VERIFICATION, VALIDATION, AND CODE REVIEW

cRAFTers

Contents

Verification

cRAFTers has created a test suite to test the DatabaseManager for its functionalities and implementation to date (2017-11-20). The test suite currently tests the backend creation of questions, assignments, and users. 5 testing methods currently implemented (please compile the project and run test suite under L01_01/App/test/backend/DatabaseManagerTest.java).

Based off of the tests and our design of the program, the tests reveal that the program is running OK and since DatabaseManager takes into account the functionality of different components in the program, our code works. However, the program still does not perform as the client has requested (to be discussed in Validation). cRAFTers has been able to successfully piece together moving pieces but it has not tied together all.

This verification test reveals that the current code in the program is functional, but requirements consistency requires improvement. To remedy this and work towards the completion of the project, cRAFTers has decided it needs to focus more on the interconnectedness of the tasks breakdown in accordance to what user stories are prioritized vs. the user stories that are epic (talked more detail in Validation).

Validation

In order to conduct the validation of our code, we had to revisit the task at hand and look back at the roots of what brought together this project. This required us to rethink the problem statement of our client and match what we've done thus far against the requirements.

Firstly, our program provides students the opportunity to retake assignments. When performing a manual test do determine the user acceptance with dummy data inserted into the database, I, as a student, am able to login, view the assignments available, the ones I've taken and my mark in those, and the ones that are available for retaking. Our client, Sohee Kang, wanted to provide the opportunity to students for retaking assignments until the end of the semester but with a catch.

The catch is what cRAFTers still needs to develop – and that is randomizing values in addition to randomizing the questions. For example, the questions in the assignments are able to randomize its order so that option A can be option D the next time you take the assignment. However, the values of the options are unchanged and is counter-intuitive to the student's learning if and when they decide to retake an assignment.

cRAFTers are currently working on an API for the program to randomize the answers with respect to the question being asked. In order to achieve this, it is required to have different algorithms in place for the program to automatically calculate the correct answer. The UI will handle the input from the user and the API will handle what to return as output to check whether the input is the correct answer based off of the randomly generated values.

Thus, in order to account for the needs of all the stakeholders, it will allow our client to simply add questions to the database and define parameters as to how the questions should be marked and the program will take care of the rest. This way, the student is able to re-try the questions with different values to aid their learning and our client will be able to focus more resources on consistently improving the program.

Code Review

To conduct the code review, cRAFTers established to keep the guidelines based off of reliability, maintainability, and readability of the code. Some of these aspects were discussed in the previous 2 sections, but here you will able to read the individual code review of all team members of cRAFTers.

Code Review Debrief Meeting: To communicate our best practices and expertise, we discussed them in the code review debrief meeting which you can view here.

Code Review Strategy: In order to conduct the code review, because everyone's code is interconnected however the team members worked at different times, before the completion of a sprint we would review each other's code and push a working copy of the program into the development branch, tying all components together. The code review strategy would involve checking if the code is understandable just by reading it, is there a trait of following proper naming conventions, is there copy & paste, how is our structured based off of the design, and most importantly, is it working?

Individual Code Reviews: Following are the findings and feedback of each of our team members, before and after our code review/debrief meeting:

- Adrian:
    - Currently is very obvious who did what just by looking at the source code, need to adopt shared conventions and standards
    - We are not making good use of reusing some UI elements, like the back button have the same functionality, but we don't have the behaviour shared
- Felicia:
    - Some functions have duplicated code in DatabaseManager, we should think about how to refactor them to minimize the duplicated code. Also we should split some of the functions to different classes for a better design pattern (e.g. DatabaseGetter and DatabaseSetter).
    - Some components have the same behaviours across all pages (e.g. all buttons change color when hovers over) so we can have an abstract/interface class for those behaviours.
- Rene:
    - Comments and docstrings in the code not there. Sometimes hard to follow along what's happening.
    - Functions have a specific role they play, but it is not explicitly stated via docstrings or comments. We need to work on making it more easier to follow so future contributors can easily get started if the need arises.
- Talha:
    - Documentation of our overall code definitely needs to be looked over by the creators of the code, sometimes it takes a while to understand the code of my teammates due to lack of commenting or javadocs.
    - Copy & paste tends to happen for UI, mainly because many parts [visually] are similar to keep consistency. However, we need to do a better job in reducing the reuse of code and introduce interfaces or abstract classes where appropriate.