Library System

Software Quality Assurance Plan and Report

Final Version

Date: November 2021

GitHub Link: https://github.com/McGill-ECSE321-Fall2021/project-group-20

Project Group 20

Signature Page

Prepared by:

Harsh Patel 2021-10-11

Reviewed by:

Alexandru Bangala 2021-10-11

Abdelmadjid Kamli 2021-10-11

Dan Hosi 2021-10-11

Ehsan Ahmed 2021-10-11

1.0 Purpose

The purpose of this Software Quality Assurance Plan and Report is to define appropriate test coverage criteria we wish to achieve for the business methods implemented in the backend. In this report we will be discussing the unit testing approaches used to achieve our target coverage criteria. We will also divulge the details about any other means of quality assurance we have used during the duration of this sprint.

2.0 Test Coverage Criteria

As a team we have decide that the appropriate test coverage criteria would be around 75%. We decided to set this goal because 75% represents a big enough percentage where we can move forward with confidence into the next sprint that most methods won't require further debugging before use. Another reason is that as a team we concluded that a coverage of 75% would be easily achievable if everyone wrote unit tests for each one of their methods in the service class and if they had written test for enough branches/edge cases for those methods. This forces team members to test edge cases and encourages the implementation of more complete tests, since team members have a set objective to work towards.

3.0 Unit Testing Approach

The service unit testing was done with Mockito, and we had access to code coverage using the EclEmma plugin. The approach taken by the team for this project was that every member should test every method inside the service class they were assigned to, since they implemented the tests and had more familiarity with them. We did this because it is important for the deliverable and the future (ease of use) that all service methods are proved to be functional. Each team member before writing a test would look at the method being tested and how it works, so they

can correctly write the tests, this is in effect Whitebox testing. After doing that the team member will consider how many edge cases the method in question has and how many he wants to test.

4.0 Report Test Coverage Per Service Class and Member

Alexandru Bangala:

- Customer: 81.2%
- Employee: 86.6%

Dan Hosi:

- Library System: 60.6%
- Address: 65.8%
- Calendar: 84.3%

Ehsan Ahmed:

- Title:84.2%
- Author: 82.9%

Abdelmadjid Kamli:

- Item: 90.5%
- Movie: 91.0%
- Book: 91.2%
- Music Album: 91.0%
- Archive: 90.5%
- Newspaper: 90.5%

Harsh Patel:

- Hour:87.3%
- Booking:82.4%
- Event: 91.0%

5.0 Explanations for Some Coverage

As we can see the test coverages above, almost everyone was able to attain the goal of 75% set by the team. However, for two classes, Address and Library System, one of our team members was not able to achieve this goal. This can be easily explained by the fact that this team member was overloaded during the weeks of the deliverable with personal and school related activities. For example, in terms of school related issues, Dan had spent over 30 hours on DPM and being the team lead for McGill Rocket Team, which made his burden way bigger than the rest. As a team we concluded that given everyone's situation, we managed to deliver something we are happy with.

6.0 Other Means of Quality Assurance

Every 3 days or so, the team members got together to review all new pushes to the main branches. Every team member will take turns sharing their screen and presenting their code to everyone. After the short presentation each team member will review the code and point out problems in the implementation and give their opinions on the quality of code. After the review, team member will decide if things should be re-done and if we should assign extra members to help another one more in trouble or if everything is good and we can all proceed with our work.