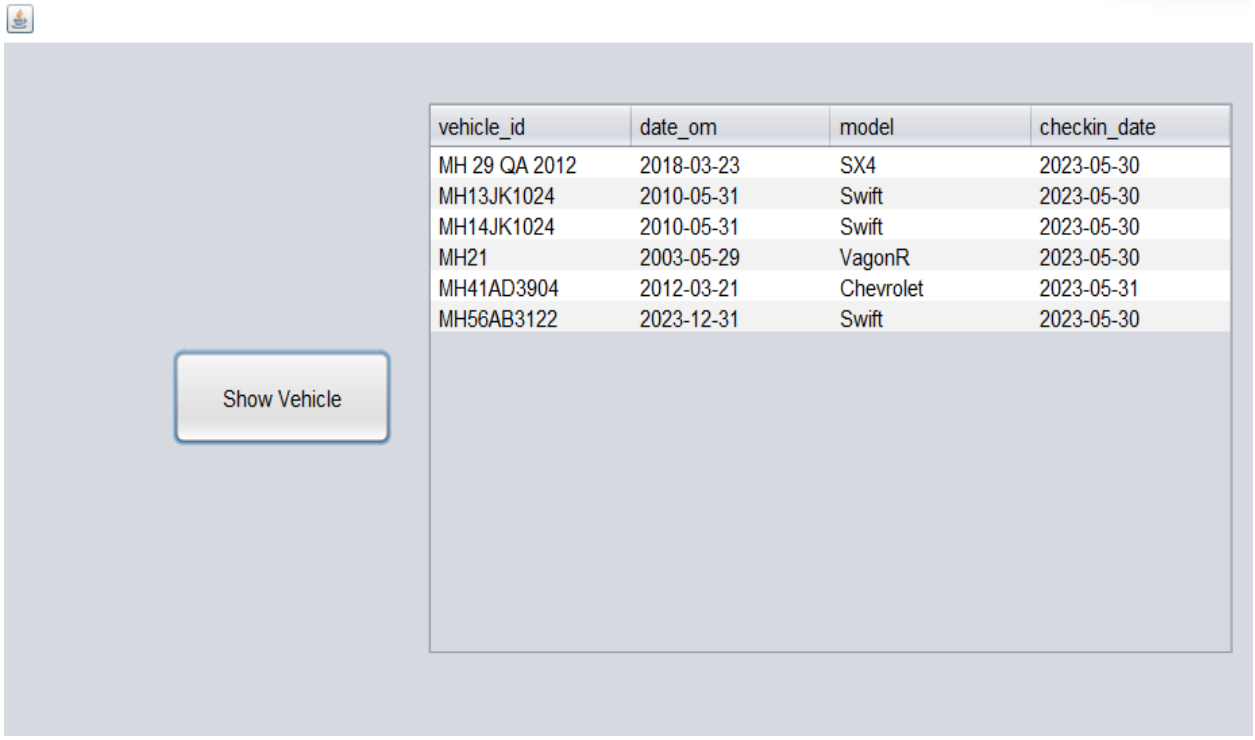

```
mysql> select * from vehicle;
+---------------+------------+-----------+--------------+
| vehicle_id    | date_om    | model     | checkin_date |
+---------------+------------+-----------+--------------+
| MH 29 QA 2012 | 2018-03-23 | SX4       | 2023-05-30   |
| MH13JK1024    | 2010-05-31 | Swift     | 2023-05-30   |
| MH14JK1024    | 2010-05-31 | Swift     | 2023-05-30   |
| MH21          | 2003-05-29 | VagonR    | 2023-05-30   |
| MH41AD3904    | 2012-03-21 | Chevrolet | 2023-05-31   |
| MH56AB3122    | 2023-12-31 | Swift     | 2023-05-30   |
+---------------+------------+-----------+--------------+
6 rows in set (0.00 sec)

mysql>
```

2: To Accept worker information such as worker ID, name etc and store it onto a table named as worker :

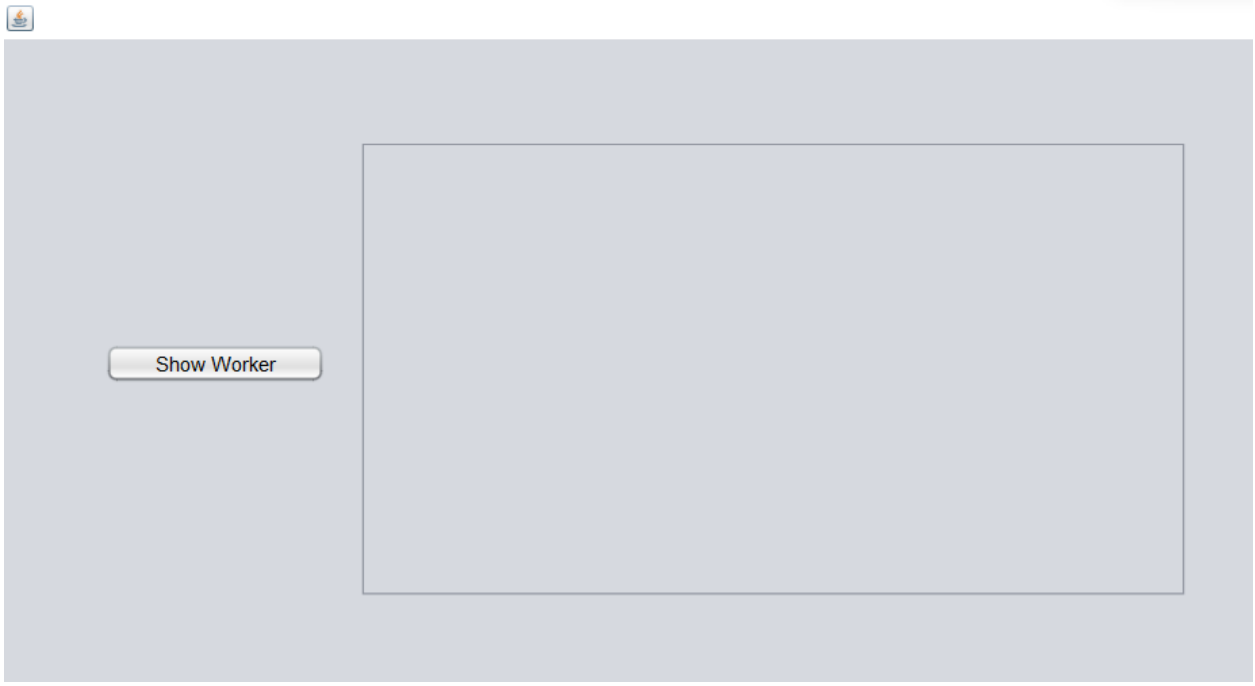


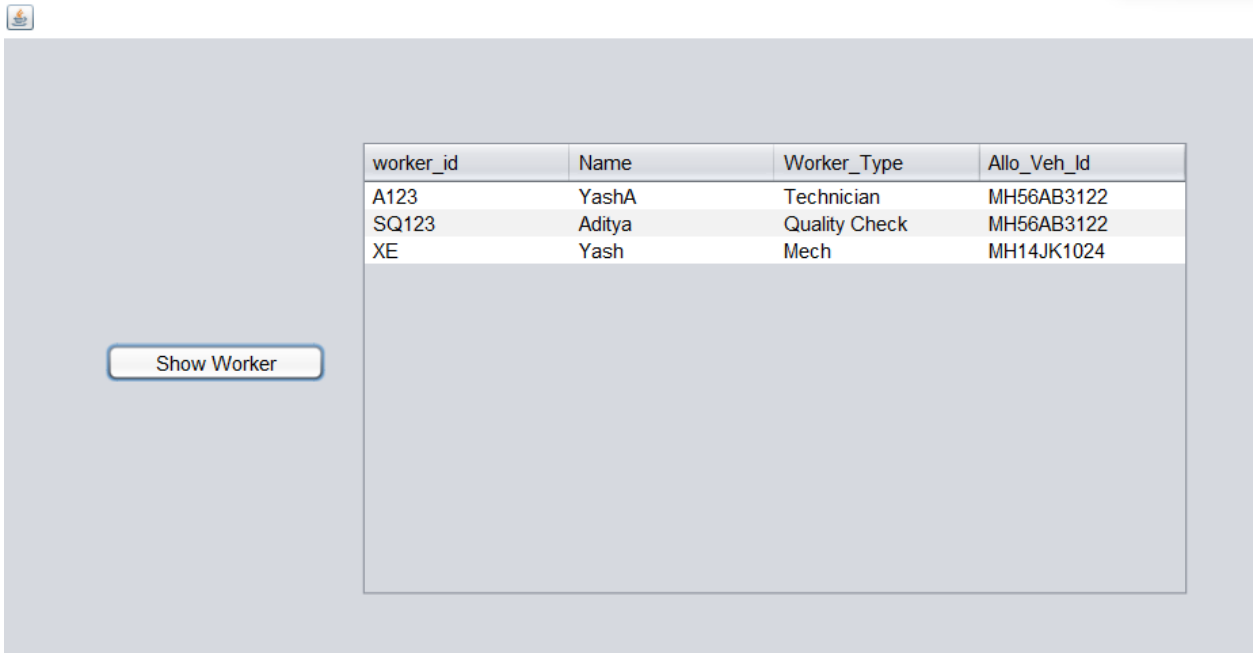

```
mysql> select * from worker;
+-----------+--------+---------------+-------------+
| worker_id | Name   | Worker_Type   | Allo_Veh_Id |
+-----------+--------+---------------+-------------+
| A123      | YashA  | Technician    | MH56AB3122  |
| SQ123     | Aditya | Quality Check | MH56AB3122  |
| XE        | Yash   | Mech          | MH14JK1024  |
+-----------+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

3: To display vehicles stored in warehouse:



4: To display worker information:





Conclusion:

The Vehicle Warehouse Management System offers a comprehensive solution for automating and streamlining warehouse management processes in the second-hand vehicle resale industry. By leveraging technology, the system improves efficiency, reduces errors, and enhances coordination among warehouse administrators, workers, and management teams. With features such as inventory management, vehicle tracking, maintenance scheduling, quality assessment, and data management, the system optimizes operations, enhances decision-making, and promotes adherence to regulatory standards. The user-friendly interface, scalability, and emphasis on security and data privacy make it a valuable tool for vehicle resellers, enabling them to effectively manage their warehouses and ultimately improve their overall business performance.