Feature: Vehicle Profile

Feature Developer: Rainer
Date Review Submitted: 3/4/2024

Peer Reviewer: Jason Barber
Date Review Completed: 3/7/2024

Major Positives

1. The flow of all the diagrams are overall clear and precise. Each layer is defined clearly along with clear naming conventions that follow coding standards. Allows for separation of concerns and layers having a clear goal.
2. Implementation of IResponse allows for quick and easy communication between layers as the error message attribute acts as both a way to communicate to the client about errors or failures in the application and as well as to developers in testing.
3. Instead of passing numerous arguments in separate parameters, the design, usually, consolidates all the attributes in a Model that is used throughout the design allowing for good reusability.

Major Negatives

1. While this is clearly an oversight of BRD requirements and lack of communication with a client, the design should include a modifying Vehicle Profile story or feature. Vehicle profiles inputs such as the color, license plate, description, and photo evidence of the car should be modifiable in order to promote client and customer satisfaction but also not resort to only having to delete a vehicle profile if they accidentally enter a wrong field.
2. In VP - 2 , while the design does show updating the data to the database based on user input, the design shows a lack of detail about the user receiving the data from the

database. In order for a user to know the current preferences selected, the user should receive the current version of his options in the database and thus can make an informed decision on his current settings.

3. Throughout the design, models are used when only certain data from those models are needed or models are not used when they should be. For example on VP - 1, the controller takes in 7 parameters of type ranging from string and int with a final IUserAccount object. While they do explicitly define what is needed, they add bloat to the method signature, less flexibility, and a dependence on the order being passed correctly. In addition, in order to add a Vehicle Profile to a user, only a username is needed, but an entire object of IUserAccount is passed instead which is not needed as we only need the attributes.
4. In the Design of VP - 4, only a IUserAccount model is passed into the controller to which a ISqlParameters model is called. The design fails to incorporate if a user decides to move past the first 10 Vehicle Profiles and only displays the first 10. By doing this, we not only unmet requirements but also a lack of a way for a user to navigate between their first 10 Vehicle Profiles.
5. Although this a nitpick and may be a lack of detail in design, the design for VP - 5 details the VehicleProfile entry point creating a call to the ServiceLog manager and is implied to call the Service Log microservice. While it is good to showcase this for future development of the Service Log feature, the design should detail or note that the ServiceLog controller is being called instead to adhere to the separation of concerns and mirco-service architecture.

Unmet Requirements

1. **Photo evidence of vehicle**
    - **Must be JPEG or PNG file types**
    - **Can be multiple photos of different angles of the vehicle**

As shown in the method signature on VP -1, a picture uploading or parameter isn't shown on the method signature. While the requirement isn't imperative to the function of the feature as a whole, the requirement is missing.

2. **Pass requirements VP - 1**
    - **For a regular authenticated user they can only make up to 50 vehicle profiles**
    - **For a vendor/renter user they can make up to 50 vehicle profiles**

As stated in the major negatives of the design, the design fails to mention checking for the current amount of Vehicle Profiles under a user in the design. This can lead to users being able to create past the 50 profile limit.

3. **Once they are done they can click "Publish" where the chosen information will become visible to any end-user and also be saved in the data store**
    - **The changes must be done within 3 seconds**

Since the Vehicle Marketplace feature has not been implemented and is scheduled to be implemented on a later date in project planning, the requirement is unable to be tested as we don't have a way for users to 'share' their vehicle profiles.

Design Recommendations

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

1. Due to the lack of ability of users being able to change their Vehicle Profiles after creation, I suggest creating a Vehicle Profile modification ‘edit’ button in order for users to make this change. The design should follow very similar to Vehicle Profile Creation and instead of creating a new entry would replace the current status of the database. If a field is left blank, a variable that can be identified as a special case will result in that field not being modified. To that point, special fields such as VIN, mileage, make, or model will not be able to be modified. While the other attributes such as VIN, make, and model are unchangeable due to the car not being able to change these, mileage can be modified but will already be updated when service logs are entered into the Vehicle Profile.
    - **Positive:** Increases customer flexibility and ability to change details of the car when needed.
    - **Negative:** Introduces a new user story and set of requirements to be checked for adding both complexity and time to the feature.

2. In order to not force customers to remember their current settings, we should display the current settings of the IsViewable to the user via a GET request to the user by grabbing the Vehicle Profile unique identifier.
    - **Positive:** Allow users to easily interact with privacy settings concerning their Vehicle Profile.

- **Negative:** Adds a small degree of complexity and has to implement a view for the items.

3. A singular model for creating a Vehicle profile should be introduced that holds the parameters listed in the method signature in VP- 1 aside from the UserAccountModel. This would allow for the model to be constructed before calling the controller and when passed allow for easy method signature and accessibility of the data. While not strictly associated with VP - 1, the design should benefit from the username being accessible in the client and inside of calling an entire object for one attribute, instead requiring a string username attribute. This allows for easily finding other data associated with that user in data tables for deleting, and creating inside of having to find a user ID.
    - **Positive:** Allows for central way to access and retrieve data related to Vehicle Profiles
    - **Negative**: If new parameters are added the model has to be updated or a new class has to inherit from it.

4. To allow for the controller to distinguish what selection of Vehicle Profiles the front - end needs, an additional parameter should be added like an **int select.** If passed as one, which would be default, the front - end would pass the number one and retrieve the first 10 Vehicle profiles. Otherwise, if a user toggles to the next page, the front - end will pass a different number depending on what page they land on with a maximum of 5. The max is because only 50 Vehicle Profiles can be created at a time and thus should only display 50 or less.

- **Positive:** Allow for users to toggle between their 50 Vehicle profiles
- **Negative:** Limits amount of pages that can be accessed and if requirements are changed and allow for more Vehicle profiles, code will have to be changed.

5. Instead of calling the Service Log Manager from the Vehicle Profile entry point, after the fetch call in the Javascript is finished for grabbing Vehicle Profile details, initiate another fetch call to the Service Log Controller.
    - **Positive :** Keeps separations of concerns and managers only access their own microservices
    - **Negative:** Have to add in handling of success and unsuccessful requests before calling Service Log service on Javascript.

Test Recommendations

1. For checking the business requirements with VP - 1 and creating a Vehicle profile, I recommend testing multiple accounts when a user creates a vehicle profile with passing a VIN for the API and when using their own make and model. The testing will test the ability of the manager layer to catch invalid make and models as well as bad api calls and proper error handling. This also includes after creating a vehicle profile to have said profile available on the page immediately after.
2. While not mentioned in the BRD, a standard held by the database is only being able to have one VIN associated with an account. Thus when another Vehicle Profile is created with a VIN, that VIN should not be able to be used for other Vehicle Profiles on other accounts.

3. For testing viewing Vehicle Profiles details, while unit testing will be important to stress end to end testing for this feature as it will include the service log later on thus the display should be functioning properly and with the feature in consideration.
4. For the Vehicle Profile creation feature, I recommend testing thoroughly that only 50 vehicle profiles can only be created through unit and end to end testing. In addition to this, properly only displaying 10 vehicles each time a view is changed thus adhering to the requirements set in the BRD.