Feature: Service Log

Feature Developer: Jason Barber

Date Review Submitted: 3/24/2024

Peer Reviewer: Rainier Getuaban

Date Review Completed: 3/25/2024

Major Positives

1. ***Use of objects and models between layers***

   Using objects and models create an easy way to transfer a sizable payload between layers. Without using the models, every parameter would have to be passed between different layers multiple times. Having the models also allows for different implementations of interfaces to access information that may or may not have been used by another implementation without having to create a new interface or change existing interfaces.

2. ***The overall flow between layers are clear***

   The small descriptions under each function call help the developer understand what each function call is supposed to do and who should call it. Also, including the “script.js” layer helps a lot with understanding that some actions would not be part of the html that the user interacts with directly. It adds another layer that the developers do not have to visualize because it is already there.

3. ***Validation of input parameters and output results***

   Validating input parameters between layers is always a good safety measure. This allows for data validity across multiple layers, meaning if one layer passes in compromised data, another layer can check and throw errors if the data is invalid. Validating the output results is also a good way to check the types of errors that are being thrown. It can allow for more context when logging operations and for users providing inputs.

Major Negatives

1. ***The fail scenarios are incomplete or missing***

   Having the fail scenarios are dictated by the functional and nonfunctional requirements written in the BRD. By not including them, it can slow down development by making the developer have to go to another document to find how a function should work in the case of a failed scenario. It also slows down testing by making the developer cross reference the BRD again to ensure that the failures are legitimate failures and not false positives.

2. ***Ambiguity where some values are retrieved from***

   Across the design, there are some places where some values are not stated where or how certain values are retrieved. For example in SL-9, it is stated “Based on the 3 most recent Maintenance Reminders average the total amount of miles driven each month,” but no context on where the 3 most recent maintenance reminders are retrieved from. There’s also the scenario where there are no recent maintenance reminders making the average time between service reminders being 0.

3. ***Oversight from the BRD to SL-6 and SL-7***

   The BRD states that a user can delete and modify an existing Service Log entry, but it never specifies that it has to be from that user. If this were to be true, then any user can delete any Service Log entry, regardless of who owns it. As an oversight of the BRD, the design of SL-6 and SL-7 should reflect that a user can only modify or delete their own vehicle’s Service Log. This would most likely require an input parameter being a user’s

account model to check both their user account and their vehicle's vin and see if that owner owns that vehicle. This portion can be done using join sql statements.

Unmet Requirements

1. ***Photo Evidence of Vehicle***

   There is no mention of photos within the design of SL-1, SL-2, SL-3, and SL-4 when it is mentioned that users can upload photos in the BRD. Assuming that the option to upload the photos is still planned to be given to the users, the design does not reflect that.

2. ***Pass Requirements of SL-5 and SL-8***

   SL-5 and SL-8 have some pass requirements that state how many logs of a vehicle that a user can view at a time and how they are sorted. This can be fixed within generating the sql using some statement like the following:

```
SELECT * FROM ServiceLog
WHERE VIN = @VIN
ORDER BY Date
OFFSET (@pageNumber * @numberOfItemsPerPage)ROWS
FETCH NEXT @numberOfItemsPerPage ROWS ONLY
```

Design Recommendations

Note: The number of the recommendation corresponds to the major negatives

1. ***Finish the Fail Scenarios***

   Even though it may be a pain to finish the fail scenarios of each of the user stories, they are important and can shorten the amount of development time. With good enough

planning, the implementation should just be a direct translation from the design to the actual code. They also help enforce any business rules such as an operation taking longer than 3 seconds or 10 seconds to execute.

2. ***Specify where some values come from***

   Try to at least say where some values are retrieved from. For example, if you call from an API, don't just say that you will retrieve from any API, but rather you will retrieve from this specific API and will use a specific call from that API or will retrieve from the main database and from a specific table. This will help in development so you don't have to go back and forth between trying to find where the right data is located.

3. ***Limit what values a user can change***

   Even though it is not explicitly stated in the BRD, limit which users can edit which service log. Again, I suggest passing in the User Account Model as a parameter throughout the layers because it can be used to see if a user owns the vehicle they are editing the Service Log for. Even though this does add more complexity when implementing modification and deletion of Service Logs, it does provide more security.

Test Recommendations

1. ***Functional and Non-Functional Requirements***

   I cannot stress this enough, but ensure that all pass requirements are tested and test any potential way a user can make the system fail, especially in the back end. Ensure that any input will be validated going in, and any response will be checked for errors going out.

2. Stress test multiple service logs

When testing SL-1, SL-2, SL-3, and SL-4, stress test the system by calling the creations multiple times. Also, stress test that you only get 25 service logs per view in the front end and that each retrieval of 25 service logs are ordered by date and can be paginated in the back end.