VEHICLE MAINTENANCE SYSTEM

A PROJECT REPORT

Submitted by

VIJAYA PRAKASH G (2303811710421176)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **“VEHICLE MAINTENANCE SYSTEM”** is the bonafide work of **VIJAYA PRAKASH G (2303811710421176)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
SUPERVISOR
ASSISTANTP ROFESSOR

[illegible]

SIGNATURE

Mr. A. Malarmannan, M.E.,

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on – 06/12/2024

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Mr.R. KARTHIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

[illegible]

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 **CGB1201 - JAVA PROGRAMMING.**

.

Signature

[illegible]

VIJAYA PRAKASH G

Place: Samayapuram

Date: 06/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (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. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, 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. A. MALARMANNAN, M.E.,** 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 serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

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

This project Vehicle Maintenance System is a robust software solution designed to assist vehicle owners and fleet managers in efficiently managing the maintenance and repair processes for their vehicles. This system is aimed at simplifying the often complex task of ensuring that vehicles receive timely service and repairs, reducing the risk of costly breakdowns and unscheduled downtime. By automating key aspects of vehicle maintenance, the system helps users track and schedule maintenance tasks, ensuring that vehicles are always in optimal working condition. A core feature of the system is its ability to generate reminders for upcoming services, such as oil changes, tire rotations, and inspections, based on predefined intervals or mileage. These timely alerts reduce the likelihood of missed or delayed services, thereby increasing the longevity and reliability of vehicles. This proactive approach not only improves vehicle performance but also enhances safety by addressing potential issues before they lead to more severe problems. The system also stores comprehensive records of all maintenance activities performed on each vehicle, including repairs, part replacements, and service costs. This detailed history serves as an invaluable resource for fleet managers, providing insights into the performance and condition of each vehicle. By having all maintenance data in one centralized location, the system facilitates better decision-making and budgeting for future repairs or replacements.

ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
It is a Java-based application designed to help vehicle owners and fleet managers efficiently track, schedule and manage vehicle maintenance and repairs.I ensures that all maintenance tasks are completed on time by providing reminders for upcoming services and stores detailed records of maintenance activities for each vehicle. This system streamlines maintenance management, reduces downtime, and improves vehicle reliability through timely service reminders and comprehensive service history tracking.	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

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

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Objective

- **Track and Schedule Maintenance:** The system allows vehicle owners and fleet managers to schedule and monitor regular maintenance tasks, ensuring timely service for each vehicle.
- **Timely Reminders**: The system provides automated reminders for upcoming services, such as oil changes, tire rotations, and inspections, reducing the risk of missed maintenance.
- **Record Maintenance Activities:** It stores detailed records of all maintenance and repair activities, including costs, service dates, and parts replaced, for easy reference and analysis.
- **Improve Vehicle Longevity and Performance:** By keeping track of all maintenance tasks and ensuring they are completed on time, the system helps maintain vehicle reliability, reduce downtime, and extend the lifespan of the vehicles.

1.2 Overview

The **Vehicle Maintenance System** is a comprehensive tool developed to assist vehicle owners and fleet managers in tracking, scheduling, and managing the maintenance and repair needs of their vehicles. The system provides a centralized platform for scheduling routine maintenance tasks, such as oil changes, tire rotations, and brake checks, ensuring that each vehicle is properly serviced at the right time. With automated reminders and alerts for upcoming services, the system helps users stay on top of maintenance schedules and avoid missed services that could lead to unexpected breakdowns. The Vehicle Maintenance System is a software solution designed to help vehicle owners and fleet managers efficiently track,vehicles.

1.3 Java Programming Concepts

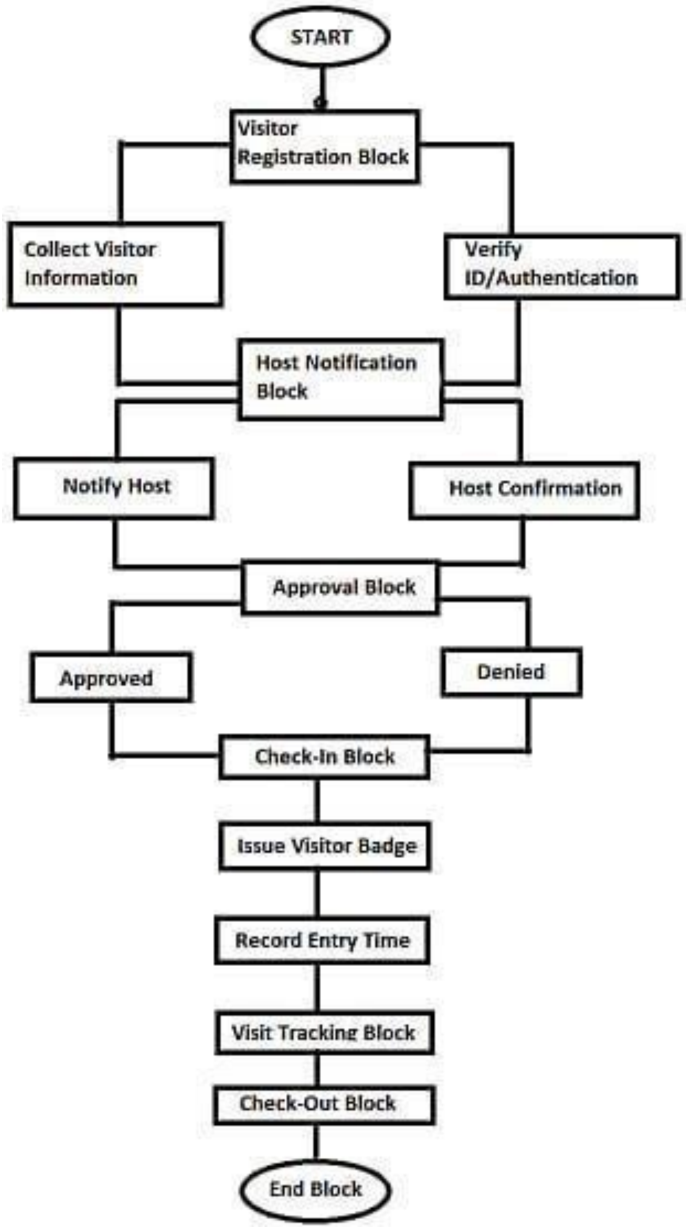
- **Classes and Objects:** The Vehicle Maintenance System uses classes such as Vehicle, MaintenanceTask, and ServiceRecord, where each class represents an entity or concept within the system. Objects of these classes are created to store and manage data for individual vehicles, maintenance tasks, and service history.
- **Encapsulation**: The system encapsulates data related to vehicles and their maintenance, such as make, model, service dates, and repair details, within classes. This ensures that the data is only accessible through well-defined methods, maintaining privacy and security.
- **Inheritance**: The system may use inheritance to extend base classes, such as creating specialized vehicle types (e.g., Car, Truck) that inherit common properties and methods from a Vehicle superclass. This allows for code reusability and easier management of different vehicle types.
- **Polymorphism**: Polymorphism allows the system to use methods in different ways based on the object type. For instance, different types of vehicles may have unique maintenance schedules, but a common scheduleService() method can be used for all vehicle types, with behavior tailored to each type through method overriding.
- **Abstraction**: The system uses abstraction to hide the complex implementation details of certain operations, such as calculating the next service date or generating service reminders, while exposing simple interfaces for users to interact with.

CHAPTER 2
PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work involves a comprehensive **Vehicle Information Management System** streamlines vehicle maintenance and management by storing essential details like make, model, VIN, and service history while allowing users to register multiple vehicles. It automates maintenance scheduling with reminders, tracks completed repairs, and provides detailed cost and performance reports. The system supports fleet management, parts inventory tracking, secure data storage with regular backups, and integration with service providers for seamless updates and invoicing. Future enhancements include predictive maintenance using AI and IoT-based real-time health monitoring, ensuring smarter decision-making and optimized costs.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Module 1: Vehicle Information Management

- Stores key details: make, model, VIN, mileage, and service history.
- Allows registration and management of multiple vehicles.
- Integrates with other modules like maintenance scheduling and logs.

3.2 Module 2: Maintenance Scheduling & Reminders

- Automates scheduling of maintenance tasks based on time or mileage.
- Sends automatic reminders via email, SMS, or app notifications.
- Customizable maintenance schedules for different tasks (e.g., oil change, tire rotation).

3.3 Module 3: Maintenance Log & Reporting

- Tracks and stores records of completed maintenance and repairs.
- Includes details like parts replaced, service dates, and costs.
- Generates detailed reports on maintenance history and costs.

3.4 Module 4: Fleet Management& Parts Tracking

- Manages maintenance schedules for multiple vehicles (fleet management).
- Prioritizes tasks and monitors vehicle health across the fleet.
- Tracks parts usage and inventory, sending reordering alerts when needed.

CHAPTER 4
CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The **Vehicle Maintenance System** is a comprehensive solution designed to streamline the management and tracking of vehicle maintenance and repairs for both individual vehicle owners and fleet managers. By automating key tasks such as scheduling, reminders, and record-keeping, the system ensures that maintenance activities are completed on time, reducing the risk of breakdowns and costly repairs. The system’s modular design allows for customization and scalability, catering to a wide range of vehicle types and fleet sizes. With features like automated maintenance schedules, real-time service reminders, and detailed logs of past services, the system not only improves operational efficiency but also helps extend the life of vehicles, minimize downtime, and optimize maintenance costs. Ultimately, the **Vehicle Maintenance System** enhances the overall management of vehicles, ensuring they remain in optimal condition and reducing the burden on owners and fleet managers.

4.2 FUTURE SCOPE

- Integration with IoT: Incorporate IoT sensors for real-time vehicle diagnostics, monitoring, and predictive maintenance alerts.
- AI-Powered Insights: Use machine learning to predict vehicle issues and optimize maintenance schedules.
- Cloud-Based Services: Enable cloud storage for maintenance records, allowing access from multiple devices.

CHAPTER 5 APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
// Vehicle class
class Vehicle
{String make;
String  model;
int year;
 ArrayList<ServiceRecord> serviceRecords;
 Vehicle(String make, String model, int year)
 {this.make = make;
 this.model = model;
 this.year = year;
 this.serviceRecords = new ArrayList<>();
 }
 void addServiceRecord(ServiceRecord record)
 {serviceRecords.add(record);
 }
 ArrayList<ServiceRecord> getServiceRecords()
 {return serviceRecords;
 }
}
// ServiceRecord class
class     ServiceRecord
{Date serviceDate;
```

```
 String description;
 double cost;
 ServiceRecord(Date serviceDate, String description, double cost)
 {this.serviceDate = serviceDate;
 this.description = description;this.cost = cost;
 }
 public String toString() {
 SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
 return "Date: " + sdf.format(serviceDate) + ", Description: " + description
+ ", Cost: $" + cost;
 }
}
// MaintenanceManager class
class     MaintenanceManager
{ArrayList<Vehicle> vehicles;
 SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
 MaintenanceManager() {
 vehicles = new ArrayList<>();
 }
 void addVehicle(String make, String model, int year)
 {vehicles.add(new Vehicle(make, model, year));
 }
 void logService(String make, String model, int year, String dateString, String
description, double cost) throws ParseException {
 Date serviceDate = dateFormat.parse(dateString);
 for (Vehicle vehicle : vehicles) {
 if (vehicle.make.equalsIgnoreCase(make) &&
vehicle.model.equalsIgnoreCase(model) && vehicle.year == year)
{vehicle.addServiceRecord(new ServiceRecord(serviceDate,
description, cost));
 return;
 }
 }
 System.out.println("Vehicle not found.");
 }
 void viewMaintenanceHistory(String make, String model, int year)
 {for (Vehicle vehicle : vehicles) {
 if (vehicle.make.equalsIgnoreCase(make) &&
vehicle.model.equalsIgnoreCase(model) && vehicle.year == year)
{System.out.println("Maintenance History for " + make + " " + model
+ " (" + year + "):");for (ServiceRecord record : vehicle.getServiceRecords())
 {System.out.println(record);
 }
 return;
 }
 }
 System.out.println("Vehicle not found.");
 }
 void calculateTotalCost(String make, String model, int year)
 {double totalCost = 0;
 for (Vehicle vehicle : vehicles) {
 if (vehicle.make.equalsIgnoreCase(make) &&
vehicle.model.equalsIgnoreCase(model) && vehicle.year == year)
{for (ServiceRecord record : vehicle.getServiceRecords())
{ totalCost += record.cost;
 }
 System.out.println("Total Maintenance Cost for " + make + " " +
model + " (" + year + "): $" + totalCost);
```

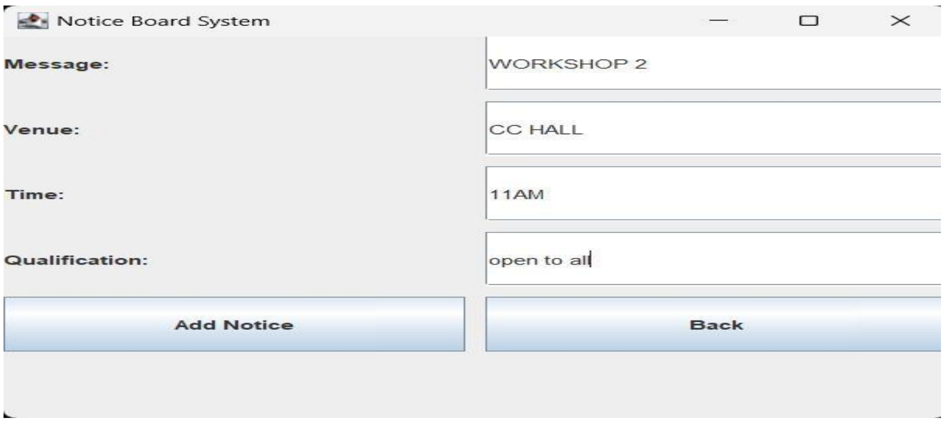
```
 return;
 }
 }
 System.out.println("Vehicle not found.");
 }
}
// MaintenanceApp class
public class MaintenanceApp {
 static MaintenanceManager manager = new MaintenanceManager();
 public static void main(String[] args) {
 Frame frame = new Frame("Vehicle Maintenance System");
 frame.setSize(400, 400);
 frame.setLayout(new GridLayout(6, 1));

 Button addVehicleButton = new Button("Add New Vehicle");
 Button logServiceButton = new Button("Log Service Record");
 Button viewHistoryButton = new Button("View Maintenance History");
Button calculateCostButton = new Button("Calculate Total Maintenance
Cost");
 Button exitButton = new Button("Exit");
 addVehicleButton.addActionListener(new ActionListener() {@Override
 public void actionPerformed(ActionEvent e) {
 Dialog dialog = new Dialog(frame, "Add New Vehicle", true);
 dialog.setLayout(new GridLayout(4, 2));
 TextField makeField = new TextField();
 TextField modelField = new TextField();
 TextField yearField = new TextField();

 dialog.add(new Label("Make:"));
 dialog.add(makeField);
 dialog.add(new Label("Model:"));
 dialog.add(modelField);
 dialog.add(new Label("Year:"));
 dialog.add(yearField);
 Button saveButton = new Button("Save");
 dialog.add(saveButton);
 saveButton.addActionListener(new ActionListener()
 {@Override
 public void actionPerformed(ActionEvent e)
 {String make = makeField.getText();
 String model = modelField.getText();
 int year = Integer.parseInt(yearField.getText());
 manager.addVehicle(make, model, year);
 dialog.setVisible(false);
 System.out.println("Vehicle added successfully!");
 }
 });
 dialog.setSize(300, 200);
 dialog.setVisible(true);
 }
 });
 logServiceButton.addActionListener(new ActionListener()
 {@Override
 public void actionPerformed(ActionEvent e) {
 Dialog dialog = new Dialog(frame, "Log Service Record", true);
 dialog.setLayout(new GridLayout(7, 2));
 TextField makeField = new TextField();
 TextField modelField = new TextField();
 TextField yearField = new TextField();
 TextField dateField = new TextField();
 TextField descField = new TextField();
 TextField costField = new TextField();dialog.add(new Label("Make:"));
 dialog.add(makeField);
 dialog.add(new Label("Model:"));
 dialog.add(modelField);
 dialog.add(new Label("Year:"));
 dialog.add(yearField);
 dialog.add(new Label("Service Date (dd-MM-yyyy):"));
 dialog.add(dateField);
 dialog.add(new Label("Description:"));
 dialog.add(descField);
 dialog.add(new Label("Cost:"));
 dialog.add(costField);
 Button saveButton = new Button("Save");
 dialog.add(saveButton);
 saveButton.addActionListener(new ActionListener()
 {@Override
 public void actionPerformed(ActionEvent e)
 {String make = makeField.getText();
 String model = modelField.getText();
 int year = Integer.parseInt(yearField.getText());
 String date = dateField.getText();
 String description = descField.getText();
 double cost = Double.parseDouble(costField.getText());
 try {
 manager.logService(make, model, year, date, description,
 cost);
```

```
 dialog.setVisible(false);
 System.out.println("Service record logged successfully!");
 } catch (ParseException ex) {
 System.out.println("Invalid date format. Please enter the date
in dd-MM-yyyy format.");
 }
 }
 });
 dialog.setSize(400, 300);
 dialog.setVisible(true);
 }
 });
 viewHistoryButton.addActionListener(new  ActionListener()
 {@Override
 public void actionPerformed(ActionEvent e) {Dialog dialog = new Dialog(frame, "View
Maintenance History",
true);
 dialog.setLayout(new GridLayout(4, 2));
 TextField makeField = new TextField();
 TextField modelField = new TextField();
 TextField yearField = new TextField();

 dialog.add(new Label("Make:"));
 dialog.add(makeField);
 dialog.add(new Label("Model:"));
 dialog.add(modelField);
 dialog.add(new Label("Year:"));
 dialog.add(yearField);
 Button viewButton = new Button("View");
```

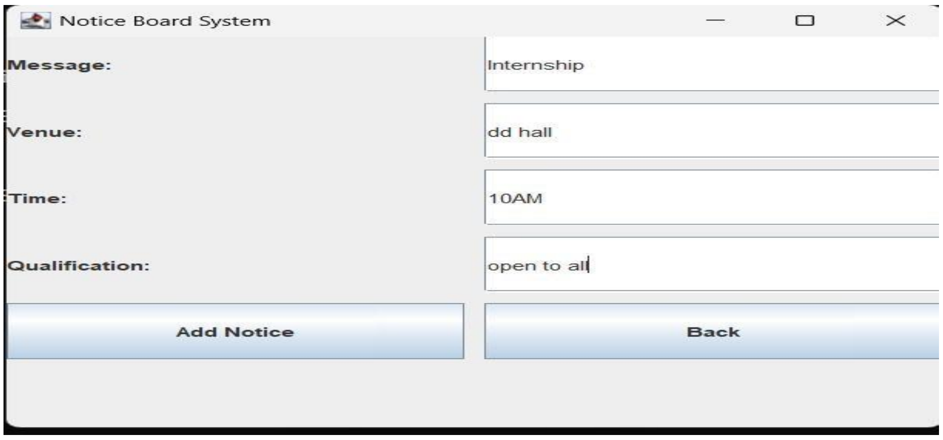
```
dialog.add(viewButton);
viewButton.addActionListener(new ActionListener()
{@Override
public void actionPerformed(ActionEvent e)
{String make = makeField.getText();
String model = modelField.getText();
int year = Integer.parseInt(yearField.getText());
manager.viewMaintenanceHistory(make, model, year);
dialog.setVisible(false);
}
});
dialog.setSize(300, 200);
dialog.setVisible(true);
}
});
calculateCostButton.addActionListener(new ActionListener()
{@Override
public void actionPerformed(ActionEvent e) {
Dialog dialog = new Dialog(frame, "Calculate Total Maintenance
Cost", true);
dialog.setLayout(new GridLayout(4, 2));
TextField makeField = new TextField();
TextField modelField = new TextField();
TextField yearField = new TextField();

dialog.add(new Label("Make:"));
dialog.add(makeField);
dialog.add(new Label("Model:"));dialog.add(modelField);
dialog.add(new Label("Year:"));
dialog.add(yearField);
Button calcButton = new Button("Calculate");
dialog.add(calcButton);
calcButton.addActionListener(new ActionListener()
{@Override
public void actionPerformed(ActionEvent e)
{String make = makeField.getText();
String model = modelField.getText();
int year = Integer.parseInt(yearField.getText());
manager.calculateTotalCost(make, model, year);
dialog.setVisible(false);
}
});
dialog.setSize(300, 200);
dialog.setVisible(true);
}
});
exitButton.addActionListener(new ActionListener()
{@Override
public void actionPerformed(ActionEvent e)
{System.out.println("Exiting...");
System.exit(0);
}
});
frame.add(addVehicleButton);
frame.add(logServiceButton);
frame.add(viewHistoryButton);
frame.add(calculateCostButton);
frame.add(exitButton);
```

```
 frame.setVisible(true);
 }
}
```

APPENDIX B
(SCREENSHOTS)



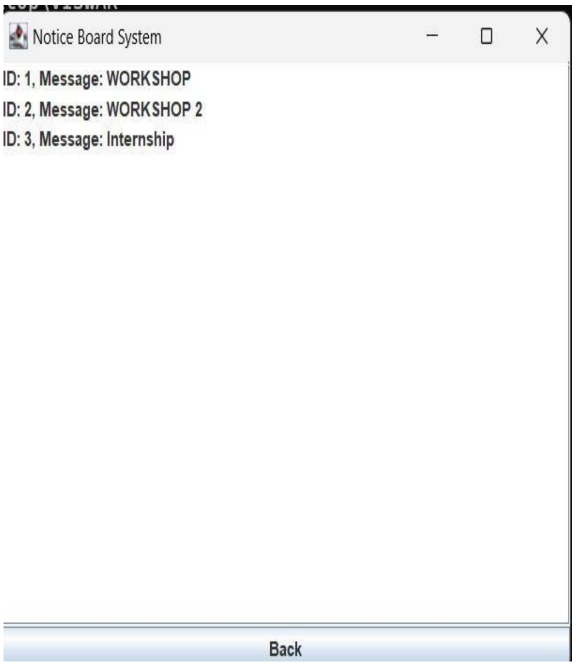
Message: WORKSHOP 2
Venue: CC HALL
Time: 11AM
Qualification: open to all



Message: Internship
Venue: dd hall
Time: 10AM
Qualification: open to all

1. GIVING INPUTS

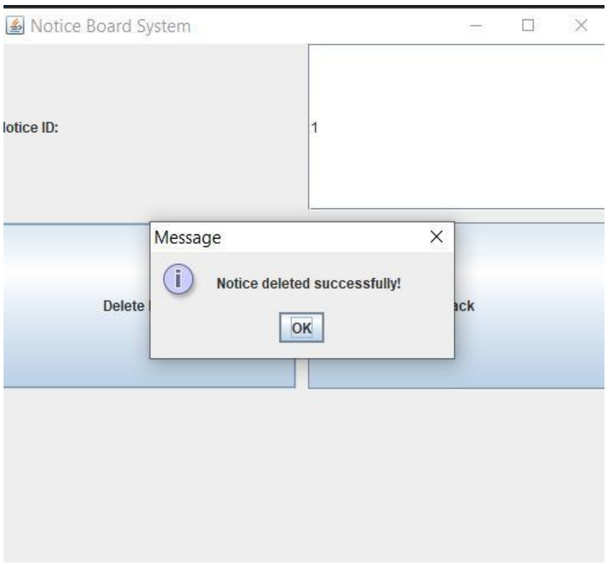
2. BEFORE UPDATE



Notice Board System

```
ID: 1, Message: WORKSHOP
ID: 2, Message: WORKSHOP 2
ID: 3, Message: Internship
```

4.DELETING NOTICES



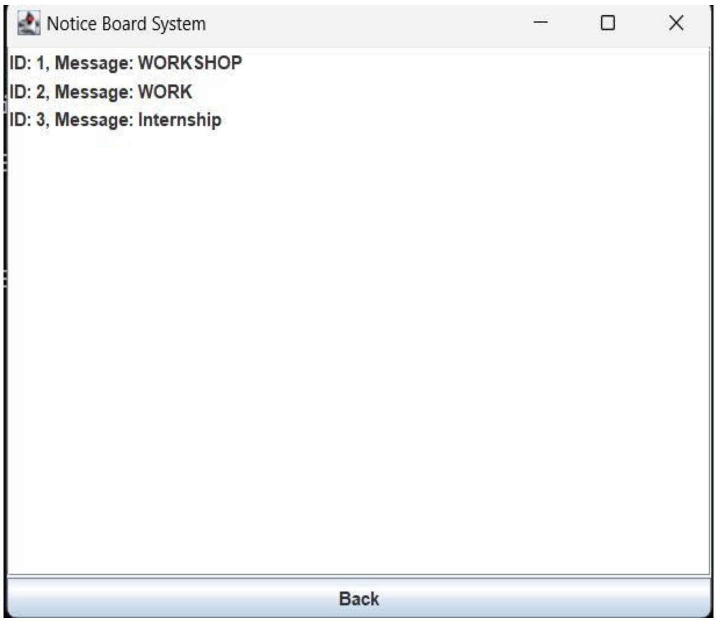
Notice Board System

lotice ID: 1

Message

Notice deleted successfully!

3. AFTER UPDATE



Notice Board System

```
ID: 1, Message: WORKSHOP
ID: 2, Message: WORK
ID: 3, Message: Internship
```

REFERENCES

1. Java Documentation Official Java Platform Documentation by Oracle https://docs.oracle.com/javase/
2. Oracle Java Documentation: Detailed reference for core Java classes and methods used in GUI and data handling.
3. GeeksforGeeks AWT Tutorial: A comprehensive guide on using AWT for building Java GUIs.