



VIT[®]
BHOPAL

Project Report

Car Care Companion

Submitted By- Vishnujith A Nair

Regd. NO.- 25MIM10110

Submitted To- Dr. AVR Mayuri

Course code- CSE 1021

1. Introduction

1.1 Project Overview

CarCare Companion is a comprehensive Python-based application designed to assist vehicle owners in maintaining their automobiles efficiently. The system integrates three crucial aspects of vehicle management: mileage optimization, problem diagnosis, and expense tracking into a single, user-friendly platform.

1.2 Problem Statement

Vehicle owners often face challenges in:

- Identifying car problems accurately
- Estimating repair costs.
- Tracking maintenance expenses
- Improving fuel efficiency
- Lack of integrated solutions for comprehensive vehicle management

1.3 Objectives

- Develop an integrated vehicle maintenance system
- Provide instant problem diagnosis and cost estimation
- Implement expense tracking with data persistence
- Offer practical mileage improvement tips
- Create an intuitive user interface

2. System Requirements

2.1 Hardware Requirements

- Processor: Intel i3 or equivalent
- RAM: 4GB minimum

- Storage: 500MB free space
- Operating System: Windows/Linux/MacOS

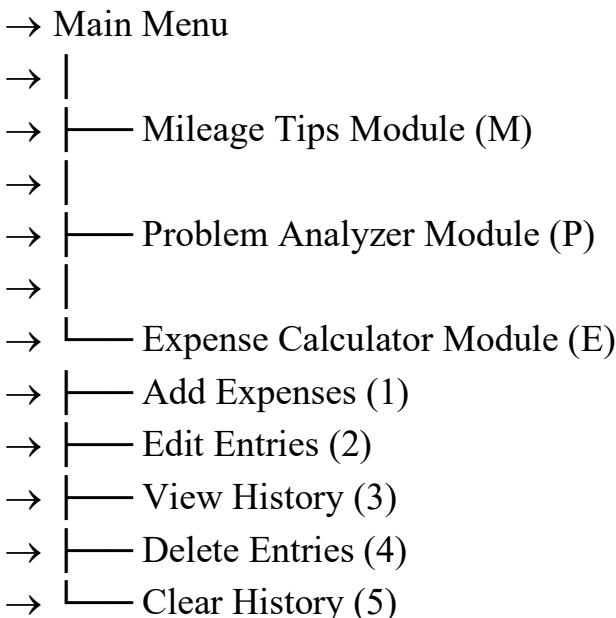
2.2 Software Requirements

- Python 3.6 or higher
- Standard Python libraries (datetime, file handling)
- Text file for data storage

3. System Design

3.1 Architecture

The system follows a modular menu-driven architecture with three main modules:



3.2 Module Description

3.2.1 Mileage Tips Module

Provides fuel efficiency improvement strategies

1. Drive Smoothly

2. Use the right gear
3. Maintain correct tyre pressure
4. Reduce extra weight
5. Regular Servicing

3.2.2 Problem Analyzer Module

Contains database of 10 common vehicle problems:

1. Engine Overheating
2. Battery Failure
3. Brake Problems
4. Low Mileage
5. Engine Misfiring
6. Tyre Issues
7. Clutch Problems
8. Suspension Problems
9. Starting Trouble
10. Fuel System Issues

3.2.3 Expense Calculator Module

- Complete CRUD operations with data persistence
- Automated calculations (total, average, highest expense)
- Date-wise expense recording

3.3 Data Storage

- File: expensehistory.txt

- Format: YYYY-MM-DD|Expense Name|₹Amount
- Advantages: Simple, human-readable, no database setup required

4 Implementation details

4.1 Core Technologies Used

i) Programming language: Python 3.x

ii) Key Libraries:

- datetime for timestamp generation
- File handling for data persistence

iii) Storage: Flat-file database system

4.2 Key Algorithm & Logics

4.2.1 Problem Matching Algorithm

python

```
if(problem=="Temperature gauge high" or problem=="steam from bonnet"):
    print("Issue = Engine Overheating")
```

#Additional Diagnosis logic

4.2.2 Expense Calculation Engine

python

```
total = total + amount
```

```
if (amount > highest):
```

```
    highest = amount
```

```
    highest_name = name
```

4.2.3 Data Management System

- Sequential file handling for persistence

- Index-based editing and deletion
- Date-stamped entries

5 Code Analysis

5.1 Strengths

1. User-Friendly Interface: Simple menu-driven navigation
2. Data Persistence: All expenses saved permanently in text file
3. Comprehensive Coverage: Handles multiple vehicle aspects
4. Cost Estimation: Provides realistic repair cost ranges (₹200 - ₹15,000)
5. Input Validation: Prevents invalid operations and errors

5.2 Technical Features

- Modular Design: Easy to maintain and extend
- Error Handling: Robust validation for user inputs
- Data Integrity: Confirmation for destructive operations
- Automatic Calculations: Real-time expense summaries

5.3 Limitations & Challenges

1. Text-based UI: No graphical user interface
2. Static Problem Database: Limited to pre-defined issues
3. Single File Storage: No backup mechanism
4. Regional Specificity: Indian Rupee based cost estimates

6. Testing & Results

6.1 Test Cases Executed

Module	Test Case	Result
--------	-----------	--------

Mileage Tips	Input 'M'	Success- displays all tips
Problem Analyzer	Input 'Car Won't Start'	Success- identifies battery issue
Expense Calculator	Add multiple expenses	Success- calculates totals correctly
File Operations	Edit/Delete entries	Success- maintains data integrity

6.2 Sample Outputs

Problem Diagnosis:

Enter your problem: screeching noise

Issue = Brake Problems

Causes = Worn brake pads, low brake oil

Estimate cost = ₹1,500 – ₹6,000

Expense Summary:

Your Expense Summary:

Total number of expense: 4

Total expenditure= ₹12,500

Highest money spent on Engine Repair is ₹6,000

Average expense= ₹3,125

7. Future Enhancements

7.1 Immediate Improvements (Next 6 Months)

1. Add graphical user interface using Tkinter
2. Implement data export to CSV/Excel
3. Expand problem database with 20+ additional issues

4. Add reminder system for regular maintenance

7.2 Advanced Features (Next 1-2 years)

1. Mobile application development (Android/iOS)

2. Cloud-based data synchronization

3. Integration with garage service databases

4. AI-based diagnostic suggestions

5. Multi-language support 6. Predictive maintenance alerts

8. Learning Outcomes

Through the development of CarCare Companion, I gained practical expertise in:

Technical Skills:

- Python programming and advanced file handling
- Menu-driven application architecture
- Data persistence and management techniques
- Input validation and error handling
- Algorithm design for problem-solving

Professional Skills:

- Project planning and execution
- Problem analysis and solution design
- Documentation and reporting
- Testing and debugging methodologies
- User-centric design approach

Domain Knowledge:

- Basic automotive maintenance principles

- Vehicle problem diagnosis patterns
- Expense management systems
- User interface design principles

9. Conclusion

CarCare Companion successfully demonstrates the practical application of programming concepts to solve real-world problems. The system effectively addresses the needs of vehicle owners by providing an integrated solution for maintenance management, problem diagnosis, and expense tracking. The project highlights the importance of user-friendly design, data persistence, and comprehensive feature sets in software development. The modular architecture ensures maintainability and provides a solid foundation for future enhancements. This project not only served as an excellent learning experience but also has practical utility for end-users, making vehicle maintenance more accessible and manageable.

10. References

1. Python Official Documentation - <https://docs.python.org/3/>
2. Vehicle Maintenance Guides - Automotive repair manuals
3. Software Engineering Principles - Academic course materials

Appendix

A. Source Code Structure

```
→ CarCare Companion
→ |
→ └── main.py (main program file)
```

```
→ └── expensehistory.txt (data storage)
→ └── project_report.pdf
→ └── README.txt (installation guide)
```

B. Installation & Execution Guide

1. Ensure Python 3.x is installed
2. Download main.py file
3. Run command: python main.py
4. Follow on-screen menu instructions
5. Data is automatically saved in expensehistory.txt

C. User Manual

Basic Operations:

- Press 'M' for Mileage Tips
- Press 'P' for Problem Diagnosis
- Press 'E' for Expense Management
- Follow numeric prompts for expense operations

Data Safety:

- All expenses automatically saved
- Confirmation required for delete operations
- Date-stamped entries for better tracking

