VERIFICATION, VALIDATION, AND CODE REVIEW

cRAFTers

Contents

Verification 2

Validation 3

Code Review 4

Verification

cRAFTers has created a test suite to test the DatabaseManager for its functionalities and implementation to date (2017-11-27). The test suite currently tests the backend creation of questions, assignments, and users. 7 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.

Since our last revisit to our validation techniques, cRAFTers acknowledged the fact that the program fails to randomize values/questions for students taking assignments. If a student were to retake an assignment with the same questions with unlimited attempts until the due date, the student would be able to score 100% regardless. Thus, cRAFTers created an API for the program in which the instructor can select from 3 types of questions (simple math, easy statistics, hard statistics), decide whether they want it to be multiple choice or short answer, and the program will automatically generate random values every time the student attempts the question (i.e. new values will be generated if the student chooses to retake that assignment). The UI handles the input from the user and the API handles 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 allows 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 cRAFTers' code review strategy and its implementation.

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: Before a team member wants to merge into the development branch (it was established in our team agreement that only working code that compiles and runs is allowed to be pushed to this branch), they would issue pull requests where every team member has to approve before their working branch is merged with development. Here, each team members can provide feedback to improve/refactor the code as necessary towards the code that has been modified since the last merge into development. Once the code has been merged into development, the team would meet and discuss over-the-phone or in-person before development is merged into master. These merges would typically occur before a major deliverable is due.