

1. INTRODUCTION:

The Regional Transport Office or Regional Transport Authority (RTO/RTA) is the organization of the Indian government responsible for maintaining a database of drivers and a database of vehicles for various states of India. The RTO issues driving licenses, organizes collection of vehicle excise duty (road tax) and sells personalized registrations.

The RTO database helps to identify unregistered vehicle which helps the department to take strict action against the owners. The database also keeps track of the RTO Officers with their valid office ID's and Location. Today, the number of vehicles manufactured is increasing in a great speed. This will lead to their registration which increases the amount of data to be handled. In previous databases there were a lot of discrepancies such as redundancy of owner details, unhandled exceptions caused by the updating of registration date etc.

To solve the above identified problems we prepared a new database which does similar transactions as that of the previous database with greater accuracy and provides better results. The designed database management system provides RTO office to maintain all records about 2-Wheeler registration, 3-Wheeler registration, LMV, HMV, learning license and driving license generation. Initially the RTO Officer will enter owner details and all the complete information will be saved into the database. To register a vehicle through this software RTO officer has to provide all the details of vehicles including vehicle number, manufacturer's name, body type, vehicle owner details, chassis number, registration date, etc. This data is going to be further retrieved by the user to see all the details specified. These transactions are now being done with higher accuracy and efficiency.

Vehicle registration system allows registration officers to register a vehicle to register owner and to provide registration details after the owner owns a vehicle. Tool has been designed to facilitate the flow of information within the organization. It also shows the information related to vehicle and the owner . The project also gives tax amount to be paid for the vehicle owned by an owner before it's due date.

This dynamic project also has a feature of changing the ownership of the vehicle. Previously, it was a bit lengthy and time taking task when a owner had to sell a vehicle to new owner officially. This project does this task very easily and saves lot of precious time. Paper work was heck of a task!

The old way of registration was totally done on papers which was not only used to take so much of time but also uses lots of paper. Hence, wastage of paper as well as hard to retrieve stored information.

1.2 PROBLEM STATEMENT:

“To reduce the paper work and totally digitalize the registration done by RTO office and to change the ownership of the vehicle. “

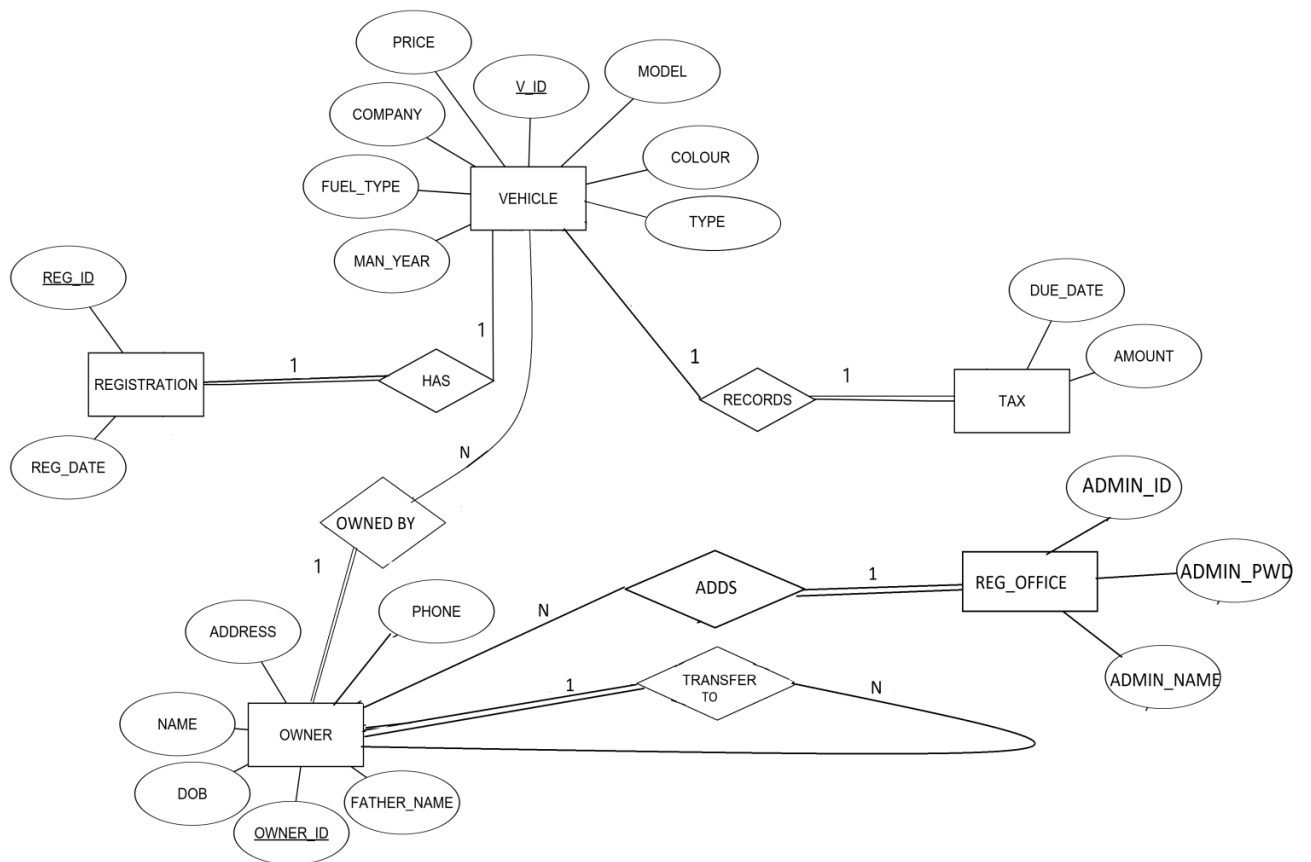
This project is used for keeping the records of the vehicle that are owned by the customers and detail of the owner who owns that particular vehicle.

FRONT-END AND BACK-END SELECTION:

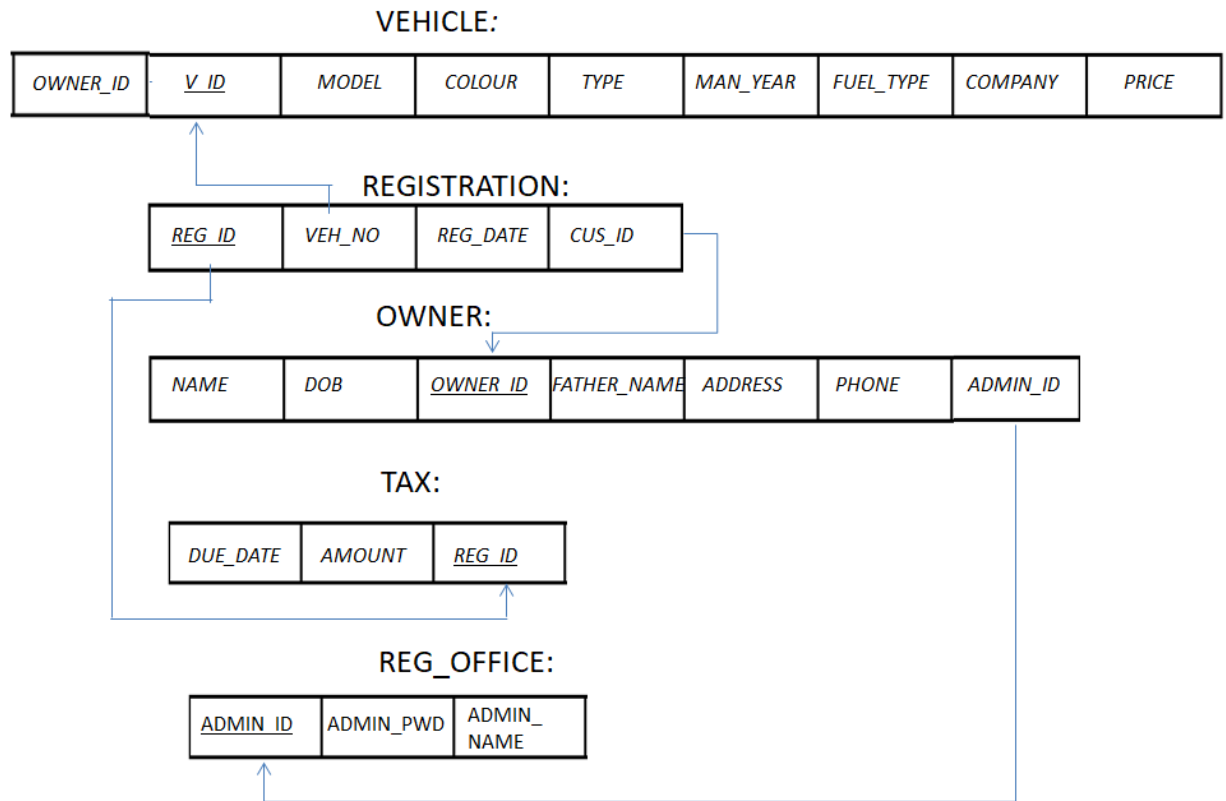
An important issue for the development of the project is the selection of suitable front-end and back-end. When we decided to develop the project, we went thorough and extensive study to determine the most suitable platform that suits the needs of the organization as well as helps in development of the project.

Front-end selection: JAVA

Back-end selection: MYSQL

2.1 CONCEPTUAL DATABASE DESIGN:

2.2 LOGICAL DATABASE DESIGN:



2.3. NORMALIZATION:

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

2.3.1: Conditions For Normalization

First Normal Form (1NF):

As per First Normal Form, no two Rows of data must contain repeating group of information i.e. each set of columns must have a unique value, such that multiple columns cannot be used to fetch the same row. Each table should be organized into rows, and each row should have a primary key that distinguishes it as unique.

Second Normal Form (2NF):

As per the Second Normal Form there must not be any partial dependency of any column on primary key. It means that for a table that has concatenated primary key, each column in the table that is not part of the primary key must depend upon the entire concatenated key for its existence. If any column depends only on one part of the concatenated key, then the table fails **Second normal form**.

Third Normal Form (3NF):

Third Normal form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this *transitive functional dependency* should be removed from the table and also the table must be in Second Normal form.

2.3.2: NORMALIZATION FOR VEHICLE:

<u>V_ID</u>	MODEL	COLOUR	TYPE	MAN_YEAR	FUEL_TYP	COMPANY	PRICE
-------------	-------	--------	------	----------	----------	---------	-------

V_ID → MODEL

V_ID → COLOUR

V_ID → TYPE

V_ID → MAN_YEAR

V_ID → FUEL_TYPE

V_ID → COMPANY

V_ID → PRICE

First Normal Form:

It is already in 1NF since each column is having a unique value, such that multiple columns cannot be used to fetch the same row.

After Second Normal Form:

<u>V_ID</u>	MODEL	COLOUR	TYPE	MAN_YEAR	FUEL_TYP	COMPANY	PRICE
-------------	-------	--------	------	----------	----------	---------	-------

It is already in 2NF form since MODEL, COLOUR, TYPE, MAN_YEAR, FUEL_TYP, COMPANY and PRICE are all dependent on primary key V_ID .

After Third Normal Form:

Since there is no transitive functional dependency, table is already in 3NF.

<u>V_ID</u>	MODEL	COLOUR	TYPE	MAN_YEAR	FUEL_TYP	COMPANY	PRICE
-------------	-------	--------	------	----------	----------	---------	-------

Hence, V_id will be the primary key for Registration table.

2.3.3: NORMALIZATION FOR REGISTRATION:

<u>REG_ID</u>	VEH_NO	REG_DATE	CUS_ID
---------------	--------	----------	--------

REG_ID → VEH_NO

REG_ID → REG_DATE

REG_ID → CUS_ID

First Normal Form:

It is already in 1NF since each column is having a unique value, such that multiple columns cannot be used to fetch the same row.

<u>REG_ID</u>	VEH_NO	REG_DATE	CUS_ID
---------------	--------	----------	--------

After Second Normal Form

<u>REG_ID</u>	VEH_NO	REG_DATE	CUS_ID
---------------	--------	----------	--------

Since REG_DATE, VEH_NO, CUS_ID depend on Primary Key REG_ID.

After Third Normal Form:

<u>REG_ID</u>	VEH_NO	REG_DATE	CUS_ID
---------------	--------	----------	--------

Since there is no transitive functional dependency, table is already in 3NF.

Hence Reg_No will be the primary key for Registration table.

2.3.4: NORMALIZATION FOR OWNER:

NAME	DOB	<u>OWNER_ID</u>	FATHER_NAME	ADDRESS	PHONE	ADMIN_ID
------	-----	-----------------	-------------	---------	-------	----------

OWNER_ID → NAME
 OWNER_ID → DOB
 OWNER_ID → FATHER_NAME
 OWNER_ID → ADDRESS
 OWNER_ID → PHONE
 OWNER_ID → ADMIN_ID

First Normal Form:

It is already in 1NF since each column is having a unique value, such that multiple columns cannot be used to fetch the same row.

NAME	DOB	<u>OWNER_ID</u>	FATHER_NAME	ADDRESS	PHONE	ADMIN_ID
------	-----	-----------------	-------------	---------	-------	----------

After Second Normal Form:

This is will be already in 2NF since every non-key attribute is fully functionally dependent on primary key.

NAME	DOB	<u>OWNER_ID</u>	FATHER_NAME	ADDRESS	PHONE	ADMIN_ID
------	-----	-----------------	-------------	---------	-------	----------

After Third Normal Form:

Since there is no transitive functional dependency, therefore , table is already in 3NF.

NAME	DOB	<u>OWNER_ID</u>	FATHER_NAME	ADDRESS	PHONE	ADMIN_ID
------	-----	-----------------	-------------	---------	-------	----------

Hence OWNER_ID will be the primary key for Owner table.

2.3.5: NORMALIZATION FOR TAX :

<i>DUE_DATE</i>	<i>AMOUNT</i>	<u><i>REG_ID</i></u>
-----------------	---------------	----------------------

REG_ID → DUE_DATE

REG_ID → AMOUNT

First Normal Form:

It is already in 1NF since each column is having a unique value, such that multiple columns cannot be used to fetch the same row.

<i>DUE_DATE</i>	<i>AMOUNT</i>	<u><i>REG_ID</i></u>
-----------------	---------------	----------------------

After Second Normal Form:

This is will be already in 2NF since every non-key attribute is fully functionally dependent on primary key.

<i>DUE_DATE</i>	<i>AMOUNT</i>	<u><i>REG_ID</i></u>
-----------------	---------------	----------------------

After Third Normal Form:

Since there is no transitive functional dependency, therefore table is already in 3NF.

<i>DUE_DATE</i>	<i>AMOUNT</i>	<u><i>REG_ID</i></u>
-----------------	---------------	----------------------

Hence, REG_ID will be the primary key for TAX table.

2.3.5: NORMALIZATION FOR REG_OFFICE :

<u>ADMIN_ID</u>	ADMIN_PWD	ADMIN_NAME
-----------------	-----------	------------

ADMIN_ID → ADMIN_PWD

ADMIN_ID → ADMIN_NAME

First Normal Form:

It is already in 1NF since each column is having a unique value, such that multiple columns cannot be used to fetch the same row.

After Second Normal Form:

This is will be already in 2NF since every non-key attribute is fully functionally dependent on primary key.

<u>ADMIN_ID</u>	ADMIN_PWD	ADMIN NAME
-----------------	-----------	------------

After Third Normal Form:

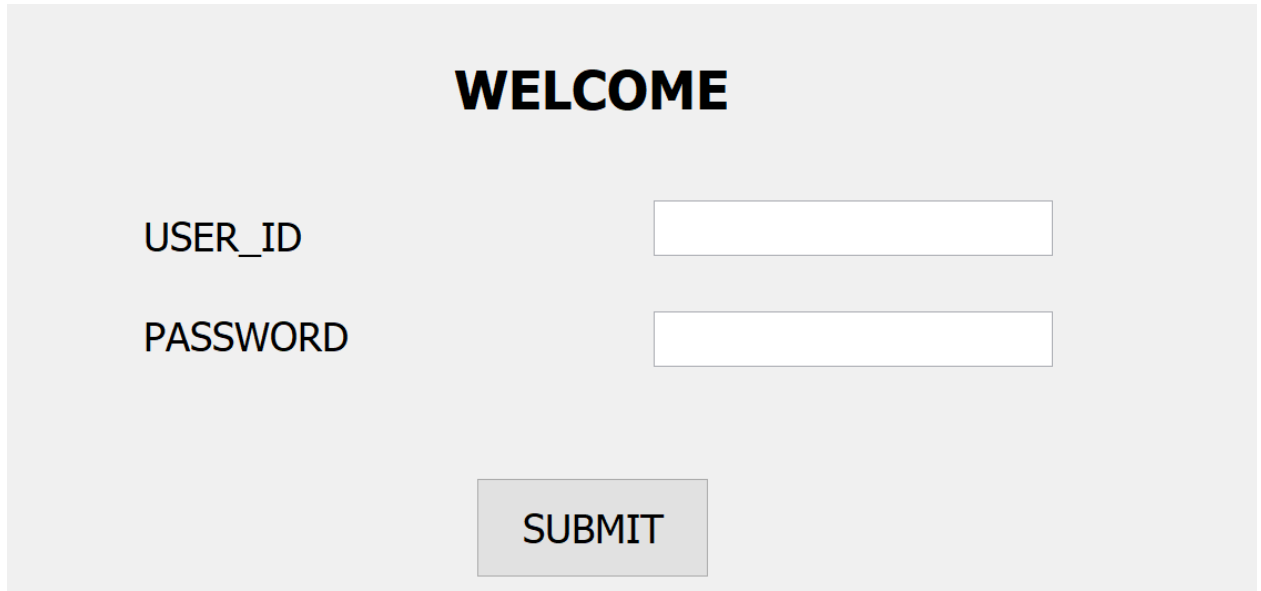
Since there is no transitive functional dependency, therefore table is already in 3NF.

<u>ADMIN_ID</u>	ADMIN_PWD	ADMIN NAME
-----------------	-----------	------------

Hence ADMIN_ID will be the primary key for REG_OFFICE table.

3.1- Screen Layout of Major Modules used in Project:

3.1.1- Initial Design of Login Frame:



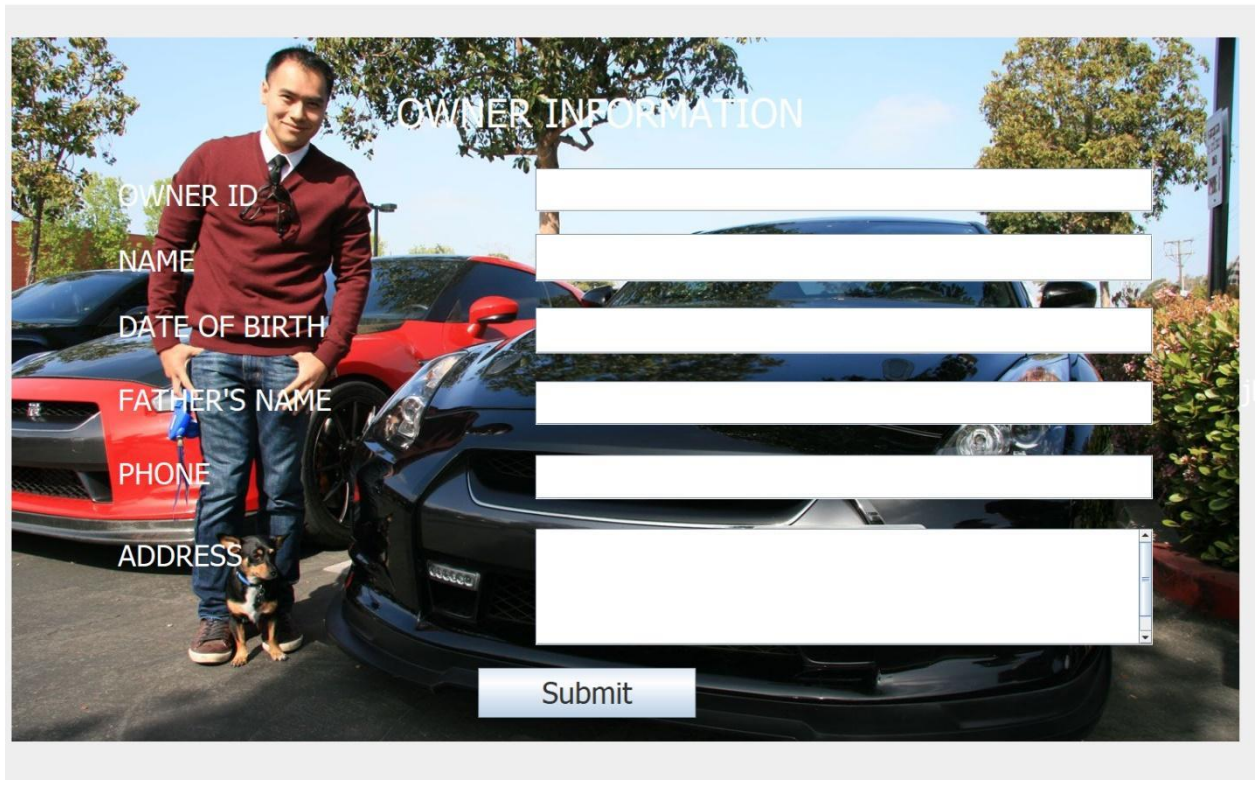
The image shows a login frame with a light gray background. At the top center, the word "WELCOME" is displayed in a large, bold, black font. Below this, there are two input fields. The first field is labeled "USER_ID" and the second is labeled "PASSWORD". Both labels are in a bold, black font. Below the input fields, there is a button labeled "SUBMIT" in a bold, black font. The button has a light gray background and a thin black border. The entire login frame is enclosed in a thin black border.

The initial/paper design of the Login Page of the Project is shown in the above figure. Initially, the officer was able to log on to the database. To log in to the database one has to provide his/her correct Login ID and Password. After clicking on Login button a connection will be established to the database and query will check the database for the respective Login ID and Password field.

The following fields which are used to enter the data from the applications:

- 1- **Login field:** For Login label, the data will be entered using a text field because the ID should be visible to the user.
- 2- **Password field:** Password field is used to get the password from the user. It is important to hide the password while entering as black dark circles for each character entered in the field. Hiding the password is important because if it is visible then another person can easily remember by looking into it and which he/she will use to access the account later.
- 3- **Login Button:** On clicking the login button if the data present in the Login and Password field matches with the data already present in the database this will redirect the user to the Admin Page.

3.1.2- Initial Design of Registration Frame:



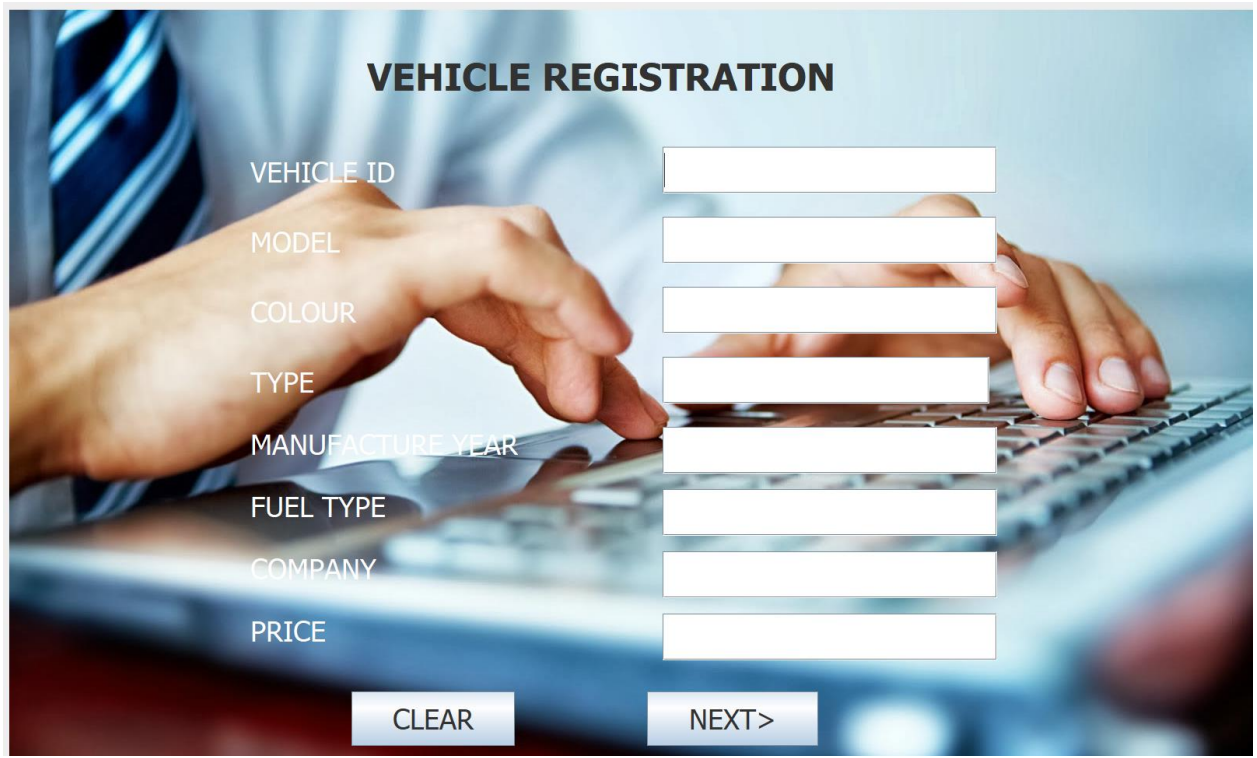
The image shows a web form titled "OWNER INFORMATION" overlaid on a background photograph of a man in a red sweater standing next to a red sports car with a small dog. The form contains the following fields and labels:

- OWNER ID
- NAME
- DATE OF BIRTH
- FATHER'S NAME
- PHONE
- ADDRESS
- Submit

The initial/paper design of the Owner details page of the Project is shown in the above figure. Here the RTO officer has to provide the details about the vehicle owner. A query will run which is going to insert the data into the database by checking all the constraints and triggers specified. The following fields which are used in this page are:

- 1. Name, Date Of Birth, Father's name, Phone, Address:** Text field is used to get the data for labels such as Name, Street, City because the data entered should easily be edited and displayed to the user.
- 2. Submit:** Button is used to submit all the details written above to their specified fields.

3.1.3- Initial Design of Vehicle Registration Page

The image shows a user interface for vehicle registration. It features a title 'VEHICLE REGISTRATION' at the top. Below the title, there are seven text input fields with labels: 'VEHICLE ID', 'MODEL', 'COLOUR', 'TYPE', 'MANUFACTURE YEAR', 'FUEL TYPE', and 'COMPANY'. At the bottom, there are two buttons: 'CLEAR' and 'NEXT>'. The background of the form is a blurred image of a person's hands typing on a laptop keyboard.

The initial/paper design of the Vehicle Registration details page of the Project is shown in the above figure. Here the RTO officer has to provide the details about the vehicle. A query will run which is going to insert the data into the database by checking all the constraints and triggers specified. The following fields which are used in this page are:

- 1- **Company:** There are multiple companies which are manufacturing the vehicle at a particular date. You can write the company name in the text field.
- 2- **Vehicle ID:** The text field which is used to show the particular model of the vehicle in the company ..
- 3- **Submit Button:** On clicking the submit button the data of the fields will be inserted into the database if all the constraints imposed are satisfied.
- 4- **Colour:** Colour TextField is used for giving details of colour of the vehicle
- 5- **Type:** It shows the type of vehicle .For example-lmv(light motor vehicle)or hmv(heavy motor vehicle).
- 6- **Reset Button:** On clicking the Reset button all the data will be cleared from the text fields used in the page.

3.2- Connectivity to the Database from Front End:

The process of connecting the Application and the Database is divided into five routines. These include:

- Loading the JDBC driver
- Connecting to the DBMS
- Creating and executing a statement
- Processing data returned by the DBMS
- Terminating the connection with the DBMS

The Steps are as follows:

3.2.1 Loading the JDBC Driver:-

The JDBC driver must be loaded before the J2EE component can connect to the DBMS. The **Class.forName()** method is used to load the JDBC driver. The developer must write a routine that loads the JDBC/ODBC Bridge driver called **sun.jdbc.odbc.JdbcOdbcDriver**. The driver is loaded by calling the **Class.forName()** method and passing it in the name of the driver, as shown in the following code:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

3.2.2 Connect to the DBMS:-

Once the driver is loaded, the J2EE component must connect to the DBMS using the **DriverManager.getConnection()** method. The **java.sql.DriverManager** class is the highest class in the **java.sql** hierarchy and is responsible for managing driver information.

The **DriverManager.getConnection()** method passes the URL as a String object that contains the driver name and the name of the database is being accessed by the J2EE component. The **DriverManager.getConnection()** method returns a **Connection** interface that is used throughout the process to reference the database. The **java.sql.Connection** interface in another member of the **java.sql** package that manages communication between the driver and the J2EE component. It is the **java.sql.Connection** interface that sends statements to the DBMS for processing. Listing 6-1 illustrates the use of the **DriverManager.getConnection()** method to load the JDBC/ODBC Bridge and connect to the **UserInfo** database.

```
String url = "jdbc:odbc:UserInfo";
```

```
String userID = "jim";
```

```
String password = "keogh";
```

```
Statement DataRequest;
```

```
Private Connection Db;
```

```
try{
```

```
    Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
    Db= DriverManager.getConnection(url,userID,password);
```

```
}
```

3.2.3 Create and Execute a SQL Statement:-

The next step, after the JDBC driver is loaded and connection is successfully made with a particular database managed by the DBMS, is to send a SQL query to the DBMS for processing. A SQL query consist of a series of SQL commands that direct the DBMS to do something such as to return rows of data to the J2EE component.

The `Connect.createStatement()` method is used to create a `Statement` object. The `Statement` object is then used to execute a query and return a `ResultSet` object that contains the response from the DBMS ,which is usually one or more rows of information requested by the J2EE component.

Typically, the query is assigned to a `String` Object, which is passed to the `Statement` object's `executeQuery()` method, which is illustrated in the next code segment. Once the `ResultSet` is received from the DBMS, the `close()` method is called to terminate the statement. Listing 6-2 retrives all the rows and columns from the column table.

```
Statement DataRequest;
```

```
ResultSet Results;
```

```
try{
```

```
String query = " SELECT * FROM USER";
```

```
DataRequest = Database.createStatement();
```



```
DataRequest = Db.createStatement();  
  
Results = DataRequest.executeQuery (query) ;  
  
DataRequest.close();  
  
}
```

3.2.4 Process Data Returned by the DBMS:-

The `java.sql.ResultSet` object is assigned the results received from the DBMS after the query is processed. The `java.sql.ResultSet` object consists of method used to interact with data that is returned by the DBMS to the J2EE component.

Later in this chapter you'll learn the details of using the `java.sql.ResultSet` object. However, the following code is an abbreviation example that gives you a preview of a commonly used routine used to extract data returned by the DBMS. Error-catching code is purposely removed from this example in order to minimize code clutter.

Assume for Listing 6-3 that a J2EE component requested user's first name and last names from a table. The `resultSet` returned by the DBMS is already assigned to the `ResultSet` object called `Results`. The first time that `next()` method of the `ResultSet` is called, the `ResultSet` pointer is positioned at first row in the `ResultSet` and return a Boolean value that if false indicates that no rows are present in the `ResultSet`. The if statement in Listing 6-3 traps this condition and displays the "End Of Data" message on the screen.

3.2.5 Terminating the connection with the DBMS:-

The connection to the DBMS is terminated using the `close()` method of connection object once the Java component is finished accessing the DBMS. The `close()` method throws an exception if a problem is encountered when disengaging the DBMS. The following is an example of calling the `close()` method. Although closing the database connection automatically closes the `ResultSet`, it is better to close the `ResultSet` explicitly before closing the connection.

```
Db.close();
```


4.1: Project modules:

Login Page Module:

Officer can Login to the System to view the vehicle registration using their Username and password as provided by senior authorities. A query will check the respective fields with the database. If the data is matched then User can view his Vehicle Registration Details.

Officer can also login as an Administrator using Username and password. The data will be matched with the database and if it is matched then User will get logged into to the Admin Page.

Vehicle Registration Module:

Two-wheeler, Three-Wheeler, Light motor Vehicle and Heavy motor vehicle and others can be registered into the database. To register a vehicle through this software Officer must provide all the details of vehicles including vehicle number, model, manufacture year, fuel type ,company,price etc.

Owner Detail Module:

This module is responsible for taking the Vehicle Owner details. The Officer has to provide Owner details like his/her Name, Address, Date of Birth, Gender etc. An Insertion Query will run which will save all the Owner data to the database with their respective Registration Card Number.

Vehicle Detail Module:

This module is responsible for taking the Vehicle details. The Officer will provide vehicle details such as vehicle model,company, body type, colour of vehicle, price etc. An Insertion Query will run and the data will be inserted to the database to the respective vehicle number of the Vehicle.

Tax Module:

This module is responsible for storing the the tax details that is tax Amount and the due date for the amount to be paid by the owner for the particular vehicle .

View Details Module:

This module is responsible for displaying the details already stored in the database. There are two things which have been provided to be displayed:

- a- Vehicle Registration Details
- b- Owner Details

The RTO Officer has to provide the Vehicle ID of the Vehicle to see the details of the Vehicle. To view the owner details the RTO Officer has to provide owner_id .Vehicle can only be registered if the owner is available for it .

5.1: Implementation for Login Module:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    int flag=0;  
  
    String s="s";  
  
    PreparedStatement st=null;  
  
    try {  
  
        Connection  
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime  
        Behavior=convertToNull","root","14011996");  
  
        st=con.prepareStatement("Select * from Login");  
  
  
        ResultSet rs=st.executeQuery();  
  
  
        while(rs.next())  
  
        {    //  
  
            if(rs.getString(1).equals(jTextField1.getText())&&rs.getString(2).equals(jPasswordFiel  
            d1.getText() )  
  
            {  
  
                flag=1;  
  
                s=rs.getString(3);  
  
                System.out.println("ER");  
  
            }  
  
        }  
  
        if(flag==1)  
  
        {  
  
            new startup(s).setVisible(true);  
  
        }  
    }  
}
```

```

this.dispose();
}

else

JOptionPane.showMessageDialog(null,"Login failed");

} catch (SQLException ex) {

Logger.getLogger(LOGIN.class.getName()).log(Level.SEVERE, null, ex);

}

}

```

5.2:Implementation of change ownership

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

Connection con=null;

PreparedStatement stm=null;

try {

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime
Behavior=convertToNull","root","14011996"); // TODO add your handling code here:

stm=con.prepareStatement("Select * from owner where owner_id=?");

stm.setString(1,t1.getText());

ResultSet rs=stm.executeQuery();

int flag=0;

while(rs.next())

{

flag=1;

```

```

    }

    if(flag==1){

        new new_owner_info(t1.getText()).setVisible(true);
        this.dispose();
    }

    else

        JOptionPane.showMessageDialog(null,"Not Found");

    } catch (SQLException ex) {

        JOptionPane.showMessageDialog(null,"OWNER DOESN'T EXIST");

        Logger.getLogger(vehicle2.class.getName()).log(Level.SEVERE, null, ex);

    }

}

```

5.3: Implementation of new owner :

```

import java.sql.*;

import java.sql.DriverManager;

import java.util.logging.Level;

import java.util.logging.Logger;

public class new_owner_info extends javax.swing.JFrame {

    String oid;

    public new_owner_info(String r) {

        oid=r;
    }

```

```

initComponents();

FillCombo();

}

/**

 * Creates new form new_owner_info

 */

public new_owner_info() {

    initComponents();

}

private void FillCombo(){

    Connection con=null;

    PreparedStatement stm=null;

    System.out.println("hry");

    try {

        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime
        Behavior=convertToNull","root","14011996");

        //System.out.println("hry"+oid);

        stm=con.prepareStatement("select * from registration where cus_id=?");

        // System.out.println("hry"+oid);

        stm.setString(1,oid);

        //System.out.println("hry");

        ResultSet rs=stm.executeQuery();

        while(rs.next())

        {

```

```
cmb.addItem(rs.getString(2));  
  
}  
  
} catch (SQLException ex) {
```

```
Logger.getLogger(vehicle2.class.getName()).log(Level.SEVERE, null, ex);  
  
}  
  
}
```

5.4: Implementation of New Owner:

```
public class ownerInfoDisplay extends javax.swing.JFrame {  
  
String oid;  
  
public ownerInfoDisplay(String oid) {  
  
oid=oid;
```

```
initComponents();  
  
setExtendedState(java.awt.Frame.MAXIMIZED_BOTH);  
  
FillCombo();
```

```
ret();  
  
txtOid.setText(oid);  
  
txtOid.setEditable(false);
```

```
txtFatherName.setEditable(false);  
  
txtDOB.setEditable(false);  
  
txtPhone.setEditable(false);  
  
txtName.setEditable(false);  
  
taAddress.setEditable(false);
```

```

}

public ownerInfoDisplay() {

initComponents();

setExtendedState(java.awt.Frame.MAXIMIZED_BOTH);

}

private void FillCombo()

{   PreparedStatement st=null;

try {

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime
Behavior=convertToNull","root","14011996");

System.out.println(owid);

st=con.prepareStatement("select * from registration where cus_id=?");

st.setString(1,owid);

System.out.println("545");

ResultSet rs=st.executeQuery();

while(rs.next())

{

cb.addItem(rs.getString("veh_no"));

}

} catch (SQLException ex) {

Logger.getLogger(ownerInfoDisplay.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void ret()

```

```

{
PreparedStatement stm=null;

try {

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime
Behavior=convertToNull","root","14011996");

stm=con.prepareStatement("Select * from owner where owner_id=?" );

stm.setString(1,owid);

System.out.println("123");

ResultSet rs1=stm.executeQuery();

while(rs1.next())
{
System.out.println("123");
txtName.setText(rs1.getString("name"));
txtDOB.setText(rs1.getString("dob"));
txtOid.setText(rs1.getString("owner_id"));
txtFatherName.setText(rs1.getString("father_name"));
taAddress.setText(rs1.getString("address"));
txtPhone.setText(rs1.getString("phone"));
}

} catch (SQLException ex) {

Logger.getLogger(ownerInfoDisplay.class.getName()).log(Level.SEVERE, null, ex);

}

}

```


5.6: Implementation of Owner Info:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
    Connection con=null;  
  
    PreparedStatement stm=null;  
  
  
    try {  
  
        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime  
Behavior=convertToNull","root","14011996");  
  
        stm=con.prepareStatement("insert into owner values(?,?,?,?);");  
  
        stm.setString(1,txtName.getText());  
  
        stm.setString(2,txtDOB.getText());  
  
        stm.setString(3,txtOid.getText());  
  
        stm.setString(4,txtFatherName.getText());  
  
        stm.setString(5,taAddress.getText());  
  
        stm.setString(6,txtPhone.getText());  
  
  
  
        int i=stm.executeUpdate();  
  
        if(i>0)  
        {  
            JOptionPane.showMessageDialog(null,"Registration Successfull");  
  
        }  
  
        else  
            JOptionPane.showMessageDialog(null,"Invalid Input");  
    }  
}
```

```

    } catch (SQLException ex) {

    Logger.getLogger(new_owner_info.class.getName()).log(Level.SEVERE, null, ex);

    }

}

```

5.7:Implementation of Tax:

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

    Connection con=null;

    String rid=txtVid.getText();

    String vid="";

    String price="";

    String date="";

    int flag=0;

    PreparedStatement stm=null;

    try {

        con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime
        Behavior=convertToNull","root","14011996");

        stm=con.prepareStatement("select * from registration where reg_id=?");

        stm.setString(1,rid);

        ResultSet rs=stm.executeQuery();

        while(rs.next())

        {   date=rs.getString("reg_date");

            vid=rs.getString("veh_no");

        }

        System.out.println(date);
    }
}

```

```

int dat=Integer.parseInt(date.substring(6));

dat=dat+1;

System.out.println(dat);

String newdat=date.substring(0, 6) + Integer.toString(dat);

System.out.println(newdat);

txtDueDate.setText(newdat);

PreparedStatement stm1=con.prepareStatement("select * from VEHICLE where
V_id=?");

stm1.setString(1,vid);


ResultSet rs1=stm1.executeQuery();

while(rs1.next())

{   flag=1;

price=rs1.getString("price");

}

// System.out.println(flag);

int i=Integer.parseInt(price);

// System.out.println(i);

i=(int) (0.01*i);

System.out.println(i);

txtTaxAmt.setText(Integer.toString(i));


} catch (SQLException ex) {

Logger.getLogger(tax.class.getName()).log(Level.SEVERE, null, ex);

}

}

```

5.6:Implementation of vehicle registration:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
    Connection con=null;  
  
    PreparedStatement stm=null;  
  
    if(flag==0) {  
        try {  
  
            con=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTime  
            Behavior=convertToNull","root","14011996");  
  
            stm=con.prepareStatement("insert into registration values(?,?,?,?)");  
  
            stm.setString(1,txtRegid.getText());  
  
            stm.setString(2,txtVid.getText());  
  
            stm.setString(3,txtRegDate.getText());  
  
            stm.setString(4,txtOid.getText());  
  
  
  
            int i=stm.executeUpdate();  
  
            System.out.print(i);  
  
            if(i>0)  
            {  
                JOptionPane.showMessageDialog(null,"Registration Successfull");  
            }  
            else  
                JOptionPane.showMessageDialog(null,"Invalid Input");  
        }  
    }  
}
```

```

    } catch (SQLException ex) {

        Logger.getLogger(new_owner_info.class.getName()).log(Level.SEVERE, null, ex);
    }
}

if(flag==1){
    PreparedStatement stm1=null;

    String rdate=txtRegDate.getText();

    String rid=txtRegid.getText();

    System.out.println(rid);

    try {
        Connection
        conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/project?zeroDateTim
        eBehavior=convertToNull","root","14011996");

        stm1=conn.prepareStatement("update                registration                set
        cus_id=?,reg_date=?,reg_id=?,veh_no=? where veh_no=?");

        System.out.println(old+"HYU");

        //system.out.println(oid);

        stm1.setString(1,oid);

        stm1.setString(2,rdate);

        stm1.setString(3,txtRegid.getText());

        stm1.setString(4,old);

        stm1.setString(5,old);

        System.out.println(old);

        System.out.println(oid);

        int i=stm1.executeUpdate();

        System.out.print(i);
    }
}

```

```
if(i>0)
{

JOptionPane.showMessageDialog(null,"UPDATION Successfull");

}

else

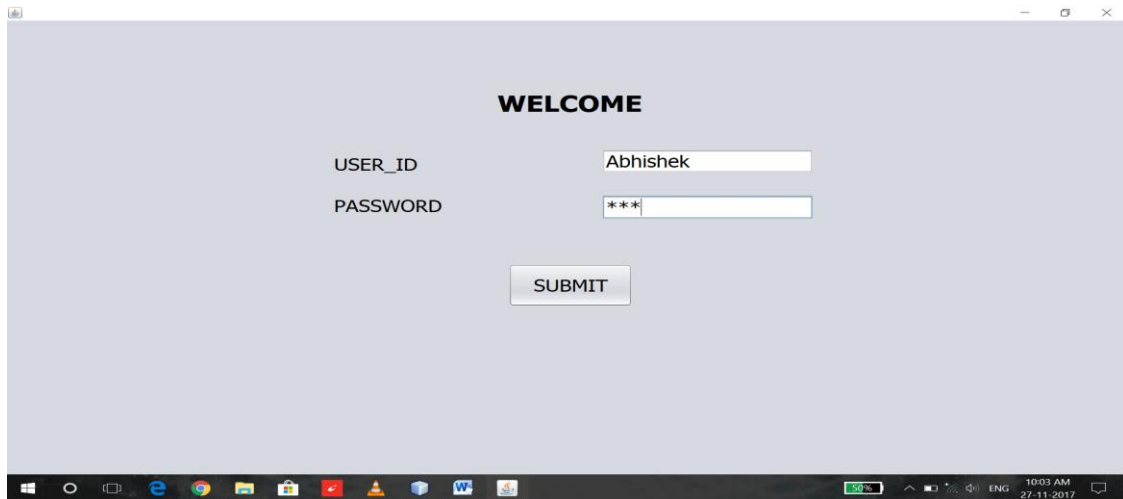
JOptionPane.showMessageDialog(null,"Invalid Input");

} catch (SQLException ex) {
Logger.getLogger(veh_reg.class.getName()).log(Level.SEVERE, null, ex);
}
}
}
```

WELCOME

USER_ID

PASSWORD



VEHICLE REGISTRATION MANAGEMENT SYSTEM

WELCOME: Abhishek Jha

TransportIndia.in

R.T.O. Transport Tourism

Welcome TransportIndia

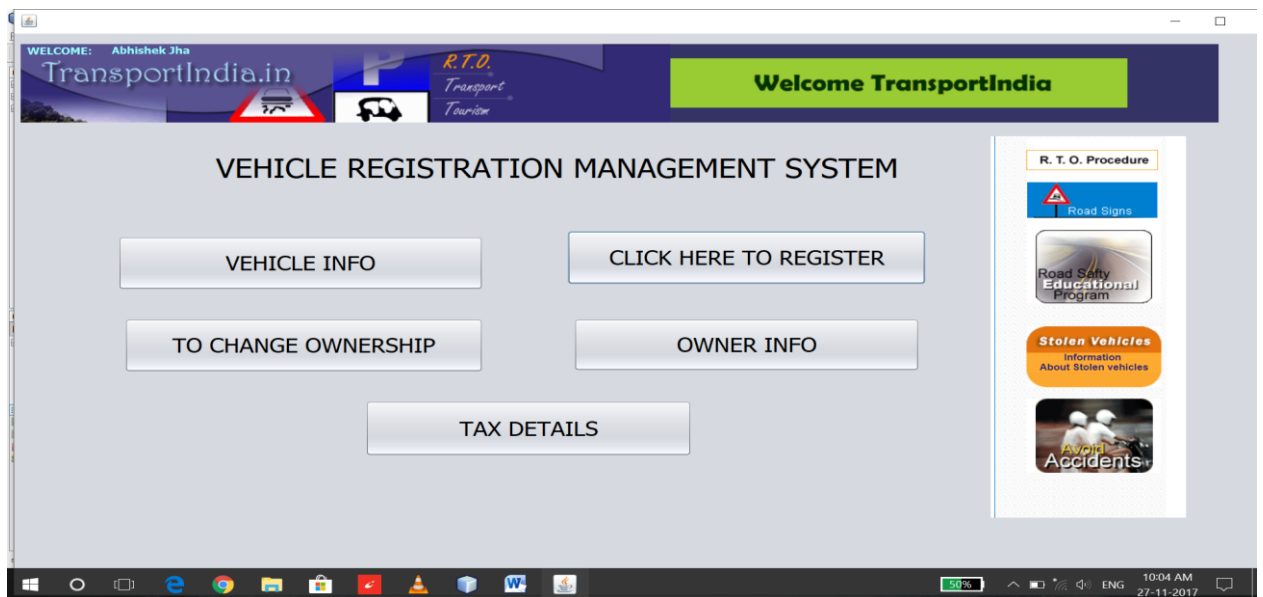
R. T. O. Procedure

Road Signs

Road Safety Educational Program

Stolen Vehicles
Information About Stolen vehicles

Avoid Accidents



OWNER INFORMATION

OWNER ID

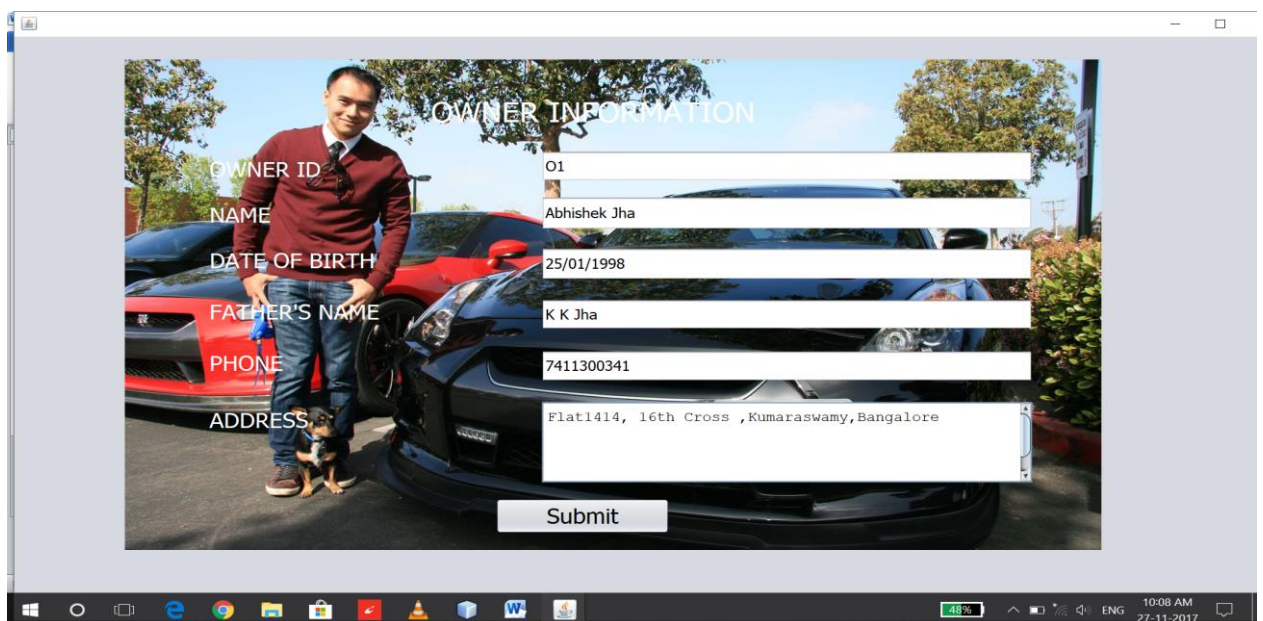
NAME

DATE OF BIRTH

FATHER'S NAME

PHONE

ADDRESS



VEHICLE REGISTRATION

VEHICLE ID	V1
MODEL	EZ1
COLOUR	BLACK
TYPE	SUV
MANUFACTURE YEAR	2015
FUEL TYPE	PETROL
COMPANY	MERCEDEZ
PRICE	800000

Registration Details

Enter Registration ID	R1
Enter Vehicle Number	V1
Enter Registration Date	22/11/2017
Enter Owner ID	O1

OWNER INFORMATION

OWNER ID: O1

NAME: Abhishek Jha

DATE OF BIRTH: 25/01/1998

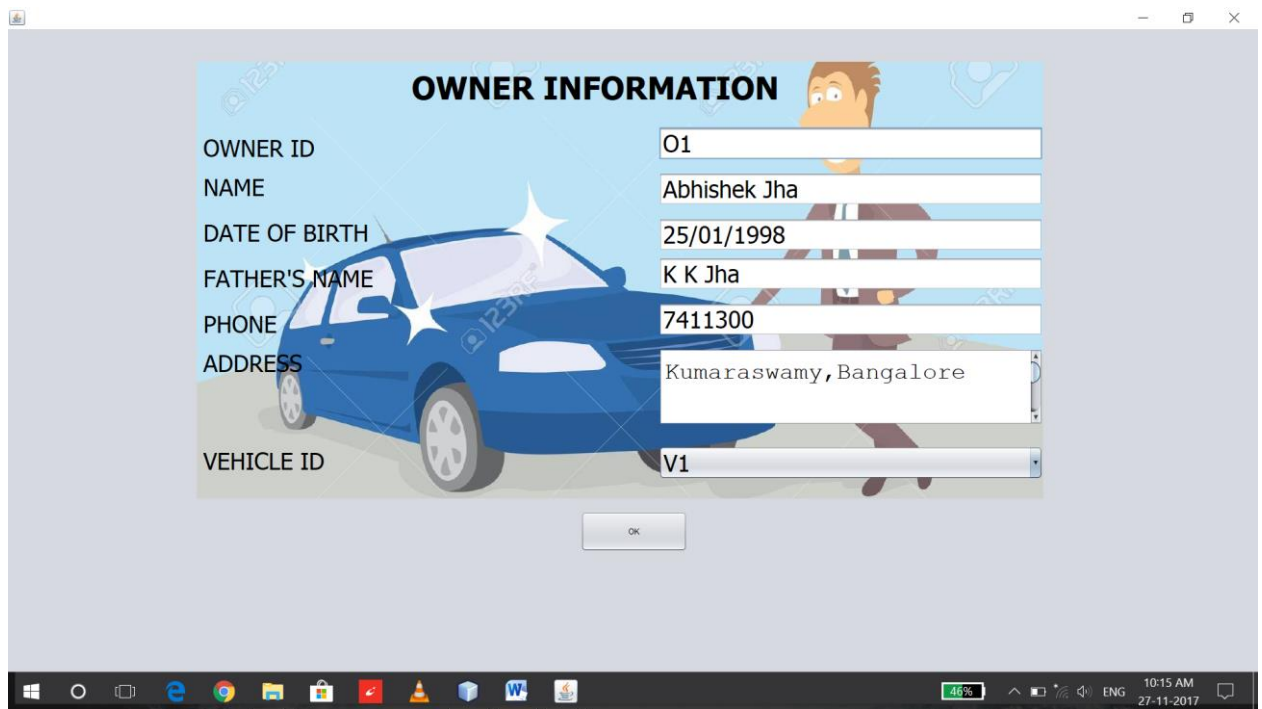
FATHER'S NAME: K K Jha

PHONE: 7411300

ADDRESS: Kumaraswamy, Bangalore

VEHICLE ID: V1

OK



VEHICLE DETAILS

VEHICLE ID: V1

MODEL: EZ1

COLOUR: BLACK

TYPE: SUV

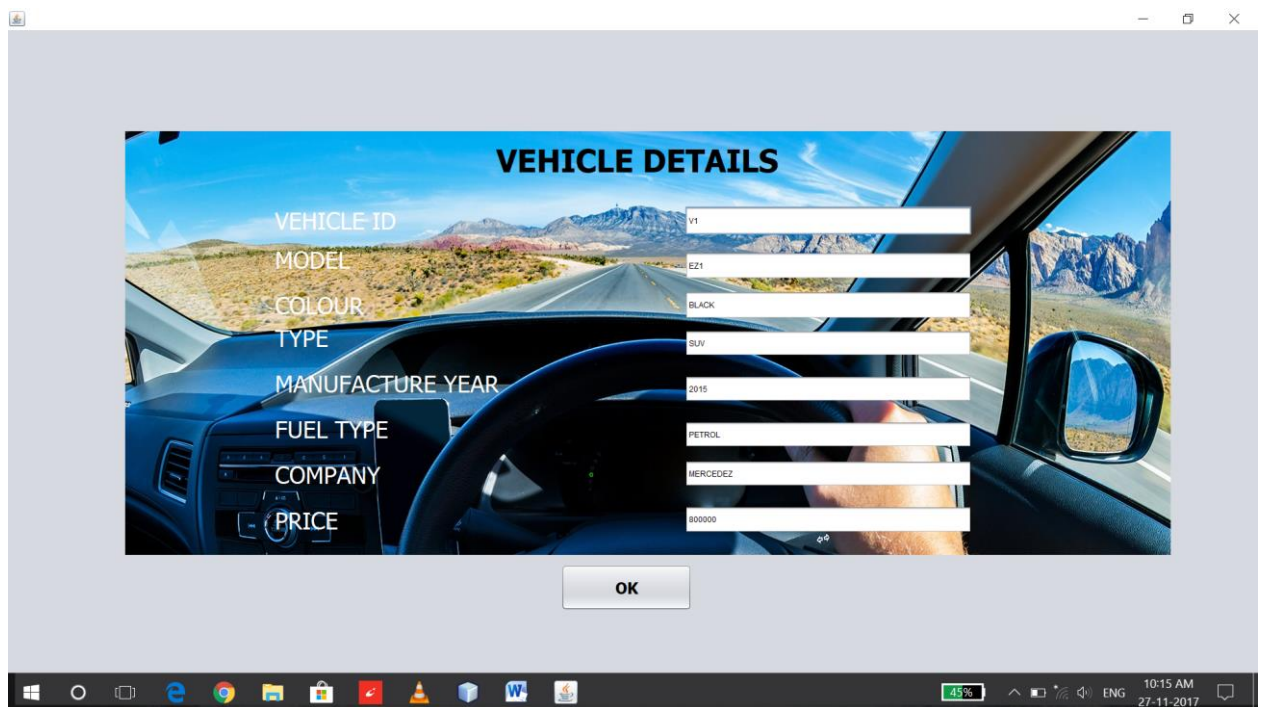
MANUFACTURE YEAR: 2015

FUEL TYPE: PETROL

COMPANY: MERCEDES

PRICE: 800000

OK

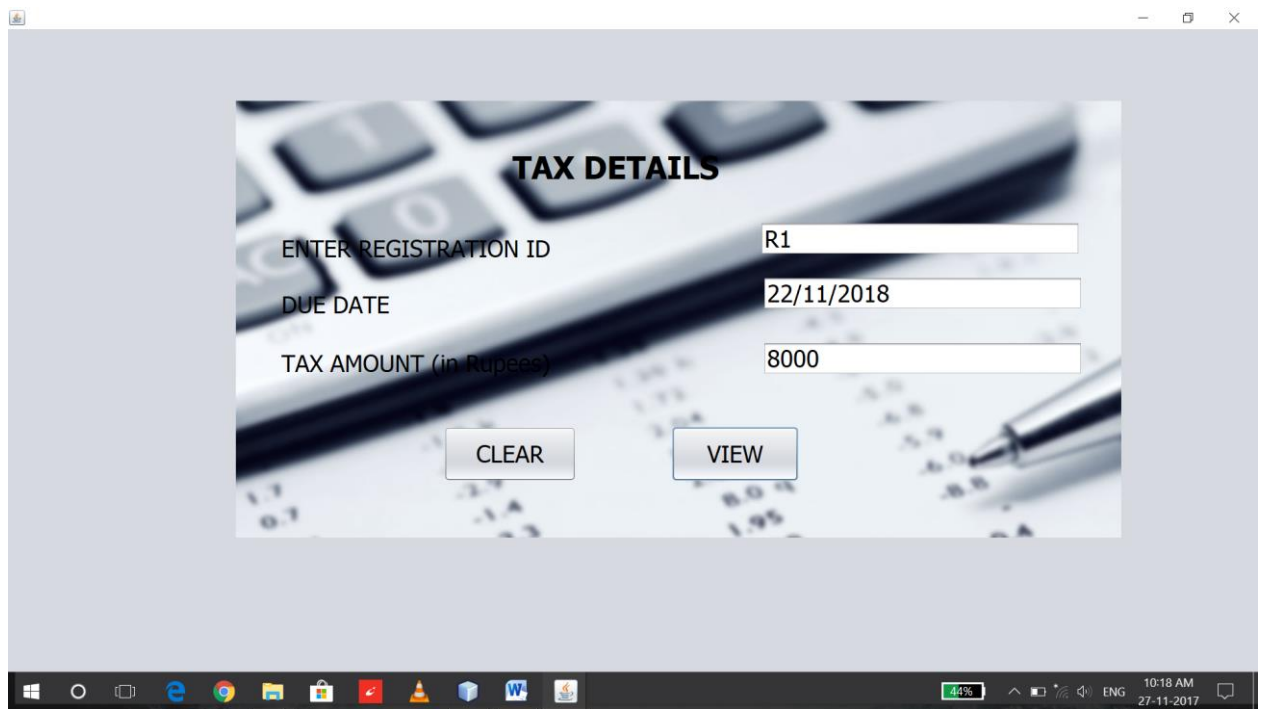


TAX DETAILS

ENTER REGISTRATION ID

DUE DATE

TAX AMOUNT (in Rupees)



The applications of vehicle registration system are as follows :-

1. Easy to register a vehicle with the owner .
2. To change the ownership of the vehicle if the vehicle is sold to the new owner easily .
3. To retrieve the details of the vehicle,owner or registration on a single click .
4. To digitalize the hectic paper work which used to consume a lot of precious time .
5. This system will maintain and manage day to day operation very smoothly with minimum errors.
6. In the main menu of our application, we have created options to Insert a New registration, Modify the existing registration, Delete an existing application.
7. Admin can find any detail related to registration by just providing registration no of vehicle.

I would like to conclude that with the use vehicle registration management system,RTO officers can effectively do the registration of the vehicle and the owner,reduce their carbon footprint by eliminating the use of paper and provide better customer service . Officers can also change the ownership of the vehicle sold to other owner . Elimination of the paper work saves lots of time and also reduces cost .

I would like to thank my professor Prof. Bhavana, Prof Manasa c and the head of the department Dr. S Nandagopalan for their continued support and feedback. I would also like to thank the teaching and non- teaching staff of the CS&E department of BIT for the resources they provided that made this project possible. Finally I would like to thank my family and friends for their support.