M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

"Vehicle Maintenance System"

Submitted in partial fulfillment of requirements for the award of the course

of

CGB1201 – JAVA PROGRAMMING

Under the guidance of

Mrs. I. Karthika M.E.,

Assistant Professor / CSE

Submitted By

KEERTHANA S (927622BCS051)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

DECEMBER 2024

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

Certified that this project report on **"Vehicle Maintenance System"** is the bonafide work of **KEERTHANA S (927622BCS051)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Mrs. I. KARTHIKA M.E.,

SUPERVISOR,

Department of Computer Science and Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Signature

Dr. D. PRADEEP M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Electronics and Communication Engineering,

M.Kumarasamy College of Engineering,

Thalavapalayam, Karur -639 113.

Submitted for the End Semester Practical Examination held on_____

M.Kumarasamy College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To achieve education and research excellence in Computer Science and Engineering

MISSION OF THE DEPARTMENT

- To excel in academic through effective teaching learning techniques
- To promote research in the area of computer science and engineering with the focus on innovation
- To transform students into technically competent professionals with societal and ethical responsibilities

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Graduates will have successful career in software industries and R&D divisions through continuous learning.

PEO 2: Graduates will provide effective solutions for real world problems in the key domain of computer science and engineering and engage in lifelong learning.

PEO 3: Graduates will excel in their profession by being ethically and socially responsible.

PROGRAM OUTCOMES

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.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1: Professional Skills:** Ability to apply the knowledge of computing techniques to design and develop computerized solutions for the problems.
- **PSO2: Successful career:** Ability to utilize the computing skills and ethical values in creating a successful career.

ABSTRACT

The **Vehicle Maintenance System** helps vehicle owners and fleet managers track and schedule maintenance tasks. It sends reminders for upcoming services, stores records of past maintenance, and connects users with service providers like mechanics. This system ensures vehicles are maintained on time, reducing downtime and extending their lifespan.

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	POs MAPPED	PSOs MAPPED
The **Vehicle Maintenance System** helps vehicle owners and fleet managers track and schedule maintenance tasks. It sends reminders for upcoming services, stores records of past maintenance, and connects users with service providers like mechanics. This system ensures vehicles are maintained on time, reducing downtime and extending their lifespan.	**PO1 [1]** **PO2 [3]** **PO3 [1]** **PO4 [1]** **PO7 [1]** **PO9 [2]** **PO10 [3]** **PO11 [1]** **PO12 [2]**	**PSO1 [2]** **PSO2 [1]**

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

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of the Vehicle Maintenance System is to develop an efficient, user-friendly, and scalable solution to streamline vehicle maintenance operations. The system is designed to simplify key processes, including vehicle tracking, maintenance task scheduling, service provider management, and user role administration. By automating maintenance scheduling and sending timely reminders, the system aims to reduce vehicle downtime and improve operational efficiency. Additionally, it provides an intuitive interface that enhances the user experience, enabling easy access to reports and notifications. The system also supports scalability, allowing for future enhancements and the integration of additional features as the needs of the users grow. Ultimately, the Vehicle Maintenance System seeks to transform manual maintenance tracking into a modern, digital solution, optimizing resources and ensuring the reliability of vehicles over time.

1.2 Overview

The Vehicle Maintenance System is a comprehensive software solution designed to manage and streamline various aspects of vehicle maintenance. It provides a user-friendly interface for vehicle tracking, maintenance task scheduling, and management of service providers. The system allows users to easily register, view, and manage vehicles, schedule maintenance tasks, and receive timely notifications about upcoming services. It also supports adding and managing users with different roles, such as Admin and User, ensuring role-based access control. The system generates detailed reports on maintenance activities, helping users monitor vehicle health and make informed decisions. Built using Java's AWT framework, the system is modular, allowing for future expansions and enhancements.

1.3 Java-programming Concepts

Object-Oriented Programming (OOP): The system is designed using the core principles of Object-Oriented Programming (OOP), where real-world entities like users, vehicles, and maintenance tasks are represented as classes and objects. Each class encapsulates the attributes and behaviors of the entity it represents, making the code modular, reusable, and easy to maintain.

Encapsulation: Encapsulation is used to protect the data within each class. By making attributes private and providing public getter and setter methods, the internal state of an object can be safely accessed and modified. This ensures that the data is controlled and manipulated only through defined methods, reducing the risk of data corruption and maintaining the integrity of the system.

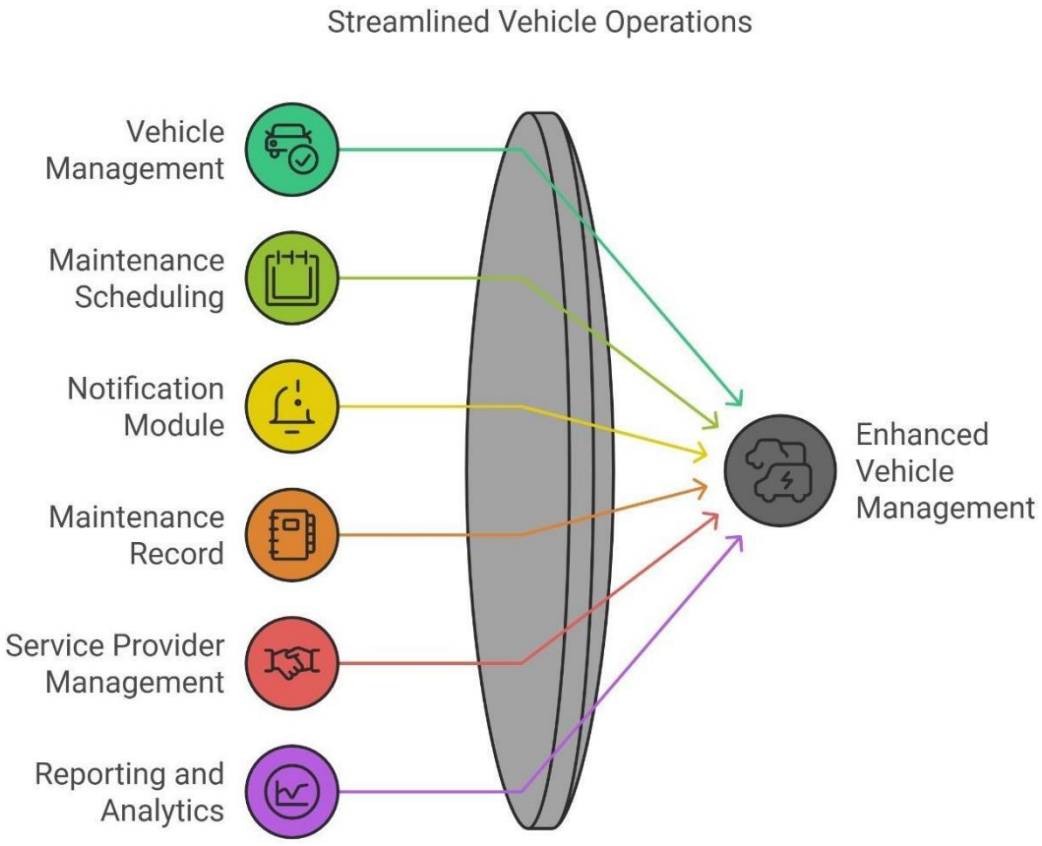
Collections Framework: The system utilizes the Collections Framework to store and manage data efficiently. Lists, like Array List, are used to manage dynamic groups of objects such as vehicles, tasks, and users. This allows the system to handle variable amounts of data without unnecessary complexity, and it enables easy manipulation and access to the data through iteration and collection operations.

Exception Handling: The system is designed with exception handling to catch and manage errors gracefully. This is especially important when dealing with user input, such as invalid dates or incorrect vehicle IDs. By using try-catch blocks, the system can prevent crashes and instead inform the user of the error, allowing them to correct their input and continue using the system without interruption

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work



2.2 Block Diagram

CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1- Management Module:

This module handles user-related tasks like registering new users, managing their login credentials, and assigning roles (e.g., Admin or User). It ensures secure and easy access to the system.

3.2 Module 2- Vehicle Management Module:

This module allows users to add, view, and manage details of vehicles, such as vehicle ID, make, and model. It helps in keeping an organized record of all vehicles

3.3 Module 3 - Maintenance Scheduling Module:

This module is used to create and track maintenance tasks for vehicles. It allows users to schedule tasks with details like the task description, vehicle ID, and due dates.

3.4 Module 4 – Notification and Reminder Module

This module sends alerts to users about upcoming maintenance tasks or important updates. It ensures that no maintenance task is missed.

3.5 Module 5 - Reporting and Analytics Module:

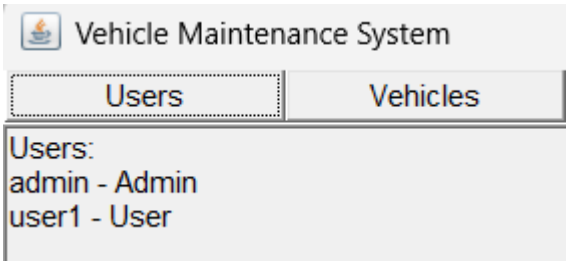
This module generates reports about maintenance tasks, vehicle usage, and other key data. It provides insights and analytics to improve decision-making

CHAPTER 4

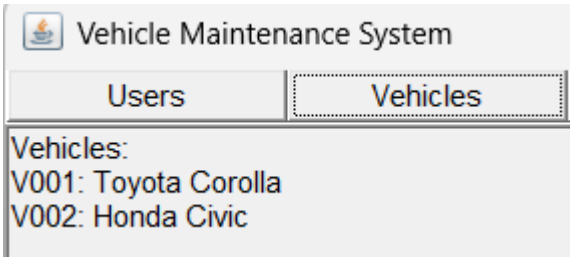
RESULTS AND DISCUSSION

RESULTS:

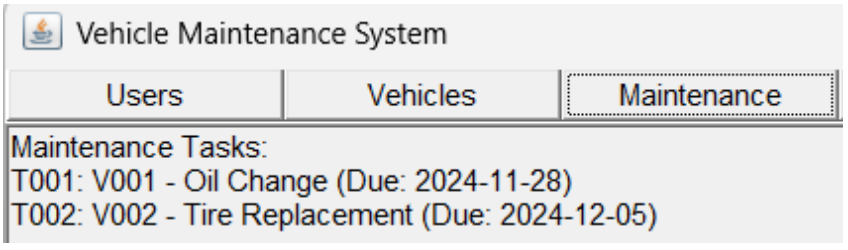
1. User Management



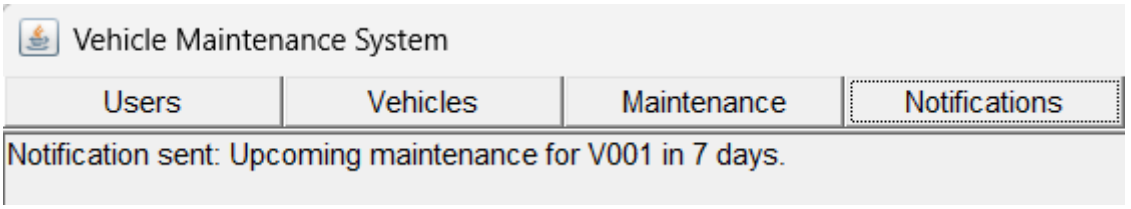
2. Vehicle Management



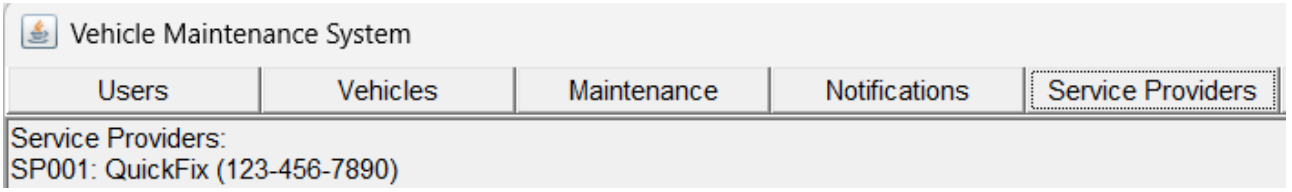
3 Maintenance Module



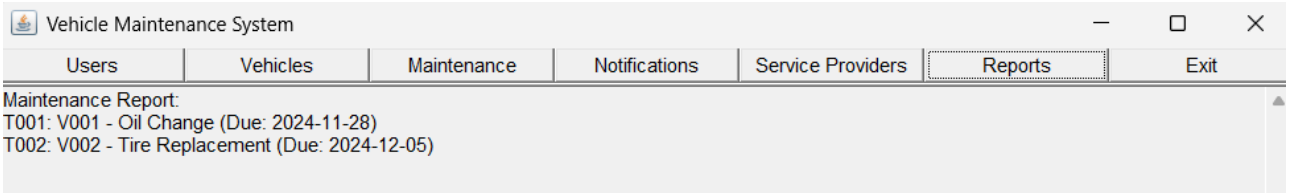
Notification and Reminder Module



5. Service Provider Management



6. Generating Report:



CHAPTER 5

CONCLUSION

The Vehicle Maintenance System is an efficient and user-friendly solution designed to streamline the management of vehicles, maintenance schedules, service providers, and user roles. By incorporating key functionalities such as user registration, task scheduling, notification services, and detailed reporting, the system ensures timely maintenance and reduces vehicle downtime. It demonstrates the application of object-oriented programming and modular design principles, offering a scalable and organized framework. This project not only addresses real-world challenges in vehicle maintenance but also provides opportunities for future enhancements, such as database integration, cloud-based services, and real-time analytics. Overall, the Vehicle Maintenance System is a practical and innovative approach to simplifying vehicle upkeep, benefiting individuals and businesses alike.

REFERENCES:

Java Programming Documentation

- Oracle. (n.d.). Java Platform, Standard Edition Documentation. Retrieved from https://docs.oracle.com/javase/

Object-Oriented Design Principles

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

Graphical User Interface Development

- Schildt, H. (2018). Java: The Complete Reference. McGraw Hill Education.

Java AWT and Event Handling

- GeeksforGeeks. (n.d.). Introduction to AWT in Java. Retrieved from https://www.geeksforgeeks.org/
- Sommerville, I. (2010). Software Engineering (9th Edition). Pearson Education.
- **Real-World Maintenance Management**
- Campbell, J. D., & Reyes-Picknell, J. V. (2015). Uptime: Strategies for Excellence in Maintenance Management. Productivity Press.

APPENDIX

```
package VEHICLE_MAINTANANCE_SYSTEM;
import java.awt.*;
import java.awt.event.*;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

// User Management Module
class User {
    private String username;
    private String password;
    private String role;

    public User(String username, String password, String role) {
        this.username = username;
        this.password = password;
        this.role = role;
    }

    public String getUsername() {
        return username;
    }

    public String getRole() {
        return role;
    }

    public boolean validatePassword(String inputPassword) {
        return this.password.equals(inputPassword);
    }
}

class UserManager {
```

```
    private List<User> users = new ArrayList<>();

    public void registerUser(String username, String password, String role) {
        users.add(new User(username, password, role));
    }

    public User login(String username, String password) {
        for (User user : users) {
            if (user.getUsername().equals(username) && user.validatePassword(password)) {
                return user;
            }
        }
        return null;
    }

    public List<User> getUsers() {
        return users;
    }
}

// Vehicle Management Module
class Vehicle {
    private String vehicleId;
    private String make;
    private String model;

    public Vehicle(String vehicleId, String make, String model) {
        this.vehicleId = vehicleId;
        this.make = make;
        this.model = model;
    }

    public String getDetails() {
        return vehicleId + ": " + make + " " + model;
    }
}
```

```
class VehicleManager {
    private List<Vehicle> vehicles = new ArrayList<>();

    public void addVehicle(String vehicleId, String make, String model) {
        vehicles.add(new Vehicle(vehicleId, make, model));
    }

    public List<Vehicle> getVehicles() {
        return vehicles;
    }
}

// Maintenance Scheduling Module
class MaintenanceTask {
    private String taskId;
    private String vehicleId;
    private String description;
    private LocalDate dueDate;

    public MaintenanceTask(String taskId, String vehicleId, String description, LocalDate dueDate) {
        this.taskId = taskId;
        this.vehicleId = vehicleId;
        this.description = description;
        this.dueDate = dueDate;
    }

    public String getDetails() {
        return taskId + ": " + vehicleId + " - " + description + " (Due: " + dueDate + ")";
    }
}

class MaintenanceScheduler {
    private List<MaintenanceTask> tasks = new ArrayList<>();

    public void scheduleTask(String taskId, String vehicleId, String description, LocalDate dueDate) {
```

```
            tasks.add(new MaintenanceTask(taskId, vehicleId, description, dueDate));
        }

        public List<MaintenanceTask> getTasks() {
            return tasks;
        }
}

// Notification and Reminder Module
class NotificationService {
        public void sendReminder(String message) {
            System.out.println("Reminder: " + message);
        }
}

// Service Provider Management Module
class ServiceProvider {
        private String providerId;
        private String name;
        private String contact;

        public ServiceProvider(String providerId, String name, String contact) {
            this.providerId = providerId;
            this.name = name;
            this.contact = contact;
        }

        public String getDetails() {
            return providerId + ": " + name + " (" + contact + ")";
        }
}

class ServiceProviderManager {
        private List<ServiceProvider> providers = new ArrayList<>();

        public void addServiceProvider(String providerId, String name, String contact) {
```

```
        providers.add(new ServiceProvider(providerId, name, contact));
    }

    public List<ServiceProvider> getProviders() {
        return providers;
    }
}

// Reporting and Analytics Module
class ReportGenerator {
    public String generateReport(List<MaintenanceTask> tasks) {
        StringBuilder report = new StringBuilder("Maintenance Report:\n");
        for (MaintenanceTask task : tasks) {
            report.append(task.getDetails()).append("\n");
        }
        return report.toString();
    }
}

// Main Application
public class VehicleMaintenanceSystemAWT extends Frame {
    private UserManager userManager = new UserManager();
    private VehicleManager vehicleManager = new VehicleManager();
    private MaintenanceScheduler scheduler = new MaintenanceScheduler();
    private ServiceProviderManager serviceProviderManager = new ServiceProviderManager();
    private NotificationService notificationService = new NotificationService();
    private ReportGenerator reportGenerator = new ReportGenerator();

    private TextArea outputArea;

    public VehicleMaintenanceSystemAWT() {
        setTitle("Vehicle Maintenance System");
        setSize(800, 600);
        setLayout(new BorderLayout());

        // Navigation Panel
```

```
Panel navigationPanel = new Panel();
navigationPanel.setLayout(new GridLayout(1, 7));
Button usersButton = new Button("Users");
Button vehiclesButton = new Button("Vehicles");
Button tasksButton = new Button("Maintenance");
Button notificationsButton = new Button("Notifications");
Button serviceProvidersButton = new Button("Service Providers");
Button reportsButton = new Button("Reports");
Button exitButton = new Button("Exit");

navigationPanel.add(usersButton);
navigationPanel.add(vehiclesButton);
navigationPanel.add(tasksButton);
navigationPanel.add(notificationsButton);
navigationPanel.add(serviceProvidersButton);
navigationPanel.add(reportsButton);
navigationPanel.add(exitButton);

// Output Area
outputArea = new TextArea();
outputArea.setEditable(false);

add(navigationPanel, BorderLayout.NORTH);
add(outputArea, BorderLayout.CENTER);

// Event Handlers
usersButton.addActionListener(e -> showUsers());
vehiclesButton.addActionListener(e -> showVehicles());
tasksButton.addActionListener(e -> showMaintenanceTasks());
notificationsButton.addActionListener(e -> sendNotification());
serviceProvidersButton.addActionListener(e -> showServiceProviders());
reportsButton.addActionListener(e -> generateReport());
exitButton.addActionListener(e -> System.exit(0));

// Sample Data
setupSampleData();
```

```
        // Window Close Action
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

        setVisible(true);
    }

    private void setupSampleData() {
        userManager.registerUser("admin", "admin123", "Admin");
        userManager.registerUser("user1", "password1", "User");

        vehicleManager.addVehicle("V001", "Toyota", "Corolla");
        vehicleManager.addVehicle("V002", "Honda", "Civic");

        scheduler.scheduleTask("T001", "V001", "Oil Change", LocalDate.now().plusDays(7));
        scheduler.scheduleTask("T002", "V002", "Tire Replacement", LocalDate.now().plusDays(14));

        serviceProviderManager.addServiceProvider("SP001",  "QuickFix",  "123-456-7890");
    }

    private void showUsers() {
        StringBuilder usersOutput = new StringBuilder("Users:\n");
        for (User user : userManager.getUsers()) {
            usersOutput.append(user.getUsername()).append(" - ").append(user.getRole()).append("\n");
        }
        outputArea.setText(usersOutput.toString());
    }

    private void showVehicles() {
        StringBuilder vehiclesOutput = new StringBuilder("Vehicles:\n");
        for (Vehicle vehicle : vehicleManager.getVehicles()) {
            vehiclesOutput.append(vehicle.getDetails()).append("\n");
```

```
            }
            outputArea.setText(vehiclesOutput.toString());
        }

        private void showMaintenanceTasks() {
            StringBuilder tasksOutput = new StringBuilder("Maintenance Tasks:\n");
            for (MaintenanceTask task : scheduler.getTasks()) {
                tasksOutput.append(task.getDetails()).append("\n");
            }
            outputArea.setText(tasksOutput.toString());
        }

        private void sendNotification() {
            notificationService.sendReminder("Upcoming maintenance for V001 in 7 days.");
            outputArea.setText("Notification sent: Upcoming maintenance for V001 in 7 days.");
        }

        private void showServiceProviders() {
            StringBuilder providersOutput = new StringBuilder("Service Providers:\n");
            for (ServiceProvider provider : serviceProviderManager.getProviders()) {
                providersOutput.append(provider.getDetails()).append("\n");
            }
            outputArea.setText(providersOutput.toString());
        }

        private void generateReport() {
            outputArea.setText(reportGenerator.generateReport(scheduler.getTasks()));
        }

        public static void main(String[] args) {
            new VehicleMaintenanceSystemAWT();
        }
    }
```