Feature: Artist Profile Calendar

Feature Developer: Jason Jitsiripol
Date Developer Submitted: 3/12/2024

Review by: Kihambo Muhumuza
Date Reviewer Completed: 3/17/2024

Major Positives:

1. The design is consistent throughout the entire layout with each case having the same layers. Also, each case is specifically depicted for clear reference to the situation at hand.
2. The design is really easy to understand, having brief descriptions of how each layer communicates with each other and through which functions they communicate with each other.
3. There are quantifiable ways to test each case. For example, there's the **Data Access Layer User DAO** that says. "If the time to delete the gig exceeds 3 seconds, then the authentication was unsuccessful," in the first case **Failed Delete Gig When Exceeding Time**.

Major Negatives:

1. The design doesn't seem to have a *lot* of quantifiable ways to test out each case. For example, in the description between **Data Access Layer User DAO** and **Data Store** in the **Failed Edit Gig on Retrieving Data** case, how will we know for sure that the gig data can't be or hasn't been sent back?
2. In our website specifically, there is an OTP that is sent to anyone trying to log into their account. The OTP isn't mentioned anywhere during the authentication.
3. The design is simple and easy to understand, but it could bite you back later if it isn't more granular. There isn't much detail in between layers for this feature, so it would help to make everything as detailed as possible.
4. The method signatures don't show the return types, nor do the return arrows contain enough information about the method to determine those return types. Like in between your **Service Layer** and **Data Access Layer** in your **Failed Delete Gig when Exceeding**

Time case, what’s the data type of “gig”? Also, what is that function returning by the time that function is passed on to the data store?

5. I’m not exactly sure what the Service Layer is doing. Elaborate more on what this layer is supposed to be doing so that it’s clear.
6. There isn’t a controller layer. There’s a UI layer, but is that the front- end? Specify where the controller and front- end communicate in the diagram to make it more clear to yourself when you’re coding.

Design Recommendations:

1. To make testing easier, make more of the in- between layer descriptions as quantifiable as possible. That way you can see if what you put in your code is up to scale with what's in your design. For example, how long should it be after the data access layer sends the gig to the data store so that we know that it's working within reasonable time and isn't buffering or slow?
2. Make everything in the design more granular so when you're coding things out, you have a more full understanding of where to direct your code and debugging. For example, you probably don't have the function yet, but what's the name of a function that would send the gig data to the data store? Having an idea of what the name of the function will be will take less time when creating the function when you're coding.
    a. Another side note in regard to granularity is implementing helper functions to help track where the program may have an error or bug when it comes up. I would create smaller functions to help divide the code up into more manageable pieces. For example, **getGigData()** could be split up into more functions like **getTitleOfGig()**, **getDateOfGig**, and **getStartTimeOfGig()**.
3. In terms of presentation of the overall design of your LLD, add a bar over the layers that are currently active. It should be a vertical bar in the middle of each active layer.

Test Recommendations:

1. Based on the order of your design's test cases, I would make it so that you start with the success case and their respective failed case first. For example, I would start with **Success Post Gig**, and then follow the case design for **Failed Post Gig** when Exceeding Time. Another example is following the design for **Success View Gig**, and then following the case design for **Failed View Gig when Exceeding Time,** and so on.