Architecture Assignment

CS 499 Fall Semester Team 5

October 1, 2018

Delta V Innovations Database

1. High Level Design

Our project is entirely focused within the bounds of the database and we will be cleaning, maintaining, and building upon it. Thanks to the efforts of last year's group, the database's tables have already been set up so we are going to expand upon the current tables and populate them with further information.

As shown in Figure 1 below, the database holds a wide variety of important details for vehicles to be accessed by work cases. The tables on the different coefficients of friction will have them calculated through a variety of factors including roadway type, vehicle speed, and road conditions. The accelerating/decelerating rates will be calculated by vehicle type and speed. Lastly, the vehicle specs table will by far be the largest and most extensive table within the entire database. Here our group will be populating this table with the vast information for each car from 1971 to present.

All our efforts will be into the database layer but it is important to realize how it'll interact with the other layers being worked on by the other groups on the Delta V Innovations project. The overall idea is to have an application that creates work cases for car accidents that can determine what exactly happened during the crash and what potentially caused it. Work cases will pull vehicle data directly from the database that'll be used in calculations that'll incorporate all the factors that caused the crash to happen.

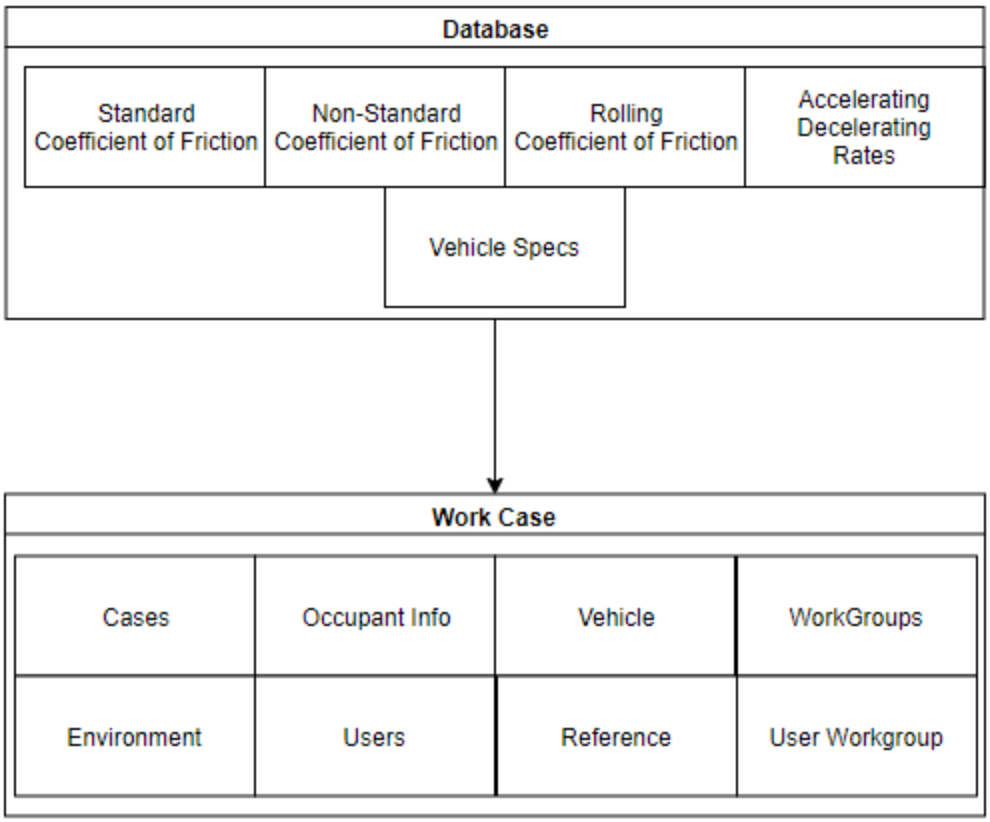

Figure 1

2. Detailed Design

Since our group is responsible for maintaining and enhancing a database, not writing code, the detailed design will be a little different. There is no class diagrams or user interface designs, but instead we created Entity-Relationship diagrams (ER diagram), relationship schematics, and a application flowchart. The ER diagram is a great way for people to organize their databases and get a visual of how everything is connected. The graph in the appendix shows the database that our group will be working with. The rectangles represent the entities, or the specific objects or “things” in the mini-world that are represented in the database. These values are generally uniquely identified and easy to represent. The diamonds are the relations, which show how two entities are linked. With our ER diagram, the ‘manages’ relation describes the

‘user’ and ‘case’ entities. It tells users the user will manage a case, and the data between the two entities will most likely be linked in the database. The ovals in the graph are the attributes, which describe properties of the entities, and in some cases the relations. The underlined values in some attributes specify that that value is a primary key. Primary keys are unique values that are able to identify the entity that they are linked to. This is the general overview of ER diagrams, and how they represent stored and linked data.

The figure below shows the relation schematic for our databases. Relation schemas are very simple models that show in greater detail the creation of the tables and how the data will be stored. The top two rows represent the ER diagram shown in the appendix, and the tables below the dark blue line represent a different database. The previous database team has already implemented this schema into the current database, our job is to populate it with information, as well as add some more columns (attributes). For the Vehicle_Specs entity we are currently working on adding model_front_overhang_mm, model_rear_overhang_mm, model_front_track_width_mm, model_rear_track_width_mm, model_weight_distribution, and model_curb_weight for vehicles from 1971 to 2019. The data is entered in through HeidiSQL or mySQL, and is mapped to look exactly like it does in the relationship schema.

Figure 2

This last design is an application flowchart. This picture was created by the previous group, and all it depicts is how the application will be used, and how the database interacts with the application itself. It also shows how the other two groups (mobile application and 3D desktop

application) will be interacting with our database. The basic overview is the mobile team is responsible for taking the crash scene data, and uploading it into the database. It is on this application, users can also access car specs such as make, model, engine type, etc. Then there is the 3D application that is responsible for crash recreation which will then work in coordination with out database to hopefully identify why a crash happened, and determine how to reduce the occurrences of crashes.

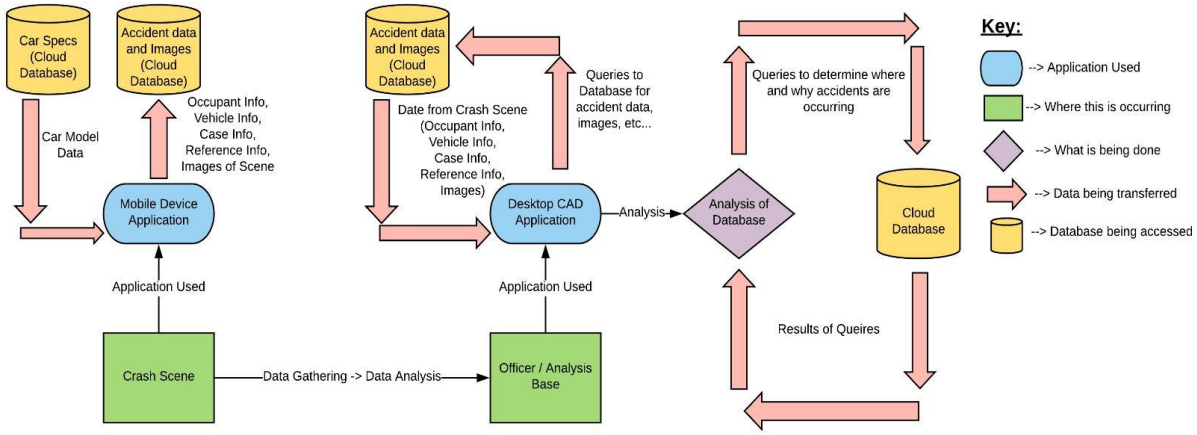

Figure 3

3. Testing

Note

Our team had a discussion with Dr. Hayes about what to do for this section due to the fact that we will not be writing any code. It was acknowledged that we will not have specific test cases for unit tests; however testing will still be conducted even if it is conducted in a different way than other team.

Note

In order to verify that the database is operating properly our team will be conducting a variety of manual tests throughout the lifecycle of the project. Each of these tests will be

conducted in order to test a potential weakness that could arise while making changes to the structure and content of the database.

The first set of tests will test to make sure that the data is mapped properly to the mobile application. Every front-end feature that accesses the back-end database in some capacity will need to be tested to ensure that the correct data is being presented to the user. These tests will allow us to know if the queries for data retrieval were constructed properly or not.

The next set of tests will verify that the database transactions are all or nothing and that the transactions are not affecting one another. All or nothing transactions means that the database calls are not operated at all if some of the parameters for the call are missing. This is important because the program will be dependent on all of the data being passed back and errors could arise if only some of the data is passed back. It is also important that the transactions are not affecting one another. It will be a major problem if importing more data into the database causes other data to become corrupt. Both of these sets of tests will be conducted by writing mock SQL statements in order to import partial sets of data into the database as well as trying to pull back partial sets of the data to the application.

The consistency and durability of the database must also be tested. When the user imports data into the database the data must first be checked to make sure it conforms to the constraints of the corresponding table column. The best example of this is variable type. In order to check this we will write mock SQL statements in order to pass in bad data into the database. Additionally, the durability of the database must be tested. It must be verified that our database will not lose data in the case of a power loss and that it is backed up properly.

The last type of tests that we will be conducting will be data security tests. Eventually, the user will be able to insert their own personal data into the database into the mobile app. This

will lead to many potential problems with users potentially having access to other users personal data. We will need to conduct manual tests using the mobile app to ensure that each user only has access to there data.

None of these types of tests have specific test cases; however in order to prove that we know how tests cases are made an example of one is provided below. This would most likely be more of a test for one of the Delta V mobile teams; however it still contains testing of the database data retrieval.

Test Case ID: 1

Test Priority: High

Module Name: Testing the "Vehicle Lookup" module of the mobile app

Test Designer: Austin Rice

Test Produced On: September 29, 2018

Summary: This test will test to make sure that when a user enters a year, make, and model of a vehicle into the mobile app the proper data is returned from the database to the user.

Test Steps:

1. Open up the "Vehicle Lookup" module on the mobile app.
2. Enter a proper year and model into the fields and hit search.
3. Choose one of the models and hit next.
4. Choose a trim and hit get vehicle info.
5. Verify that data was pull back and is listed on the screen. This data should include Model Weight, Model Length, Model Height, Model Wheelbase, Model Drive, Top Speed, and 0 to 60 mph.

4. Quality Assurance Review

Our team found a few issues when conducting our quality assurance review of parts 1-3. One of the workcases were missing on the high level overview. This was added in order to make sure all the work case scenarios are correct for the vehicle spec database table. Additionally, in the detailed design a few of the cardinalities were incorrect in the ER diagram. Lastly, it was identified that a few more test cases should be added; however there was not time to add them. They will be added in the following week.

5. Metrics

Complexity of Overall System

Doesn't really apply to our application, but our database has 13 tables populated with lots of data.

Product Size

We have planned our product based on 6 user stories. The user stories cover the 13 current tables in the database. We have one current test case for our project.

Product Effort

	Hours	Word Count
Daniel Palmer	9	2367
Austin Rice	11	2765
Matthew Stipsits	8	1800

Defects

There have been no noted defects.

6. Developer Notebook

Our team has a website through github to track the progress of our project. It contains all of the projects up to this point including information about the team member. The site can be found here: https://acri232.github.io/CS499Team5/

Appendix

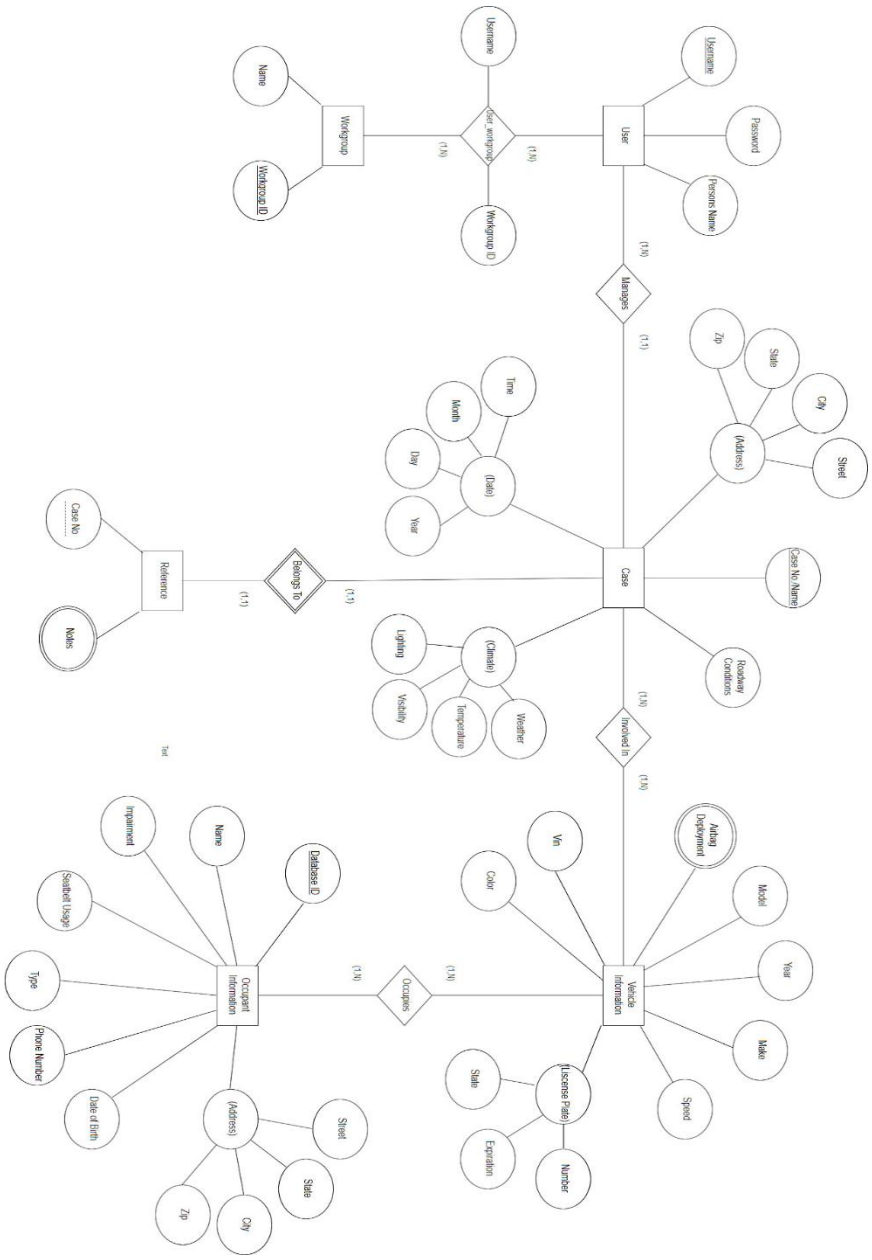

Figure 4