Team A

Delta V Math Calculations System Specification

Version <1.0>

Table of Contents

System Specification

1. Introduction

1.1 Purpose

This document provides a comprehensive overview of Delta V Calculator and the process that was used to construct it. The document captures the importance of the solution, system requirements, a high-level schedule, system architecture, and the software quality plan. This can assist the customer and future teams who wish to continue this project, as well as users who wish to use this application.

1.2 Scope

This document covers topics such as the scheduling used to complete this application, the architecture used to structure our code and our testing.

1.3 References

Meeting notes
https://confluence.cs.uky.edu/x/oIIR

Project Plan
https://confluence.cs.uky.edu/x/dIMR

Coding Assignment (Contains developer's guide and administrative manual)
https://confluence.cs.uky.edu/x/-wA-/

Database setup instructions
https://confluence.cs.uky.edu/x/GQA-/

High-level architecture
https://confluence.cs.uky.edu/x/tIQR

Creating a standalone executable
https://github.com/jlre249/CS499_Project/wiki/Creating-a-Standalone-Executable

1.4 Overview

The system specification goes on to cover an overview of the project, its requirements, schedule, architecture, architecture, maintenance and quality assessment.

2. Project Overview

This section provides an overview of the project. The project creates a software application to automate tedious hand-calculated formulas to help investigators of a crash save time. See the following sections for more details about the problem, vision, and stakeholders.

2.1 Problem Statement

According to the National Highway Traffic Safety Administration, there were 6,734,000 crashes reported in the United States in 2018 (https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812951). These crashes had an estimated economic impact of $242 billion. Due to the large number of crashes and the need to investigate crashes for insurance purposes, much time is spent investigating crashes. Currently, investigations involve recreating the accident by hand calculating formulas such as the weight distribution of the vehicle. This project will provide an application that makes calculations based on vehicle data to save the investigator from doing hours of hand calculations.

2.2 Vision Statement

The purpose of this product is to assist in crash scene data analysis by performing automatic calculations on data input by the user and generating a report. The positive change is that it eliminates the need for performing the necessary calculations by hand, and prevents mathematical errors as a result. The table below shows the Product Vision Board which details the vision, target group, needs, product goals, and business goals for the product.

Vision

The purpose of this product is to assist in crash scene data analysis by performing automatic calculations on data input by the user and generating a report. The positive change is that it eliminates the need for performing the necessary calculations by hand, and prevents mathematical errors as a result.

Target Group	Needs	Product	Business Goals
• Insurance agencies • Law enforcement • Private investigators • Car manufacturers • Civil and mechanical engineers • Attorneys	• Saves users time by automatically doing tedious calculations • Improves accuracy by preventing user errors since calculations are done automatically • Allows user to easily generate a report of the data analysis	• A software application that takes user input, performs calculations, and outputs a report • The product provides ease of use for the user • This product stands out because the market is currently not offering anything of its kind, and the current solution for calculating is done by pen and paper	• The product will benefit the company by allowing the users of their software to save weeks of time by performing automatic calculations on vehicle data • Provides accurate results for the solutions from the math calculations

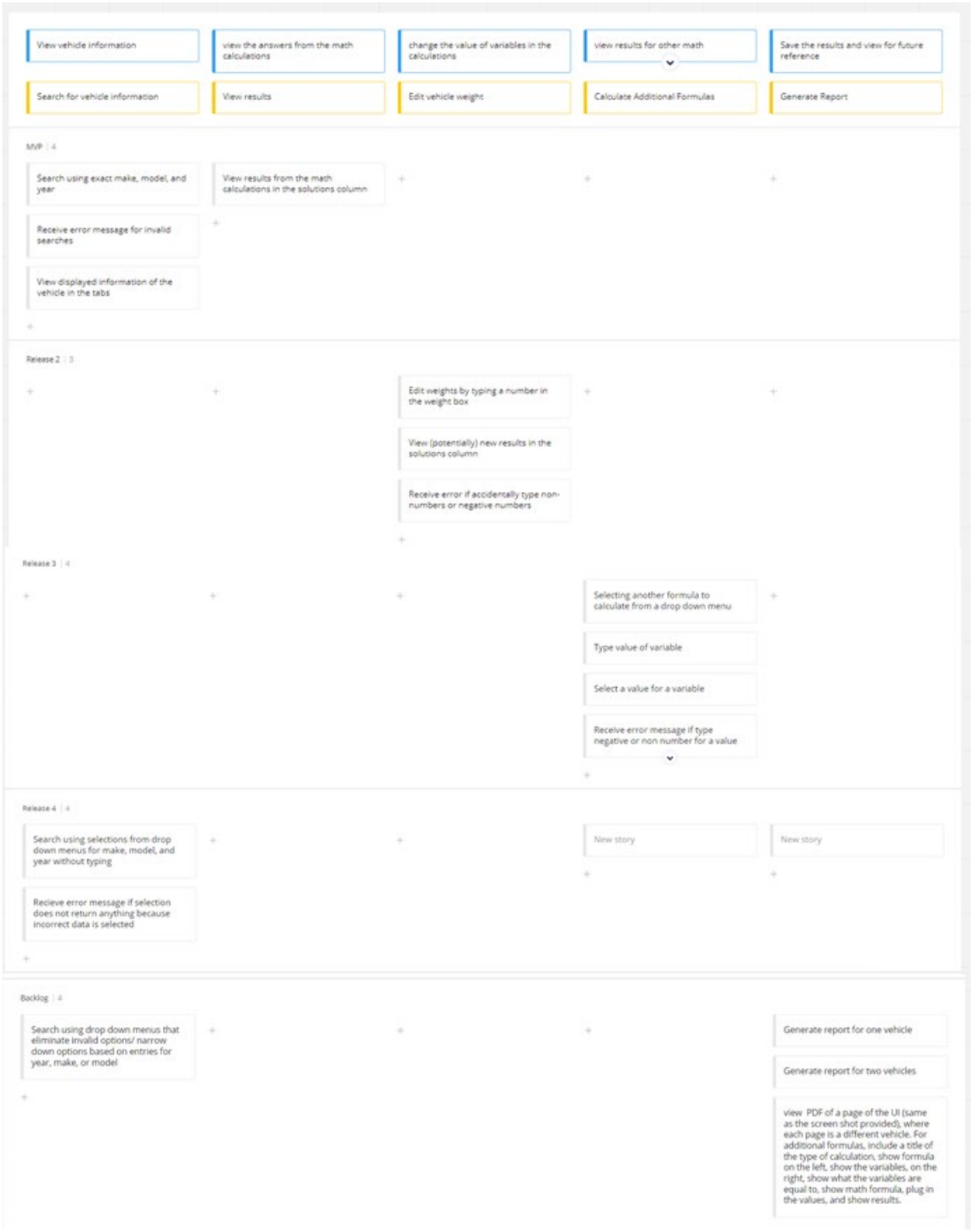
2.3 Stakeholders

Mike Flamm is the client and a stakeholder for the system. Mike expects the application to meet the needs of the other stakeholders and be completed on schedule. Another stakeholder group is the users of the application. Users expect the application to be easy to use and reliable. The final group of stakeholders is future teams. Future teams expect the application to be easily modifiable to assist with maintenance and future development.

3. Project Requirements

This section describes the requirements for the project, such as functional overview, user stories, and quality goals. In section 3.1, there is an overview of the users, their goals, and the features that will aid them in reaching their goals. Section 3.2.1 displays the user stories that have been implemented, including updates, if any, for those user stories. Section 3.2.2 shows the user stories that have yet to be implemented, and the reason why they remain as planned. Section 3.2.3 shows the in-progress user stories, including the reason for why the user story in progress.

3.1 Functional Overview



Above shows the user story map for the project. The user story map shows that all users should be able to search for vehicle information, view results from the math calculations, edit vehicle weights, calculate additional formulas, and generate a report of all results. In the MVP, the goal is to implement the functionality of searching for a vehicle, and viewing the information of that vehicle in tabs. The MVP also

includes the ability to view results from math calculations. The user is limited to only providing valid information in order to search for a vehicle. Release two involves the user's ability to modify vehicle weights by typing a value into a text box. The user is limited to only typing in non-negative numbers in order to change a vehicle weight. In release three, to add an additional formula, the user should be allowed to select a formula from a drop down, and type values into text boxes, leading to viewing the calculation results. The user is limited to only typing in non-negative numbers in order to provide information for the variables. In release four, the goal for searching for vehicle data should be updated so that the user can search for information using dropdowns.

One story that remains in the backlog is the dropdowns, used for searching for a vehicle, eliminates invalid options based on input from the previous dropdowns. Another story, that remains in the backlog, is the user's ability to generate a pdf report of the results for one vehicle, and results for two vehicles.

3.2 User Stories

This section shows the implemented, planned and in-progress user stories.

3.2.1 Implemented

User Story	Acceptance Criteria	Update(s)
As an investigator I want to be able to search for vehicle information by using year, model, and trim, so that information about that vehicle can be filled into the tabs below	The Investigator should be able to search for a vehicle by year, model, make, and trim, and when they click a button, the tabs below should be filled with the additional information about the vehicle. ~~If the investigator uses incorrect information (misspellings, incorrect info, etc.), then they will receive an error message~~. ~~If the investigator does not fill out all fields required, then they will receive an error message saying to fill in the required fields.~~ ~~If the investigator provides vehicle information that is not available, they will receive an error message.~~ Only one vehicle should ever be returned from the search.	The user story was implemented using combo boxes; therefore, the investigator cannot provide incorrect information, nor can they provide information of a vehicle that is not available. Not only so, but if the investigator neglects filling out a field, then the application will not take any action.
As an investigator I want to be able to view results from the math calculations for one vehicle, so that I will be able see the answers of one vehicle without having to do the math by hand.	Solution boxes get filled in with correct results from calculations. Results should match ~~example search~~ and calculation results.	The implementation for an example search was decided on not being implemented.
As an investigator I want to be able to edit a vehicle's weight, so that I can see how the results from the calculations change.	Investigator can change a weight in the tabs on the vehicle picture. Solution boxes are updated with new results from new weights. ~~If the investigator enters a letter, negative numbers, or special character, they will receive an error message.~~	The investigator is prevented from typing any negative numbers, letters, and special characters.
As an investigator I	Investigator can click a button and a PDF report	The additional formulas section, for

want to be able to generate a report of the results, so that I can review for future reference.	with the vehicle information and results is created. PDF should have a page that matches the UI and includes the information for the vehicle. Additional formulas ~~will be~~ ~~on a separate page~~ with a title, the formula on the left, the variables in the calculation, the variables' values, the math formula, and the results.	the report, will not be on a separate page.

3.2.2 Planned

User Story	Acceptance Criteria	Reason
As an investigator, I want to be able to perform math calculations for multiple vehicles, and see multiple solutions to compare the results.	The investigator should be able to add vehicles, and for each vehicle, calculate math formulas and display the results of the math calculations.	We were never able to get around to figuring out how to implement the functionality.
As an investigator, I do not want the executable to crash whenever inappropriate data is added as an input	If a user provides inappropriate input, then application should prevent itself from crashing, while also warning the user about the error.	Our client suggested that this should not be our highest priority for the project.

3.2.3 In Progress

User Story	Acceptance Criteria	Reason
As an investigator I want to be able to use the vehicle information to calculate additional formulas to get more information.	Investigator can select another formula from a drop-down menu. Investigator can fill in needed variables by typing or selecting values. If the investigator types values for a variable, and they provide non-numbers, then they will be provided an error message. If the investigator does not provide any value, then they will be asked to fill out the field. Solution boxes are updated with formula solutions. Results should match calculation results.	The formulas can be calculated, if the investigator enters data manually, but, due to time constraints, the ability to select a value has not been implemented. Also, due to time constraints, the ability to add a formula, and handle inappropriate inputs, is not implemented.

3.3 Quality Goals

Quality Goals	Target
Usability	Searching for a vehicle will display vehicle data and formula solutions faster than obtaining the results by hand.
Modifiable	Documentation provides example changes to

	the code to help future developers update the program.
Reliability	The completed project will not have any known crashes and all error cases will be handled.

4. Project Schedule

Sprint	Sprint duration	Schedule
0	9/15/2020 - 9/29/2020	▪ Spike: ▪ Figure out old team's code and how to start adding to it. ▪ We plan to communicate with past team members ▪ Figure out how to use QT and MySQL ▪ study up on Sql. How to query for and update data
1	9/30/2020 - 10/14/2020	• Two user stories: • As a user I want to be able to search for vehicle information by using year, model, and trim, so that information about that vehicle can be filled into the tabs below. • As a user I want to be able to view results from the math calculations for one vehicle, so that I will be able see the answers of one vehicle without having to do the math by hand • At the end of SECOND WEEK: • MVP • Release 1 • **accomplish:** • Search using exact make, model, year, trim • Receive error message for invalid searches • View displayed information of the vehicle in the tabs • View results from the math calculations in the solutions column

2	10/15/2020 - 10/29/2020	Complete user stories: ▪ As a user I want to be able to edit a vehicle's weight, so that I can see how the results from the calculations change ▪ As a user I want to be able to use the vehicle information to calculate additional formulas to get more information ▪ At the end of SECOND WEEK: ▪ release 2 ▪ **accomplish:** ▪ Edit vehicle's weight, and edit math calculations to update the results. ▪ edit weights by typing numbers ▪ receiving error message if using non-numbers or negative numbers ▪ results in the solutions column gets updated once the weight has changed ▪ release 3 ▪ **accomplish:** ▪ selecting another formula to calculate ▪ type/select values for the variables ▪ receive an error message if the investigator inputs invalid information
3	10/30/2020 - 11/13/2020	Complete user stories: ▪ As a user I want to be able to generate a report of the results, so that I can review for future reference ▪ At the end of SECOND WEEK: ▪ release 4 ▪ **accomplish:** ▪ Search using selections from drop down menus for make, model, and year without typing ▪ Receive error message if selection does not return anything because incorrect data is selected

As shown above in the product schedule, we planned to start with a spike to familiarize ourselves with Qt c++ and SQL, release our minimum viable product at the end of our first sprint and add onto to it in the following sprints.

After meeting with our customer, our group had the understanding that this project was heavily dependent on the Qt c++ framework and database technology. Due to our limited experience with these topics, the group decided to start our schedule with a two-week spike with an emphasis on SQL for the purpose of retrieving necessary data from a database. In hindsight, the spike was not as effective as it could have been. The focus should have been more on learning the Qt c++ framework and not on SQL because the necessary queries were not so complex. The majority of the code we produced dealt with things other than retrieving data. In this spike we also set up meetings with past team members from this project. This was very valuable because the former members were able to assist us in setting up our developing environments and answer questions we had about the project at the time.

The plan was to release a minimum viable product by the end of the first release. We wanted the product to be able to retrieve a specific vehicle upon the user providing year, make, model, and trim by typing. Then upon the retrieval of the vehicle, to use information specific to that vehicle to perform calculations and show the results to the user. During this sprint we learned that it would be beneficial and not too complex to provide dropdown menus with appropriate options for the user to perform a vehicle search rather than type their specifications. This was a feature that we had planned to implement in our last sprint but we decided to do it release it in our MVP.

Providing the capabilities of performing and displaying the calculations was carried over into the second sprint. In this sprint the calculations user story was completed along with the user story of editing the weights and calculations to update the results. At this point in our sprint, after meeting with our customer we decided to address the issue of incomplete data from the database which was not considered at the time of planning.

The user story for selecting a desired formula from a dropdown menu was carried over into the third sprint along with the tasks of providing missing information on vehicles. The user story of generating a pdf report was quickly completed. The issue of missing data originated from past teams trying to consolidate data from several data sources. We decided to circumvent this problem by having the user search for information using two separate data sources.

There were several changes that came from not fully understanding Qt's capabilities, such as using dropdown menus for the search. Issues such as filling in missing information came from not fully understanding what was in the database. If anything was to be changed from, from a scheduling stand point, we could have used the spike to better understand our provided database and the Qt framework to avoid having to change our schedule.

5. System Architecture Overview

The system architecture shows the components of the system and how they interact with each other. This section provides the architecture constraints, system scope and context, and the solution strategy for the system. Section 6 provides details about each component of the system.

5.1 Architecture Constraints

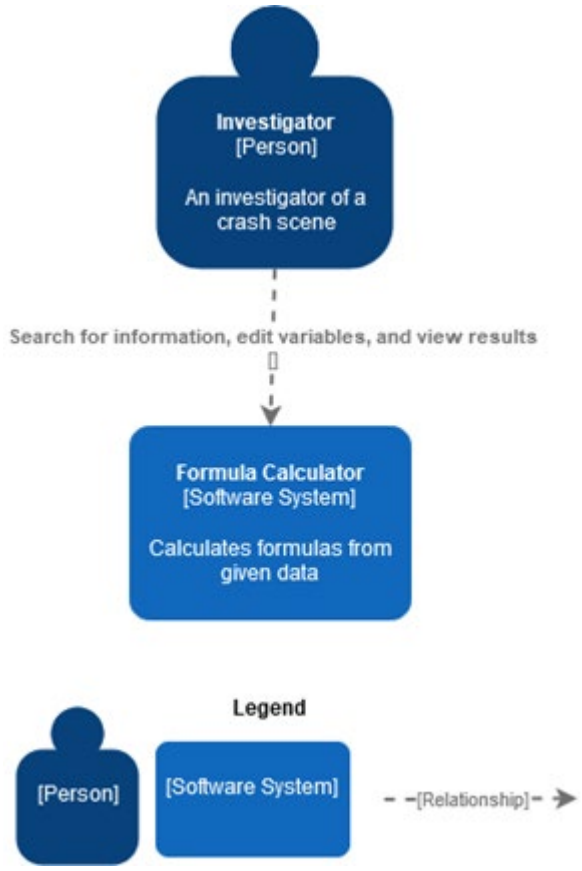
The system had three main technical constraints: software to use in development, programming language to use, and database to use. The system had to be developed using Qt and MySQL since previous teams used them. The system had to be programmed in C++ since previous teams had used that and future teams should be familiar with the language. Finally, the database for the system had already been populated, so the system is limited to the information already in the database.

The system had two organizational constraints: the team and the time schedule. The team was limited to the senior design team members Itai Kumengisa, Tabitha Holloway, and Christopher Butler. The time schedule was limited to the time given to the class for the projects which was late September to early November.

5.2 System Scope and Context

The system will be used by investigators of a crash scene to reconstruct details of the crash such as the minimum speed of the vehicle. Investigators will interact with the formula calculator by searching for vehicle information, editing variables, and viewing results. The formula calculator will calculate the solutions to formulas and show the results to the investigator. The system context is shown in the system context diagram below and is followed by details of the diagram.

System Context Diagram



The investigator selects make, model, trim, and year to search for a vehicle, and after clicking the search button, the text boxes below will automatically fill with the found vehicle data from the database. The investigator can also modify information, like the weight of the vehicle, and solutions from math calculations will update automatically. The investigator can also input data for additional formulas that will be calculated and the solutions shown.

The application allows the investigator to input search data for a vehicle, change weights for a vehicle, and input data to calculate additional formulas. It also displays the solutions of the math formula calculations to the investigator.

5.3 Solution Strategy

The system uses C++, MySQL, and Qt since they were used previously. The database and user interface were already created, so the system only needed to interface with them and perform calculations on the data.

The system had three main quality goals: usability, modifiable, and reliability. To ensure the system is easy to use, we tried to avoid adding clutter to the user interface, to make the interface responsive, and to provide error messages to the user when appropriate. To enable easy modification of the system, we used an object orientated approach to programming, documented the code with comments, and provided a developer's guide to assist future teams with development. To make sure the system is reliable, we implemented input verification and error messages to ensure that the investigator does not crash the system with invalid input.

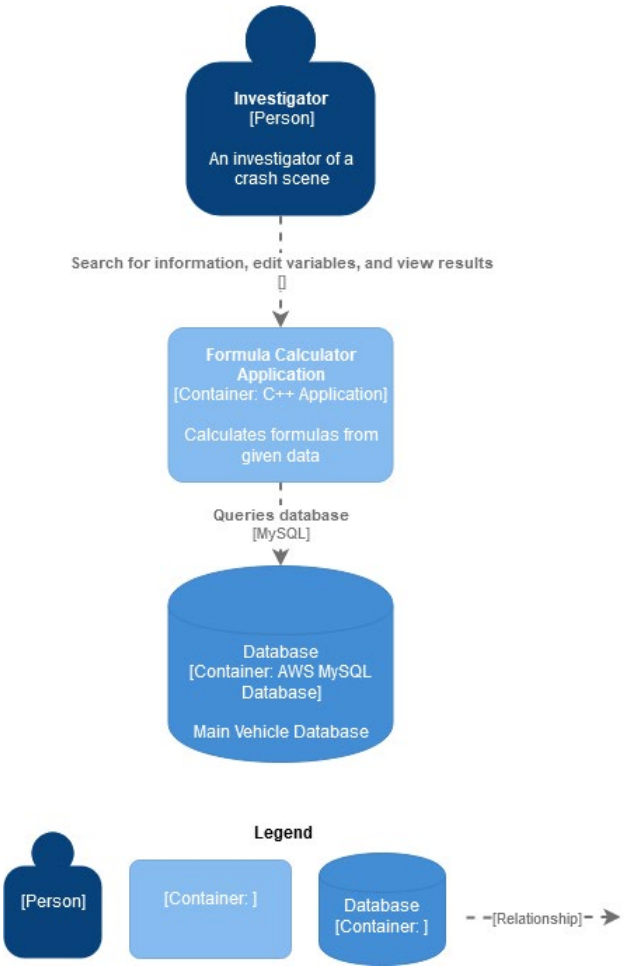
6. System Architecture

This section provides details about the system architecture. It begins with the container diagram which is a more detailed version of the system context diagram provided in Section 5.2. Details are given about each

container in the container diagram in the subsections of Section 6.2. Details are then given about the important interfaces in the system. Finally, a component view of the system is given with details of each component

The container diagram below shows the interactions between the investigator using the system, the formula calculator application which calculates and displays the formula solutions, and the database that stores the information about the vehicles. Details about each container are provided in the subsections of Section 6.2.

Container Diagram



6.1 Motivation

The structure of the container diagram is straightforward. The investigator only interacts with the formula calculator to prevent the investigator from directly manipulating the database and allow the investigator to easily search for information about vehicles without knowledge of the structure of the database. The formula calculator processes the investigator's requests, obtains the information from the database, calculates formulas, and displays the results to the investigator. The database simply responds to the queries from the formula calculator with the appropriate information.

6.2 Container Level

Each subsection in this section provides details about one of the containers in the container diagram above. For details about how the investigator interacts with the system, see the description in Section 5.2.

6.2.1 Formula Calculator

The formula calculator receives search requests from the investigator and uses this data to calculate the

solutions for the math formulas. It also receives variable changes from the investigator, determines which of the formulas the variables belong to, and updates the solutions after re-calculating the formulas. The formula calculator also queries the vehicle database to get information the investigator wants to search for. If certain information cannot be retrieved from the database, then the data will not be used.

6.2.2 Database

The vehicle database stores vehicle information to be displayed to the investigator and used in calculations by the formula calculator. The formula calculator queries this database to obtain the vehicle information the investigator is searching for.

6.3 Important Interfaces

This section details some of the important interfaces that are used in the system.

6.3.1 Line Edit Interface

The line edits, Qt's name for textboxes, are accessed in the code using pointers to the line edit. These pointers are obtained using Qt's *findChild()* function which is documented at https://doc.qt.io/qt-5/qobject.html#findChild. The values of the line edits are read using the line edit function *text()* and values are written to the line edits using the line edit function *setText()*. These functions are documented at https://doc.qt.io/qt-5/qlineedit.html. Currently, these interfaces are spread throughout the code and used in whichever function needs to access that line edit. Future teams should consider consolidating these accesses into a separate layer that handles line edit accesses.

6.3.2 Database Interface

Use the interface template above.

The database interface is handled by the *DatabaseConnection* class. This class was developed by a previous team and uses the *QSqlDatabase*, *QSqlQuery*, and *QSqlRecord* classes to send MySQL queries to the database and parse the query result. These classes are documented at https://doc.qt.io/qt-5/qsqldatabase.html, https://doc.qt.io/qt-5/qsqlquery.html, and https://doc.qt.io/qt-5/qsqlrecord.html.

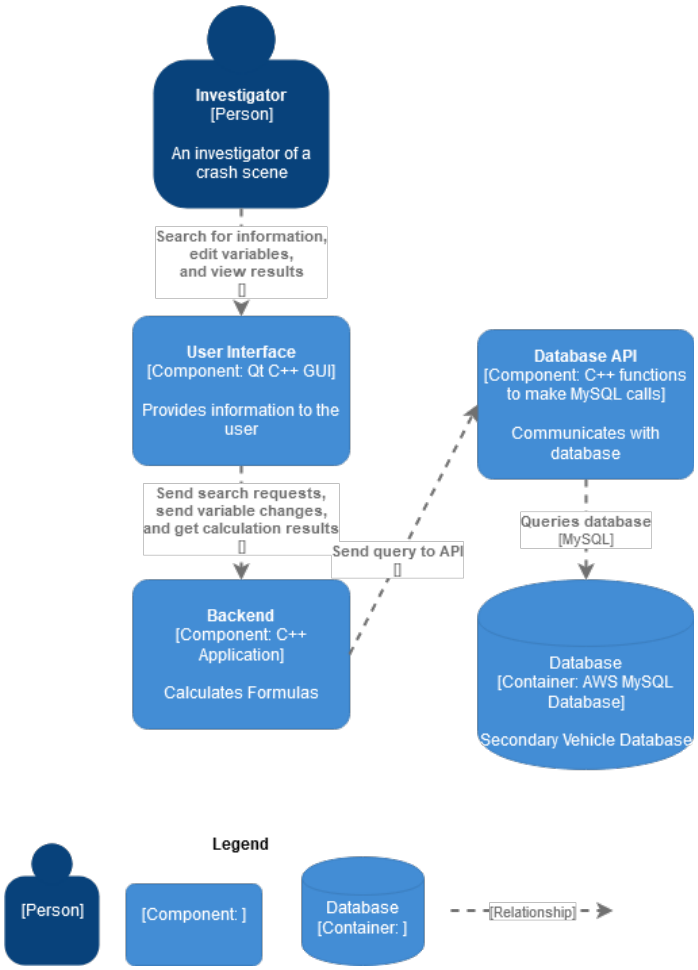
6.4 Component Level

This section gives the component level diagram for the formula calculator container. For details about the investigator see Section 5.2. For details about the database container, see Section 6.2.2.

6.4.1 Inside Formula Calculator

The component diagram breaks down the components of the formula calculator container in the container diagram. The investigator interacts with the user interface component. The user interface interacts with the backend which calculates the formulas. The backend sends queries to the database API which queries the database.

Component Diagram



The component diagram is structured to let each component interact with only two other components. This allows clear separation between components and keeps the flow of information simple.

The investigator interacts with the user interface in the same way that the investigator interacts with the system as a whole. For more detail on this interaction, see Section 5.2.

The user interface displays data and solutions to the investigator. The user interface also allows the investigator to input a vehicle to search for and to change values for variables. The user interface sends this information to the backend for processing. This interface includes the line edit interface described in Section 6.3.1. The user interface ended up being entangled with the backend of the system. Ideally, they would be better separated, so future teams should consider separating the user interface more from the backend.

The backend takes search requests and variable changes from the user interface. The backend also sends search requests to the database API and calculates solutions to the formulas based on the vehicle data. Finally, the backend sends solutions and search results back to the user interface to be displayed.

The database API takes a search request from the backend and queries the database for the information using MySQL. The database API returns the results of the query to the backend. This interface is described in more detail in Section 6.3.2.

The vehicle database stores information about the vehicles, receives queries from the database API, and sends query results back to the database API. For more information about the database container, see

Section 6.2.2.

7. Software Quality Plan

The software testing plan involves using both acceptance and unit testing in order to make sure the application is working as required. Section 7.1 has the quality review for both the unit tests and the acceptance tests. Section 7.2.1 discusses the test plan for acceptance testing. Section 7.2.2 involves discussing the test plan for the unit tests. Section 7.3.1 describes test cases for acceptance testing, while explaining the reason for using each test case. Section 7.3.2 describes test cases for unit testing, while explaining the reason for using each test case.

7.1 Quality Review

One quality review, that was conducted, was performed to determine if the user story acceptance criteria were met. The process for this review was done by the team members, as a group, went through the application by following the tasks required by the user stories. The process, and results, from following those tasks were compared with acceptance criteria. We learned that one of the user stories, such as the ability to add additional formulas, was not followed completely, but since our client was not requiring the complete implementation of this user story, we determined that the failure was not significant. We also learned of a few bugs in the program. For instance, if the user searched for a vehicle without providing information, the application would crash. Another bug we learned of was if the user selects a year, but then selects “Select a year,” the model combo box is still populated with models from the year the user first selected. We learned that if a user searched for a vehicle with at least one field selected, the program would crash.

Another quality review, that was conducted, was performed to determine if the unit tests passed in order to confirm that the math calculations were done properly, and that certain values would not create unexpected values. The process for this review was done by running the unit tests. We learned that the unit tests did not cover enough edge cases, but because this was not a priority for our client, we determined it was not significant. We also learned that, in general, there were not enough unit tests to determine if the formula calculations are reliable.

7.2 Test Plan

There will be two types of testing that will be executed, which will include acceptance testing and unit testing.

7.2.1 Acceptance Testing

The acceptance test will be used in order to determine if the application can work the same way that the acceptance criteria, from the user stories, requires the application to run from the user’s point of view. Therefore, the results from the acceptance tests will be compared with the acceptance criteria, for each of the user stories, in order to determine if the tests pass. The acceptance criteria include:

- The investigator can enter year, model, make, and trim, and after clicking a button, the text boxes get filled with the selected vehicle's information.
- The solution boxes, that display results from the math calculations, should display correct results.
- The investigator can change the different weights for a vehicle, in which the edits lead to updating the redistributed weights.
- The investigator can click button to add a new formula, fill in the needed variables by typing or selecting a value. As a result, the solution boxes are updated with formula solutions.
- The investigator can click a button and a pdf report with the vehicle information and results is

created.

- The investigator can add additional vehicles, and be able to calculate the math formulas for each of those vehicles.
- The investigator is incapable of breaking the application, despite providing inappropriate or incorrect data.

Therefore, we will be testing how the application responds to user input, because this helps determine if the application is properly searching for data, placing the searched data into the appropriate textboxes, while also performing the math calculations using that data, and displaying the results of the math calculations in the proper textboxes. We will also test if the values in the textboxes update whenever the user inserts or the changes the values for textboxes, mainly for the weights and height of the wheels. This is being tested for because it will show that the application can update values of variables when given new inputs. Testing will also be done the applications ability to create a pdf report of the vehicle information, and results from the math calculations. This is because an acceptance criterion allows the investigator to create a report of such data. Testing will not cover the ability for the application to add multiple vehicles, and calculate the math formulas for each, because the group never intended accomplish the user story with this done criteria. Testing will also not cover edge cases, such as dividing by zero, because our customer decided that it shouldn't be our biggest priority. While the acceptance tests will be performed throughout the progress of the project, we plan to spend about an hour, to perform and compare acceptance tests, at the end of each sprint.

7.2.2 Unit Testing

The goal of the unit tests is to determine if the calculations for the math formulas are correctly performed. In order to determine if the calculations are accurate, the results for each calculation from the application and the unit tests will be compared. Therefore, testing will focus mainly on the formula calculations. This is because the formula calculations are an important part of the program because using the formulas it is the purpose for the application. The tests will determine the accuracy for the formulas that calculate the weight distributions, horizontal center of mass, center of mass height, minimum speed, and velocity. The tests will cover these formulas because these are the formulas that the customer provided. Although it is needed, the testing will not cover many of the edge cases, such as dividing by zero, because our customer decided that it shouldn't be our biggest priority. For unit testing, we plan to spend about a week to set up and create the tests, and two hours to run and check all test results.

7.3 Test Cases

7.3.1 Acceptance Test Cases

One test case, for acceptance testing, is that the investigator selects a make, model, year and trim from the first set of dropdowns, and then selects the get vehicle info button. The reason this test case will be used is because it will show if the application is using the inputs for searching data, and displaying the returned searched data properly.

Another test case, for acceptance testing, is given that the investigator first used the first set of dropdowns for searching, the investigator selects a make, model, year and trim from the second set of dropdowns, and then selects the get vehicle info button. The reason this test case will be used because it will show if the application is using the inputs for searching data, and displaying the returned searched data properly. It will also show if the second search helps fill out some of the blank textboxes that the first search could not fill.

Another test case, for acceptance testing, is given that the investigator has selected a make, model, trim, and year, but has yet to select the get vehicle info button, the investigator selects "Select a year" in the year dropdown, and then selects the get vehicle info button. The reason this test case will determine if the application is capable of receiving unexpected inputs from the user.

Another test case, for acceptance testing, is when the investigator changes the value in the textbox that contains the overall weight for a vehicle. The reason this test case will help determine if the application will update existing calculated values based upon the changes the user makes to the inputs.

Another test case, for acceptance testing, is when the investigator changes the value in the textbox that contains the weight on the front left tire. The reason this test case will also determine if the application will update existing calculated values based upon the changes the user makes to the inputs.

Another test case, for acceptance testing, is when the investigator selects the save as pdf button. The reason for using this test case is that it will show if the application can save the all of its information in a report that will be saved as a pdf.

Another test case, for acceptance testing, is given that the application has inputs for the curb weight, wheel base, track width front and rear, overall length, width, height, and weight distribution, the investigator inputs data for the height of center of the front wheels, height of center of the right wheels raised, and weight on the front right wheels raised. The reason for using this test case is that it will show if the application can calculate formulas without needing complete input from the program.

A final test case, for acceptance testing, is when the investigator inserts data for the skid distance and the acceleration drag. The reason for using this test case is that it shows that the application can calculate additional formulas, and that it can calculate using input data only from the user.

7.3.2 Unit Test Cases

The test cases, for unit testing, will only include the math calculations. The calculations will include weight distributions, horizontal center of mass, center of mass height, minimum speed, and velocity. The reason that the unit tests will focus on the formulas calculations is because using the formulas is the purpose of the application.

8. Maintenance

Starting this project took longer than we had hoped for. Specifically, we struggled getting our qt Application to connect to a database stored in the cloud. To help shorten the time it takes to set up supporting systems, we've attempted to document the steps that we took. We've documented the process for downloading Qt, for installing MySQL, MariaDB, and connecting our application to the database through Qt.

In our program, the user searches for a vehicle and the search returns information on that vehicle. That information is then taken and used to perform several calculations. We did this by filling in text boxes on the user interface and taking the values from the textboxes to use in our calculations. This worked but it is not the best way to go about it, it's better that the calculations are not so dependent on the user interface. To remedy this, it would be a good idea to make a vehicle class that has its variables populated with information from the query results. The vehicle class can have methods that perform the desired calculations and the results of calls to those methods can be provided to the user interface.

9. Glossary

Term	Definition

<Term-1> <definition-1>

<Term-2> <definition-2>

10. Word Count per student

Christopher Butler: 1872
Tabitha Holloway: 2687
Itai Kumengisa: 1249
Links to other documents: 5741