High-Level Architecture

Introduction and Goals

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 amount 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. Our project will provide an application that makes calculations based on vehicle data to save the investigator from doing hours of hand calculations.

Requirements:

- The application will be able to provide accurate solutions for the math formulas from the given user input.
- The application will be able to handle changes in the input from the user, eliminating the process of redoing the math by hand.
- The application will be able to pull vehicle information from the database and display the data to the investigator.

Three 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.

Stakeholders	Expectations
Users	Application needs to be easy to use and reliable.
Future Teams	Application needs to be easily modifiable.
Mike Flamm	Application needs to meet the needs of other stakeholders and be completed on schedule.

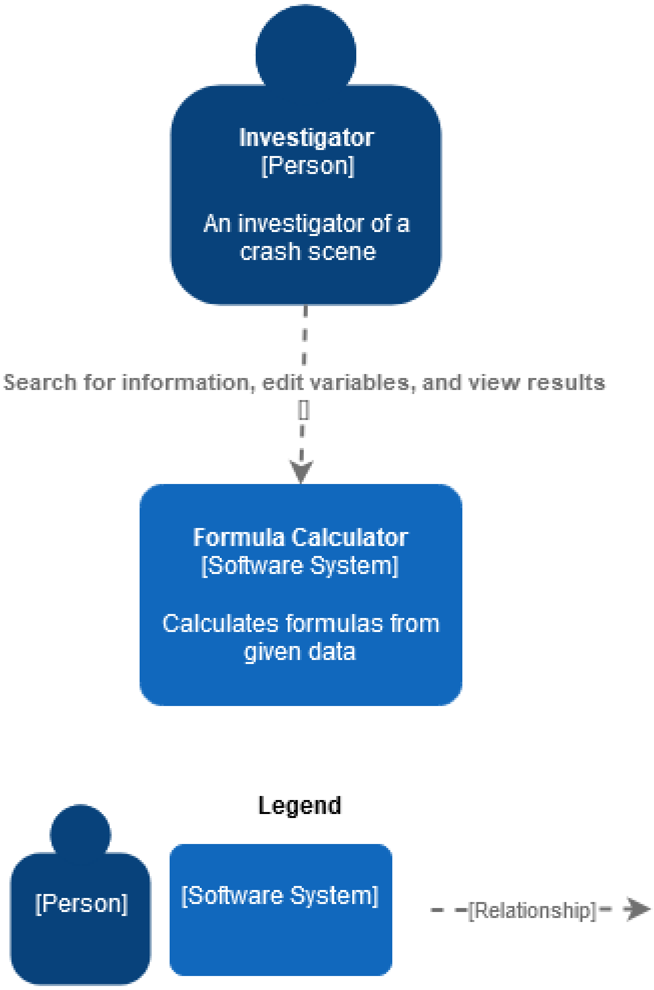
Constraints

Technical constraints	Constraint	Background / motivation
TC1	Software and programming	We will be constrained to the languages and software used by the previous teams. These include Qt, MySQL, and C++
TC2	Database	The database we are given is already populated, so our information will be restricted to what's provided there

Organizational constraints	Constraint	Background / motivation
OC1	Team	Itai Kumengisa, Tabitha Holloway, Christopher Butler, Tyler Dewitt
OC2	Time schedule	Starting late September and will do four releases in three sprints ending in early November
OC3	Configuration/ version control management	Will use Github to maintain consistent code and keep track of releases
OC4	Code Guidelines	Limiting our code structure to using tabs instead of spaces and the naming of variables to camel casing

Context and Scope

System Context Diagram


Investigator
[Person]
An investigator of a crash scene
Search for information, edit variables, and view results
[]
Formula Calculator
[Software System]
Calculates formulas from given data
Legend
[Person]
[Software System]
[Relationship]

Partner	Explanation
Investigator	The Investigator selects make, model, trim, and year, and after the selections, tabs below will automatically fill with additional vehicle data. The Investigator can also modify information, like the weight of the vehicle, and solutions from math calculations will update. The Investigator can also input data for additional formulas, in which calculations will be performed and solutions will be shown.
Formula Calculator	The application allows the user to input search data for a vehicle, change weights for a vehicle, and input data to calculate additional formulas. It also displays the solutions, after calculations from using the math formulas are finished, to the user.

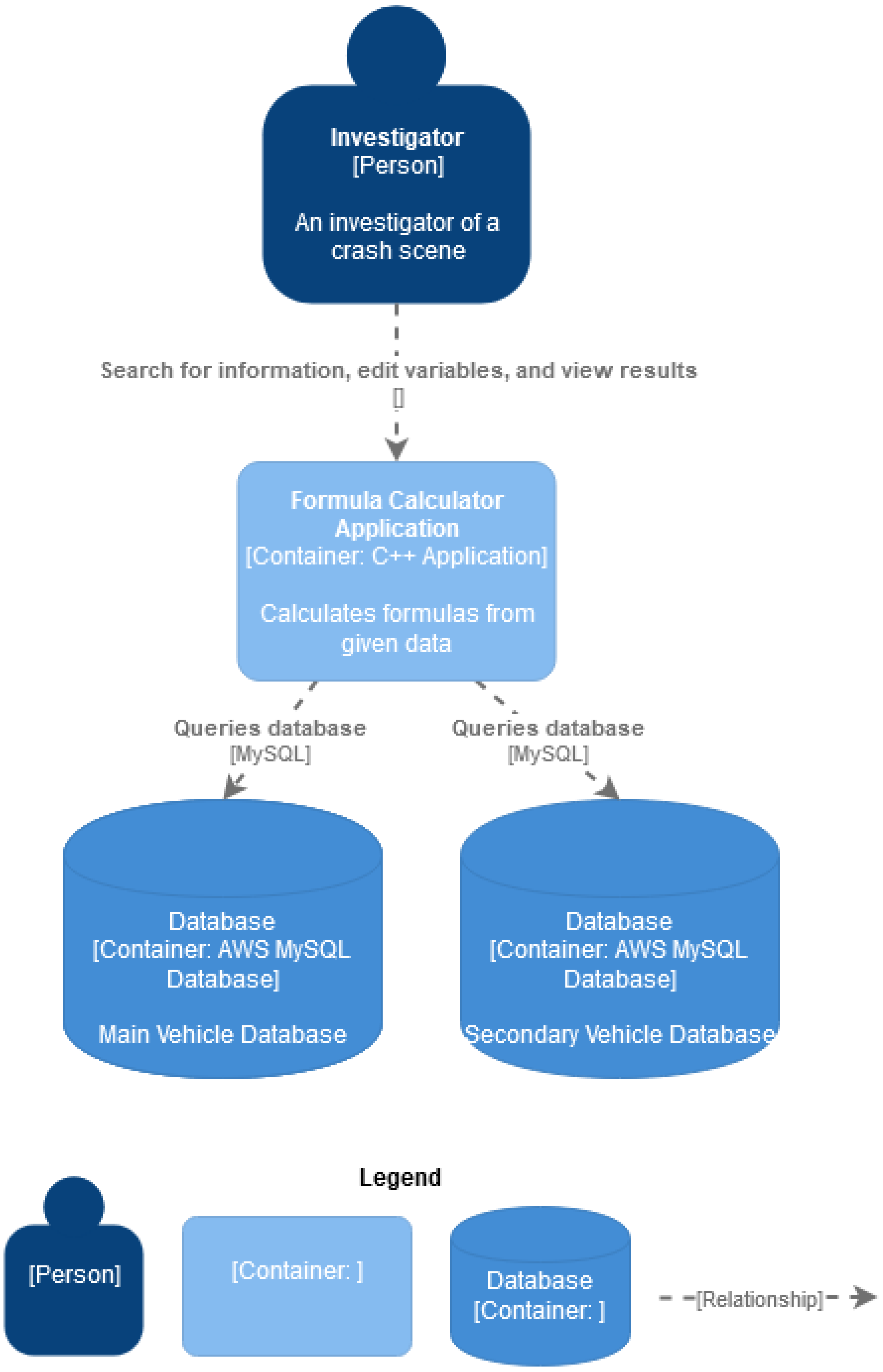
Solution Strategy

We will be using C++, MySQL, and Qt since they were used previously. The database and user interface were already created, so we only need to interface with them and perform calculations on the data.

Quality Goal	Solution Approach
Usability	To ensure the application is easy to use, we will avoid adding clutter to the user interface and will make the interface responsive and provide error messages to the user when appropriate.
Modifiable	To enable easy modification of the system, we will use an object orientated approach to programming and will ensure that all functions are documented and provide an overview for how to do an example calculation with the functions we made.
Reliability	To make sure the application is reliable, we will implement input verification and error messages to ensure that the investigator does not crash the system with invalid input.

Building Block View

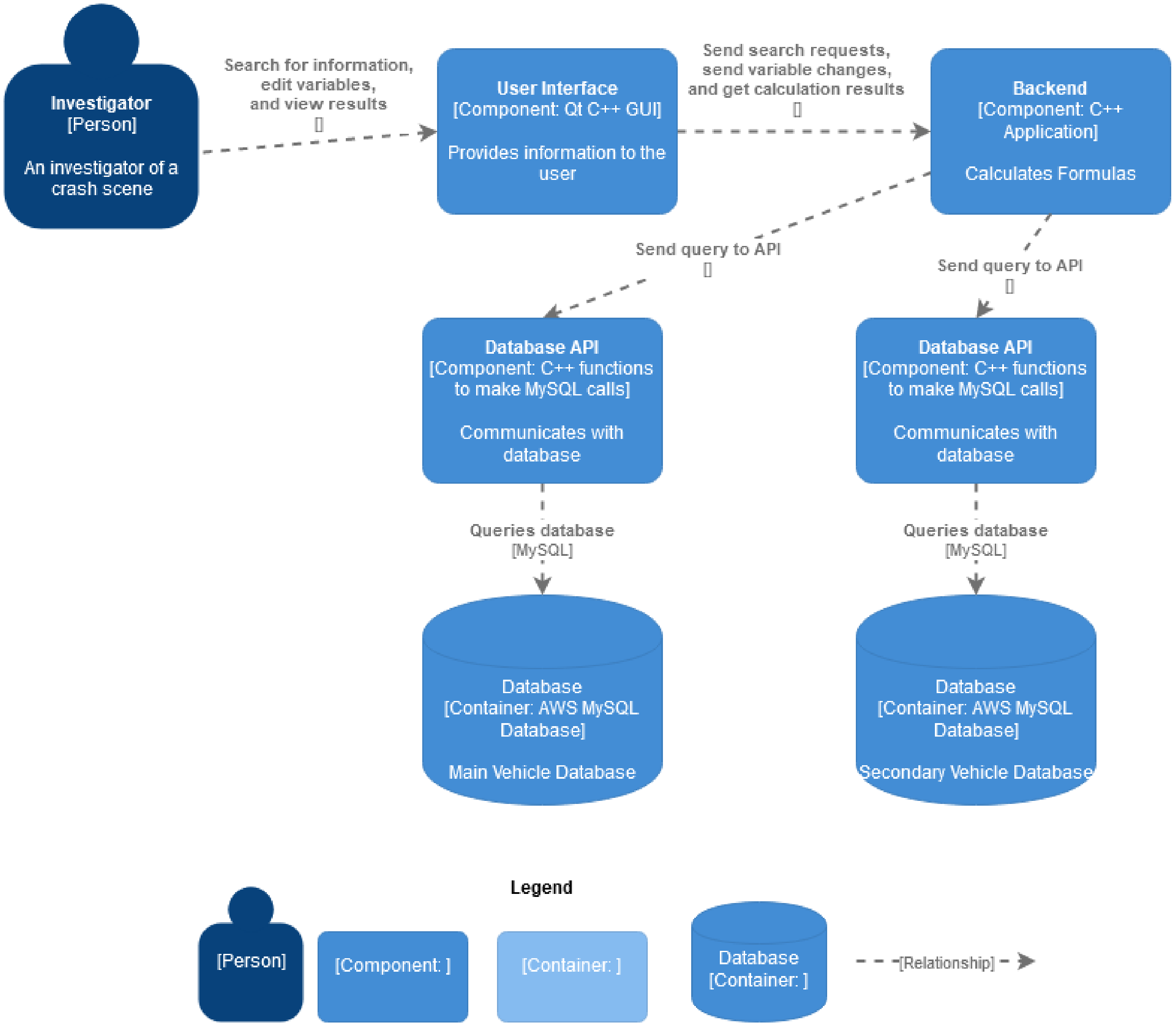
Container Diagram


Investigator
[Person]
An investigator of a crash scene
Search for information, edit variables, and view results
[]
Formula Calculator Application
[Container: C++ Application]
Calculates formulas from given data
Queries database
[MySQL]
Queries database
[MySQL]
Database
[Container: AWS MySQL Database]
Main Vehicle Database
Database
[Container: AWS MySQL Database]
Secondary Vehicle Database
Legend
[Person]
[Container:]
Database
[Container:]
[Relationship]

Technical Interface	Explanation
Investigator	The Investigator selects make, model, trim, and year, and after the selections, tabs below will automatically fill with additional vehicle data. The Investigator can also modify information, like the weight of the vehicle, and solutions from math calculations will update. The Investigator can also input data for additional formulas, in which calculations will be performed and solutions will be shown.
Formula Calculator	The Formula Calculator receives search requests, from the Investigator, and by using this data, will calculate the solutions for the math formulas. It will also receive variable changes, from the Investigator, in which the Formula Calculator will determine which of the formulas the variables belong to, and update the solutions after re-calculating. The Formula Calculator also queries the main vehicle database first, and then the secondary vehicle database in order to get information not retrieved from the main vehicle database. If certain information cannot be retrieved from any of the databases, then the data will not be used.
Main Vehicle Database	The main 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 first to obtain vehicle data.

Secondary Vehicle Database	The secondary 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 vehicle data if the main database did not have the requested information.

Component Diagram


Investigator
[Person]
An investigator of a crash scene
Search for information, edit variables, and view results
[]
User Interface
[Component: Qt C++ GUI]
Provides information to the user
Send search requests, send variable changes, and get calculation results
[]
Backend
[Component: C++ Application]
Calculates Formulas
Send query to API
[]
Send query to API
[]
Database API
[Component: C++ functions to make MySQL calls]
Communicates with database
Database API
[Component: C++ functions to make MySQL calls]
Communicates with database
Queries database
[MySQL]
Queries database
[MySQL]
Database
[Container: AWS MySQL Database]
Main Vehicle Database
Database
[Container: AWS MySQL Database]
Secondary Vehicle Database
Legend
[Person]
[Component:]
[Container:]
Database
[Container:]
[Relationship]

Component Diagram	Explanation
Investigator	The Investigator selects make, model, trim, and year, and after the selections, tabs below will automatically fill with additional vehicle data. The Investigator can also modify information, like the weight of the vehicle, and solutions from math calculations will update. The Investigator can also input data for additional formulas, in which calculations will be performed and solutions will be shown.
User Interface	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.
Backend	The backend takes search requests and variable changes from the user interface. The backend also sends search requests to the database APIs 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.
Database API	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.
Main Vehicle Database	The main vehicle database stores vehicle information to be displayed to the investigator and used in calculations by the backend. The backend queries this database first to obtain vehicle data.

Secondary Vehicle Database	The secondary vehicle database stores vehicle information to be displayed to the investigator and used in calculations by the backend. The backend queries this database to obtain vehicle data if the main database did not have the requested information.

Word Count per student:

Christopher Butler: blue

Tabitha Holloway: orange

Itai Kumengisa: green

Tyler Dewitt: