GUI Vehicle Fleet Management System

Workshop 2

Course: APD545 - Winter2026

Last Name: Maheswaran

First Name: Suganya

ID: 048298137

This assignment represents my own work in accordance with Seneca Academic Policy.

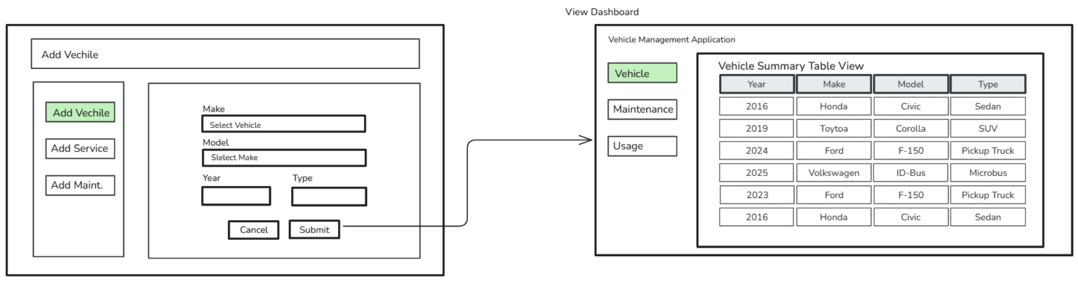
Date: Feb 17, 2026

Introduction

In this workshop, I designed and implemented a Vehicle Fleet Management System. The desktop application allows users to efficiently manage vehicle information, track maintenance schedules, and log vehicle usage. The system follows a multi-layered architecture based on the MVC design pattern with a Service Layer, ensuring a clear separation between the user interface, business logic, and data storage.

Wireframe

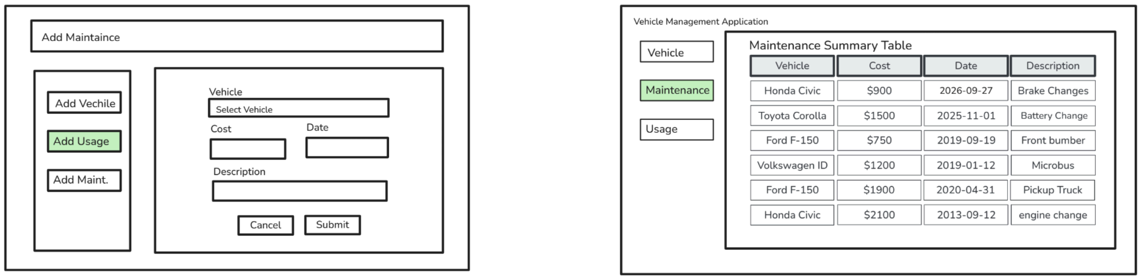
The following section presents the wireframes designed for the Vehicle Fleet Management System. These wireframes illustrate the layout, navigation flow, and key user interface elements of the main and summary windows, including forms for adding vehicles, maintenance records, and usage logs, as well as tables for viewing summaries. Due to time constraints and other commitments, these wireframes were not pre-approved by the user prior to development. They serve as a visual guide to the intended design and provide context for the implemented front-end interface.



Vehicle Summary Table View

Year	Make	Model	Type
2016	Honda	Civic	Sedan
2019	Toytoa	Corolla	SUV
2024	Ford	F-150	Pickup Truck
2025	Volkswagen	ID-Bus	Microbus
2023	Ford	F-150	Pickup Truck
2016	Honda	Civic	Sedan

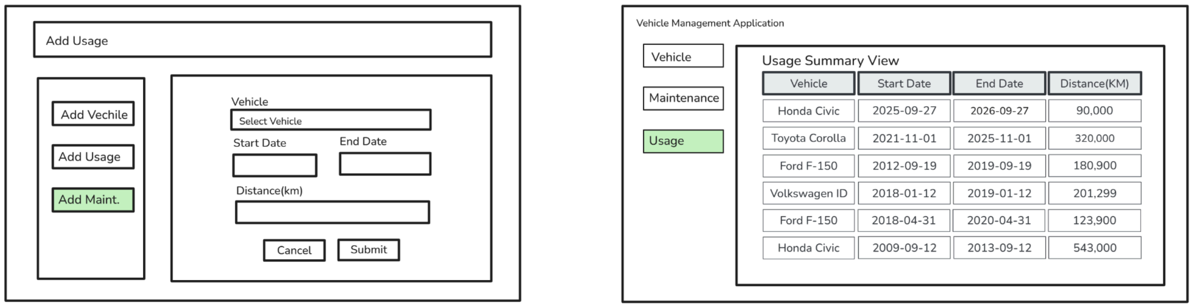
Figure 1 - Wireframe showing the Add Vehicle form alongside the Vehicle Summary view



Maintenance Summary Table

Vehicle	Cost	Date	Description
Honda Civic	$900	2026-09-27	Brake Changes
Toyota Corolla	$1500	2025-11-01	Battery Change
Ford F-150	$750	2019-09-19	Front bumber
Volkswagen ID	$1200	2019-01-12	Microbus
Ford F-150	$1900	2020-04-31	Pickup Truck
Honda Civic	$2100	2013-09-12	engine change

Figure 2 - Wireframe showing the Add Maintenance form alongside the Maintenance Summary view



Usage Summary View

Vehicle	Start Date	End Date	Distance(KM)
Honda Civic	2025-09-27	2026-09-27	90,000
Toyota Corolla	2021-11-01	2025-11-01	320,000
Ford F-150	2012-09-19	2019-09-19	180,900
Volkswagen ID	2018-01-12	2019-01-12	201,299
Ford F-150	2018-04-31	2020-04-31	123,900
Honda Civic	2009-09-12	2013-09-12	543,000

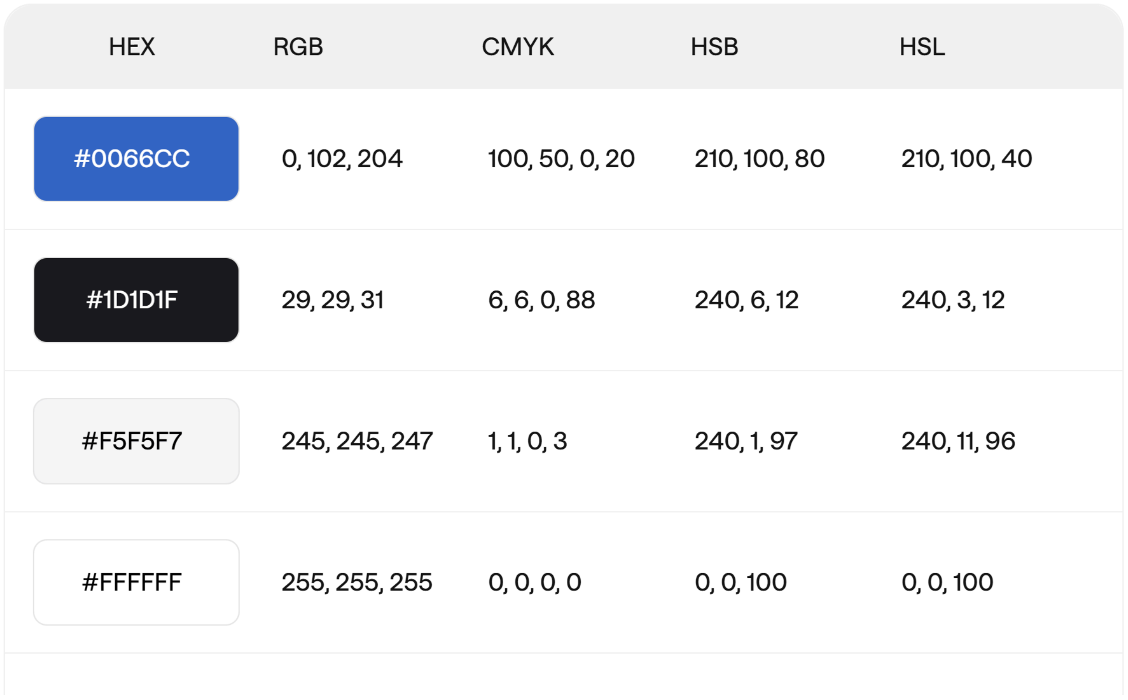
Figure 3 - Wireframe showing the Add Usage form alongside the Usage Summary view

Design and Colour Palette

For the VMUMS application, the design was inspired by Apple's clean and modern interface. The colour scheme was directly adopted from Apple's brand guidelines (source: Mobbin Apple Colors), ensuring a professional and visually appealing UI.

- **Primary Blue (#0066CC)**: Used for selected buttons, active elements to draw user attention.
- **Dark Gray/Black (#1D1D1F)**: Used for submit, cancel, unselcted buttons and text for high contrast.
- **Light Gray (#F5F5F7)**: Applied to form backgrounds and table headers to separate content without being visually overwhelming.
- **White (#FFFFFF)**: Used for table bodies and main form backgrounds to maintain clarity and readability.

This palette provides a clean, minimalist, and consistent interface, improving usability while giving the application a professional and modern look.



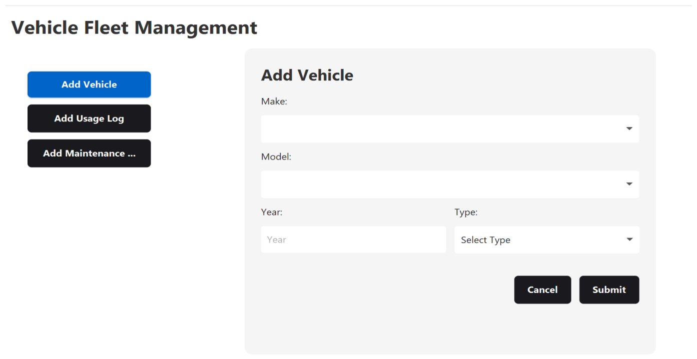
HEX	RGB	CMYK	HSB	HSL
#0066CC	0, 102, 204	100, 50, 0, 20	210, 100, 80	210, 100, 40
#1D1D1F	29, 29, 31	6, 6, 0, 88	240, 6, 12	240, 3, 12
#F5F5F7	245, 245, 247	1, 1, 0, 3	240, 1, 97	240, 11, 96
#FFFFFF	255, 255, 255	0, 0, 0, 0	0, 0, 100	0, 0, 100

Figure 4 – Colour Palette Used in UI

Application UI

This section highlights the final user interface and experience of the application. The screenshots show the key screens, workflows, and design decisions made to enhance usability and visual appeal.

1. Form Input View



Vehicle Fleet Management

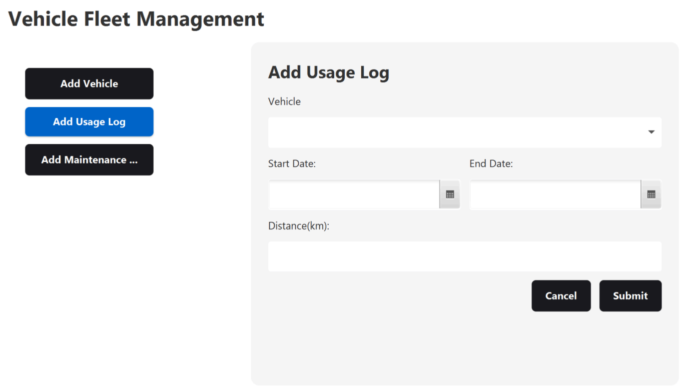
Add Vehicle

Make:

Model:

Year: Year

Type: Select Type



Vehicle Fleet Management

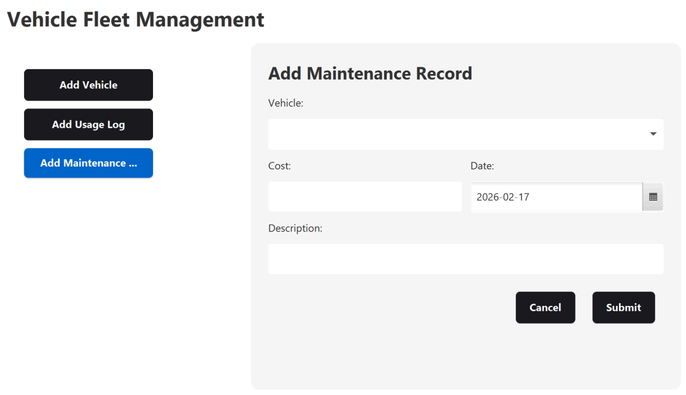
Add Usage Log

Vehicle

Start Date:

End Date:

Distance(km):



Vehicle Fleet Management

Add Maintenance Record

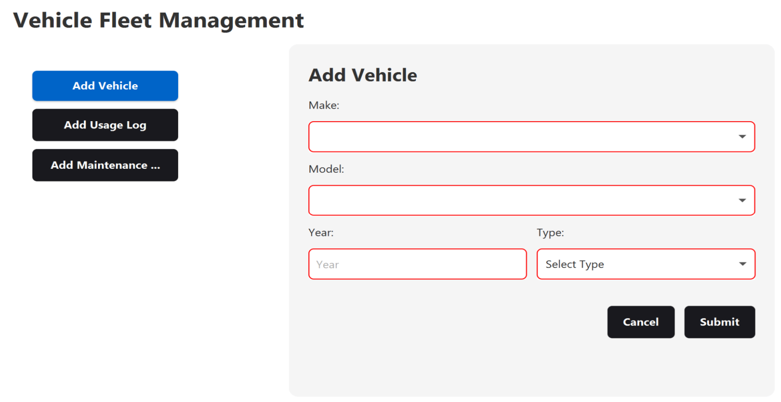
Vehicle:

Cost:

Date: 2026-02-17

Description:

2. Invalid Input View


Vehicle Fleet Management

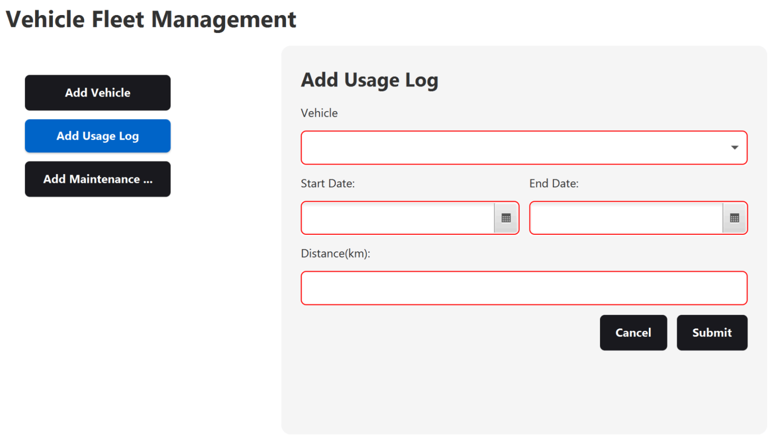
Add Vehicle

Make:

Model:

Year: Year

Type: Select Type


Vehicle Fleet Management

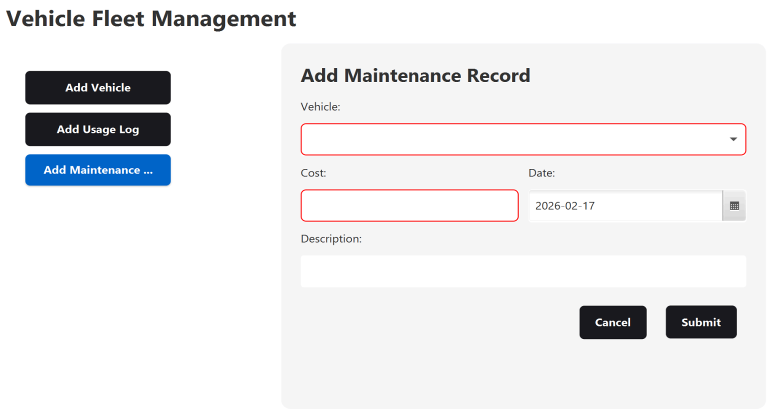
Add Usage Log

Vehicle

Start Date:

End Date:

Distance(km):


Vehicle Fleet Management

Add Maintenance Record

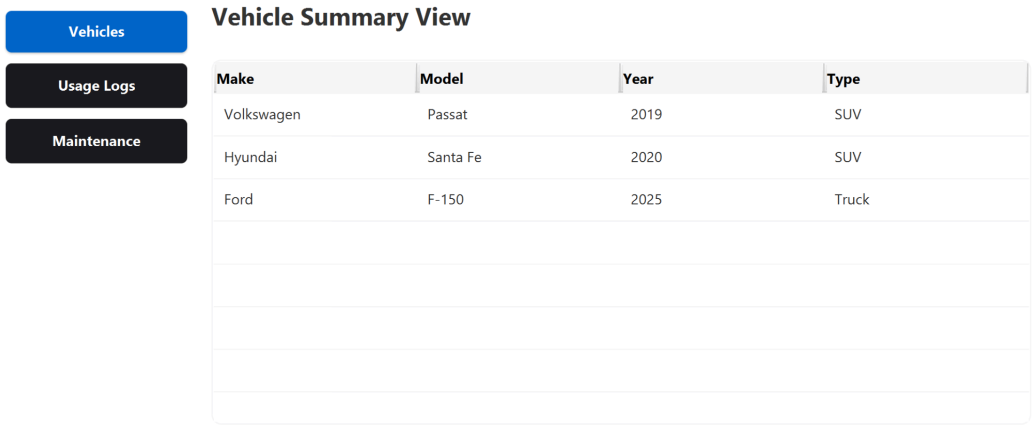
Vehicle:

Cost:

Date: 2026-02-17

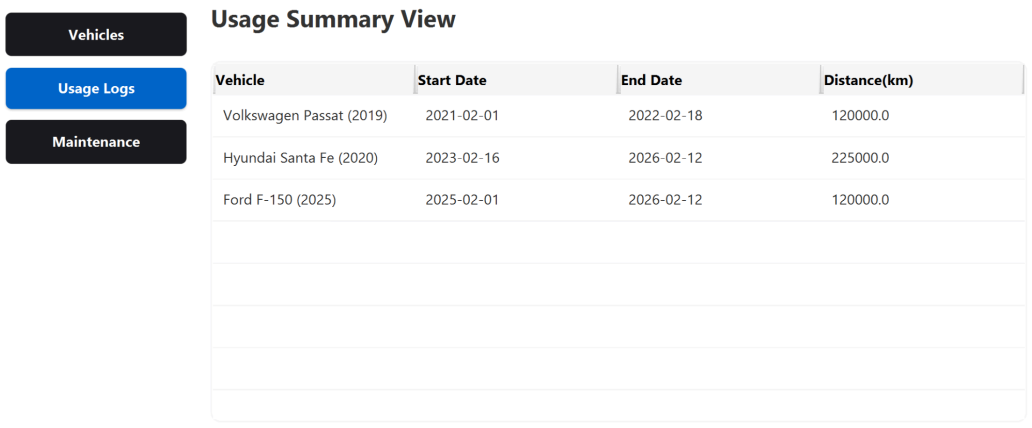
Description:

3. Table View



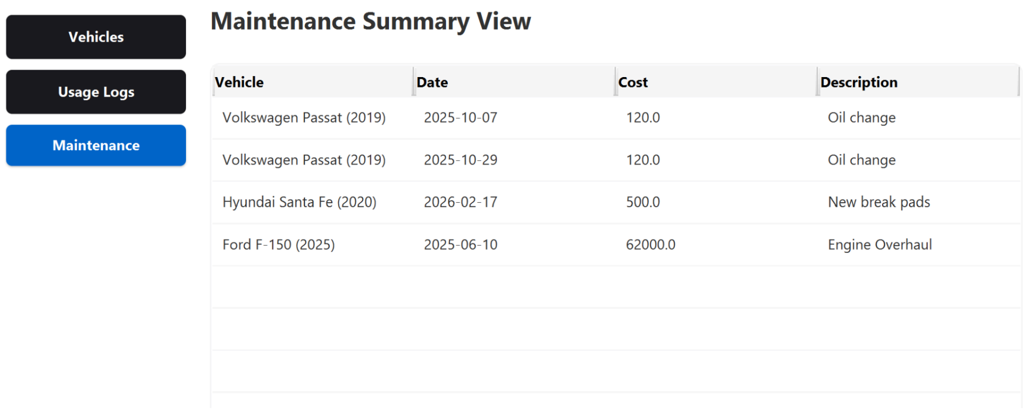
Vehicle Summary View

Make	Model	Year	Type
Volkswagen	Passat	2019	SUV
Hyundai	Santa Fe	2020	SUV
Ford	F-150	2025	Truck



Usage Summary View

Vehicle	Start Date	End Date	Distance(km)
Volkswagen Passat (2019)	2021-02-01	2022-02-18	120000.0
Hyundai Santa Fe (2020)	2023-02-16	2026-02-12	225000.0
Ford F-150 (2025)	2025-02-01	2026-02-12	120000.0



Maintenance Summary View

Vehicle	Date	Cost	Description
Volkswagen Passat (2019)	2025-10-07	120.0	Oil change
Volkswagen Passat (2019)	2025-10-29	120.0	Oil change
Hyundai Santa Fe (2020)	2026-02-17	500.0	New break pads
Ford F-150 (2025)	2025-06-10	62000.0	Engine Overhaul

Limitations

While this application demonstrates the core functionality, there are several limitations to note:

1. **Data Display Bug:**
   When navigating from the Maintenance Summary view to the Usage Summary view, existing data does not always display correctly. This is a known issue that would need to be addressed in a production-ready version.

2. **Code Structure and Maintainability:**
   Much of the code is "spaghetti code," making it difficult to read, maintain, and debug. With better upfront planning, reusable code could have been identified and modularized earlier. Waiting until the end to refactor led to increased debugging time and errors.

3. **Development Workflow:**
   The initial implementation of the Java desktop application was straightforward, but the limited time available meant that features were built in large, monolithic chunks. Breaking the project into smaller, incremental batches would have improved modularity, testing, and overall cohesiveness.

4. **Prototype vs. Production:**
   As it stands, the application functions as a prototype but is not suitable for production. The logic flow is inconsistent in some areas, and the overall structure is messy, which could lead to errors or maintenance challenges.

5. **CSS and Styling:**
   A significant portion of the styling is done using inline CSS. This makes changes repetitive and error-prone, as updates in one place may require changes elsewhere. A more modular approach to styling would improve maintainability and scalability.

Overall, these limitations highlight areas for improvement in planning, modularity, and workflow. Addressing these issues would make the application more robust and suitable for real-world use.

Reflection

I really enjoyed designing and building this application. Through this project, I gained hands-on experience in building a Java desktop application and developed a solid understanding of GUI development. Although the application did not include a database for persistent data storage or connect to Wi-Fi, it provided a strong foundation in interface design and application logic.

Working on this project also allowed me to exercise my UI/UX skills, which is not my strongest area. I tend to get bogged down in design details, but I was able to find a workflow that worked well. I approached the UI/UX design toward the end of the project, selecting a cohesive color scheme inspired by Apple’s palette, which is proven and visually appealing.

I noticed many similarities between front-end development in JavaFX and other front-end frameworks, particularly in terms of styling, layout management, and modular design. One of the biggest challenges for me was resisting the urge to constantly rewrite and improve the code. Better planning and design upfront would help, but I also feel that after each iteration, my understanding of code structure and design improves.

Despite being a rushed project completed past the due date, it gave me a thorough understanding of application development and strengthened my existing knowledge base. Regarding the MVC architecture, I largely maintained the separation of Model, View, and Controller. However, there were instances where I accessed the Model directly from the Controller. While this is not strictly aligned with MVC principles, I believe it was acceptable given the scope and scale of the project.

Overall, this project was a valuable learning experience, enhancing both my technical skills and my approach to designing cohesive, user-friendly applications.