VEHICLE MAINTENANCE SYSTEM

A PROJECT REPORT

Submitted by

VIDHULA T R (8115U23EC119)

in partial fulfillment of requirements for the award of the course

EGB1201 - JAVA PROGRAMMING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

DECEMBER - 2024

K. RAMAKRISHNAN COLLEGE OF ENGINEERING

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **" VEHICLE MAINTENANCE SYSTEM"** is the bonafide work of **VIDHULA T R (8115U23EC119)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

SIGNATURE	**SIGNATURE**
Dr. T. M. NITHYA, M.E.,Ph.D.,	Mr. K.SWAMINATHAN, M.E.,MBA.,(Ph.D).,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
ASSOCIATE PROFESSOR	ASSISTANT PROFESSOR
Department of CSE	Department of CSE
K.Ramakrishnan College of Engineering (Autonomous)	K.Ramakrishnan College of Engineering (Autonomous)
Samayapuram–621112.	Samayapuram–621112.

Submitted for the viva-voce examination held on ……………

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

I declare that the project report on **“VEHICLE MAINTENANCE SYSTEM”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **EGB1201 - JAVA PROGRAMMING.**

.

Signature

____

VIDHULA T R

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Engineering (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. D. SRINIVASAN, B.E, M.E., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mr. K. SWAMINATHAN, M.E.,MBA.,(Ph.D).,**Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To achieve a prominent position among the top technical institutions

MISSION OF THE INSTITUTION

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

VISION OF THE DEPARTMENT

To create eminent professionals of Computer Science and Engineering by imparting quality education.

MISSION OF THE DEPARTMENT

M1: To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

M2: To engage the students in research and development activities in the field of Computer Science and Engineering.

M3: To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firmware solutions addressing real life problems.

- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

The **Vehicle Maintenance System** is a Java-based application designed to help manage vehicle information and schedule maintenance tasks. The system allows users to add vehicle details such as registration number, make, model, and year through a graphical user interface (GUI) built using Swing components. Users can also schedule maintenance tasks, specifying the vehicle, task description, and due date. The system stores the vehicles and tasks in memory, with an option to view all scheduled maintenance tasks. The GUI is intuitive, featuring buttons for adding vehicles, scheduling tasks, and viewing the list of all tasks.The application further enhances its functionality by allowing users to filter and view tasks that are due in the next three days, helping to keep track of upcoming maintenance needs. Using a simple timer, the system checks and reminds users of upcoming due tasks. The GUI is designed to provide an easy-to-use interface for managing vehicles and their maintenance schedules, and it helps ensure timely maintenance by displaying relevant tasks and deadlines. Through the combination of these features, the system ensures effective tracking of vehicle maintenance and reminders.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Vehicle Maintenance System automates the process of managing vehicle records and maintenance tasks, providing a streamlined, user-friendly solution	PO1 – 2	PSO1 – 2
The system identifies real-world problems and develops a modular design using Java Swing and OOP principles to solve them.	PO2 – 3	PSO2 – 3
Users can add vehicles, schedule maintenance tasks, and view due tasks efficiently through a well-organized interface.	PO3 – 3	PSO3 – 3
The project integrates advanced Java concepts like date-time libraries, collections, and event-driven programming to enhance functionality.	PO4 – 3	PSO2 – 2
By facilitating task reminders and maintaining maintenance history, the system ensures proper vehicle upkeep and timely servicing.	PO5 - 2	PSO1 - 3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the **Vehicle Maintenance System** is to provide an efficient and user-friendly solution for managing vehicle information and scheduling maintenance tasks. This system aims to ensure timely upkeep of vehicles by allowing users to record details, schedule tasks, and track upcoming maintenance deadlines. By offering an intuitive GUI and features to view both scheduled and due tasks, the application helps users streamline vehicle maintenance and prevent delays in essential services.

Additionally, the system aims to improve the overall organization and monitoring of vehicle-related responsibilities. It is designed to cater to individual users or organizations that maintain a fleet of vehicles, reducing the risk of missed maintenance schedules and improving vehicle longevity and performance.

1.2 Overview

The Vehicle Maintenance System is a desktop application implemented in Java, designed to streamline the management of vehicle information and maintenance schedules. Through an intuitive Swing-based graphical user interface, the program enables users to add detailed records of vehicles, including their registration number, make, model, and year of manufacture. Users can also schedule maintenance tasks by associating them with a specific vehicle, providing task descriptions, and specifying due dates. These details are stored in memory for easy access and display within the application.

The system provides two primary views for users to manage maintenance tasks effectively:

1. **View Scheduled Tasks:** Displays all scheduled maintenance tasks, including vehicle details, task descriptions, and their respective due dates.
2. **Show Due Tasks:** Filters and displays only tasks that are due within the next three days, ensuring users are aware of imminent maintenance needs.

By organizing vehicle data and maintenance tasks in one place, the system helps users ensure timely upkeep and improved vehicle performance, making it an essential tool for individuals or organizations managing multiple vehicles.

1.3 Java Programming Concepts

1.3.1 Object-Oriented Programming (OOP):

- Classes and Objects: The program defines **Vehicle** and **MaintenanceTask** as separate classes to encapsulate vehicle and task details, making the design modular and reusable.
- Encapsulation: Private fields in the **Vehicle** and **MaintenanceTask** classes are accessed and modified through public getter methods, maintaining data integrity.
- Instance Management: Objects are created for vehicles and maintenance tasks and stored in lists for easy management.

1.3.2 Java Swing (GUI Programming):

- **Swing Components:** The program uses Swing components such as JFrame, JPanel, JLabel, JButton, JTextField, JTable, and JComboBox to create the graphical user interface.
- **Layouts:** Layout managers like **GridLayout** and **BorderLayout** are used to arrange GUI components systematically.
- **Event Handling:** Action listeners **(ActionListener)** are implemented to handle button clicks and perform corresponding actions, such as saving data or displaying tasks.

1.3.3 Collections Framework:

- **List Interface:** The program uses ArrayList to maintain a dynamic list of vehicles and tasks, enabling efficient storage and retrieval.
- **Iteration:** Enhanced for loops are used to iterate over the lists to populate tables and filter tasks.

1.3.4 Date and Time API:

- **LocalDate** and **DateTimeFormatter:** These classes from the **java.time** package are used to parse and format dates, ensuring accurate handling of due dates and date-based calculations.
- **Date Filtering:** Logic to determine tasks due in the next three days is implemented using LocalDate comparisons.

1.3.5 Data Validation and Error Handling:

- **Input Validation:** Input fields such as the year and due date ensure the correct format and valid values are entered.
- **Exception Handling:** Although not explicitly included in this version, try-catch blocks can be added for robust error handling during parsing and other operations.

1.3.6 Timers:

- **Timers and Scheduling:** A Timer and TimerTask were initially used for reminders, but the program now allows manual retrieval of due tasks for better flexibility.

1.3.7 Java AWT (Abstract Window Toolkit):

- **Basic UI Support:** The program integrates AWT concepts like Color and Font for styling and enhancing the visual appeal of the GUI.

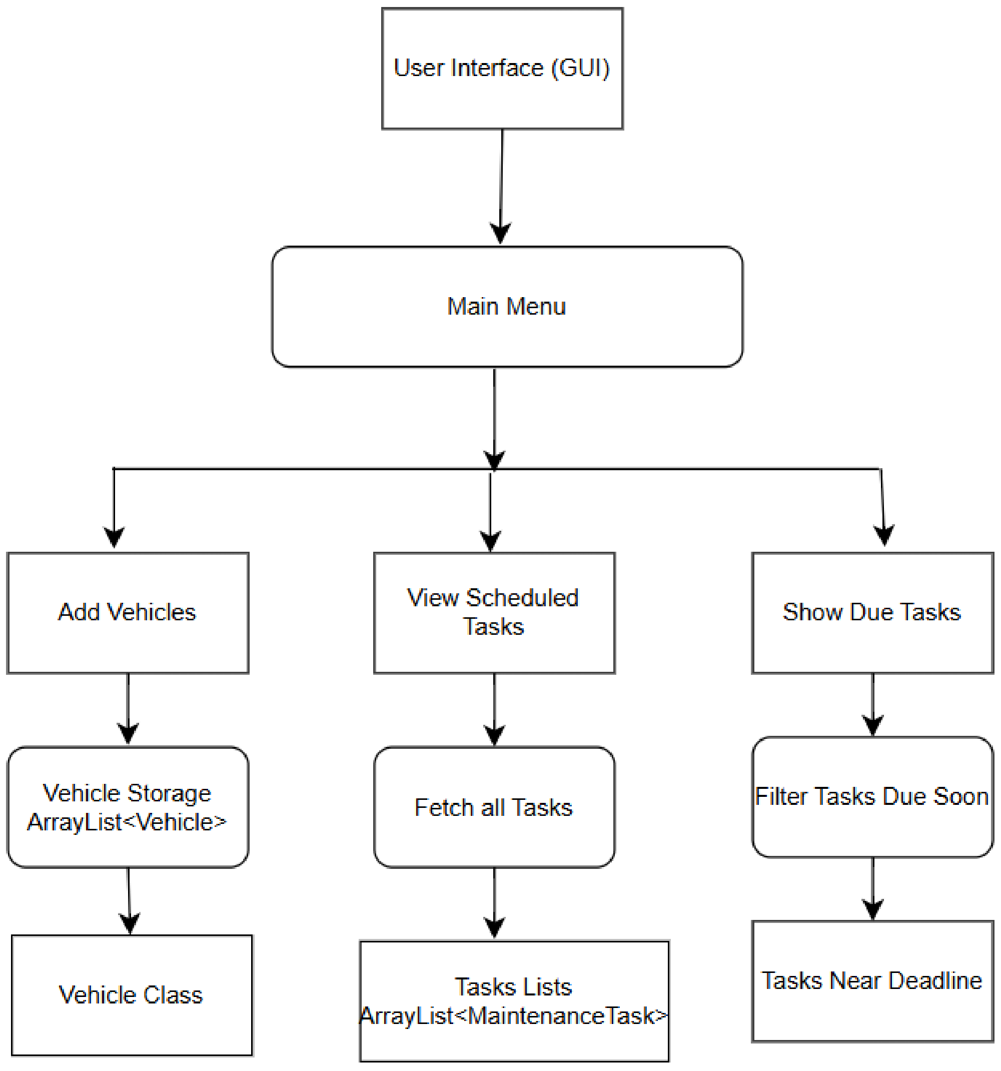
CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work of the **Vehicle Maintenance System** is to provide a comprehensive solution for managing vehicle information and scheduling maintenance tasks effectively. The application is designed to address the challenges of keeping track of multiple vehicles and their respective maintenance schedules, ensuring timely servicing and reducing the risk of vehicle downtime. The system enables users to store detailed vehicle records, schedule maintenance tasks with specific due dates, and retrieve both comprehensive and filtered task lists to aid in maintenance planning.

Key proposed functionalities include the ability to view all scheduled maintenance tasks and a separate feature to filter and display tasks due in the next three days. This approach helps users prioritize immediate maintenance needs while maintaining a clear overview of all scheduled activities. The program also aims to deliver an easy-to-use interface, built with Java Swing, to make vehicle and maintenance management accessible for both individuals and organizations managing vehicle fleets. By integrating features like task filtering, data storage, and user-friendly interfaces, the system proposes a reliable tool to streamline vehicle maintenance processes.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 User Interface Module

Description:

This module manages the graphical interface of the application, enabling users to interact with the system.

Key Components:

- **Main Menu:** Displays the primary options: Add Vehicle, View Scheduled Tasks, and Show Due Tasks.
- **Forms and Tables:**
  - Add Vehicle Form: Captures details of a vehicle.
  - Task Display Table: Shows scheduled tasks or tasks near their due date.

3.2 Vehicle Management Module

Description:

Handles the addition and management of vehicles in the system.

Key Components:

- **Add Vehicle Form:** Collects user input for vehicle registration number, make, model, and year.
- **Vehicle Class:** Represents vehicle data (attributes like registration, make, etc.).

3.3 Task Scheduling Module

Description:

Manages the scheduling of maintenance tasks for vehicles.

Key Components:

- **Task Scheduling Form:** Captures task description, associated vehicle, and due date.
- **MaintenanceTask Class:** Represents tasks with attributes like vehicle registration, description, and due date.

3.4 Task Retrieval Module

Description:

Retrieves and displays maintenance tasks based on user requests.

Key Features:

- **View Scheduled Tasks:** Displays all tasks in the system.
- **Show Due Tasks:** Filters and displays tasks that are due within the next three days.

3.5 Data Management Module

Description:

Handles the internal storage and management of vehicle and task data.

Key Components:

- **Vehicle List (ArrayList<Vehicle>):** Stores vehicle details.
- **Task List (ArrayList<MaintenanceTask>):** Stores maintenance task details**.**

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Vehicle Maintenance System successfully demonstrates the application of software engineering principles and Java programming to solve a real-world problem. By automating the process of adding vehicles, scheduling maintenance tasks, and tracking upcoming due dates, the system reduces manual efforts and improves efficiency. Its intuitive graphical user interface, developed using Java Swing, ensures ease of use for end-users, while the modular design supports scalability for further enhancements. The program has been structured to meet both personal and small-scale fleet management needs, ensuring timely servicing and better vehicle upkeep. Overall, this project highlights the integration of object-oriented programming concepts and modern tool usage in addressing practical challenges.

4.2 FUTURE SCOPE

1. Integration with Databases: Future versions of the system can include database integration (e.g., MySQL or SQLite) for persistent data storage, enabling scalability and better data management.

2. Mobile Application Development: A mobile version of the application could be developed for enhanced accessibility and convenience for users on the go.

3. Notification System: The system could be enhanced to send automated email or SMS reminders for upcoming maintenance tasks.

4. Analytics and Reporting: Adding analytics features to track maintenance trends, expenses, and vehicle performance could further aid users in managing their vehicles efficiently.

APPENDIX A (SOURCE CODE)

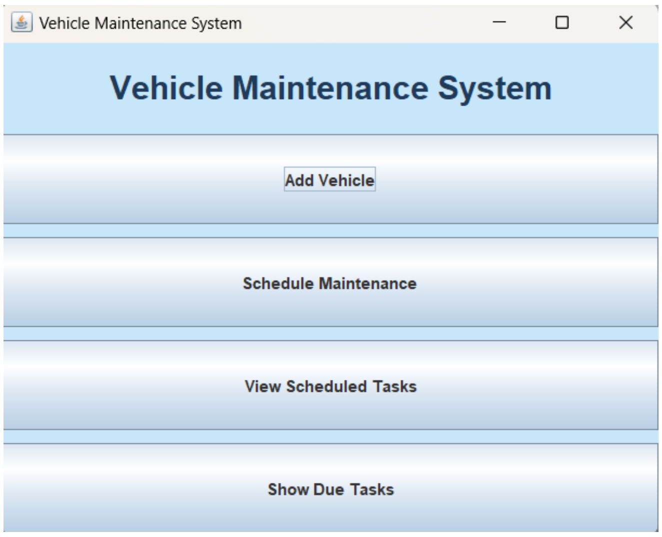
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.List;
import java.time.*;
import java.time.format.DateTimeFormatter;

public class VehicleMaintenanceSystem {
    private JFrame frame;
    private JTextField regNumberField, modelField, yearField, taskDescField,
dueDateField;
    private JComboBox<String> makeDropdown;
    private List<Vehicle> vehicles = new ArrayList<>();
    private List<MaintenanceTask> tasks = new ArrayList<>();

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new
VehicleMaintenanceSystem().createAndShowGUI());
    }

    // Create and show the main GUI
    private void createAndShowGUI() {
        frame = new JFrame("Vehicle Maintenance System");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500, 400);

        // Set a background color and layout
        JPanel mainPanel = new JPanel();
        mainPanel.setBackground(new Color(200, 230, 250)); // Light blue background
        mainPanel.setLayout(new BorderLayout());

        // Title label
        JLabel titleLabel = new JLabel("Vehicle Maintenance System",
JLabel.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));
        titleLabel.setForeground(new Color(30, 60, 90)); // Dark blue text
        titleLabel.setBorder(BorderFactory.createEmptyBorder(20, 0, 20, 0));
        mainPanel.add(titleLabel, BorderLayout.NORTH);

        // Buttons panel
        JPanel buttonsPanel = new JPanel();
        buttonsPanel.setLayout(new GridLayout(4, 1, 10, 10)); // Updated layout for 4
```

```
buttons
        buttonsPanel.setBackground(new Color(200, 230, 250)); // Match main
background color

        JButton addVehicleButton = new JButton("Add Vehicle");
        JButton scheduleTaskButton = new JButton("Schedule Maintenance");
        JButton viewTasksButton = new JButton("View Scheduled Tasks");
        JButton showDueTasksButton = new JButton("Show Due Tasks"); // New
button

        addVehicleButton.addActionListener(e -> showAddVehicleForm());
        scheduleTaskButton.addActionListener(e ->
showScheduleMaintenanceForm());
        viewTasksButton.addActionListener(e -> showScheduledTasks());
        showDueTasksButton.addActionListener(e -> showDueTasks()); // Add event
for "Show Due Tasks"

        buttonsPanel.add(addVehicleButton);
        buttonsPanel.add(scheduleTaskButton);
        buttonsPanel.add(viewTasksButton);
        buttonsPanel.add(showDueTasksButton);

        mainPanel.add(buttonsPanel, BorderLayout.CENTER);

        // Add everything to the main frame
        frame.add(mainPanel);
        frame.setVisible(true);
    }

    // Show the "Add Vehicle" form
    private void showAddVehicleForm() {
        JFrame addVehicleFrame = new JFrame("Add Vehicle");
        addVehicleFrame.setSize(300, 250);
        addVehicleFrame.setLayout(new GridLayout(5, 2));

        addVehicleFrame.add(new JLabel("Registration Number:"));
        regNumberField = new JTextField();
        addVehicleFrame.add(regNumberField);

        addVehicleFrame.add(new JLabel("Make:"));
        String[] makes = {"Toyota", "Honda", "Ford", "BMW", "Audi", "Chevrolet"};
        makeDropdown = new JComboBox<>(makes);
        addVehicleFrame.add(makeDropdown);

        addVehicleFrame.add(new JLabel("Model:"));
        modelField = new JTextField();
        addVehicleFrame.add(modelField);

        addVehicleFrame.add(new JLabel("Year:"));
        yearField = new JTextField();
        addVehicleFrame.add(yearField);

        JButton saveButton = new JButton("Save");
        saveButton.addActionListener(e -> saveVehicle());
        addVehicleFrame.add(saveButton);

        addVehicleFrame.setVisible(true);
    }

    // Save the new vehicle information
    private void saveVehicle() {
        String regNumber = regNumberField.getText();
        String make = (String) makeDropdown.getSelectedItem();
        String model = modelField.getText();
        int year = Integer.parseInt(yearField.getText());

        Vehicle newVehicle = new Vehicle(regNumber, make, model, year);
        vehicles.add(newVehicle);

        JOptionPane.showMessageDialog(frame, "Vehicle added successfully!");

        // Clear the fields after saving
        regNumberField.setText("");
        modelField.setText("");
        yearField.setText("");
    }

    // Show the "Schedule Maintenance" form
    private void showScheduleMaintenanceForm() {
        JFrame scheduleFrame = new JFrame("Schedule Maintenance");
        scheduleFrame.setSize(300, 250);
        scheduleFrame.setLayout(new GridLayout(5, 2));

        scheduleFrame.add(new JLabel("Select Vehicle:"));
        JComboBox<String> vehicleDropdown = new JComboBox<>();
        for (Vehicle vehicle : vehicles) {
            vehicleDropdown.addItem(vehicle.getRegistration());
        }
        scheduleFrame.add(vehicleDropdown);

        scheduleFrame.add(new JLabel("Task Description:"));
        taskDescField = new JTextField();
        scheduleFrame.add(taskDescField);

        scheduleFrame.add(new JLabel("Due Date (YYYY-MM-DD):"));
        dueDateField = new JTextField();
        scheduleFrame.add(dueDateField);

        JButton saveButton = new JButton("Save Task");
        saveButton.addActionListener(e -> saveMaintenanceTask(vehicleDropdown));
        scheduleFrame.add(saveButton);

        scheduleFrame.setVisible(true);
    }

    // Save the maintenance task
    private void saveMaintenanceTask(JComboBox<String> vehicleDropdown) {
        String vehicleReg = (String) vehicleDropdown.getSelectedItem();
        String description = taskDescField.getText();
        String dueDate = dueDateField.getText();

        MaintenanceTask task = new MaintenanceTask(vehicleReg, description,
dueDate);
        tasks.add(task);

        JOptionPane.showMessageDialog(frame, "Maintenance task scheduled!");

        // Clear the fields after saving
        taskDescField.setText("");
        dueDateField.setText("");
    }

    // Show all scheduled tasks (no filtering)
    private void showScheduledTasks() {
        JFrame tasksFrame = new JFrame("All Scheduled Maintenance Tasks");
        tasksFrame.setSize(400, 300);
        String[] columnNames = {"Vehicle", "Task", "Due Date"};
        Object[][] data = new Object[tasks.size()][3];

        // Populate data with all tasks
        for (int i = 0; i < tasks.size(); i++) {
            MaintenanceTask task = tasks.get(i);
            data[i][0] = task.getVehicleRegistration();
            data[i][1] = task.getDescription();
            data[i][2] = task.getDueDate();
        }
        JTable table = new JTable(data, columnNames);
        tasksFrame.add(new JScrollPane(table));
        tasksFrame.setVisible(true);
    }

    // Show tasks that are due in the next three days
    private void showDueTasks() {
        JFrame dueTasksFrame = new JFrame("Tasks Due Soon");
        dueTasksFrame.setSize(400, 300);
        String[] columnNames = {"Vehicle", "Task", "Due Date"};
        List<Object[]> dueData = new ArrayList<>();

        LocalDate today = LocalDate.now();
        for (MaintenanceTask task : tasks) {
            LocalDate dueDate = LocalDate.parse(task.getDueDate(),
DateTimeFormatter.ofPattern("yyyy-MM-dd"));
            if (!dueDate.isBefore(today) && dueDate.isBefore(today.plusDays(3))) { //
Filter tasks due in 3 days
                dueData.add(new Object[]{task.getVehicleRegistration(),
task.getDescription(), task.getDueDate()});
            }
        }

        // Convert filtered data to a 2D array for JTable
        Object[][] data = dueData.toArray(new Object[0][0]);
        JTable table = new JTable(data, columnNames);
        dueTasksFrame.add(new JScrollPane(table));
        dueTasksFrame.setVisible(true);
    }

    // Class to represent a vehicle
    class Vehicle {
        private String registration;
        private String make;
        private String model;
        private int year;

        public Vehicle(String registration, String make, String model, int year) {
            this.registration = registration;
            this.make = make;
            this.model = model;
            this.year = year;
        }

        public String getRegistration() {
            return registration;
        }

        public String getMake() {
            return make;
        }

        public String getModel() {
            return model;
        }

        public int getYear() {
            return year;
        }
    }

    // Class to represent a maintenance task
    class MaintenanceTask {
        private String vehicleRegistration;
        private String description;
        private String dueDate;

        public MaintenanceTask(String vehicleRegistration, String description, String
dueDate) {
            this.vehicleRegistration = vehicleRegistration;
            this.description = description;
            this.dueDate = dueDate;
        }

        public String getVehicleRegistration() {
            return vehicleRegistration;
        }

        public String getDescription() {
            return description;
        }

        public String getDueDate() {
            return dueDate;
        }
    }
}
```

APPENDIX B (SCREENSHOTS)

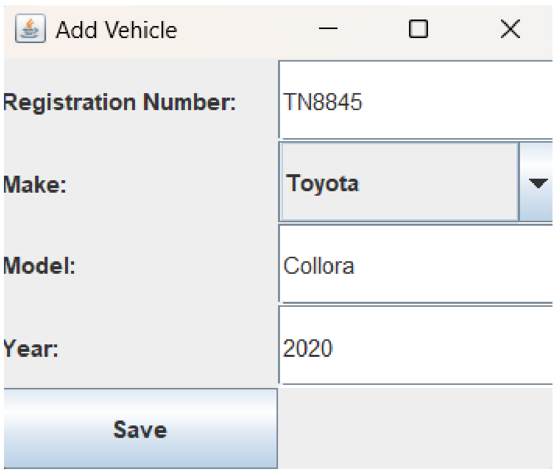
Step 1: Launching the Application

1. Launch the program, and the main interface of the **Vehicle Maintenance System** is displayed.
2. The main menu consists of the following options:
   - **Add Vehicle**
   - **Schedule Maintenance**
   - **View Scheduled Tasks**
   - **Show Due Tasks**

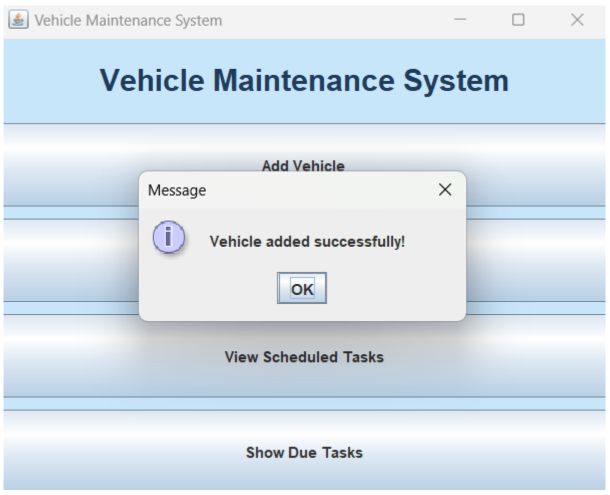


Step 2: Adding a New Vehicle

1. Then click the **Add Vehicle** button.
2. A new form appears, prompting the user to enter vehicle details:
   - **Registration Number**
   - **Make** (selected from a dropdown list)
   - **Model**
   - **Year**
3. After entering the details, the user clicks the **Save** button.
4. A confirmation message appears: *"Vehicle added successfully!"*.
5. The vehicle details are stored in the program's internal data structure (ArrayList).



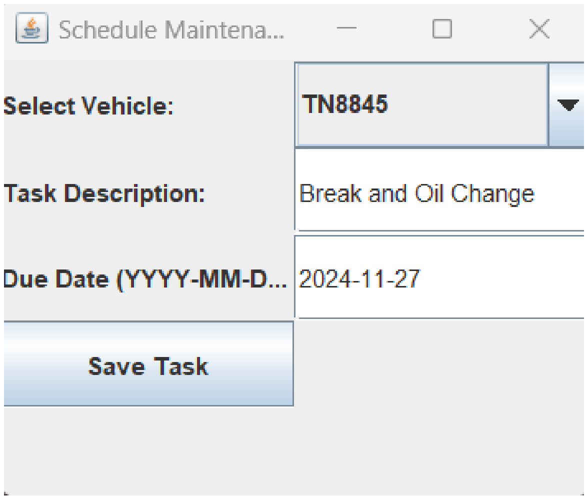
Registration Number:	TN8845
Make:	Toyota
Model:	Collora
Year:	2020



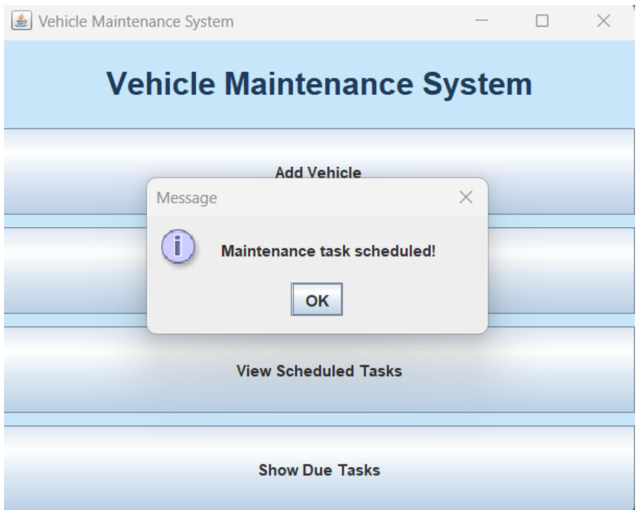
Vehicle added successfully!

Step 3: Scheduling Maintenance

1. Next click the **Schedule Maintenance** button.
2. A new form opens, asking for:
   - **Vehicle Selection**: Dropdown showing registered vehicles.
   - **Task Description**: A text field for entering the maintenance task.
   - **Due Date**: A field for entering the due date in YYYY-MM-DD format.
3. After filling in the details, the user clicks the **Save Task** button.
4. A confirmation message appears: *"Maintenance task scheduled!"*.
5. The task details are stored in the system's task list (ArrayList).



Select Vehicle:	TN8845
Task Description:	Break and Oil Change
Due Date (YYYY-MM-D...	2024-11-27



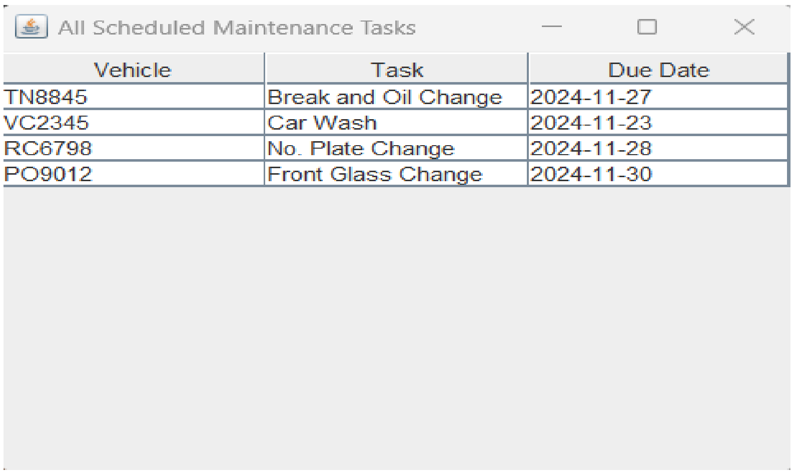
Maintenance task scheduled!

Step 4: Viewing Scheduled Tasks

1. Then click the **View Scheduled Tasks** button.
2. A table is displayed, showing all scheduled tasks with the following columns:
   - **Vehicle Registration Number**
   - **Task Description**

- **Due Date**

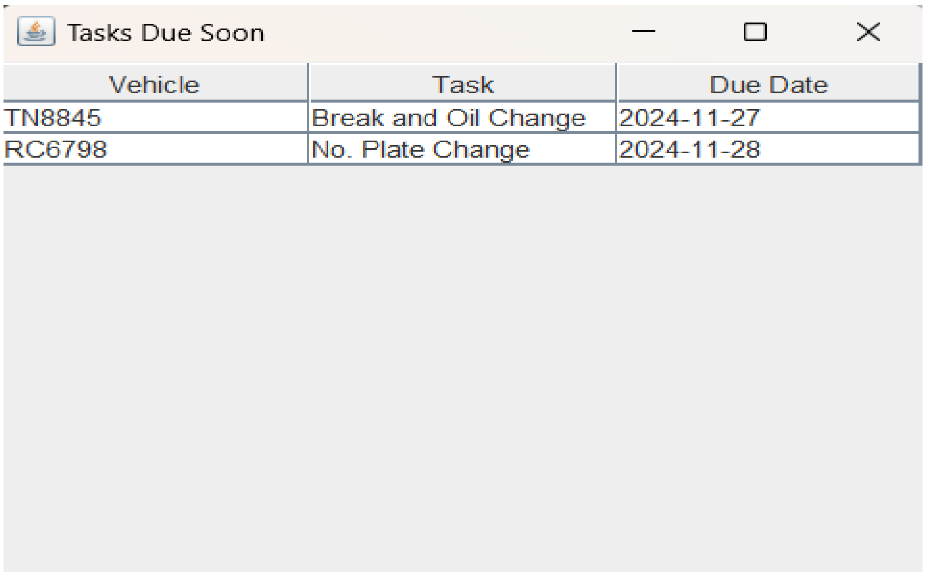
3. The user can scroll through the table to view all tasks in the system.


All Scheduled Maintenance Tasks

Vehicle	Task	Due Date
TN8845	Break and Oil Change	2024-11-27
VC2345	Car Wash	2024-11-23
RC6798	No. Plate Change	2024-11-28
PO9012	Front Glass Change	2024-11-30

Step 5: Viewing Tasks Due Soon

1. Next click the **Show Due Tasks** button.
2. A table appears, displaying tasks with due dates within the next three days. The columns include:
   - **Vehicle Registration Number**
   - **Task Description**
   - **Due Date**
3. If no tasks are near the deadline, the table remains empty or shows a message like *"No tasks are due soon."*.


Tasks Due Soon

Vehicle	Task	Due Date
TN8845	Break and Oil Change	2024-11-27
RC6798	No. Plate Change	2024-11-28

REFERENCES

1. **Deitel, H. M., Deitel, P. J., & Nieto, T. R.**, "Java How to Program," Pearson, 11th ed., pp. 12-60, 2017. ISBN-13: 978-0134743354.
2. **Herbert Schildt**, "Java: The Complete Reference," McGraw-Hill Education, 10th ed., pp. 30-75, 2018. ISBN-13: 978-1260440232.
3. **McMillan, L.**, "Java Programming," Pearson, 4th ed., pp. 100-135, 2013. ISBN-13: 978-0133250564.
4. **N. S. Kumbhar, S. V. Kumbhar, and V. V. Gudivada**, "Software Engineering: A Practitioner's Approach to Software Systems Development," CRC Press, 1st ed., pp. 220-240, 2018. ISBN-13: 978-1138030749.
5. **M. S. P. S. P. S. B. Jain, R. K. Gupta**, "Programming with Java: A Primer," Tata McGraw-Hill, 5th ed., pp. 54-102, 2017. ISBN-13: 978-1259008237.