Feature: Inventory Management

Feature Developer: Rainier Marlone Getuaban
Date Review Submitted: 4/7/2024

Peer Reviewer: Jason Barber
Date Review Completed: 4/10/2024

Major Positives

1. A major positive based on the sequence diagram is the inclusion of including how Security works in the application. While it only shows a successful case of being authorized to access the microservice, previous LLD's didn't represent this and made an issue when retrieving usernames in the application since we didn't have the JWT. In addition, additional security is added by validation of parameters in the manager layer though additional validation in the front - end would prove beneficial in the checks of simple syntax checks and such.
2. The sequence diagram includes ready to use SQL statements which allows for easy transfer to the Sql Target in the code later on. By doing this, the developer has an idea of the SQL should execute and even troubleshoot ahead of time if the developer notices an issue with the statement.
3. An issue we ran into recently was the extraction of data from our IResponse return values attribute which the Inventory Management Sequence diagrams covers now by extracting the information and converting to a JSON object between the service and manager layers. This allows for ease of use with extracting the information in the object whether in the back - end or front - end of the application.

Major Negatives

1. As explained during class, the use of Async functions may prove beneficial. The current status of Inventory involves no mathematical or complex functions (in other words CPU

bound work) thus including Async functionality for our I/O bound work would increase how our machine optimizes our code.

2. Aside from noting that a user initiates an event or HTML is injected into the front - end, the design lacks any details of how the Front - End will interact, display, or modify with the data. By doing this, searching and viewing look very similar on inspection aside from inclusion of different arguments, yet the AJAX call or rendering of view is incomplete or missing entirely. This also extends to unmet requirement of certain front - end logging that is required
3. While a folder is included detailing “Editing Inventory Management” , an BRD oversight we missed was the actual modifying of statuses. We mentioned how the status should be displayed but never of where and how they will be changed leaving it ambiguous.

Unmet Requirements

- IM - 1
    1. If a Vehicle Profile is assigned the Needs Service status, the Vehicle profile must display:
        a. Service Needed
    2. If a Vehicle Profile is assigned the Unassigned status, the Vehicle profile must display:
        a. Warning message that Vehicle has not been assigned a status
- IM - 2
    1. If a Vehicle Profile is assigned the Needs Service status, the Vehicle profile must display:
        - Service Needed
    2. If a Vehicle Profile is assigned the Unassigned status, the Vehicle profile must display:
        - Warning message that Vehicle has not been assigned a status

- General
    - IM - 1 and IM - 2 Front - End logging messages are not detailed in the LLD

Design Recommendations

Note: Design recommendations correlate to the number of major negatives.

1. Based on our in class discussion, if time is allotted, to change the feature to include async functionality would prove beneficial for the application and for the feature itself to be good experience in asynchronous programming.
2. The front - end generation and calls need to be detailed in the LLD in order to give the developers a solid understanding of how the application should behave on a lower level. By not including view generation, checks and such, the front - end is left ambiguous and may not correlate to the design we have used so far in the application. That being said, for the front - end, I would recommend creating the view based on the IActionResult returned so we can generate different views depending if it is unauthorized, bad request, or other responses.
3. Due to not including a crucial user story into our BRD, I would recommend making a meeting with our client to discuss adding a user story to the BRD in order to cover missing bases in the feature. If it is approved, we should implement back and front end checks that make sure the correct values are being shown and returned/

Test Recommendations

1. Based on the nature of the feature, I would implore to focus on front - end testing. In particular I would focus on these items: editing statuses, correctly generated views, and producing warning messages.
2. Make sure to test the service layer that the correct objects or values are being returned based on the function call as they are essential to generating the correct views