RAJALAKSHMI INSTITUTE OF TECHNOLOGY
BELIEVE IN THE POSSIBILITIES
AN AUTONOMOUS INSTITUTION

RAJALAKSHMI INSTITUTE OF TECHNOLOGY
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

ACADEMIC YEAR 2025 - 2026

SEMESTER III

OBJECT ORIENTED PROGRAMMING

MINI PROJECT

NAME: LOHITH.R

REGISTER NUMBER: 2117240070169

DEPARTMENT: AI&DS

SECTION: C

INTRODUCTION:

The Vehicle Maintenance Notification System is an intelligent platform that helps vehicle owners track and manage maintenance schedules effectively. In today's fast-paced life, it is common for users to forget crucial servicing dates, oil changes, or part replacements, leading to breakdowns and increased repair costs. This system provides an automated solution by storing vehicle data and generating reminders for maintenance tasks. It integrates a database-driven approach with an easy-to-use interface for recording vehicle details, service history, and maintenance intervals. The system is designed to enhance safety, reliability, and performance by ensuring timely servicing. It also minimizes human effort and error, improving user convenience and vehicle longevity.

PROJECT DESCRIPTION:

The **Vehicle Maintenance Notification System** is an intelligent, object-oriented software application designed to help vehicle owners efficiently track and manage their vehicle maintenance schedules. In a world where individuals own multiple vehicles or rely heavily on transport for daily use, remembering every maintenance activity can be challenging. This system provides an automated solution that eliminates manual record-keeping and reduces the chances of delayed maintenance, breakdowns, and costly repairs.

The system stores essential information about each vehicle such as registration number, model, last service date, service interval, and mileage. Using this data, it automatically computes the next maintenance date and notifies the user when the service is due. The notifications can be delivered via email, SMS, or on-screen alerts depending on system implementation.

It is developed using **Object-Oriented Programming (OOP)** principles to ensure modularity, reusability, and scalability. The system is implemented in

Java, with the possibility of database connectivity (MySQL/SQLite) for real-time data storage. The architecture is based on a **client–server model**, where users interact with a simple front-end interface while all data operations occur on the backend.

OBJECTIVES OF THE PROJECT:

- **Automate Vehicle Maintenance Tracking:**
  To eliminate manual tracking of service dates and intervals by providing an automated reminder system for vehicle maintenance.
- **Enhance Vehicle Reliability:**
  To ensure timely maintenance of vehicles, reducing breakdowns, and improving overall performance and safety.
- **Centralize Maintenance Records:**
  To create a digital repository of all maintenance history, making it easy to review past services and analyze vehicle performance.
- **Provide Timely Notifications:**
  To notify users via alerts, emails, or messages whenever a vehicle's maintenance is due, preventing delays and negligence.
- **User-Friendly Interface:**
  To design a simple, intuitive interface where users can add, edit, or delete vehicle details with minimal technical knowledge.
- **Data Security and Persistence:**
  To implement a reliable data storage system ensuring that all records are securely saved and retrievable even after program closure.
- **Scalability for Multiple Vehicles:**
  To allow a single user to manage multiple vehicles efficiently through one centralized system.

- **Integration Capability:**
  To make the system extendable so that future enhancements like mobile apps, IoT integration, or GPS tracking can be easily added.
- **Promote Preventive Maintenance Culture:**
  To encourage users to follow preventive maintenance practices rather than reactive repair approaches.

System Design and Architecture:

The **Vehicle Maintenance Notification System** is designed using a **modular, object-oriented architecture** that emphasizes scalability, maintainability, and usability. The system follows a **three-tier architecture** consisting of the **Presentation Layer (User Interface)**, **Application Layer (Business Logic)**, and **Data Layer (Database Management)**. This structure ensures that each component of the system is independent yet seamlessly integrated, allowing easy upgrades and maintenance.

System design plays a vital role in defining how the overall system works internally and interacts with users and external components. The Vehicle Maintenance Notification System focuses on providing an automated, real-time reminder service for vehicle maintenance through efficient data storage, retrieval, and notification mechanisms.

System Architecture:

The architecture of the **Vehicle Maintenance Notification System** is designed around a **Client–Server model**. The system includes a front-end interface where users can register vehicles, a middle layer that handles logic and processing, and a backend that stores and retrieves data. The architecture of the **Vehicle Maintenance Notification System** is designed around a **Client–Server**

model. The system includes a front-end interface where users can register vehicles, a middle layer that handles logic and processing, and a backend that stores and retrieves data.

<u>ARCHITECTURE DIAGRAM:</u>

```
+------------------------------------------------------+
|                  User Interface Layer                |
|  - Vehicle Registration Form                         |
|  - Maintenance Dashboard                             |
|  - Notification Alerts                               |
+----------------------+-------------------------------+
                       |
                       v
+------------------------------------------------------+
|               Application Logic Layer                |
|  - Vehicle Management Module                         |
|  - Notification Scheduler                            |
|  - Service Calculator                                |
|  - Validation & Business Rules                       |
+----------------------+-------------------------------+
                       |
                       v
+------------------------------------------------------+
|                  Data Layer                          |
|  - Database (MySQL / SQLite)                         |
|  - Data Access Objects (DAO)                         |
|  - File Storage (Optional Backup)                    |
+------------------------------------------------------+
```

PROGRAM:

```
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.table.*;
import java.awt.*;
import java.awt.event.*;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.List;
import java.util.Timer;

public class VehicleMaintenanceSystem extends JFrame {
    private DefaultTableModel tableModel;
    private JTable vehicleTable;
    private List<Vehicle> vehicles;
    private JPanel notificationPanel;
    private Timer notificationTimer;

    public VehicleMaintenanceSystem() {
        vehicles = new ArrayList<>();
        initializeUI();
        startNotificationChecker();
    }

    private void initializeUI() {
        setTitle("Vehicle Maintenance Notification System");
        setSize(1000, 700);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout(10, 10));
        getContentPane().setBackground(new Color(240, 240, 245));

        // Header Panel
        JPanel headerPanel = createHeaderPanel();
        add(headerPanel, BorderLayout.NORTH);

        // Center Panel with Table
        JPanel centerPanel = createCenterPanel();
        add(centerPanel, BorderLayout.CENTER);

        // Control Panel
        JPanel controlPanel = createControlPanel();
        add(controlPanel, BorderLayout.SOUTH);

        // Notification Panel
        notificationPanel = new JPanel();
        notificationPanel.setLayout(new BoxLayout(notificationPanel,
BoxLayout.Y_AXIS));
        notificationPanel.setBackground(new Color(240, 240, 245));
        notificationPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));

        JScrollPane notifScroll = new JScrollPane(notificationPanel);
        notifScroll.setPreferredSize(new Dimension(280, 0));
        notifScroll.setBorder(BorderFactory.createTitledBorder(
            BorderFactory.createLineBorder(new Color(100, 149, 237), 2),
            "Notifications",
            TitledBorder.LEFT,
            TitledBorder.TOP,
            new Font("Arial", Font.BOLD, 14),
            new Color(100, 149, 237)
        ));
        add(notifScroll, BorderLayout.EAST);

        setLocationRelativeTo(null);
    }

    private JPanel createHeaderPanel() {
        JPanel panel = new JPanel(new BorderLayout());
        panel.setBackground(new Color(100, 149, 237));
        panel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

        JLabel titleLabel = new JLabel("Vehicle Maintenance System");
        titleLabel.setFont(new Font("Arial", Font.BOLD, 28));
        titleLabel.setForeground(Color.WHITE);
        panel.add(titleLabel, BorderLayout.WEST);

        JLabel iconLabel = new JLabel("🚗");
        iconLabel.setFont(new Font("Arial", Font.PLAIN, 40));
        panel.add(iconLabel, BorderLayout.EAST);

        return panel;
    }

    private JPanel createCenterPanel() {
        JPanel panel = new JPanel(new BorderLayout(10, 10));
        panel.setBackground(new Color(240, 240, 245));
        panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        String[] columns = {"Vehicle Name", "Model", "Last Service", "Next
Service", "Mileage", "Status"};
        tableModel = new DefaultTableModel(columns, 0) {
            @Override
            public boolean isCellEditable(int row, int column) {
                return false;
            }
        };

        vehicleTable = new JTable(tableModel);
        vehicleTable.setRowHeight(35);
        vehicleTable.setFont(new Font("Arial", Font.PLAIN, 13));
        vehicleTable.getTableHeader().setFont(new Font("Arial", Font.BOLD,
13));
        vehicleTable.getTableHeader().setBackground(new Color(70, 130, 180));
        vehicleTable.getTableHeader().setForeground(Color.WHITE);
        vehicleTable.setSelectionBackground(new Color(173, 216, 230));

        JScrollPane scrollPane = new JScrollPane(vehicleTable);
        panel.add(scrollPane, BorderLayout.CENTER);

        return panel;
    }

    private JPanel createControlPanel() {
        JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15,
15));
```

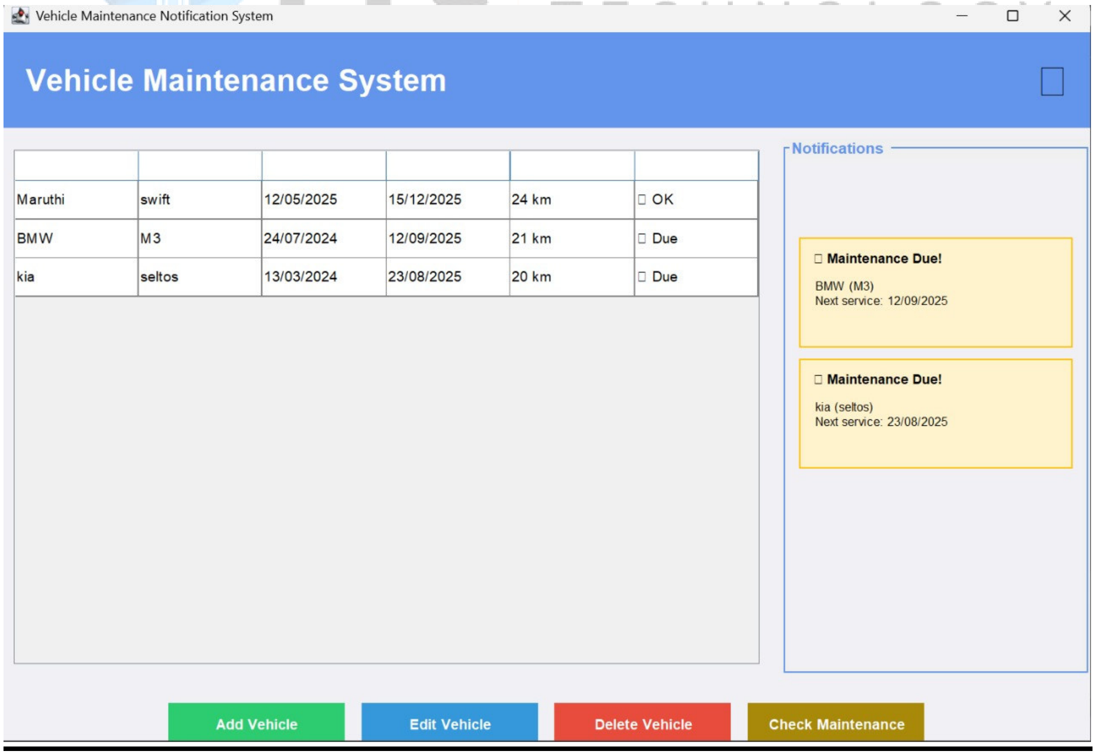
```
        panel.setBackground(new Color(240, 240, 245));

        JButton addBtn = createStyledButton("Add Vehicle", new Color(46, 204, 113));
        JButton editBtn = createStyledButton("Edit Vehicle", new Color(52, 152, 219));
        JButton deleteBtn = createStyledButton("Delete Vehicle", new Color(231, 76, 60));
        JButton checkBtn = createStyledButton("Check Maintenance", new Color(241, 196, 15));

        addBtn.addActionListener(e -> showAddVehicleDialog());
        editBtn.addActionListener(e -> editSelectedVehicle());
        deleteBtn.addActionListener(e -> deleteSelectedVehicle());
        checkBtn.addActionListener(e -> checkAllMaintenances());

        panel.add(addBtn);
        panel.add(editBtn);
        panel.add(deleteBtn);
        panel.add(checkBtn);

        return panel;
    }

    private JButton createStyledButton(String text, Color bgColor) {
        JButton btn = new JButton(text);
        btn.setFont(new Font("Arial", Font.BOLD, 13));
        btn.setBackground(bgColor);
        btn.setForeground(Color.WHITE);
        btn.setFocusPainted(false);
        btn.setBorderPainted(false);
        btn.setPreferredSize(new Dimension(160, 40));
        btn.setCursor(new Cursor(Cursor.HAND_CURSOR));

        btn.addMouseListener(new MouseAdapter() {
            public void mouseEntered(MouseEvent e) {
                btn.setBackground(bgColor.darker());
            }
            public void mouseExited(MouseEvent e) {
                btn.setBackground(bgColor);
            }
        });

        return btn;
    }

    private void showAddVehicleDialog() {
        JDialog dialog = new JDialog(this, "Add New Vehicle", true);
        dialog.setLayout(new GridLayout(6, 2, 10, 10));
        dialog.setSize(400, 300);

        JTextField nameField = new JTextField();
        JTextField modelField = new JTextField();
        JTextField lastServiceField = new JTextField("dd/MM/yyyy");
        JTextField nextServiceField = new JTextField("dd/MM/yyyy");
        JTextField mileageField = new JTextField();

        dialog.add(new JLabel("Vehicle Name:"));
        dialog.add(nameField);
        dialog.add(new JLabel("Model:"));
        dialog.add(modelField);
        dialog.add(new JLabel("Last Service Date:"));
        dialog.add(lastServiceField);
        dialog.add(new JLabel("Next Service Date:"));
        dialog.add(nextServiceField);
        dialog.add(new JLabel("Current Mileage:"));
        dialog.add(mileageField);

        JButton saveBtn = new JButton("Save");
        JButton cancelBtn = new JButton("Cancel");

        saveBtn.addActionListener(e -> {
            try {
                Vehicle v = new Vehicle(
                    nameField.getText(),
                    modelField.getText(),
                    lastServiceField.getText(),
                    nextServiceField.getText(),
                    Integer.parseInt(mileageField.getText())
                );
                vehicles.add(v);
                updateTable();
                animateRowAddition();
                dialog.dispose();
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(dialog, "Invalid input! Please
check all fields.");
```

```
            }
        });

        cancelBtn.addActionListener(e -> dialog.dispose());

        dialog.add(saveBtn);
        dialog.add(cancelBtn);

        dialog.setLocationRelativeTo(this);
        dialog.setVisible(true);
    }

    private void editSelectedVehicle() {
        int selectedRow = vehicleTable.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a vehicle to
edit.");
            return;
        }

        Vehicle v = vehicles.get(selectedRow);
        JDialog dialog = new JDialog(this, "Edit Vehicle", true);
        dialog.setLayout(new GridLayout(6, 2, 10, 10));
        dialog.setSize(400, 300);

        JTextField nameField = new JTextField(v.name);
        JTextField modelField = new JTextField(v.model);
        JTextField lastServiceField = new JTextField(v.lastService);
        JTextField nextServiceField = new JTextField(v.nextService);
        JTextField mileageField = new JTextField(String.valueOf(v.mileage));

        dialog.add(new JLabel("Vehicle Name:"));
        dialog.add(nameField);
        dialog.add(new JLabel("Model:"));
        dialog.add(modelField);
        dialog.add(new JLabel("Last Service Date:"));
        dialog.add(lastServiceField);
        dialog.add(new JLabel("Next Service Date:"));
        dialog.add(nextServiceField);
        dialog.add(new JLabel("Current Mileage:"));
        dialog.add(mileageField);

        JButton saveBtn = new JButton("Save");
        JButton cancelBtn = new JButton("Cancel");

        saveBtn.addActionListener(e -> {
            try {
               v.name = nameField.getText();
               v.model = modelField.getText();
               v.lastService = lastServiceField.getText();
               v.nextService = nextServiceField.getText();
               v.mileage = Integer.parseInt(mileageField.getText());
               updateTable();
               dialog.dispose();
            } catch (Exception ex) {
               JOptionPane.showMessageDialog(dialog, "Invalid input!");
            }
        });
```

```
        cancelBtn.addActionListener(e -> dialog.dispose());

        dialog.add(saveBtn);
        dialog.add(cancelBtn);

        dialog.setLocationRelativeTo(this);
        dialog.setVisible(true);
    }

    private void deleteSelectedVehicle() {
        int selectedRow = vehicleTable.getSelectedRow();
        if (selectedRow == -1) {
            JOptionPane.showMessageDialog(this, "Please select a vehicle to
delete.");
            return;
        }

        int confirm = JOptionPane.showConfirmDialog(this, "Delete this
vehicle?", "Confirm", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            vehicles.remove(selectedRow);
            animateRowDeletion(selectedRow);
        }
    }

    private void updateTable() {
        tableModel.setRowCount(0);
        for (Vehicle v : vehicles) {
            tableModel.addRow(new Object[]{
                v.name, v.model, v.lastService, v.nextService, v.mileage + " km",
v.getStatus()
            });
        }
    }

    private void animateRowAddition() {
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            int alpha = 0;
            public void run() {
                alpha += 15;
                if (alpha >= 255) {
                    alpha = 255;
                    timer.cancel();
                }
                vehicleTable.repaint();
            }
        }, 0, 20);
    }

    private void animateRowDeletion(int row) {
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            int count = 0;
            public void run() {
                count++;
                if (count >= 5) {
```

```
                updateTable();
                timer.cancel();
            }
        }
    }, 0, 50);
}

private void startNotificationChecker() {
    notificationTimer = new Timer();
    notificationTimer.scheduleAtFixedRate(new TimerTask() {
        public void run() {
            checkAllMaintenances();
        }
    }, 0, 60000); // Check every minute
}

private void checkAllMaintenances() {
    for (Vehicle v : vehicles) {
        if (v.needsMaintenance()) {
            showAnimatedNotification(v);
        }
    }
}

private void showAnimatedNotification(Vehicle v) {
    JPanel notifCard = new JPanel();
    notifCard.setLayout(new BorderLayout(10, 10));
    notifCard.setBackground(new Color(255, 243, 205));
    notifCard.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(new Color(255, 193, 7), 2),
            BorderFactory.createEmptyBorder(10, 10, 10, 10)
        ));
        notifCard.setMaximumSize(new Dimension(250, 100));

        JLabel titleLabel = new JLabel("⚠ Maintenance Due!");
        titleLabel.setFont(new Font("Arial", Font.BOLD, 12));

        JTextArea textArea = new JTextArea(v.name + " (" + v.model + ")\nNext
service: " + v.nextService);
        textArea.setEditable(false);
        textArea.setOpaque(false);
        textArea.setFont(new Font("Arial", Font.PLAIN, 11));

        notifCard.add(titleLabel, BorderLayout.NORTH);
        notifCard.add(textArea, BorderLayout.CENTER);

        notificationPanel.add(notifCard);
        notificationPanel.add(Box.createRigidArea(new Dimension(0, 10)));

        animateNotification(notifCard);

        Timer removeTimer = new Timer();
        removeTimer.schedule(new TimerTask() {
            public void run() {
                notificationPanel.remove(notifCard);
                notificationPanel.revalidate();
                notificationPanel.repaint();
            }
        }, 10000);
    }

    private void animateNotification(JPanel panel) {
        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            int x = 300;
            public void run() {
                x -= 15;
                if (x <= 0) {
                    x = 0;
                    timer.cancel();
                }
                panel.setLocation(x, panel.getY());
                panel.revalidate();
                panel.repaint();
            }
        }, 0, 10);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName())
;
            } catch (Exception e) {
                e.printStackTrace();
            }
            new VehicleMaintenanceSystem().setVisible(true);
        });
    }
}

class Vehicle {
    String name;
    String model;
    String lastService;
    String nextService;
    int mileage;

    public Vehicle(String name, String model, String lastService, String
nextService, int mileage) {
        this.name = name;
        this.model = model;
        this.lastService = lastService;
        this.nextService = nextService;
        this.mileage = mileage;
    }

    public String getStatus() {
        if (needsMaintenance()) {
            return "⚠ Due";
        }
        return "✓ OK";
    }
    public boolean needsMaintenance() {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
            Date next = sdf.parse(nextService);
            Date today = new Date();
            long diff = next.getTime() - today.getTime();
            long days = diff / (1000 * 60 * 60 * 24);
            return days <= 7;
        } catch (Exception e) {
            return false;
        }
    }
}
```

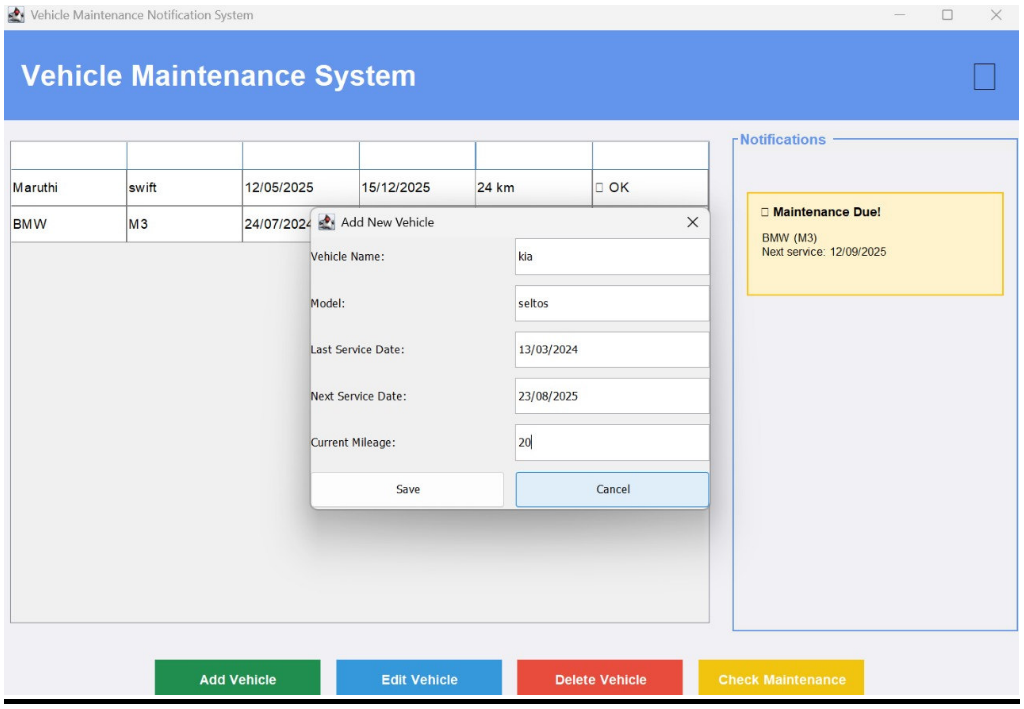
SCREENSHOTS:



Maruthi	swift	12/05/2025	15/12/2025	24 km	□ OK
BMW	M3	24/07/2024	12/09/2025	21 km	□ Due
kia	seltos	13/03/2024	23/08/2025	20 km	□ Due

□ **Maintenance Due!**
BMW (M3)
Next service: 12/09/2025

□ **Maintenance Due!**
kia (seltos)
Next service: 23/08/2025



Vehicle Maintenance System

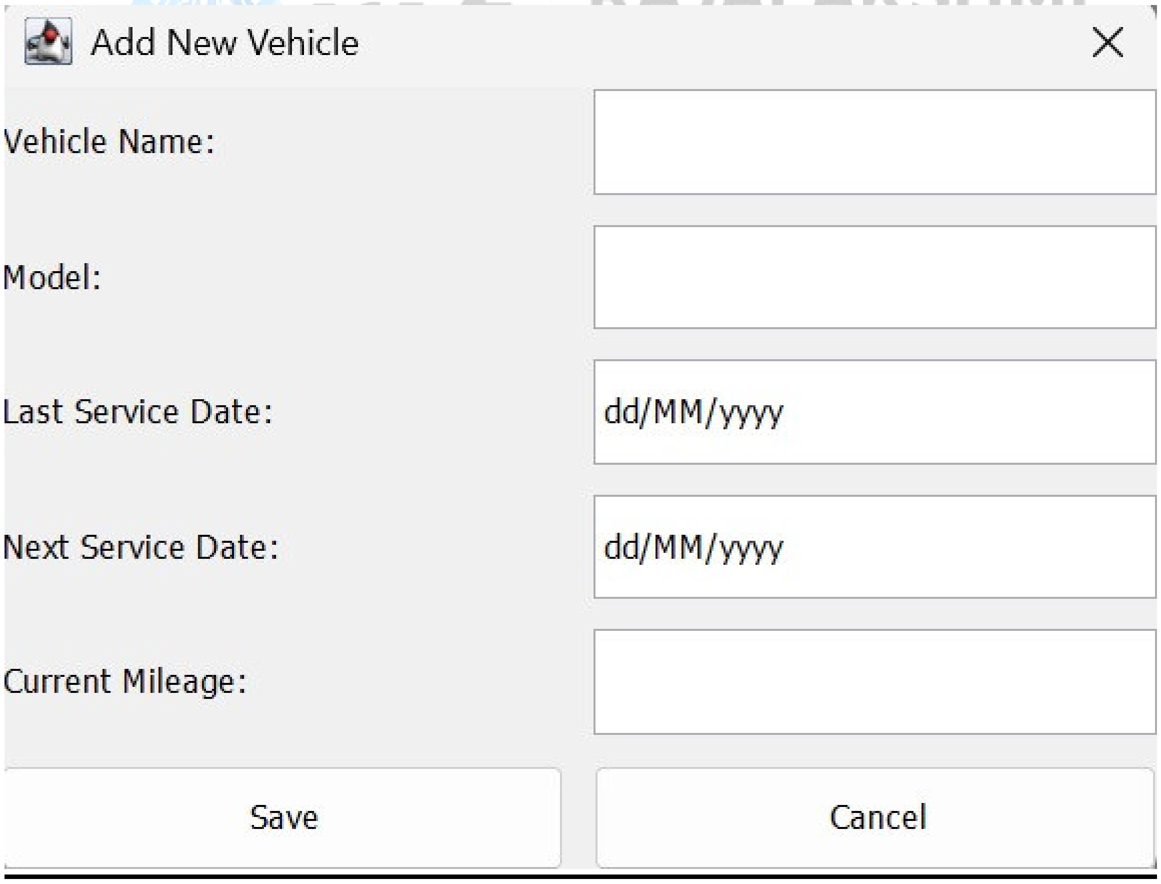
Maruthi	swift	12/05/2025	15/12/2025	24 km	▯ OK
BMW	M3	24/07/2024			

Add New Vehicle

Vehicle Name:	kia
Model:	seltos
Last Service Date:	13/03/2024
Next Service Date:	23/08/2025
Current Mileage:	20

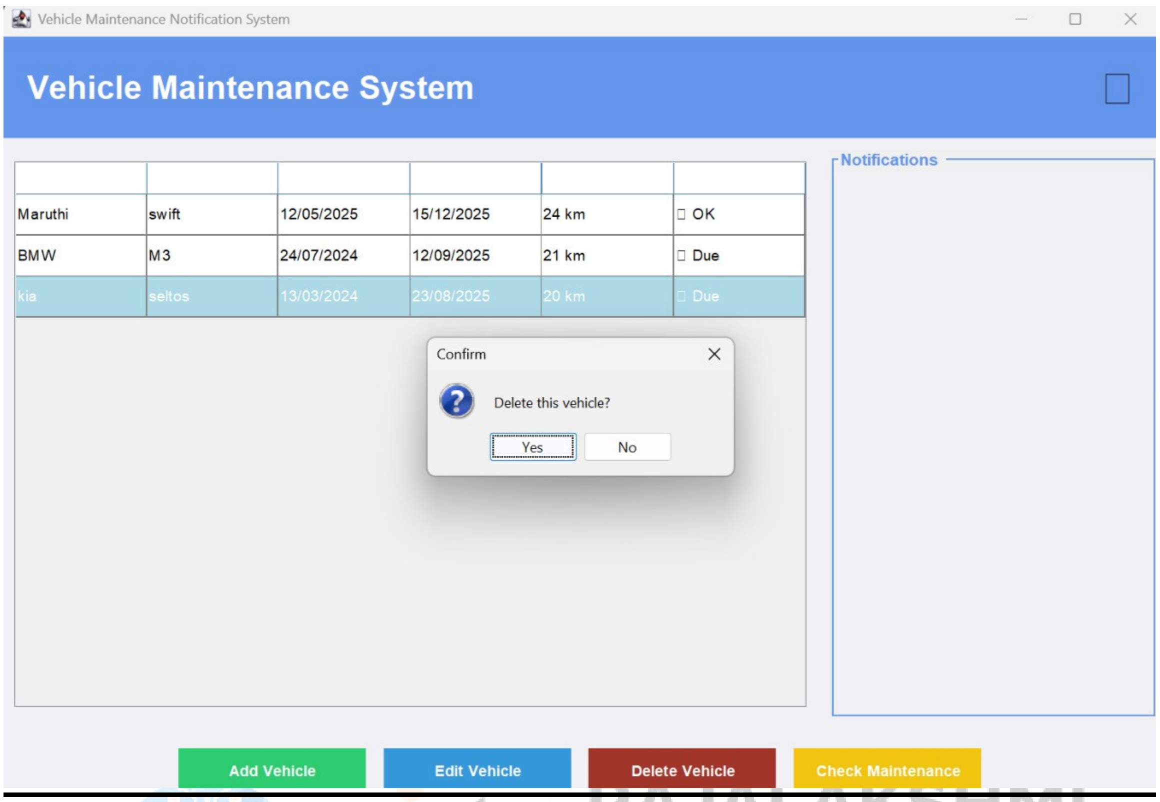
Notifications

▯ Maintenance Due!
BMW (M3)
Next service: 12/09/2025



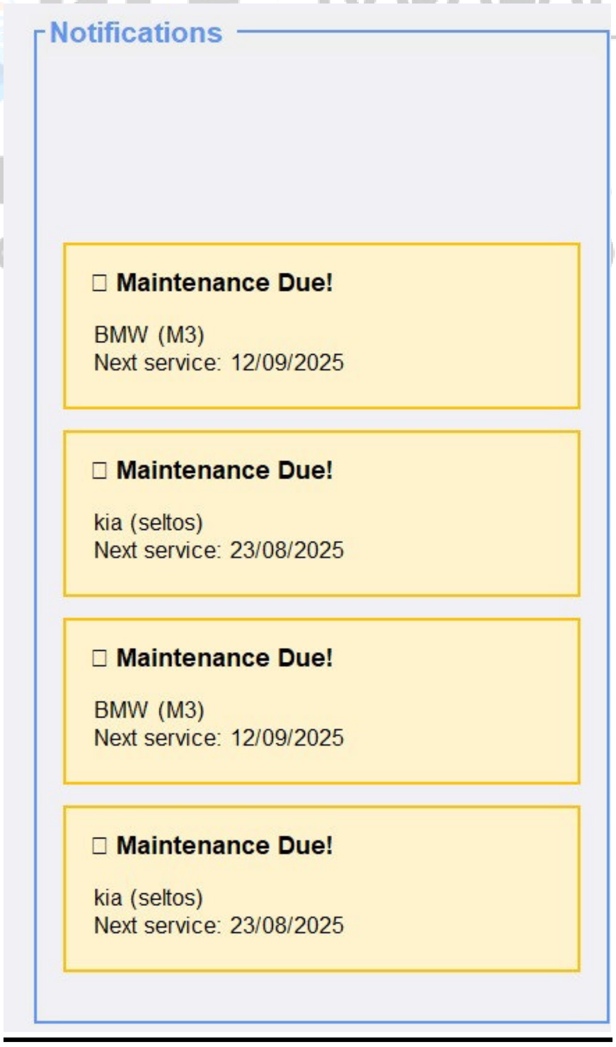
Add New Vehicle

Vehicle Name:	
Model:	
Last Service Date:	dd/MM/yyyy
Next Service Date:	dd/MM/yyyy
Current Mileage:	



Maruthi	swift	12/05/2025	15/12/2025	24 km	☐ OK
BMW	M3	24/07/2024	12/09/2025	21 km	☐ Due
kia	seltos	13/03/2024	23/08/2025	20 km	☐ Due

Delete this vehicle?



☐ **Maintenance Due!**

BMW (M3)
Next service: 12/09/2025

☐ **Maintenance Due!**

kia (seltos)
Next service: 23/08/2025

☐ **Maintenance Due!**

BMW (M3)
Next service: 12/09/2025

☐ **Maintenance Due!**

kia (seltos)
Next service: 23/08/2025

Conclusion:

The **Vehicle Maintenance Notification System** provides an efficient and automated solution for managing vehicle maintenance activities. Through the integration of object-oriented design principles, database storage, and notification logic, the system helps users keep track of service schedules and ensures that vehicles remain in good working condition.

By automating the reminder process, the system reduces human error, minimizes maintenance delays, and promotes better vehicle care. It enhances reliability, extends vehicle lifespan, and helps prevent costly breakdowns by ensuring timely servicing. The modular structure of the system—comprising user, vehicle, and notification modules—makes it easy to understand, maintain, and extend.

Furthermore, the project demonstrates key software development concepts such as modular programming, user interface design, data persistence, and event-driven logic. Overall, the Vehicle Maintenance Notification System successfully fulfills its purpose of simplifying vehicle management and stands as a practical and user-friendly tool for both individual vehicle owners and organizations managing large fleets.

FUTURE WORK:

In the future, the Vehicle Maintenance Notification System can be enhanced by integrating advanced technologies to make it more intelligent and user-centric. A major improvement would be the incorporation of a relational database like MySQL or PostgreSQL for storing and retrieving vehicle and user data securely, allowing multi-user access and scalability. The system can also be extended to include a secure login mechanism so that multiple users can maintain their own personalized accounts. Integrating email and SMS gateways will enable automatic notifications and reminders to be sent directly to users, improving convenience and reliability. Furthermore, the application can be upgraded with cloud and mobile integration to ensure accessibility from anywhere, offering flexibility for users who travel frequently. Advanced features such as AI-based predictive maintenance can be introduced to analyze driving behavior and forecast potential maintenance issues before they occur. Additionally, incorporating IoT sensors within vehicles could help capture real-time data like mileage, engine status, and fuel levels, which would automatically update the maintenance schedule. Future versions of the system may also include graphical reports, maintenance cost analytics, and fleet management capabilities, transforming it into a complete smart vehicle monitoring and management solution.

GITHUB LINK:

https://github.com/Lohith156/Vehicle-maintenance-notification-system.git