A

Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

By

PHANI MEGHANA<1602-21-737-028>

Under the Guidance of

B. Leelavathy

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

2022-2023

BONAFIDE CERTIFICATE

This is to certify that this project report titled

‘AUTOMATIC FUEL TRACKER IN VEHICLES’

is a project work of Phani Meghana bearing roll no. 1602-21-737-028 who carried out this project under my supervision in the IV semester for the academic year 2022- 2023.

Signature	Signature
External Examiner	Internal Examiner

AUTOMATIC FUEL TRACKER IN VEHICLES

ABSTRACT

A fuel tracker system is required to help vehicle owners keep track of their fuel usage and efficiency. By monitoring and analyzing fuel consumption data, users can make informed decisions about their driving habits, vehicle maintenance, and fuel expenses. The "Automatic Fuel Tracker in Vehicles" project aims to develop a database system that can automatically track fuel usage and efficiency in vehicles. The database includes tables for vehicles, fueling records, fuel types, and users. The relationships among the tables are established through primary and foreign keys, ensuring data integrity and consistency. The system enables users to view and analyze fuel usage data for their vehicles, helping them make informed decisions about their driving habits and vehicle maintenance.

List of Tables:

- Users table
- Vehicle table
- Fuelling table
- Fuel_type table

Attributes and Domain Types:

Users table:

- user_id number(10)
- vehicle_id number(10)
- user_name varchar2(20)
- city varchar2(20)

Vehicle table:

- user_id number(10)
- vehicle_id number(10)
- year number(5)
- brand varchar2(20)
- model varchar2(20)

Fuelling table:

- vehicle_id number(10)
- fuelling_id number(10)
- day DATE
- liters number(5,2)

Fuel_type table:

- vehicle_id number(10)
- fuelling_id number(10)
- name varchar2(20)
- price_per_liter number(5,2)

Key constraints:

Users table:

- user_id – primary key

Vehicle table:

- vehicle_id – primary key
- user_id – foreign key

Fuelling table:

- fuelling_id – primary key
- vehicle_id – foreign key

Fuel_type table:

- (vehicle_id, fuelling_id) – primary key
- vehicle_id – foreign key
- fuelling_id – foreign key

AIM AND PRIORITY OF THE PROJECT

To create a Java GUI-based desktop application that connects students looking for career choices with skills and Interest. It takes values like student name, username, Age, Skills, etc through forms which are then updated in the database using JDBC connectivity.

ARCHITECTURE AND TECHNOLOGY

Software used:

Java, Oracle 11g Database, Java SE version 14, Run SQL.

Java SWING:

Java SWING is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) - an API for providing a graphical user interface (GUI) for Java programs.

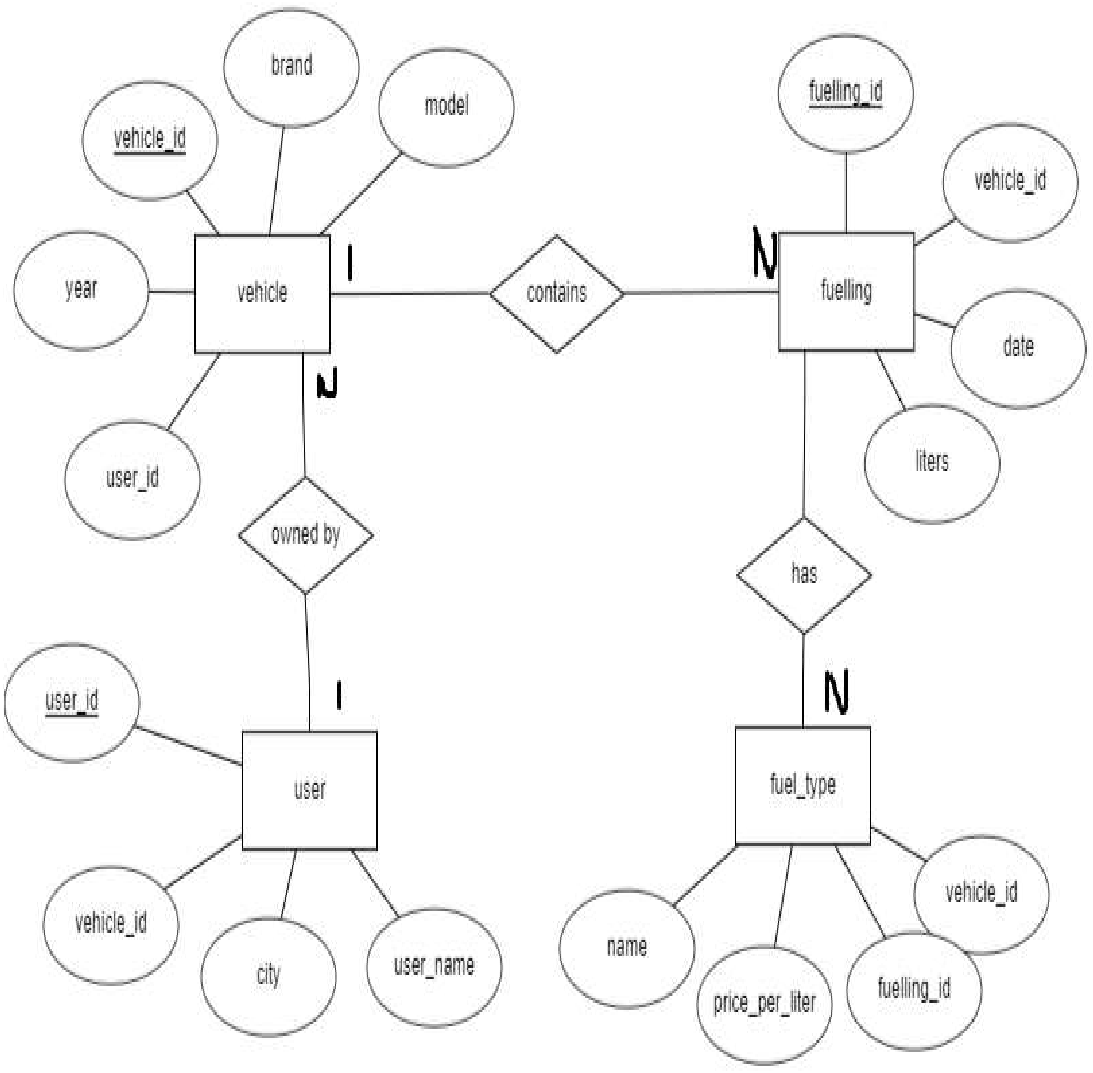
Swing was developed to provide a more sophisticated set of GUI components than the earlier AWT. Swing provides a look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists

SQL:

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS

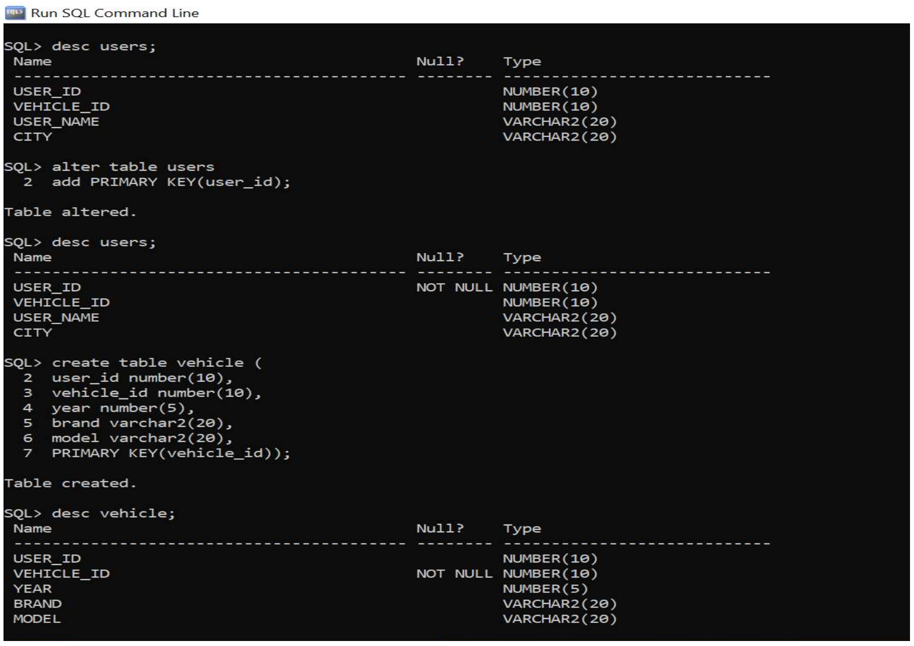
DESIGN

ER DIAGRAM



DDL COMMANDS:

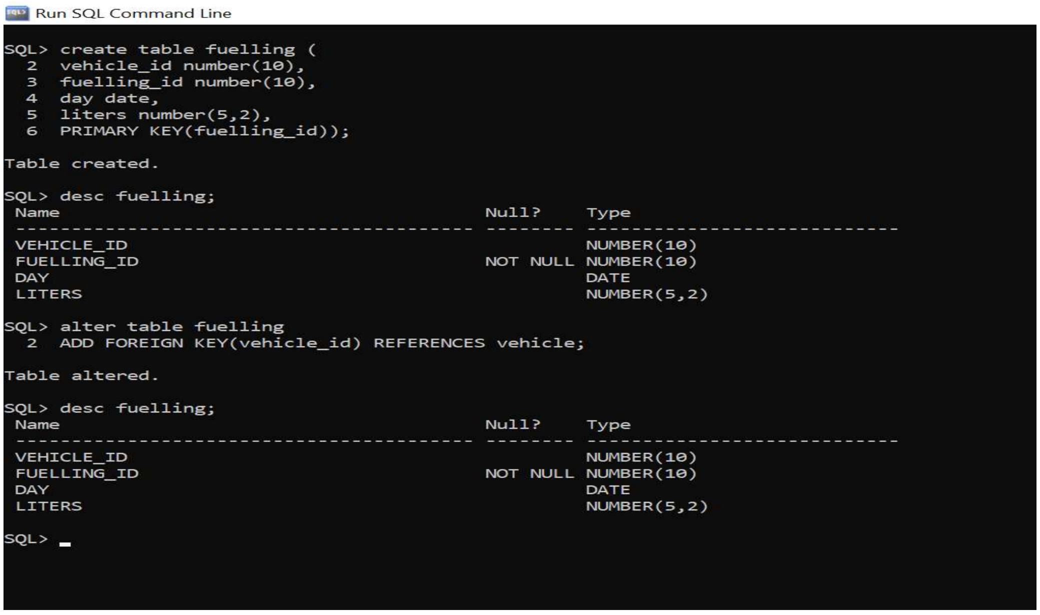
Users and Vehicle tables



Run SQL Command Line

```
SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                            NUMBER(10)
 VEHICLE_ID                                         NUMBER(10)
 USER_NAME                                          VARCHAR2(20)
 CITY                                               VARCHAR2(20)

SQL> alter table users
  2  add PRIMARY KEY(user_id);

Table altered.

SQL> desc users;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                   NOT NULL NUMBER(10)
 VEHICLE_ID                                         NUMBER(10)
 USER_NAME                                          VARCHAR2(20)
 CITY                                               VARCHAR2(20)

SQL> create table vehicle (
  2  user_id number(10),
  3  vehicle_id number(10),
  4  year number(5),
  5  brand varchar2(20),
  6  model varchar2(20),
  7  PRIMARY KEY(vehicle_id));

Table created.

SQL> desc vehicle;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 USER_ID                                            NUMBER(10)
 VEHICLE_ID                                NOT NULL NUMBER(10)
 YEAR                                               NUMBER(5)
 BRAND                                              VARCHAR2(20)
 MODEL                                              VARCHAR2(20)
```

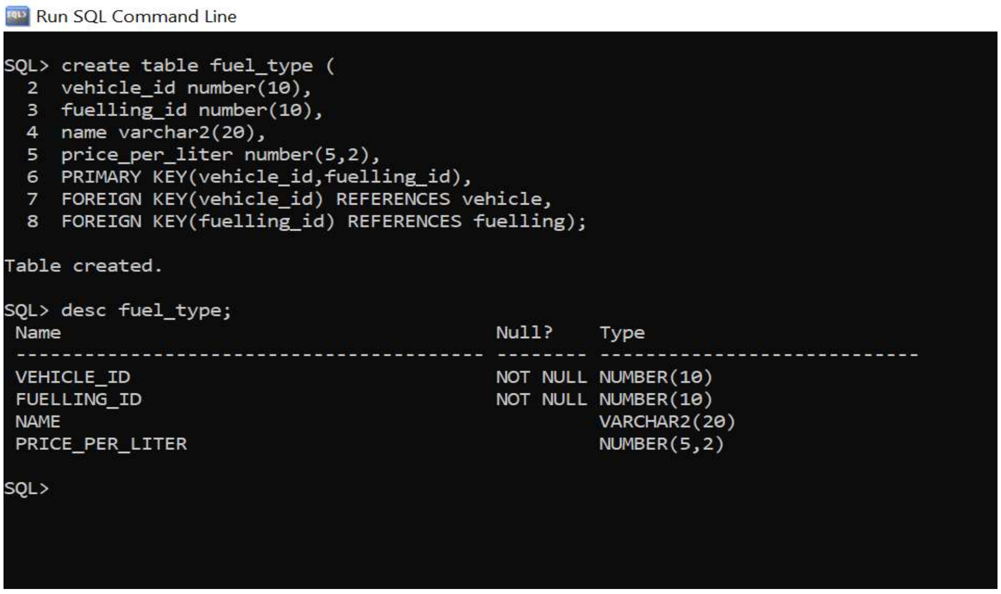
Fuelling Table



Run SQL Command Line

```
SQL> create table fuelling (
  2  vehicle_id number(10),
  3  fuelling_id number(10),
  4  day date,
  5  liters number(5,2),
  6  PRIMARY KEY(fuelling_id));

Table created.

SQL> desc fuelling;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                         NUMBER(10)
 FUELLING_ID                               NOT NULL NUMBER(10)
 DAY                                                DATE
 LITERS                                             NUMBER(5,2)

SQL> alter table fuelling
  2  ADD FOREIGN KEY(vehicle_id) REFERENCES vehicle;

Table altered.

SQL> desc fuelling;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                         NUMBER(10)
 FUELLING_ID                               NOT NULL NUMBER(10)
 DAY                                                DATE
 LITERS                                             NUMBER(5,2)

SQL> _
```

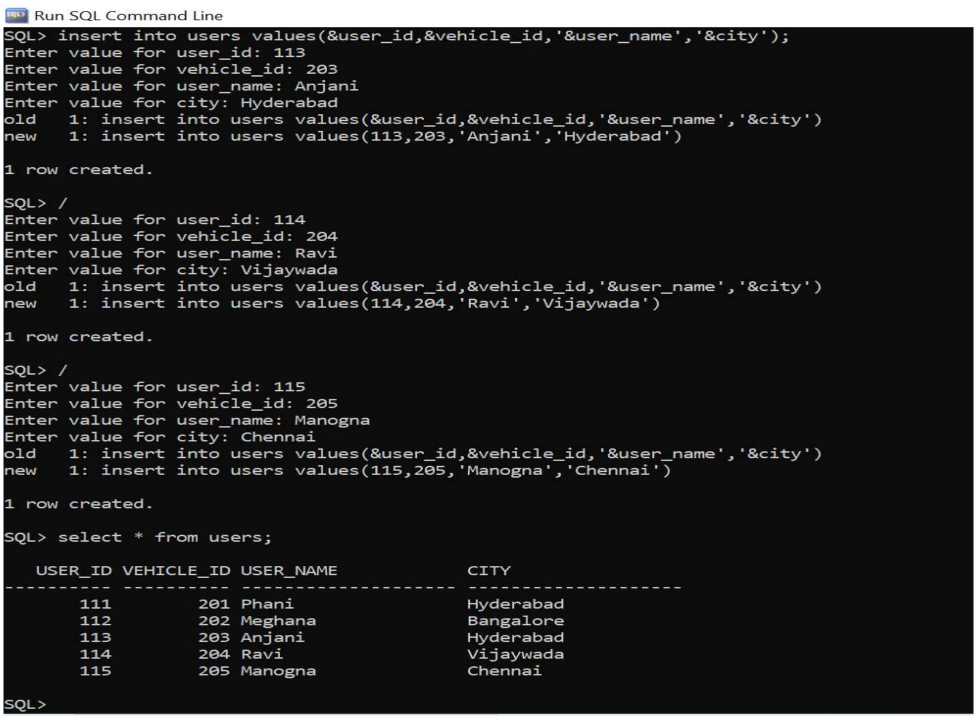
Fuel_type table



```
SQL> create table fuel_type (
  2  vehicle_id number(10),
  3  fuelling_id number(10),
  4  name varchar2(20),
  5  price_per_liter number(5,2),
  6  PRIMARY KEY(vehicle_id,fuelling_id),
  7  FOREIGN KEY(vehicle_id) REFERENCES vehicle,
  8  FOREIGN KEY(fuelling_id) REFERENCES fuelling);

Table created.

SQL> desc fuel_type;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                NOT NULL NUMBER(10)
 FUELLING_ID                               NOT NULL NUMBER(10)
 NAME                                               VARCHAR2(20)
 PRICE_PER_LITER                                    NUMBER(5,2)

SQL>
```

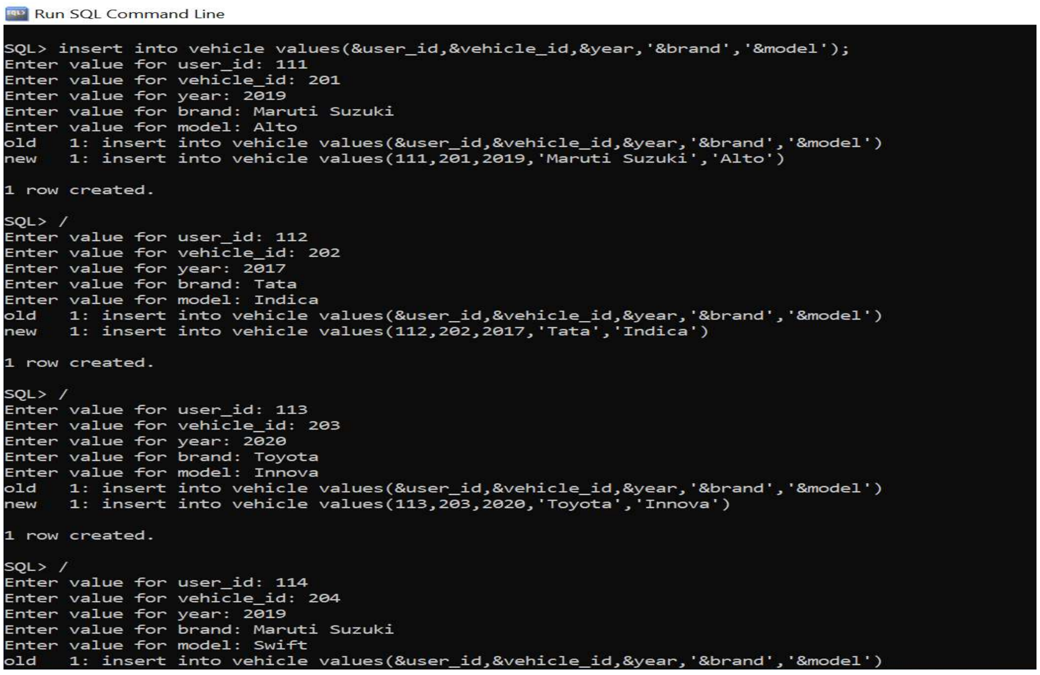
DML COMMANDS:

Users table:

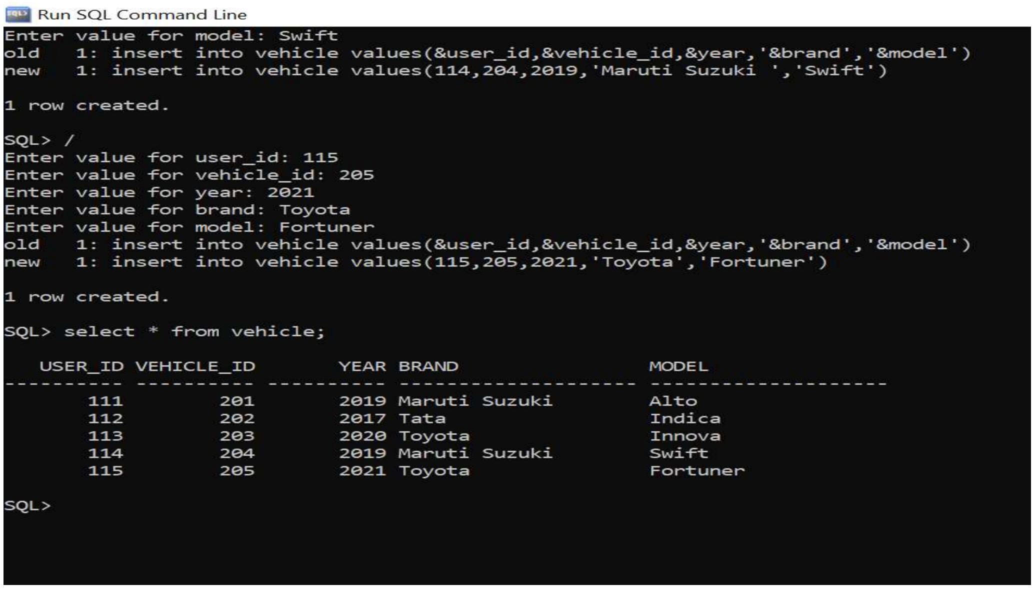


```
SQL> insert into users values(&user_id,&vehicle_id,'&user_name','&city');
Enter value for user_id: 113
Enter value for vehicle_id: 203
Enter value for user_name: Anjani
Enter value for city: Hyderabad
old   1: insert into users values(&user_id,&vehicle_id,'&user_name','&city')
new   1: insert into users values(113,203,'Anjani','Hyderabad')

1 row created.

SQL> /
Enter value for user_id: 114
Enter value for vehicle_id: 204
Enter value for user_name: Ravi
Enter value for city: Vijaywada
old   1: insert into users values(&user_id,&vehicle_id,'&user_name','&city')
new   1: insert into users values(114,204,'Ravi','Vijaywada')

1 row created.

SQL> /
Enter value for user_id: 115
Enter value for vehicle_id: 205
Enter value for user_name: Manogna
Enter value for city: Chennai
old   1: insert into users values(&user_id,&vehicle_id,'&user_name','&city')
new   1: insert into users values(115,205,'Manogna','Chennai')

1 row created.

SQL> select * from users;

   USER_ID VEHICLE_ID USER_NAME            CITY
---------- ---------- -------------------- --------------------
       111        201 Phani                Hyderabad
       112        202 Meghana              Bangalore
       113        203 Anjani               Hyderabad
       114        204 Ravi                 Vijaywada
       115        205 Manogna              Chennai

SQL>
```

Vehicle table:

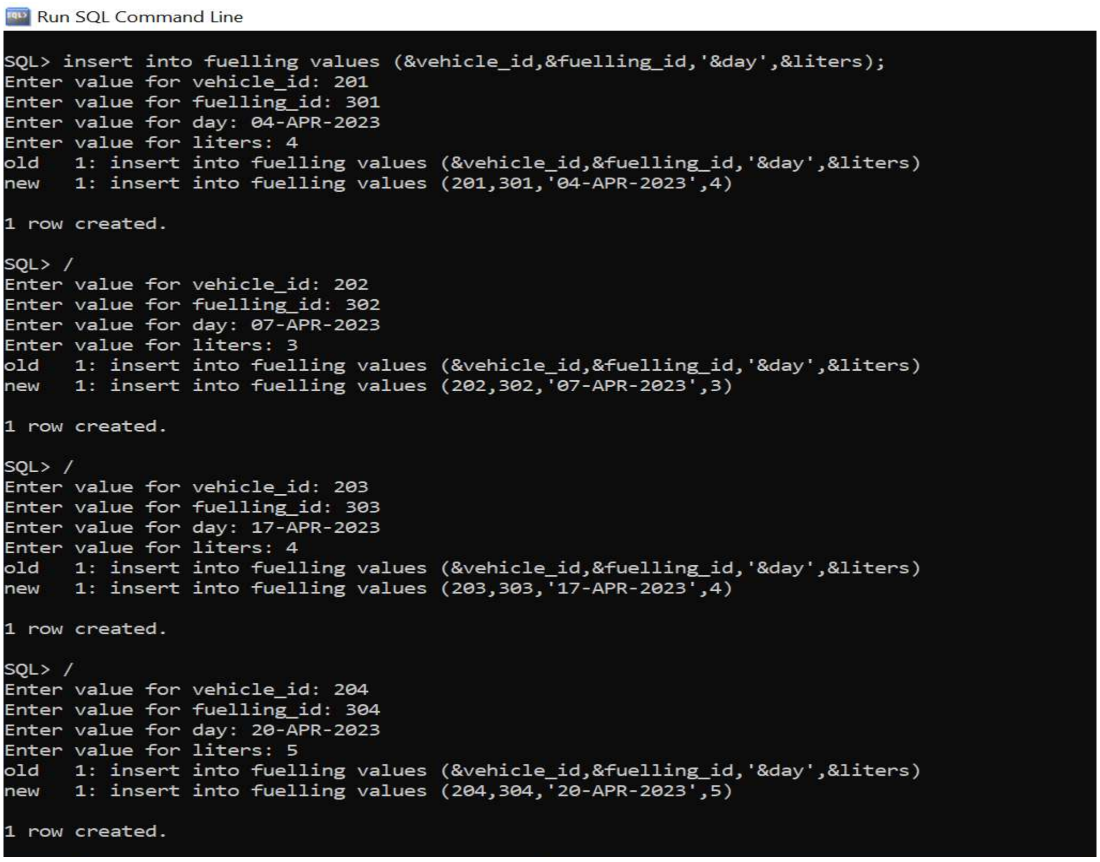


```
SQL> insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model');
Enter value for user_id: 111
Enter value for vehicle_id: 201
Enter value for year: 2019
Enter value for brand: Maruti Suzuki
Enter value for model: Alto
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
new   1: insert into vehicle values(111,201,2019,'Maruti Suzuki','Alto')

1 row created.

SQL> /
Enter value for user_id: 112
Enter value for vehicle_id: 202
Enter value for year: 2017
Enter value for brand: Tata
Enter value for model: Indica
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
new   1: insert into vehicle values(112,202,2017,'Tata','Indica')

1 row created.

SQL> /
Enter value for user_id: 113
Enter value for vehicle_id: 203
Enter value for year: 2020
Enter value for brand: Toyota
Enter value for model: Innova
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
new   1: insert into vehicle values(113,203,2020,'Toyota','Innova')

1 row created.

SQL> /
Enter value for user_id: 114
Enter value for vehicle_id: 204
Enter value for year: 2019
Enter value for brand: Maruti Suzuki
Enter value for model: Swift
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
```

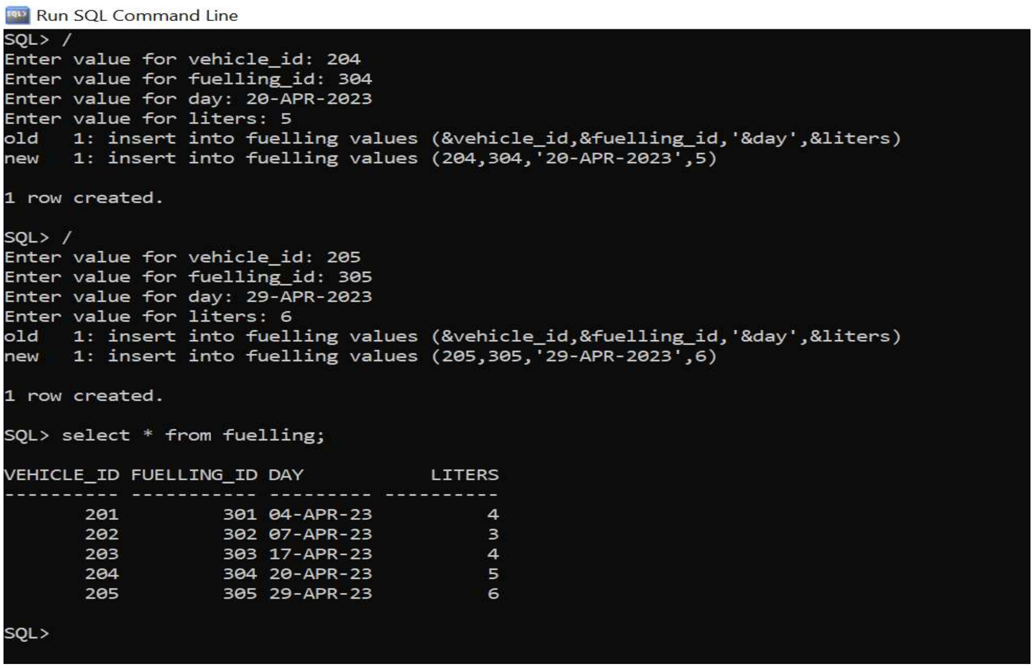


```
Enter value for model: Swift
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
new   1: insert into vehicle values(114,204,2019,'Maruti Suzuki ','Swift')

1 row created.

SQL> /
Enter value for user_id: 115
Enter value for vehicle_id: 205
Enter value for year: 2021
Enter value for brand: Toyota
Enter value for model: Fortuner
old   1: insert into vehicle values(&user_id,&vehicle_id,&year,'&brand','&model')
new   1: insert into vehicle values(115,205,2021,'Toyota','Fortuner')

1 row created.

SQL> select * from vehicle;

   USER_ID VEHICLE_ID       YEAR BRAND                MODEL
---------- ---------- ---------- -------------------- --------------------
       111        201       2019 Maruti Suzuki        Alto
       112        202       2017 Tata                 Indica
       113        203       2020 Toyota               Innova
       114        204       2019 Maruti Suzuki        Swift
       115        205       2021 Toyota               Fortuner

SQL>
```

Fuelling table:

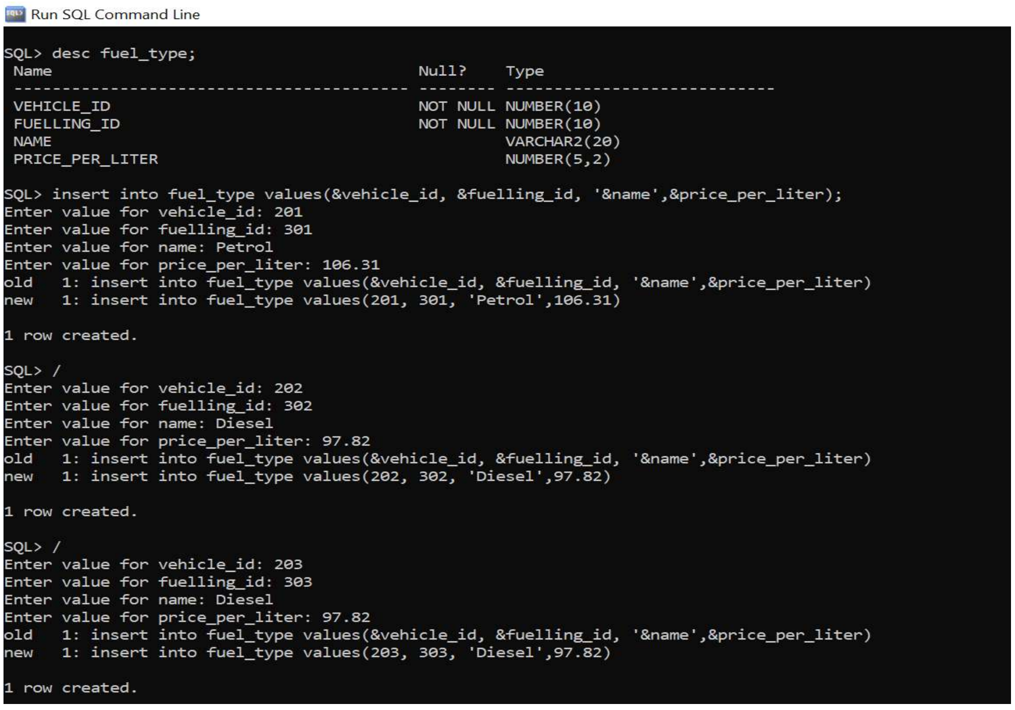

Run SQL Command Line

```
SQL> insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters);
Enter value for vehicle_id: 201
Enter value for fuelling_id: 301
Enter value for day: 04-APR-2023
Enter value for liters: 4
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (201,301,'04-APR-2023',4)

1 row created.

SQL> /
Enter value for vehicle_id: 202
Enter value for fuelling_id: 302
Enter value for day: 07-APR-2023
Enter value for liters: 3
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (202,302,'07-APR-2023',3)

1 row created.

SQL> /
Enter value for vehicle_id: 203
Enter value for fuelling_id: 303
Enter value for day: 17-APR-2023
Enter value for liters: 4
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (203,303,'17-APR-2023',4)

1 row created.

SQL> /
Enter value for vehicle_id: 204
Enter value for fuelling_id: 304
Enter value for day: 20-APR-2023
Enter value for liters: 5
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (204,304,'20-APR-2023',5)

1 row created.
```

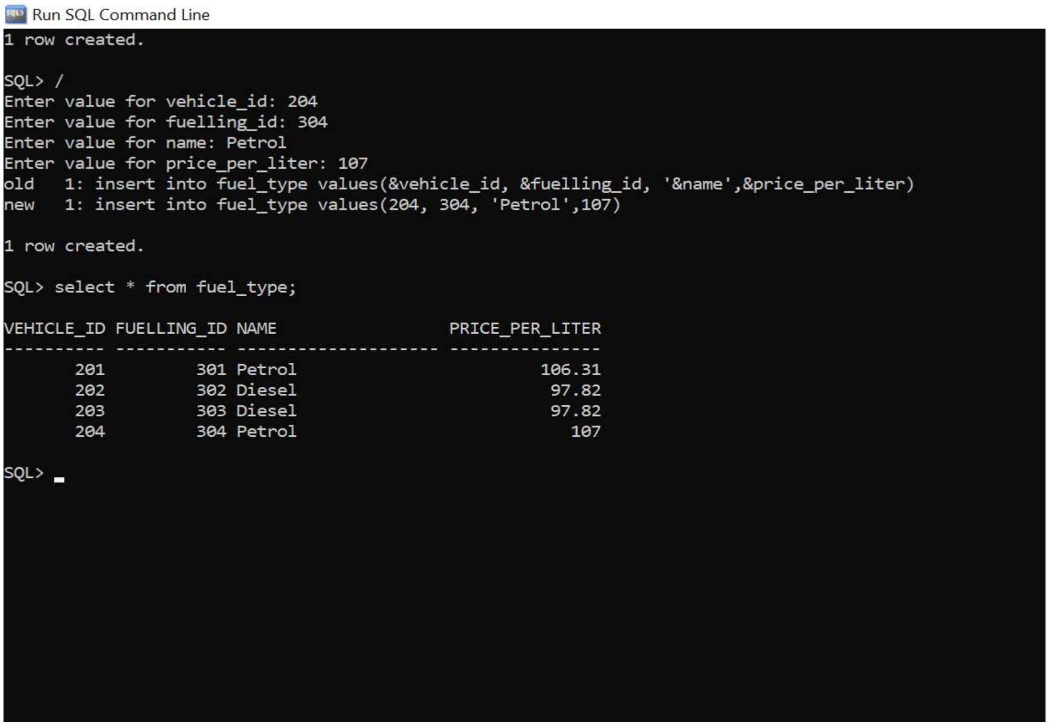

Run SQL Command Line

```
SQL> /
Enter value for vehicle_id: 204
Enter value for fuelling_id: 304
Enter value for day: 20-APR-2023
Enter value for liters: 5
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (204,304,'20-APR-2023',5)

1 row created.

SQL> /
Enter value for vehicle_id: 205
Enter value for fuelling_id: 305
Enter value for day: 29-APR-2023
Enter value for liters: 6
old   1: insert into fuelling values (&vehicle_id,&fuelling_id,'&day',&liters)
new   1: insert into fuelling values (205,305,'29-APR-2023',6)

1 row created.

SQL> select * from fuelling;

VEHICLE_ID FUELLING_ID DAY           LITERS
---------- ----------- --------- ----------
       201         301 04-APR-23          4
       202         302 07-APR-23          3
       203         303 17-APR-23          4
       204         304 20-APR-23          5
       205         305 29-APR-23          6

SQL>
```

Fuel_type table:



```
SQL> desc fuel_type;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 VEHICLE_ID                                NOT NULL NUMBER(10)
 FUELLING_ID                               NOT NULL NUMBER(10)
 NAME                                               VARCHAR2(20)
 PRICE_PER_LITER                                    NUMBER(5,2)

SQL> insert into fuel_type values(&vehicle_id, &fuelling_id, '&name',&price_per_liter);
Enter value for vehicle_id: 201
Enter value for fuelling_id: 301
Enter value for name: Petrol
Enter value for price_per_liter: 106.31
old   1: insert into fuel_type values(&vehicle_id, &fuelling_id, '&name',&price_per_liter)
new   1: insert into fuel_type values(201, 301, 'Petrol',106.31)

1 row created.

SQL> /
Enter value for vehicle_id: 202
Enter value for fuelling_id: 302
Enter value for name: Diesel
Enter value for price_per_liter: 97.82
old   1: insert into fuel_type values(&vehicle_id, &fuelling_id, '&name',&price_per_liter)
new   1: insert into fuel_type values(202, 302, 'Diesel',97.82)

1 row created.

SQL> /
Enter value for vehicle_id: 203
Enter value for fuelling_id: 303
Enter value for name: Diesel
Enter value for price_per_liter: 97.82
old   1: insert into fuel_type values(&vehicle_id, &fuelling_id, '&name',&price_per_liter)
new   1: insert into fuel_type values(203, 303, 'Diesel',97.82)

1 row created.
```



```
1 row created.

SQL> /
Enter value for vehicle_id: 204
Enter value for fuelling_id: 304
Enter value for name: Petrol
Enter value for price_per_liter: 107
old   1: insert into fuel_type values(&vehicle_id, &fuelling_id, '&name',&price_per_liter)
new   1: insert into fuel_type values(204, 304, 'Petrol',107)

1 row created.

SQL> select * from fuel_type;

VEHICLE_ID FUELLING_ID NAME                 PRICE_PER_LITER
---------- ----------- -------------------- ---------------
       201         301 Petrol                        106.31
       202         302 Diesel                         97.82
       203         303 Diesel                         97.82
       204         304 Petrol                           107

SQL> _
```

IMPLEMENTATION

JAVA-SQL Connectivity using JDBC:

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database and is oriented towards relational databases.

The connection to the database can be performed using Java programming (JDBC API) as:

```
public DBAccess(){
            try{
                  Class.forName("oracle.jdbc.OracleDriver");
      con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","phani","phani");
                  con.setAutoCommit(false);
                  stmt=con.createStatement();
                  savePoint = con.setSavepoint("lastSave");
            }
            catch(ClassNotFoundException ex){
                  System.out.println(ex);
            }
            catch(SQLException ex){
                  System.out.println(ex);
            }
      }
```

Front End Programs for user interface:

1. **DBAccess.java**

```
//package vehicledb;
import java.sql.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import javax.swing.table.DefaultTableModel;

public class DBAccess{
        String uid;
        static Connection con;
        Statement stmt;
        ResultSet rs,rsUpdate;
        Savepoint savePoint,savePoint1;
        int flag;
        public DBAccess(){

                try{
                        //String classpath = "C:\\path\\to\\ojdbc8.jar";
        //System.setProperty("java.class.path", classpath);
                        Class.forName("oracle.jdbc.OracleDriver");


con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","phani","phani");
                        con.setAutoCommit(false);
                        stmt=con.createStatement();
                        savePoint = con.setSavepoint("lastSave");
                }
                catch(ClassNotFoundException ex){
                        System.out.println(ex);
                }
                catch(SQLException ex){
                        System.out.println(ex);
                }
        }

        //TO CHECK IF USER EXISTS
        public boolean checkUserExistence(String userId) {
      try {
        String query = "SELECT * FROM Users WHERE user_id = ?";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, userId);
        ResultSet resultSet = preparedStatement.executeQuery();
        return resultSet.next();
      } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error checking user existence: " +
e.getMessage(), "Error",
              JOptionPane.ERROR_MESSAGE);
        return false;
      }
        }
        // Function to check if a vehicle exists in the Vehicle table
        public boolean checkVehicleExistence(String vehicleId) {
                try {
                        String query = "SELECT COUNT(*) AS count FROM vehicle WHERE
vehicle_id = ?";
```

```
                PreparedStatement pstmt = con.prepareStatement(query);
                pstmt.setString(1, vehicleId);
                ResultSet rs = pstmt.executeQuery();
                rs.next();
                int count = rs.getInt("count");
                rs.close();
                pstmt.close();
                return count > 0;
            }
            catch (SQLException e) {
                System.out.println("Error checking vehicle existence: " + e.getMessage());
                return false;
            }
        }


        // INSERT INTO USERS TABLE
  public void insertData(String user_name, String user_id, String vehicle_id, String city) {
      try {
        String query = "INSERT INTO users (user_id,vehicle_id,user_name,city) VALUES
(?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, user_id);
        pstmt.setString(2, vehicle_id);
        pstmt.setString(3, user_name);
        pstmt.setString(4, city);
        pstmt.executeUpdate();
        pstmt.close();
        System.out.println("Data inserted successfully!");
      } catch (SQLException e) {
        System.out.println("Error inserting data: " + e.getMessage());
      }
   }

        //DELETE FROM USERS TABLE
        public void deleteData(String user_id) {
      try {
        String query = "DELETE FROM users WHERE user_id = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, user_id);
        int rowsAffected = pstmt.executeUpdate();
        pstmt.close();

        if (rowsAffected > 0) {
          System.out.println("Data deleted successfully!");
        } else {
          System.out.println("No data found with the given user ID.");
        }
      } catch (SQLException e) {
        System.out.println("Error deleting data: " + e.getMessage());
      }
   }

        //UPDATE IN USERS TABLE
        public void updateData(String user_name, String user_id, String vehicle_id, String city) {
      try {
        String query = "UPDATE users SET vehicle_id = ?, user_name = ?, city = ? WHERE
user_id = ?";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, vehicle_id);
        pstmt.setString(2, user_name);
        pstmt.setString(3, city);
        pstmt.setString(4, user_id);
        int rowsAffected = pstmt.executeUpdate();
        pstmt.close();

        if (rowsAffected > 0) {
          System.out.println("Data updated successfully!");
        } else {
          System.out.println("No data found with the given user ID.");
        }
      } catch (SQLException e) {
        System.out.println("Error updating data: " + e.getMessage());
      }
   }

        //VIEW USERS TABLE
        public void viewData() {
   try {
      String query = "SELECT user_name, vehicle_id, city FROM users";
      PreparedStatement pstmt = con.prepareStatement(query);
      ResultSet rs = pstmt.executeQuery();

      // Create a table model to hold the data
      DefaultTableModel tableModel = new DefaultTableModel();
      tableModel.addColumn("User Name");
      tableModel.addColumn("Vehicle ID");
      tableModel.addColumn("City");

      while (rs.next()) {
        String userName = rs.getString("user_name");
        String vehicleId = rs.getString("vehicle_id");
        String city = rs.getString("city");

        tableModel.addRow(new Object[] { userName, vehicleId, city });
      }

      rs.close();
      pstmt.close();
      JTable dataTable = new JTable(tableModel);


      JScrollPane scrollPane = new JScrollPane(dataTable);


      JFrame frame = new JFrame("View Data");
      frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      frame.add(scrollPane);
      frame.pack();
      frame.setVisible(true);
   }
        catch (SQLException e)
        {
      System.out.println("Error viewing data: " + e.getMessage());
   }
        }

        // INSERT INTO VEHICLES TABLE
        public void insertVehicleData(String userId, String vehicleId, String year, String brand,
String model) {
      try {
        String query = "INSERT INTO Vehicle (user_id, vehicle_id, year, brand, model)
VALUES (?, ?, ?, ?, ?)";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, userId);
        preparedStatement.setString(2, vehicleId);
        preparedStatement.setString(3, year);
        preparedStatement.setString(4, brand);
        preparedStatement.setString(5, model);
        preparedStatement.executeUpdate();
        JOptionPane.showMessageDialog(null, "Vehicle data inserted successfully.", "Success",
              JOptionPane.INFORMATION_MESSAGE);
      } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, "Error inserting vehicle data: " + e.getMessage(),
"Error",
              JOptionPane.ERROR_MESSAGE);
      }
   }

   // UPDATE VEHICLES TABLE
   public void updateVehicleData(String userId, String vehicleId, String year, String brand, String
model) {
      try {
        String query = "UPDATE Vehicle SET year = ?, brand = ?, model = ? WHERE user_id
= ? AND vehicle_id = ?";
        PreparedStatement preparedStatement = con.prepareStatement(query);
        preparedStatement.setString(1, year);
        preparedStatement.setString(2, brand);
```

```
            preparedStatement.setString(3, model);
            preparedStatement.setString(4, userId);
            preparedStatement.setString(5, vehicleId);
            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(null, "Vehicle data updated successfully.", "Success",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "No records found for the specified user ID and
vehicle ID.",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error updating vehicle data: " + e.getMessage(),
"Error",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    // DELETE FROM VEHICLES TABLE
    public void deleteVehicleData(String userId, String vehicleId) {
        try {
            String query = "DELETE FROM Vehicle WHERE user_id = ? AND vehicle_id = ?";
            PreparedStatement preparedStatement = con.prepareStatement(query);
            preparedStatement.setString(1, userId);
            preparedStatement.setString(2, vehicleId);
            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                JOptionPane.showMessageDialog(null, "Vehicle data deleted successfully.", "Success",
                        JOptionPane.INFORMATION_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(null, "No records found for the specified user ID and
vehicle ID.",
                        "Error", JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null, "Error deleting vehicle data: " + e.getMessage(),
"Error",
                    JOptionPane.ERROR_MESSAGE);
        }
    }

    public void viewVehicleData() {
    try {
        String query = "SELECT * from vehicle";
        PreparedStatement pstmt = con.prepareStatement(query);
        ResultSet rs = pstmt.executeQuery();

        // Create a table model to hold the data
        DefaultTableModel tableModel = new DefaultTableModel();
        tableModel.addColumn("User ID");
        tableModel.addColumn("Vehicle ID");
        tableModel.addColumn("Year");
                    tableModel.addColumn("Brand");
                    tableModel.addColumn("Model");


        while (rs.next()) {
            String userID = rs.getString("user_id");
            String vehicleId = rs.getString("vehicle_id");
            int year = rs.getInt("year");
                        String brand = rs.getString("brand");
                        String model = rs.getString("model");

            tableModel.addRow(new Object[] { userID, vehicleId, year, brand, model });
        }

        rs.close();
        pstmt.close();

        // Create a JTable with the table model
        JTable dataTable = new JTable(tableModel);

        // Create a scroll pane to contain the table
        JScrollPane scrollPane = new JScrollPane(dataTable);

        // Create a new window to display the table
        JFrame frame = new JFrame("View Data");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.add(scrollPane);
        frame.pack();
        frame.setVisible(true);
    }
            catch (SQLException e)
            {
        System.out.println("Error viewing data: " + e.getMessage());
    }
            }
            // INSERT INTO FUEL_TYPE TABLE
            public void insertFuelTypeData(String vehicleId, String fuellingId, String name, String
pricePerLiter) {
    try {
        // Check if the vehicle ID exists in the Vehicle table
        if (!checkVehicleExistence(vehicleId)) {
            System.out.println("Vehicle ID does not exist. Cannot add Fuel_type data.");
            return;
        }

        String query = "INSERT INTO fuel_type (vehicle_id, fuelling_id, name, price_per_liter)
VALUES (?, ?, ?, ?)";
```

```
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, vehicleId);
        pstmt.setString(2, fuellingId);
        pstmt.setString(3, name);
        pstmt.setString(4, pricePerLiter);
        pstmt.executeUpdate();
        pstmt.close();
        System.out.println("Data inserted successfully into Fuel_type table.");
    } catch (SQLException e) {
        System.out.println("Error inserting data into Fuel_type table: " + e.getMessage());
    }
}

    // Function to delete data from the Fuel_type table
    public void deleteFuelTypeData(String vehicleId, String fuellingId) {
        try {
            String query = "DELETE FROM fuel_type WHERE vehicle_id = ? AND fuelling_id = ?";
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setString(1, vehicleId);
            pstmt.setString(2, fuellingId);
            int rowsAffected = pstmt.executeUpdate();
            pstmt.close();
            System.out.println(rowsAffected + " row(s) deleted from Fuel_type table.");
        } catch (SQLException e) {
            System.out.println("Error deleting data from Fuel_type table: " + e.getMessage());
        }
    }

    // Function to update data in the Fuel_type table
    public void updateFuelTypeData(String vehicleId, String fuellingId, String name, String
pricePerLiter) {
        try {
            String query = "UPDATE fuel_type SET name = ?, price_per_liter = ? WHERE vehicle_id
= ? AND fuelling_id = ?";
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setString(1, name);
            pstmt.setString(2, pricePerLiter);
            pstmt.setString(3, vehicleId);
            pstmt.setString(4, fuellingId);
            int rowsAffected = pstmt.executeUpdate();
            pstmt.close();
            System.out.println(rowsAffected + " row(s) updated in Fuel_type table.");
        } catch (SQLException e) {
            System.out.println("Error updating data in Fuel_type table: " + e.getMessage());
        }
    }

    // Function to view data from the Fuel_type table
    public void viewFuelTypeData() {
        try {
            String query = "SELECT * FROM fuel_type";
            PreparedStatement pstmt = con.prepareStatement(query);
            ResultSet rs = pstmt.executeQuery();

            DefaultTableModel tableModel = new DefaultTableModel();
            tableModel.addColumn("Vehicle ID");
            tableModel.addColumn("Fuelling ID");
            tableModel.addColumn("Name");
            tableModel.addColumn("Price per Liter");

            while (rs.next()) {
                String vehicleId = rs.getString("vehicle_id");
                String fuellingId = rs.getString("fuelling_id");
                String name = rs.getString("name");
                String pricePerLiter = rs.getString("price_per_liter");
                tableModel.addRow(new Object[]{vehicleId, fuellingId, name, pricePerLiter});
            }

            rs.close();
            pstmt.close();

            JTable dataTable = new JTable(tableModel);
            JScrollPane scrollPane = new JScrollPane(dataTable);

            JFrame frame = new JFrame("View Fuel Type Data");
            frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            frame.add(scrollPane);
            frame.pack();
            frame.setVisible(true);
        } catch (SQLException e) {
            System.out.println("Error viewing Fuel_type data: " + e.getMessage());
        }
    }

    // Function to insert data into the Fuelling table
    public void insertFuellingData(String vehicleId, String fuellingId, String day, String liters) {
    try {
        // Check if the vehicle ID exists in the Vehicle table
        if (!checkVehicleExistence(vehicleId)) {
            System.out.println("Vehicle ID does not exist. Cannot add Fuelling data.");
            return;
        }

        String query = "INSERT INTO fuelling (vehicle_id, fuelling_id, day, liters) VALUES
(?, ?, ?, ?)";
        PreparedStatement pstmt = con.prepareStatement(query);
        pstmt.setString(1, vehicleId);
        pstmt.setString(2, fuellingId);
        pstmt.setString(3, day);
        pstmt.setString(4, liters);
        pstmt.executeUpdate();
        pstmt.close();
        System.out.println("Data inserted successfully into Fuelling table.");
    } catch (SQLException e) {
        System.out.println("Error inserting data into Fuelling table: " + e.getMessage());
    }
}

    // Function to delete data from the Fuelling table
    public void deleteFuellingData(String vehicleId, String fuellingId) {
        try {
            String query = "DELETE FROM fuelling WHERE vehicle_id = ? AND fuelling_id = ?";
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setString(1, vehicleId);
            pstmt.setString(2, fuellingId);
            int rowsAffected = pstmt.executeUpdate();
            pstmt.close();
            System.out.println(rowsAffected + " row(s) deleted from Fuelling table.");
        } catch (SQLException e) {
            System.out.println("Error deleting data from Fuelling table: " + e.getMessage());
        }
    }

    // Function to update data in the Fuelling table
    public void updateFuellingData(String vehicleId, String fuellingId, String day, String liters) {
        try {
            String query = "UPDATE fuelling SET day = ?, liters = ? WHERE vehicle_id = ? AND
fuelling_id = ?";
            PreparedStatement pstmt = con.prepareStatement(query);
            pstmt.setString(1, day);
            pstmt.setString(2, liters);
            pstmt.setString(3, vehicleId);
            pstmt.setString(4, fuellingId);
            int rowsAffected = pstmt.executeUpdate();
            pstmt.close();
            System.out.println(rowsAffected + " row(s) updated in Fuelling table.");
        } catch (SQLException e) {
            System.out.println("Error updating data in Fuelling table: " + e.getMessage());
        }
    }

    // Function to view data from the Fuelling table
    public void viewFuellingData() {
        try {
            String query = "SELECT * FROM fuelling";
            PreparedStatement pstmt = con.prepareStatement(query);
            ResultSet rs = pstmt.executeQuery();

            DefaultTableModel tableModel = new DefaultTableModel();
            tableModel.addColumn("Vehicle ID");
            tableModel.addColumn("Fuelling ID");
            tableModel.addColumn("Day");
            tableModel.addColumn("Liters");

            while (rs.next()) {
                String vehicleId = rs.getString("vehicle_id");
                String fuellingId = rs.getString("fuelling_id");
                String day = rs.getString("day");
                String liters = rs.getString("liters");
                tableModel.addRow(new Object[]{vehicleId, fuellingId, day, liters});
            }

            rs.close();
            pstmt.close( );

            JTable dataTable = new JTable(tableModel);
            JScrollPane scrollPane = new JScrollPane(dataTable);

            JFrame frame = new JFrame("View Fuelling Data");
            frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
            frame.add(scrollPane);
            frame.pack();
            frame.setVisible(true);
        } catch (SQLException e) {
            System.out.println("Error viewing Fuelling data: " + e.getMessage());
        }
    }
            public boolean closeConnection(){
                    try{
                            con.commit();
                            if(!con.isClosed())
                                    con.close();
                            return true;
                    }
                    catch(SQLException e){
                            System.out.println(e);
                            return false;
                    }
            }

}
```

2. MAINPAGE

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.regex.*;
public class Mainpage{
	JFrame frame;
	JLabel lbluname,lblvecid,lblcity,lbluid,lbltitle,lbloption;
	JButton btnadd,btndelete,btnupdate,btnvehicle;
	JTextField tfuname, tfuid, tfvecid;
	JMenuItem miView;
    JMenu actions;
	JComboBox cityList;
	String[] cities = {"Hyderabad", "Bangalore", "Chennai",
"Vijaywada","Kolkata","Mumbai","Delhi","Ahmedabad"};
	JMenuBar menubar;
	JMenu about,lkfl;
	JMenuItem miabtproject,miabtstudent,miMotif,miNimbus,miCross;
	JPanel p1,p2,p3,p4,p5,p6,p7,p8;
	DBAccess db;
	public Mainpage(){
		super();
	}
	public Mainpage(DBAccess db){
		this.db = db;
		initializeComponents();
		registerListeners();
		addComponentsToFrame();
		frame.setBackground(Color.lightGray);
		frame.setLayout(new GridLayout(8,1));
		frame.setSize(500,500);
		frame.setVisible(true);
		frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
	}
	public void initializeComponents(){
		frame = new JFrame("USER DATA");
		menubar = new JMenuBar();

		about = new JMenu("About");
		miabtproject = new JMenuItem("About Project");
		miabtstudent = new JMenuItem("About Student");
		lkfl = new JMenu("LAF");
		miMotif = new JMenuItem("Motif");
		miNimbus = new JMenuItem("Nimbus");
		miCross = new JMenuItem("Cross Platform");
		miView = new JMenuItem("View");
      actions = new JMenu("Actions");
		lbltitle = new JLabel("AUTOMATIC FUEL TRACKER IN VEHICLES");
		lbluname = new JLabel("USER NAME: ");
		tfuname = new JTextField(12);
		lbluid = new JLabel("USER ID: ");
		tfuid= new JTextField(4);
		lblcity = new JLabel("Select city: ");
		cityList = new JComboBox(cities);
		cityList.setSelectedIndex(1);
		lblvecid = new JLabel("Vehicle ID:");
```

```
                    tfvecid = new JTextField(15);
                    btnadd = new JButton("SUBMIT");
                    btndelete = new JButton("Delete");
        btnupdate = new JButton("Update");
                    btnvehicle = new JButton("VEHICLES");
                    lbloption = new JLabel("Click here for vehicle data");
                    p1 = new JPanel();
                    p2 = new JPanel();
                    p3 = new JPanel();
                    p4 = new JPanel();
                    p5 = new JPanel();
                    p6 = new JPanel();
                    p7 = new JPanel();
                    p8 = new JPanel();
          }
          public void registerListeners(){
          miabtproject.addActionListener (new ActionListener () {
                         public void actionPerformed(ActionEvent e) {
                              JOptionPane.showMessageDialog(frame, "Project: AUTOMATIC
FUEL TRACKER IN VEHICLES","INFORMATION",
JOptionPane.INFORMATION_MESSAGE);
                         }
                    });
                    miabtstudent.addActionListener (new ActionListener () {
                         public void actionPerformed(ActionEvent e) {
                              JOptionPane.showMessageDialog(frame, "Name of Student: Phani
Meghana\nRoll number: 1602-21-737-028","INFORMATION",
JOptionPane.INFORMATION_MESSAGE);
                         }
                    });
                    miMotif.addActionListener(new ActionListener()
                    {
                         public void actionPerformed(ActionEvent evt)
                         {
                              try{

          UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
                                   SwingUtilities.updateComponentTreeUI(frame);
                              }
                              catch(ClassNotFoundException|InstantiationException|
IllegalAccessException|UnsupportedLookAndFeelException ex)
                              {}
                         }
                    });
                    miNimbus.addActionListener(new ActionListener()
                    {
                         public void actionPerformed(ActionEvent evt)
                         {
                              try{
```

```
		UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
						SwingUtilities.updateComponentTreeUI(frame);
					}
					catch(ClassNotFoundException|InstantiationException|
IllegalAccessException|UnsupportedLookAndFeelException ex)
					{}
				}
			});

			miCross.addActionListener(new ActionListener()
			{
				public void actionPerformed(ActionEvent evt)
				{
					try{

		UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
						SwingUtilities.updateComponentTreeUI(frame);
					}
					catch(ClassNotFoundException|InstantiationException|
IllegalAccessException|UnsupportedLookAndFeelException ex)
					{}
				}
			});
			miView.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
		  db.viewData();
		}
	  });
			frame.addWindowListener(new WindowAdapter() {
				public void windowClosing(WindowEvent e) {
					if(db.closeConnection()){
						JOptionPane.showMessageDialog(frame, "Connection
closing.\n Exit?","INFORMATION", JOptionPane.INFORMATION_MESSAGE);;
					}
					System.exit(0);
				}
			});
			btnadd.addActionListener(new ActionListener() {
	  public void actionPerformed(ActionEvent e) {
		String userName = tfuname.getText();
		String userID = tfuid.getText();
		String vehicleID = tfvecid.getText();
		String city = (String) cityList.getSelectedItem();

		// Call the insertData method in the DBAccess class
		db.insertData(userName, userID, vehicleID, city);
	  }});
			btndelete.addActionListener(new ActionListener() {
		public void actionPerformed(ActionEvent e) {
```

```
            String userID = tfuid.getText();
            db.deleteData(userID);
         }
      });
               btnupdate.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String userName = tfuname.getText();
            String userID = tfuid.getText();
            String vehicleID = tfvecid.getText();
            String city = (String) cityList.getSelectedItem();
                              db.updateData(userName, userID, vehicleID, city);
         }
      });
               btnvehicle.addActionListener(new ActionListener() {
                       public void actionPerformed(ActionEvent e) {
                              String userId = tfuid.getText();
                              if (db.checkUserExistence(userId)) {
                                     new VehicleData(db);
                              }
                              else {
                                     JOptionPane.showMessageDialog(frame, "User does not
exist. Submit user data first!", "Error", JOptionPane.ERROR_MESSAGE);
                              }
                       }
               });




         }
         public void addComponentsToFrame(){
               about.add(miabtproject);
               about.add(miabtstudent);
               lkfl.add(miCross);
               lkfl.add(miMotif);
               lkfl.add(miNimbus);
               menubar.add(about);
               menubar.add(lkfl);
               actions.add(miView);
               menubar.add(actions);
               frame.setJMenuBar(menubar);
               p1.add(lbltitle);
               p2.add(lbluname);
               p2.add(tfuname);
               p3.add(lbluid);
               p3.add(tfuid);
               p4.add(lblvecid);
               p4.add(tfvecid);
               p5.add(lblcity);
               p5.add(cityList);
```

```
                p6.add(btnadd);
                p6.add(btndelete);
                p6.add(btnupdate);
                p7.add(lbloption);
                p8.add(btnvehicle);

                frame.add(p1);
                frame.add(p2);
                frame.add(p3);
                frame.add(p4);
                frame.add(p5);
                frame.add(p6);
                frame.add(p7);
                frame.add(p8);

        }
        public static void main(String[] args){
                DBAccess db = new DBAccess();
                //String classpath = "." + System.getProperty("path.separator") + "ojdbc8.jar";
    //System.setProperty("java.class.path", classpath);
                new Mainpage(db);
        }
}
```

3. Vehicle Data Table

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

public class VehicleData {
    private JFrame frame;
    private JTextField tfUserId, tfVehicleId, tfYear, tfBrand, tfModel;
    private JButton btnSubmit, btnUpdate, btnDelete, btnView;
    private DBAccess db;
        private JMenu actions;
        private JMenuBar menubar;
        private JMenuItem mifuelling,mifueltype;

    public VehicleData(DBAccess db) {
        this.db = db;
        initializeComponents();
        registerListeners();
        addComponentsToFrame();
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setVisible(true);
    }
```

```
private void initializeComponents() {
            frame = new JFrame("Vehicle Data");
            menubar = new JMenuBar();
            actions = new JMenu("Actions");
            //mifuelling = new JMenu("Fuelling");
            //mifueltype = new JMenu("Fuel type");
  mifuelling = new JMenuItem("Fuelling");
            mifueltype = new JMenuItem("Fuel type");
  JLabel lblUserId = new JLabel("User ID:");
  tfUserId = new JTextField(10);
  JLabel lblVehicleId = new JLabel("Vehicle ID:");
  tfVehicleId = new JTextField(10);
  JLabel lblYear = new JLabel("Year:");
  tfYear = new JTextField(5);
  JLabel lblBrand = new JLabel("Brand:");
  tfBrand = new JTextField(20);
  JLabel lblModel = new JLabel("Model:");
  tfModel = new JTextField(20);
  btnSubmit = new JButton("Submit");
  btnUpdate = new JButton("Update");
  btnDelete = new JButton("Delete");
  btnView = new JButton("View");
}

private void registerListeners() {
  btnSubmit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String userId = tfUserId.getText();
      String vehicleId = tfVehicleId.getText();
      String year = tfYear.getText();
      String brand = tfBrand.getText();
      String model = tfModel.getText();
      db.insertVehicleData(userId, vehicleId, year, brand, model);
    }
  });

  btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String userId = tfUserId.getText();
      String vehicleId = tfVehicleId.getText();
      String year = tfYear.getText();
      String brand = tfBrand.getText();
      String model = tfModel.getText();
      db.updateVehicleData(userId, vehicleId, year, brand, model);
    }
  });

  btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String userId = tfUserId.getText();
```

```
            String vehicleId = tfVehicleId.getText();
            db.deleteVehicleData(userId, vehicleId);
        }
    });

    btnView.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            //String userId = tfUserId.getText();
            //String vehicleId = tfVehicleId.getText();
            db.viewVehicleData();
        }
    });
                /*mifueltype.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                FuelType fueltypewindow = new FuelType(db);
                                fueltypewindow.setVisible(true);

                        }
                });

                mifuelling.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                Fuelling fuellingwindow = new Fuelling(db);
                                fuellingwindow.setVisible(true);
                        }
                });*/
                mifuelling.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Fuelling fuellingWindow = new Fuelling(db);
            fuellingWindow.setVisible(true);
        }
    });

    mifueltype.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            FuelType fuelTypeWindow = new FuelType(db);
            fuelTypeWindow.setVisible(true);
        }
    });
}

private void addComponentsToFrame() {
    JPanel panel = new JPanel();
                frame.setJMenuBar(menubar);
                menubar.add(actions);
                actions.add(mifuelling);
                actions.add(mifueltype);
                //menubar.add(mifuelling);
                //menubar.add(mifueltype);
    panel.setLayout(new GridLayout(7, 2));
```

```
        panel.add(new JLabel("User ID:"));
        panel.add(tfUserId);
        panel.add(new JLabel("Vehicle ID:"));
        panel.add(tfVehicleId);
        panel.add(new JLabel("Year:"));
        panel.add(tfYear);
        panel.add(new JLabel("Brand:"));
        panel.add(tfBrand);
        panel.add(new JLabel("Model:"));
        panel.add(tfModel);
        panel.add(btnSubmit);
        panel.add(btnUpdate);
        panel.add(btnDelete);
        panel.add(btnView);

        frame.add(panel);
    }
}
```

4. Fuelling table

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

public class Fuelling extends JFrame {
    private JFrame frame;
    private JTextField tfVehicleId, tfFuellingId, tfDay, tfLiters;
    private JButton btnSubmit, btnUpdate, btnDelete, btnView;
    private DBAccess db;

    public Fuelling(DBAccess db) {
        this.db = db;
        initializeComponents();
        registerListeners();
        addComponentsToFrame();
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setVisible(true);
    }

    private void initializeComponents() {
        frame = new JFrame("Fuelling");
        JLabel lblVehicleId = new JLabel("Vehicle ID:");
        tfVehicleId = new JTextField(10);
        JLabel lblFuellingId = new JLabel("Fuelling ID:");
        tfFuellingId = new JTextField(10);
        JLabel lblDay = new JLabel("Day:");
        tfDay = new JTextField(10);
        JLabel lblLiters = new JLabel("Liters:");
        tfLiters = new JTextField(10);
        btnSubmit = new JButton("Submit");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnView = new JButton("View");
    }

    private void registerListeners() {
        btnSubmit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                String day = tfDay.getText();
                String liters = tfLiters.getText();
                db.insertFuellingData(vehicleId, fuellingId, day, liters);
            }
        });

        btnUpdate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                String day = tfDay.getText();
                String liters = tfLiters.getText();
                db.updateFuellingData(vehicleId, fuellingId, day, liters);
            }
        });

        btnDelete.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                db.deleteFuellingData(vehicleId, fuellingId);
            }
        });

        btnView.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                db.viewFuellingData();
            }
        });
    }

    private void addComponentsToFrame() {
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2));
        panel.add(new JLabel("Vehicle ID:"));
        panel.add(tfVehicleId);
        panel.add(new JLabel("Fuelling ID:"));
        panel.add(tfFuellingId);
        panel.add(new JLabel("Date:"));
        panel.add(tfDay);
        panel.add(new JLabel("Liters:"));
        panel.add(tfLiters);
        panel.add(btnSubmit);
        panel.add(btnUpdate);
        panel.add(btnDelete);
        panel.add(btnView);

        frame.add(panel);
    }
}
```

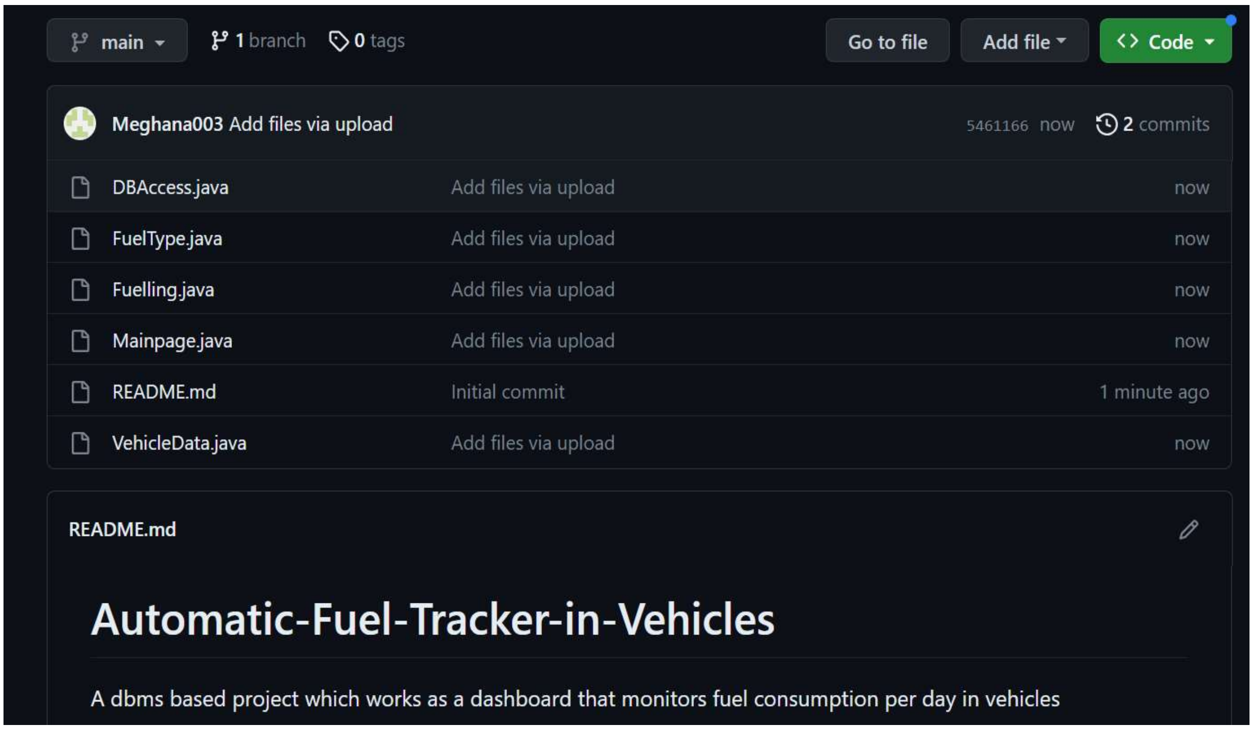
5. FuelType Table

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

public class FuelType extends JFrame {
    private JFrame frame;
    private JTextField tfVehicleId, tfFuellingId, tfName, tfPricePerLiter;
    private JButton btnSubmit, btnUpdate, btnDelete, btnView;
    private DBAccess db;

    public FuelType(DBAccess db) {
        this.db = db;
        initializeComponents();
        registerListeners();
        addComponentsToFrame();
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        frame.setVisible(true);
    }

    private void initializeComponents()
{
        frame = new JFrame("Fuel Type");
        JLabel lblVehicleId = new JLabel("Vehicle ID:");
        tfVehicleId = new JTextField(10);
        JLabel lblFuellingId = new JLabel("Fuelling ID:");
        tfFuellingId = new JTextField(10);
        JLabel lblName = new JLabel("Name:");
        tfName = new JTextField(20);
        JLabel lblPricePerLiter = new JLabel("Price per Liter:");
        tfPricePerLiter = new JTextField(10);
        btnSubmit = new JButton("Submit");
        btnUpdate = new JButton("Update");
        btnDelete = new JButton("Delete");
        btnView = new JButton("View");
    }

    private void registerListeners() {
        btnSubmit.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                String name = tfName.getText();
                String pricePerLiter = tfPricePerLiter.getText();
                db.insertFuelTypeData(vehicleId, fuellingId, name, pricePerLiter);
            }
        });

        btnUpdate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                String name = tfName.getText();
                String pricePerLiter = tfPricePerLiter.getText();
                db.updateFuelTypeData(vehicleId, fuellingId, name, pricePerLiter);
            }
        });

        btnDelete.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String vehicleId = tfVehicleId.getText();
                String fuellingId = tfFuellingId.getText();
                db.deleteFuelTypeData(vehicleId, fuellingId);
            }
        });

        btnView.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e)
  {
                db.viewFuelTypeData();
            }
        });
    }

    private void addComponentsToFrame()
{

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2));
        panel.add(new JLabel("Vehicle ID:"));
        panel.add(tfVehicleId);
        panel.add(new JLabel("Fuelling ID:"));
        panel.add(tfFuellingId);
        panel.add(new JLabel("Name:"));
        panel.add(tfName);
        panel.add(new JLabel("Price per Liter:"));
        panel.add(tfPricePerLiter);
        panel.add(btnSubmit);
        panel.add(btnUpdate);
        panel.add(btnDelete);
        panel.add(btnView);

        frame.add(panel);
    }
}
```

GitHub link and Folder Structure

Link: https://github.com/Meghana003/Automatic-Fuel-Tracker-in-Vehicles

Folder structure:



Meghana003 Add files via upload 5461166 now 2 commits

DBAccess.java	Add files via upload	now
FuelType.java	Add files via upload	now
Fuelling.java	Add files via upload	now
Mainpage.java	Add files via upload	now
README.md	Initial commit	1 minute ago
VehicleData.java	Add files via upload	now

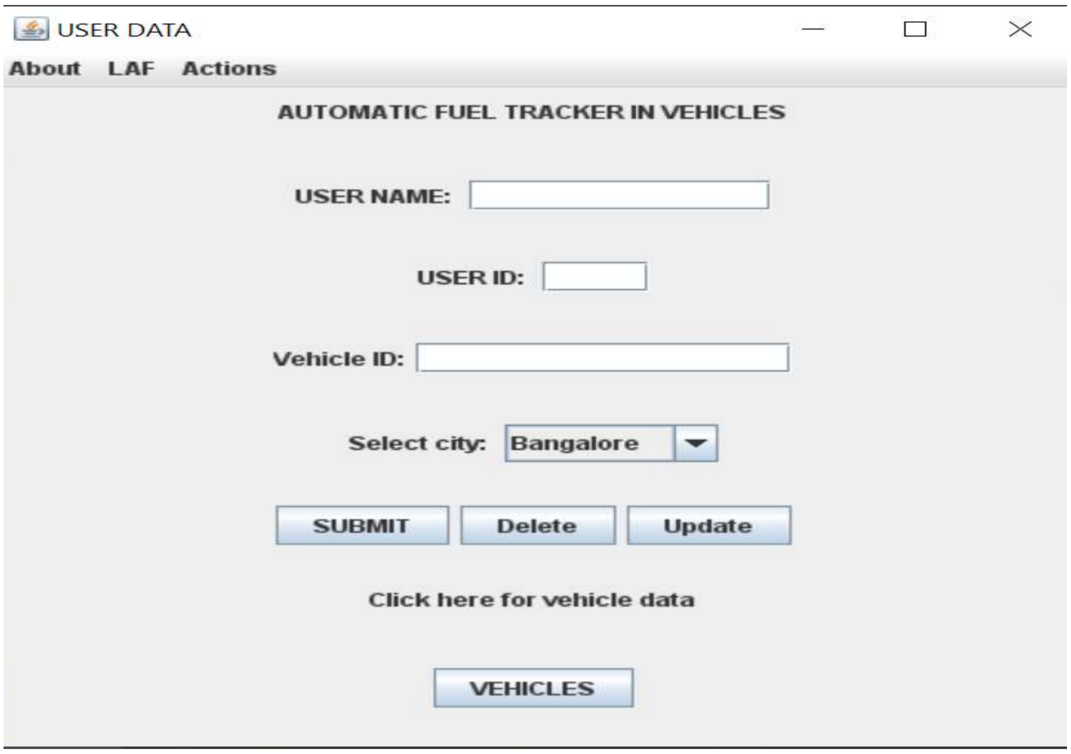
README.md

Automatic-Fuel-Tracker-in-Vehicles

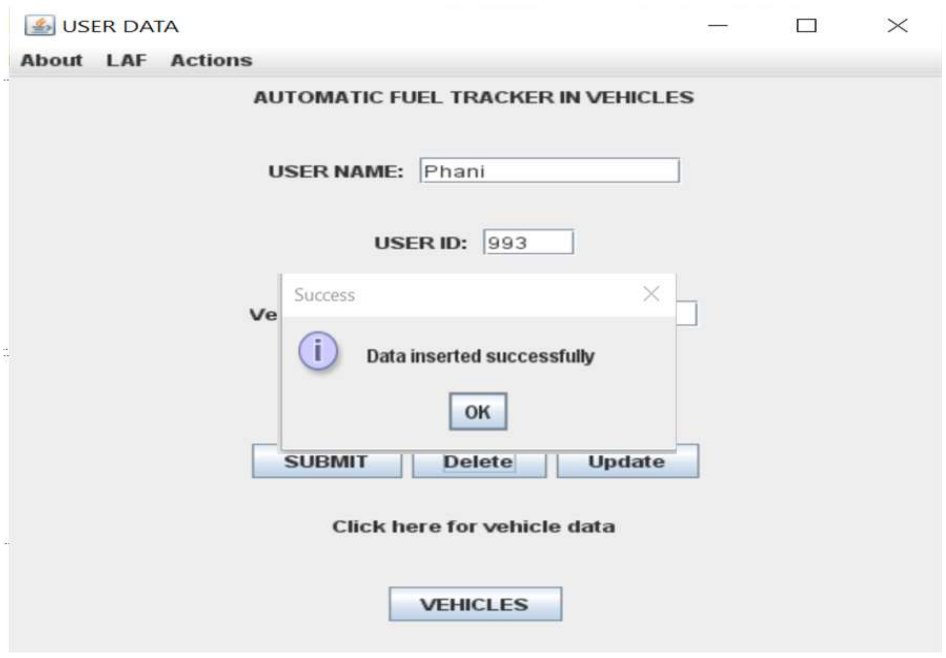
A dbms based project which works as a dashboard that monitors fuel consumption per day in vehicles

TESTING

MAIN PAGE AND USER DATA COLLECTION:



INSERTING DATA INTO USERS



USER NAME: Phani

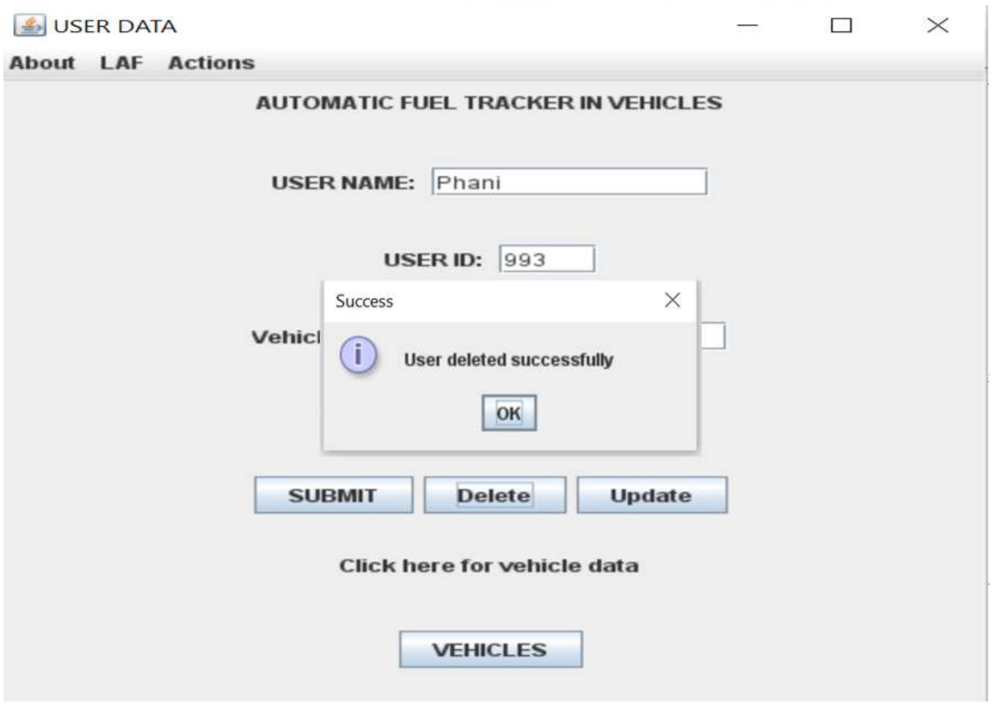
USER ID: 993

Success

Data inserted successfully

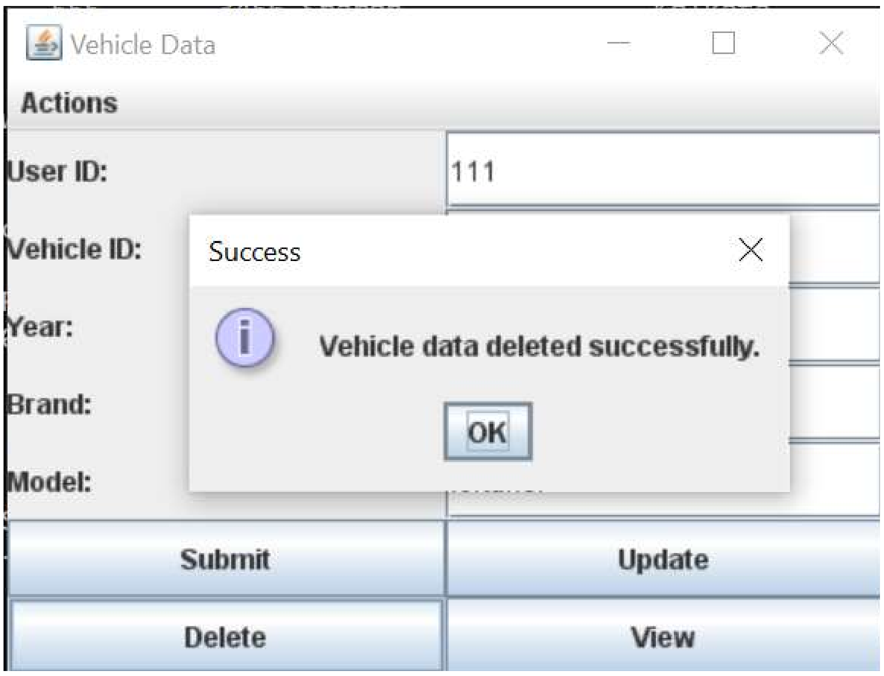
USER DATA
About LAF Actions
AUTOMATIC FUEL TRACKER IN VEHICLES
Ve
OK
SUBMIT
Delete
Update
Click here for vehicle data
VEHICLES

DELETING DATA FROM USERS:

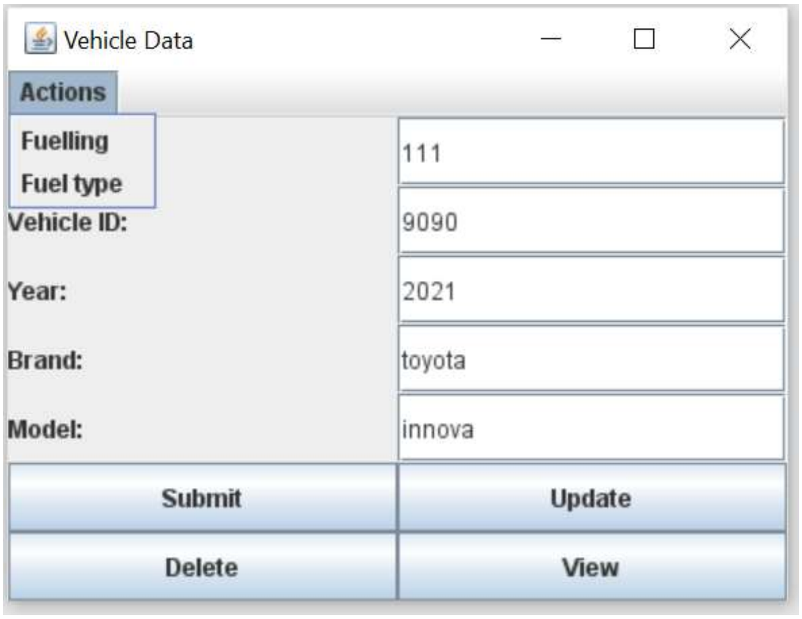


User deleted successfully

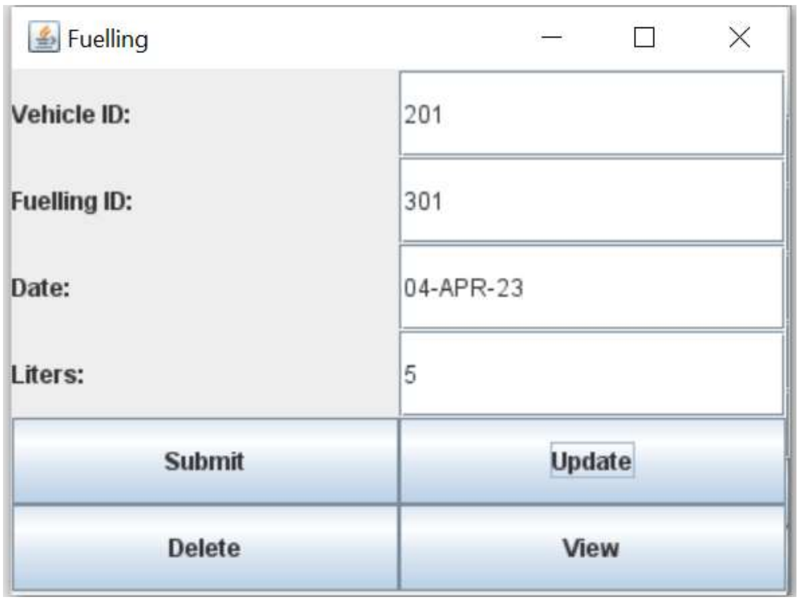
INSERTING INTO VEHICLES



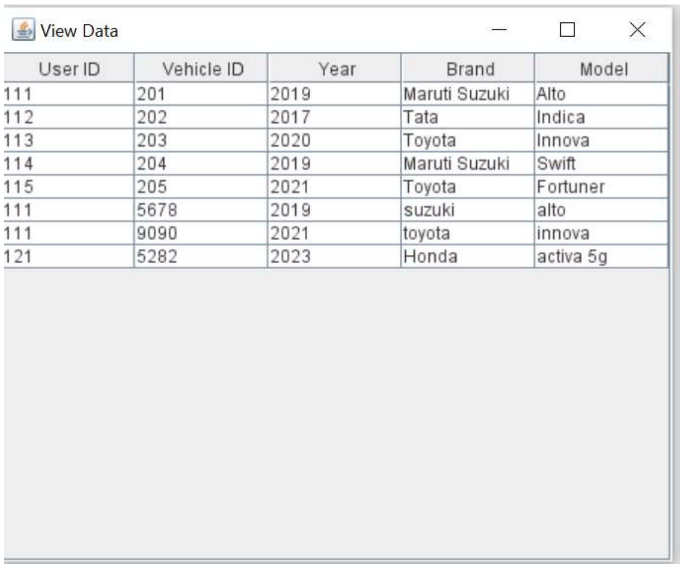
Vehicle data deleted successfully.



	111
Vehicle ID:	9090
Year:	2021
Brand:	toyota
Model:	innova



Vehicle ID:	201
Fuelling ID:	301
Date:	04-APR-23
Liters:	5



User ID	Vehicle ID	Year	Brand	Model
111	201	2019	Maruti Suzuki	Alto
112	202	2017	Tata	Indica
113	203	2020	Toyota	Innova
114	204	2019	Maruti Suzuki	Swift
115	205	2021	Toyota	Fortuner
111	5678	2019	suzuki	alto
111	9090	2021	toyota	innova
121	5282	2023	Honda	activa 5g

RESULTS

I have successfully completed the mini project "AUTOMATIC FUEL TRACKER IN VEHICLES"

DISCUSSION AND FUTURE WORK

The project has a promising future with potential avenues for expansion and improvement. By integrating additional features such as user authentication, data analytics, and reporting capabilities, the system can provide enhanced insights into vehicle management and fuel consumption patterns. Furthermore, integrating with external APIs or services can enable real-time fuel price updates and automated data retrieval. The project can also be extended to support a mobile application, allowing users to access and manage vehicle data on the go. With continuous enhancements and adaptations to meet user needs, the project holds the potential to become a comprehensive and user-friendly vehicle management solution.

REFERENCES

- **https://docs.oracle.com/en/java/**
- **https://docs.oracle.com/javase/tutorial/uiswing/**
- **https://docs.oracle.com/javase/tutorial/jdbc/**