Test Summary Report

For

IUS Nursing Checklist

1 Purpose 2

2 Application Overview 2

3 Testing Scope 2

4 Metrics 2

5 Types of testing performed 2

6 Test Environment & Tools 3

7 Lessons LearnED 3

8 Recommendations 3

9 Best Practices 3

10 Exit Criteria 4

11 Conclusion/Sign Off 4

# Purpose

To explain the testing of the IUS Nursing Skills Checklist website.

# Application Overview

The application is for students and administrators of the Nursing department. The students can signup/login to the website to manage skills by use of a form on the website. The administrators can lookup student data and track their progress.

# Test Scope

The testing is aimed at defining long lasting usage with minimal application management over the course of the websites use. This means the user interface, ease of use, data access layers should all be extensively tested to assure they perform as intended.

# Metrics

There are five main test cases that must be performed for the desired functionality. The five main cases are: student signup, student login, admin signup, admin login, and updated database values from the UI. From these 5 test cases, there can be multiple branching variations of the test case. For example, we may wish to test the happy path of creating a user using a name we know is not in the database. However, we may also test a name that already exists in the database to assure that it does not create another user when the same information is given. We have tested the happy path as well as all other branching edge cases from the five main test cases to assure functionality is sound.

# Types of Tests Performed

All testing performed is from manual testing from all developers on the team. Our project is not robust and requiring extensive automated testing; however, if time was not a factor we had planned to implement Jasmine unit testing, a Javascript test suite that automates the test experience. As a standard of how any application should be tested, this website was tested with end-to-end testing, regression testing, and user-experience testing was performed.

# Test Environment & Tools

The great thing about our team is we were all familiar with popular version control in GitHub. GitHub has taken over the industry standard of version control and will not be going anywhere for a very long time. We all worked in separated branches/environments to perform our own testing and implementation; our main branch only had code “pushed” into it with a code review from another team developer. So before any code was merged into the main application branch, we had already reviewed and tested the application and the incoming code. As stated, we utilized manual testing since our application is very lightweight and a single purpose application.

# Lessons Learned

Something Professor (possibly Doctor now?) Sexton said to me in one of his courses is that testing should start before code has been written. The statement has a lot of truth and quality to it because testing becomes harder to implement in an existing code base than it would otherwise. There are so many dependencies typically to mock testing when the code has already been written without testing in mind. So this was a lesson learned, as we did not think about testing until our application was mostly finished.

# Recommendations

For future optimizations of the website, the team suggests implementing a testing suite for Javascript, such as Jasmine. This will automate existing cases and provide insight to how to minimize dependencies within the code. Best practice in a industry setting is not making the most elegent and abstracted code that nobody can understand; Instead, the use of laymen and easy-to-read variables/code and keeping track of how to minimize dependencies is worth its weight in gold.

# Best Practices

As stated in the Recommendations section, best practice is different where you find your home. In an industry setting, where a new developer may be taking your spot in a years time, it is best to not severely abstract code and use complicated structures that “make sense” at the time. It is better to design with future maintainers in mind, ie “Will Bob the newly hired junior dev be able to read this code I am writing without scrapping it?” A project that is thought out in this way can also inject test cases to depict important logic details, especially when an application must become abstract for whatever purpose, that may have otherwise gone other the radar.

# Exit Criteria

All of our test cases have been executed. As new code was written, some defects were presented that were retested by use of regression testing. A thorough end-to-end testing has been performed on the finished product and the application is working as intended with no pressing issues in sight.

# Conclusion/Sign Off

The project is mostly finished; however, there are optimizations to be made. As far as testing is concerned, implementing a automated testing suite is something to be done in the future.