# Sample Table of Contents

1. Version History (Should include Version #, Name of the Author and Date)
2. Introduction (About the system and what does the document is about)
3. Targeted Audience (Who are the targeted audience for this document)
4. Scope (What is in and out of the scope of this document)
5. Application Architecture (High level architecture/overview of entire system/ main component etc.)

*Sub heading 1*

*Sub heading 2*

1. Business Architecture (Use Case diagrams along with the description)

*Sub heading 1*

*Sub heading 2*

1. Detailed Design (Class diagrams, Component diagrams etc.)

*Sub heading 1*

*Sub heading 2*

1. Data Architecture (Database structures, ERD, Physical/Logical Data Model)

*Sub heading 1*

*Sub heading 2*

1. Security Architecture (What are the security consideration in your designs)

*Sub heading 1*

*Sub heading 2*

1. *Deployment Architecture (Infrastructure, deployment model etc.) Sub heading 1*

*Sub heading 2*

1. Testing Model (How are you testing your application. Junit, API testing etc)

*Sub heading 1*

*Sub heading 2*

1. References
2. Acronyms/Abbreviation

# List of Figures

Should include clickable links for all figures used in your entire report

Version History

This is the first version of the document. This document was written by Stephanie Prystupa-Maule, Mario Valencia, John Tieu and Mathew Chebet on Monday, March 17, 2025.

Introduction

The system described in this document is a web based Public Transit Fleet Management System (PTFMS) written in HTML and Java. This system is meant to handle tasks revolving around operator management, such as performance tracking, and managing breaks. This system also deals with vehicle management, which includes fuel consumption, maintenance and GPS tracking. This system will allow Operator Managers to manage their teams and vehicle fleets more effectively. This document contains all the information related to the architecture of this system from its overall design, security, data and deployment.

Targeted Audience

The target audience for this document are municipalities who are looking for a new transit management system for their transportation infrastructure, as well as transport staff such as Managers and Operators as they would be the primary users of this system.

Scope

This project solely deals with the management of vehicle fleets in terms of their maintenance, vehicle locations, fuel consumption and management of operators in terms of their ability to be on time and the times they are actively operating a vehicle. Some assumptions that were made are that transport vehicles can either operate normal or express routes and that the distances between each stop would be constant.

Application Architecture

This project was designed to use a 3-layer architecture design meaning that it will consist of a user interface(UI), a business logic layer, and finally a layer for the database to store and retrieve data. The user interface will simply consist of HTML forms and simple HTML pages. The business logic layer will be written in Java and will make use of servlets. Finally, the database layer will use MySQL.

Business Architecture

Detailed Design

Data Architecture

A diagram of a flowchart

AI-generated content may be incorrect.Figure . The conceptual diagram for the database used for the system.

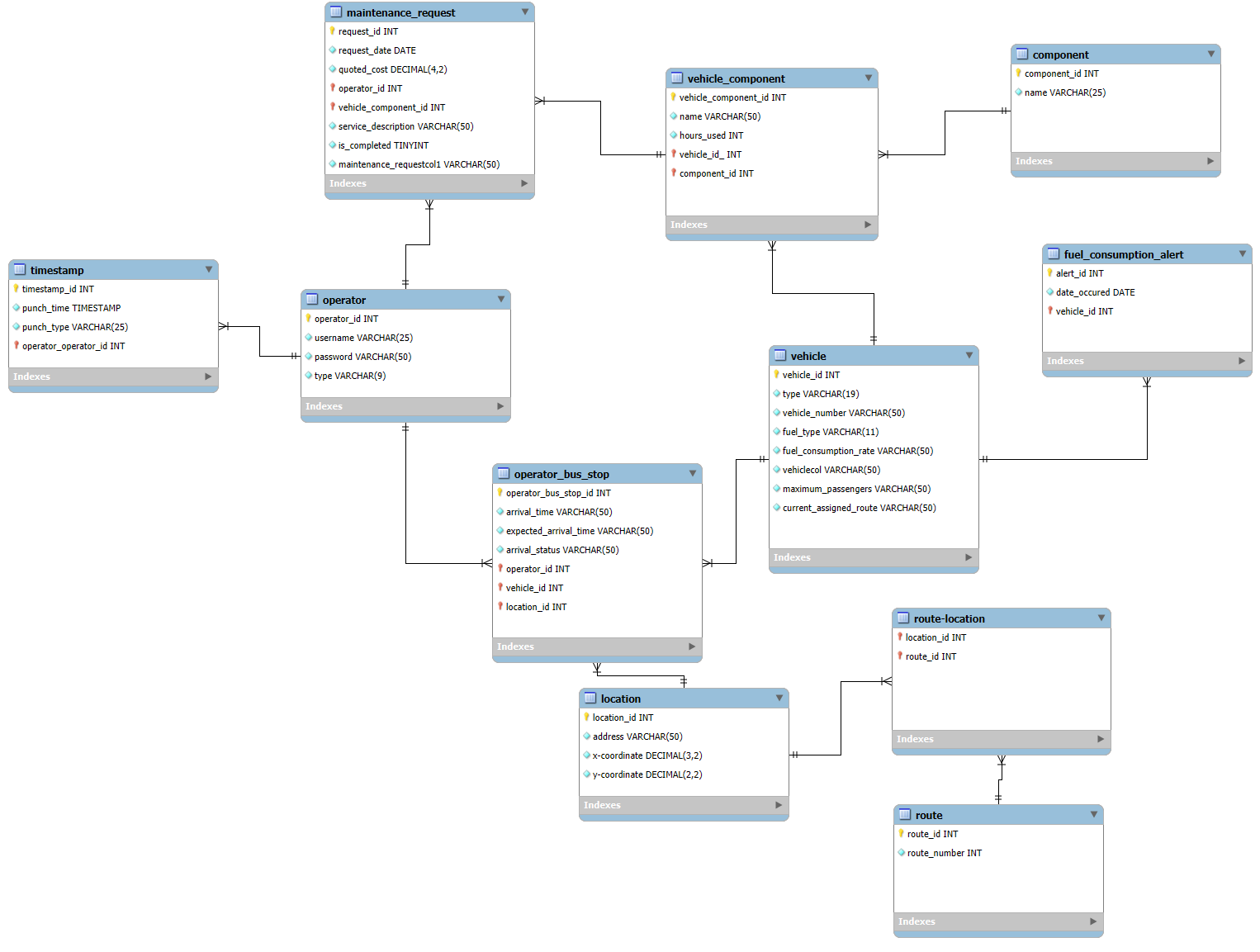


Figure . The physical diagram for the database used for the system.

Security Architecture

The security design plays a significant role in ensuring the data transmitted between the layers are intact. For example, a few security designs that can be implemented are:

* Least Privileges
* Minimal Coupling
* Data & Input Validation

Assigning modules to minimal privileges needed to run the system will help to prevent unnecessary access to the data involved and potentially breaking other features down the line due to systematic errors or user errors. To implement this, we would have to consider using *private*, and *protected* access modifiers in most cases, and *public* strictlyfor features that must execute as part of the overall design to ensure that the effects of bad inputs and corrupted data are minimal to the vehicle and operation management system.

By implementing the least amount of coupling between the internal subsystems, we can prevent issues going further down the chain of events involving multiple functions that make up the behaviour of the entire application. For example, it may be better to determine the efficiency of a vehicle in its own module using data obtained directly from the source, rather than involving data from a third-party feature, due to easier maintenance when troubleshooting the cause of issues.  
 As mentioned earlier, data and input values can go bad due to many factors, such as a bad actor, or corrupted data from a data source, such as a hardware issue. To ensure the data impacts are minimized with the operations of the public transit system, we must cleanse any incoming data. For example, there will always be the potential of attempts to break the internal database system using invalid data inputs or commands that are not accounted for the typical use case of the whole program. By creating various validation stages, we can reduce the reliance on the inputs directly that can cause issues with the backend systems.

Deployment Architecture

Testing Model