PSU SWENG 500 Team 3 Agile Test Summary Report

8/7/13

Contents

1 Background and Overview

This document provides the Test Summary for the testing activities for three iterations of Team 3's Brewers Buddy project. Brewers Buddy Iterations 1, 2, and 3 Tests were completed on 8/6/2013. During testing, the entire solution was tested. Testing was conducted using various computers containing Microsoft Visual Studio. NUnit Test Adapter was also attached to Visual Studio and assisted the integrated testing process. NUnit is a unit-testing framework for all .Net languages, and is one of many programs in the xUnit family.

2 Testing Approach

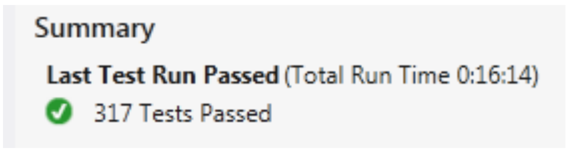
Our testing approach for the brewers buddy system was to automate as much of the testing as possible. This way, running a system regression would simply require running an automated test suite. To accomplish this goal, we attempted to write automated NUnit tests for each new feature or use case we implemented. This helped us to achieve a high level of test coverage and quality. In some cases, we performed manual testing. We determined whether to use automated or manual testing by examining the difficulty or creating an automated test. If an automated test would take several hours to develop, but a simple manual test could be performed in a few minutes, then we opted for the manual test. In most cases, we were able to generate automated tests quite easily. Using this mix allowed us to minimize the time spent testing and the time spent developing tests. Overall, we feel this approach lead to a high quality system.

3 Test Results

Test Coverage Overall for Custom Code (**Note:** We did not do testing on things created already within the framework such as migrations, filters, utilities, etc.)

Name	***Not Covered (Blocks)***	***Not Covered (Blocks %)***	***Covered (Blocks)***	***Covered(Blocks %)***
BrewersBuddy.Controllers	1071	49.3%	1098	50.7%
BrewersBuddy.Models	8	2.5%	318	97.5%
BrewersBuddy.Services	56	11.7%	420	88.3%
Total	1135	38.2%	1836	61.8%

Tests Run: 317 tests were run and passed.

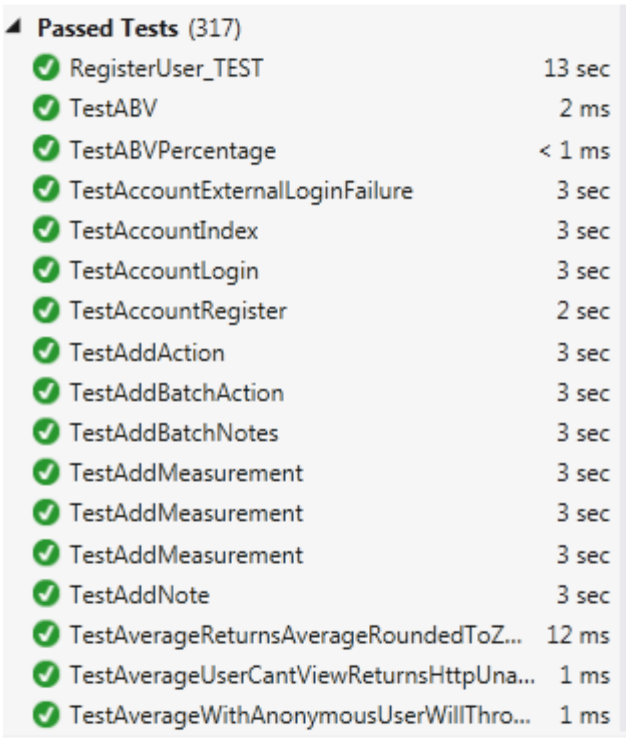


Summary

Last Test Run Passed (Total Run Time 0:16:14)

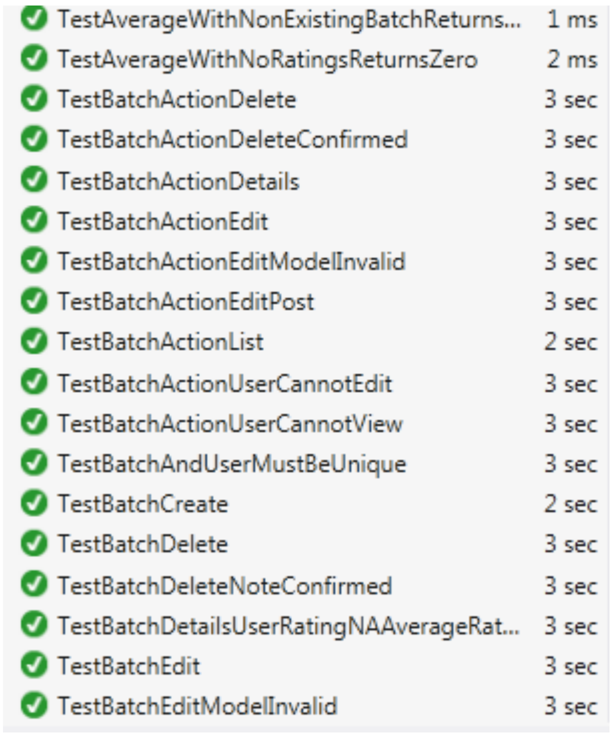
✓ 317 Tests Passed

Selected Examples of Passed Tests

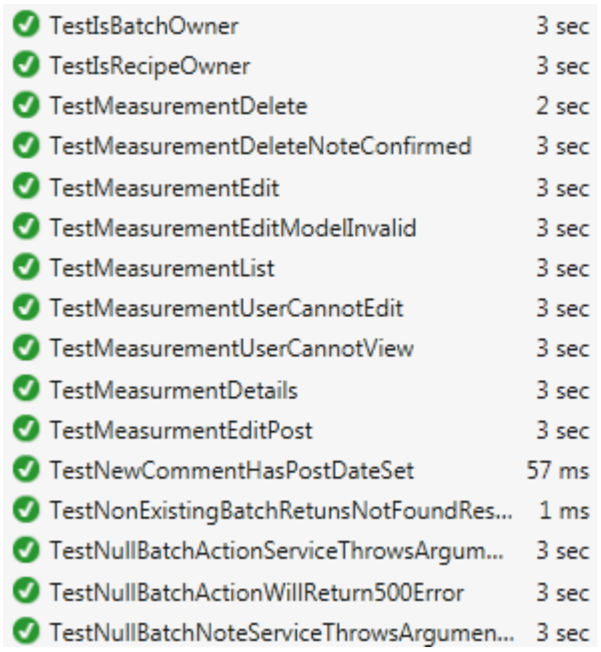


Passed Tests (317)

✓ RegisterUser_TEST	13 sec
✓ TestABV	2 ms
✓ TestABVPercentage	< 1 ms
✓ TestAccountExternalLoginFailure	3 sec
✓ TestAccountIndex	3 sec
✓ TestAccountLogin	3 sec
✓ TestAccountRegister	2 sec
✓ TestAddAction	3 sec
✓ TestAddBatchAction	3 sec
✓ TestAddBatchNotes	3 sec
✓ TestAddMeasurement	3 sec
✓ TestAddMeasurement	3 sec
✓ TestAddMeasurement	3 sec
✓ TestAddNote	3 sec
✓ TestAverageReturnsAverageRoundedToZ...	12 ms
✓ TestAverageUserCantViewReturnsHttpUna...	1 ms
✓ TestAverageWithAnonymousUserWillThro...	1 ms



```
✓ TestAverageWithNonExistingBatchReturns...   1 ms
✓ TestAverageWithNoRatingsReturnsZero         2 ms
✓ TestBatchActionDelete                       3 sec
✓ TestBatchActionDeleteConfirmed              3 sec
✓ TestBatchActionDetails                      3 sec
✓ TestBatchActionEdit                         3 sec
✓ TestBatchActionEditModelInvalid             3 sec
✓ TestBatchActionEditPost                     3 sec
✓ TestBatchActionList                         2 sec
✓ TestBatchActionUserCannotEdit               3 sec
✓ TestBatchActionUserCannotView               3 sec
✓ TestBatchAndUserMustBeUnique                3 sec
✓ TestBatchCreate                             2 sec
✓ TestBatchDelete                             3 sec
✓ TestBatchDeleteNoteConfirmed                3 sec
✓ TestBatchDetailsUserRatingNAAverageRat...   3 sec
✓ TestBatchEdit                               3 sec
✓ TestBatchEditModelInvalid                   3 sec
```



```
✓ TestIsBatchOwner                            3 sec
✓ TestIsRecipeOwner                           3 sec
✓ TestMeasurementDelete                       2 sec
✓ TestMeasurementDeleteNoteConfirmed          3 sec
✓ TestMeasurementEdit                         3 sec
✓ TestMeasurementEditModelInvalid             3 sec
✓ TestMeasurementList                         3 sec
✓ TestMeasurementUserCannotEdit               3 sec
✓ TestMeasurementUserCannotView               3 sec
✓ TestMeasurmentDetails                       3 sec
✓ TestMeasurmentEditPost                      3 sec
✓ TestNewCommentHasPostDateSet               57 ms
✓ TestNonExistingBatchRetunsNotFoundRes...    1 ms
✓ TestNullBatchActionServiceThrowsArgum...    3 sec
✓ TestNullBatchActionWillReturn500Error       3 sec
✓ TestNullBatchNoteServiceThrowsArgumen...    3 sec
```

4 Variances

No testing variances were found 100% of the 317 tests passed

5 Other Code Related Metrics

Lines of Code: 3073

Maintainability Index: 85

MAX(0,(171 - 5.2 * ln(Halstead Volume) - 0.23 * (Cyclomatic Complexity) - 16.2 * ln(Lines of Code))*100 / 171) According to Microsoft in between 20 – 100 is Green - Overall Green

Depth of Inheritance: All values 4 or under. A value greater than 4 would compromise encapsulation and increase complexity. - Overall Green

Cyclomatic Complexity: All code members are between 1-10, which means the code is structured and well written code and highly testable - Overall Green

6 Function Testing

In addition to all of the automated unit tests, the team also performed functional testing as part of each sprint. At the end of each sprint, team members used each major function of the site and logged any defects they found in our issue tracking system. These defects were then added to the sprint backlog for the next sprint. In several cases this type of functional testing was required because writing unit tests proved difficult. In these cases, the team simply performed functional testing on these parts of the system instead of attempting to write unit tests. For example, the add friend functionality was difficult to fully unit test because it required sending an e-mail and then clicking a link in this e-mail. We used functional testing to test this functionality.