Team 5 (Delta V Innovation Database Team)
Maintenance Assignment
12/03/18

Team Members:
Austin Rice
Matthew Stipsits
Daniel Palmer

Customer:
Mike Flamm
deltaVinnovationsinc@gmail.com

1. Analysis

1.a Requirements or User Stories

The requirement specification document found on the team webpage has been updated to reflect what we were able to complete throughout the semester. The update is located under the corresponding sections in italics. The updated document can be found here:

https://acri232.github.io/CS499Team5/

1.b Paper Interface Prototype

When completing this portion of the assignment earlier in the semester we did not complete this since our project did not contain an interface. Therefore, this was unable to be updated.

1.c Updated Acceptance Plan

Our team used our story points created earlier in the semester as an acceptance plan for our project. The original story points were updated in order to show what we were able to complete and what was unable to be completed. The updates for the user stories can be found in italics under the “Metrics” page on the team website. The link is:

https://acri232.github.io/CS499Team5/metrics.html

2. High Level Design

2.a Architecture or package diagram

The high level design has not been altered throughout the course of the project. An update was provided for it in the “High Level Design” section of the Architecture Assignment. The update is in italics. Additionally, the update Architecture Assignment document can be found here:
https://acri232.github.io/CS499Team5/

3. Detailed Design

3.a Updated detailed design

The detailed design contained in the Architecture Assignment document found on the team webpage has been updated. A new table was added to the database and the relational schema for it was added to the detailed design. The update is in italics under the detailed design section. The updated document can be found here:

https://acri232.github.io/CS499Team5/

3.b Updated user interface design

When completing this portion of the assignment earlier in the semester we did not complete this since our project did not contain an interface. Therefore, this was unable to be updated.

3.c Updated design pattern use - If none used, so state

No design pattern was used for our project this semester.

3.d Quality Assurance Review

Our team found a few issues when conducting our quality assurance review of the update user stories, updated high level design, and updated detailed design. 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. While going through the user stories we realized that the ability for users to insert data was never fully implemented. An update on the progress of this was added in order for future teams to know what was completed. Lastly, it was identified that a few more test cases should be added. They were added in order to complete the testing portion of this assignment.

4. Implementation

4.a Source Code Listing

The code that we have written is stored on an AWS server that is not easily accessible. Copies of the php files that we wrote can be found at the google drive link located below or on the team website. The updated script code is also available there. All of the PHP files and script were written after the Coding Assignment was complete. Therefore, there is no indication of what is changed since it is all new.

https://drive.google.com/drive/u/1/folders/1Clt0SgT6HZs2oA6gZZvSecVvWS7jEqUG

As for our database host, we are using Amazon Web server where we have a single database instance that is allotted 20 Gigabytes of storage. Inserting and accessing data can be done on mySQL or HeidiSQL.

4.b Updated User Manual

The user manual is a pdf that can be found on the team website. Due to the fact that our team was working with a database and not creating a product, our user guide is different than other teams. It is very important in helping future Delta V teams understand how the database can be accessed and how it works. Since the coding assignment we have added documentation on how to access the PHP files on the AWS server instance. The Update User Manual can be found on the team website:

https://github.com/acri232/CS499Team5/blob/master/index.html

4.c Updated Admin Manual

The mobile app is available on the Apple App Store and Google Play Store. It is called “Delta V Field Lite”. Internet access is required for use. A user does not need to login in order to use the app at this time.

4.d Code Review

There were a few checklists that our group referenced in order to make sure the database we are working on met certain standards. Ncsu.edu was a great website that ran through all the basics on database management. It is really good for tips on preserving and maintaining your database and giving tips on the standards that any database should meet. Along with that website, we used scribd.com’s database design review checklist to help enhance our database. This checklist was more in depth and provided a lot of minor details that helped improve the database we are working on.

5. Testing

5.a Test Plan

Over the span of the last few weeks, plenty of precautionary measures and a lot of testing has been done to ensure our methods are correct, and the database is serving its purpose. When creating and maintaining a database, it is important to create structural tests, functional tests, and nonfunctional tests.

The structural database tests deal with table and column testing, schema testing, stored procedures and views testing. These kind of tests typically involve the elements inside the data repositories and also the storage of data elements. This kind of testing was done the most since the database is still in its early stages.

Functional tests were also important and used frequently. This kind of testing ensures that the actions performed by the end users is consistent and lines up with the desired output. The nonfunctional tests are the type that test the integrity and optimize the current database. When completing this project, it is important to include a mixture of all these kinds of testing methods so that the database has no potential holes or security risks.

5.b Test Cases and Results

Test Case ID: 1
Test Priority: High
Module Name: Testing the correctness of “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.

Result: Success. Current data is pulled back correctly and there is no discrepancies with the presented data. Current tables have not been brought to the applications GUI yet, so that testing is yet to happen. Same process of testing will occur once the GUI is updated.

Test Case ID: 2
Test Priority: Moderate
Module Name: Data Type and Size Verification
Test Designer: Daniel Palmer
Test Produced On: October 20, 2018
Summary: Each attributes type and size must correspond to its value and not exceed its limits

Test Steps: We must check the size and type of the attributes and data when:

1. We create a new table or attribute.
2. We add data into the database.

Result: We created the new table Vehicle_Specs_Additional, and in it we have 18 attributes. All attributes are given the type, varchar, as that is what the previous semester had done to most of their tables. The length was set to a default value of 45, which the previous semesters group also did with their tables. When inputting data, we made sure that the roughly 22,000 entries were within these limits, and correlated with the given type.

Test Case ID: 3
Test Priority: Moderate
Module Name: Bad Data Inputs
Test Designer: Daniel Palmer
Test Produced On: October 20, 2018
Summary: Every database needs to be able to handle bad data entries and respond in an appropriate manner.
Test Steps:

1. Insert improper data types and sizes to see how the software and database handle it.
2. Determine a template each file should look like that gets imported into the database so that the process is easier.

Result: We created the new table Vehicle_Specs_Additional, and in it we have 18 attributes. All attributes are given the type, varchar, as that is what the previous semester had done to most of their tables. The length was set to a default value of 45, which the previous semesters group also did with their tables. When inputting data, we made sure that the roughly 22,000 entries were within these limits, and correlated with the given type. As for error handling, given the file has more data than the database accepts, HeidiSQL's software is able to handle that by parsing excessive lines of data. Each file that was imported into the database was through HeidiSQL, and followed the format given in the current database.

Test Case ID: 4
Test Priority: Low
Module Name: Data Mapping
Test Designer: Daniel Palmer

Test Produced On: October 15, 2018

Summary: Whenever the user submits a form on the application UI, it triggers a CRUD (Create/Retrieve/Update/Delete) event at the backend.

Test Steps:

1. User populates field with credentials or vehicle information.
2. Backend verifies that information and one of the following different actions will happen.
3. If information has not been created, then it creates it.
4. If information has already been created, then it updates previous information.
5. If user needs information, then it retrieves it.

Result: The user database tables are on low priority and not been created yet, and this is mainly where we would do this testing. Other areas could be when a user requests information about a certain car, and we have already verified that this works correctly. New tables that are created will need new php scripts, that should be able to be copies of other similar php scripts. Those scripts have yet to be created.

Test Case ID: 5

Test Priority: Low

Module Name: NULL vs empty vs N/A data representation

Test Designer: Daniel Palmer

Test Produced On: October 22, 2018

Summary: When given data values that are empty strings, there are many ways to express this value, and knowing how to handle them is very important.

Test Steps: Upon creation of a new table, test the use of NULL vs N/A data representation to see which better suites the table we have created.

Result: For the table we created, we decided on using N/A to represent empty data. We did not set a default value so that it would not default to null. Instead we will put N/A to represent a Null. Null values can be messy to use especially when trying to query something and using relational algebra to get certain values.

Test Case ID: 6

Test Priority: High

Module Name: Additional Vehicle Spec Data Population

Test Designer: Austin Rice

Test Produced On: December 2, 2018

Summary: Make sure that the Vehicle Spec Additional data is being populated in the mobile app.

Test Steps: Navigate to the mobile app tab and choose a year, make and model. Verify that the correct data is pulled back.

Result: This test was a success after altering the SQL queries a few times. All of the data is being pulled back properly.

6. Technical Metric Collection

6.a Estimated Lines of Code changed

After learning our “twist” from our customer we realized that a lot of code would need to be written. We estimated that the script in order to update the files would be about 500 lines of code. Additionally, we had to write SQL queries in PHP files. We estimated that we would need to write 3 php files with each having roughly 50 lines of code. Therefore, our overall estimation for lines of code changed/inserted was roughly 650 lines.

6.b Actual Lines of Code

Files:
DBConfig.php - 15 lines
getModel.php - 51 lines
getVehicleInfo.php - 57 lines
getYear.php - 38 lines
File Script - 325 lines
Total New Lines : 488 lines

6.c Complexity of Each Module

Within the database, we exclusively worked with two tables throughout the semester:

VEHICLE_SPECS: 12.5 MB (This table was already created at the start)
VEHICLE_SPECS: 3.5 MB (This is the extra table we had to create and populate with the new data)

6.d Overall Complexity

The DeltaV Test Database has a total of 22 tables within totaling 16.4 MB.

6.e Product Size

We have a total of 6 user stories and 5 test cases.

6.f Product Effort

	Hours	Word Count
Daniel Palmer	45	5359
Austin Rice	36	6100
Matthew Stipsits	20	5028

6.g Defects

To accomplish our primary objective of populating our database tables with the relevant data on vehicle specs, our customer provided us with a collection of vehicle information from 1971 to 2019 collected by the Canadian Association of Road Safety Professionals. Our main conflict is that the format already created within the database differs from how all the vehicle specs gathered is formatted. We built a script to properly convert the data, however, there are still some discrepancies with vehicle make and models not matching in the database.

Demonstration

The demonstration of our project was done in video form during our presentation to the class. Additionally, the Youtube screencast and final poster slide can be found in the found in the "Project Outcomes" flash drive and on the team website.

Developers Notes

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/

References

https://www.scribd.com/doc/10452236/Database-Design-Review-Checklist
https://www.lib.ncsu.edu/sites/default/files/dmp_checklist_Resources_sept2014.pdf