Final Report

Software Testing and Quality assurance

Office Hours System

Andrew McCracken and James Gruss

# Software Quality Activities and their Value

During our project, multiple software quality assurance activities have taken place. One of the first activities was the informal review of our prior code base. We made some decisions on which features were the most important, which features were most feasible to work on, and which features were currently poorly implemented. Those features that were most important and feasible were then tested using Unit Testing.

In completing a preliminary code review of the initial implementation, we also noticed some small user interface design errors. Since we were planning on doing system acceptance testing, we decided to fix these issues as well, even though we were not planning on unit testing them, per se. We did this because usability is an important quality attribute.

Version management was also an important aspect of our quality assurance activities. Everything that we did to change the code, we did through Git. Documentation was completed either on Google Drive, backed up in a team folder with versioning available, or on Git as well. This allowed us to work in different locations and at different times while ensuring code integration was successful. Furthermore, it allowed for continuous integration of code.

Formulating and completing test plans were, clearly, an important aspect of our project that we completed, as well. For much of our testing, we formalized the process to ensure that our code was tested in the correct way. For example, we created control flow diagrams for our Login Manager functionality, mapped our system test cases to user-expected functionality. We also made use of automated testing tools to ensure that quality will continue to be ensured into the future, and that it takes no more time than necessary.

# Quality Assurance Metrics

Metrics are how we can judge if our project has been successful or not. A main feature of our system is usability, so this metric focuses on the usability of our system. If a user completes a task without help without stopping, it is a success.

**Completion Rate** (as defined in STQA Plan)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Goal | Nate | Justin | Chip | Even | Nisha | Total |
| Register and Log into the system | 1 | 1 | 1 | 1 | 1 | 4/5 |
| Find your Dean’s Office Hours | 1 | 0 | 1 | 1 | 1 | 4/5 |
| Send your TA an office hour’s request | 1 | 1 | 1 | 1 | 1 | 4/5 |
| Student receives email | 1 | 1 | 1 | 1 | 1 | 4/5 |
| Totals | 4 | 3 | 4 | 4 | 1 | 19/20 |

Rate: 95%

Metric Result: Success

Summary: Our system, in the perspective, seemed to be fairly usable. One aspect in which it failed was the ease in finding and viewing a professor's office hours. One user was unable to finish in a reasonable amount of time because he simply didn’t understand the interface. It needs work.

**Bugs Per Line of Code Per Unit (Tested)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Unit** | **Buggy Lines** | **LOC (No Spaces)** | **Result** |
| **Send Email** | **4** | **185** | **.02** |
| **Check Password** | **0** | **20** | **0** |
| **Create User** | **2** | **22** | **.09** |

Rate: 0.0264

Metric Result: Success

Summary: Although this metric was a success based on our goal, these

# Testing the Product

System Acceptance Test cases and Unit Test cases with results can be found in respective excel spreadsheets and unit testing files detailing each test case.

Testing Files: “\test\_cases\stqa\_NEW\...”

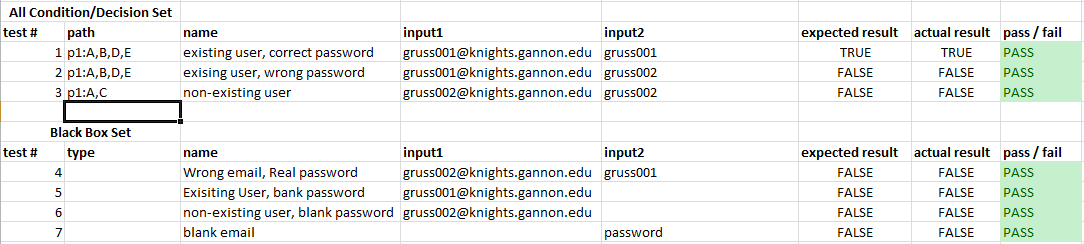
**Unit Testing**

We found unit testing to be a key resource in our software quality process. The isolation of concerns that it presents allows for developers to separate concerns. We did find some cases where our actual functionality differed from our expected functionality. We also demonstrated functionality that is working correctly, and put into place procedures for continuing to test these areas into the future.

### Login Manager

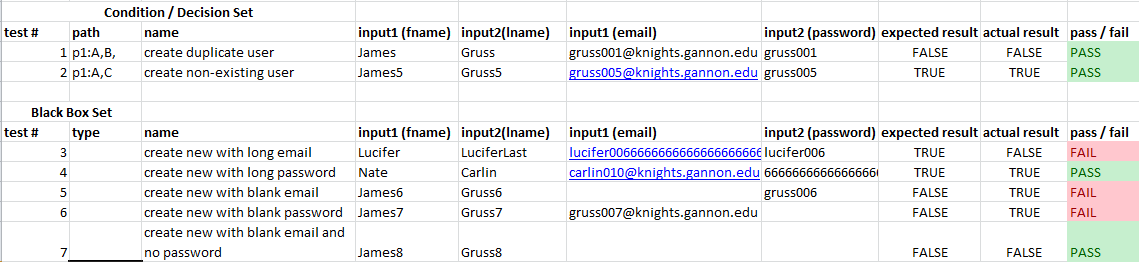
As stated in our test plan, we have tested the Check Password and the Create New User functions. The results are included in our test cases folder. Additionally, below are the results:

Check Password Results



As the reader can see, every test cases has passed. This means that every basis path, condition, decision, and combination has been tested and produces the correct functionality. Please refer to our testing plan for more details on test adequacy.

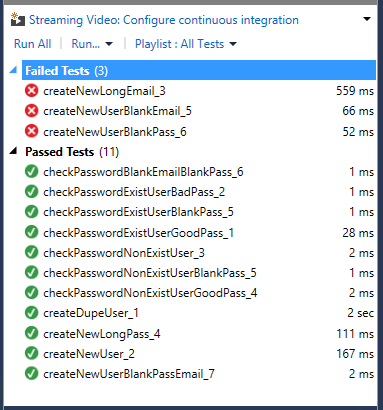
Create New User Results



Create New User, however, failed in three ways. Every basis, condition, and decision, however, were tested and passed, in general, successfully. The issue comes in defensive input validation. As this function is the one tasked with directly validating input, it needs to handle these types of unexpected inputs. First, it allowed a user with a blank email to create an account. This is clearly not expected functionality and goes against security. Similarly, it allows a blank password. Finally, it fails if a longer than handleable email is entered. THis should be considered. Either input checking needs to be given to another component to abstract this problem, or we must add functionality.

#### Visual Studio Report

These test cases can only be ran from the development server in ITS, which necessitates usernames, passwords, and, maybe training. The VS unit testing file can be found in the test cases folder To illustrate the testing tool, here is the Visual Studio report:



### Email Sender

Below is a baseline test case and template for all other test cases for the EmailSender:



The test cases created showed that the functions behave mostly as expected, but also revealed some flaws. The system handles NULL values correctly, but does not handle empty strings as expected. This is a serious flaw in the system because the email and ICS invitation are imperative to its success.

## System Acceptance Testing

All tests suites and data for system acceptance tests can be located at cis315\_OfficeHours\test\_cases\stqa\_NEW\System Acceptance Tests. Here you will find an individual folder for each functionality being tested. Each contains a Selenium test case suite and a spreadsheet detailing each test case.

#### System Test Cases for Registration:

|  |  |
| --- | --- |
| Requirement (Found in Vision and Scope doc) | Test Cases that test the requirement |
| **BR-1.1**: Due to Unavailable LDAP system, Students must create an account with an email and password before using the system. They must use their Gannon email. | Test cases: 1, 2, 3, 4, 5 |
| System interaction and boundary analysis | Test cases: 6, 7, 8, 9, 10, 11 |

These test cases were successful in validating that the system meets requirements. They also discovered a bug in the code. Test case 7 found that a user can create a password, but if the confirmation password field is left blank, the user can still successfully register. The user can then log in with the account, as shown in test case 11. This functionality is clearly not expected; the user interface uses two password fields in order to ensure that the password a user enters the password that he intends. This bug is a result of user input validation, however, and not of the underlying system that has been unit tested at this stage.

#### 

#### System Test Cases for Make Appointment:

|  |  |
| --- | --- |
| Requirements (Found in Vision and Scope doc) | Test Cases that test the requirement |
| Allow students to select department | Test case 1 |
| Allow students to select professor | Test case 2 |
| Make appointment with any professor in CAFE | Test case 3 |
| Message body empty (generic will be created) | Test case 4 |
| Message body long | Test case 5 |
| Message body custom | Test case 6 |

These test cases were successful in validating that the system meets requirements. It is clear that all aspects of creating an appointment work as expected in relation to the requirements. These higher level test cases, combined with the unit tests for the EmailSender, show that the functionality of creating and sending an appointment are working as expected.