# SFT221 SCRUM Report and Reflections

This report should be completed in the class and submitted at the end of class. Late submissions cannot be accepted without prior approval of the instructor.

**GROUP**: \_\_\_\_\_\_\_\_\_\_\_5\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1.Thomas Lewis | 4.Kiki |
| 2.Asem M | 5. |
| 3.Hla Myint Myat | 6. |

## Milestone 5 Tasks

In this milestone, you should write, implement, and execute integration tests. Integration tests test how multiple functions work together to complete a task. Depending on what is being tested, you might be able to write unit tests to do the testing and automatically compare the results. In other cases, you might need to manually check the output to check it. This will all be stated in the tests where it discusses how they should be run.

As you update the function-test matrix, you will need to add a very brief description for each integration test so the matrix will clearly show what the tests are testing. Acceptance tests will be tested against actual user requirements and will list all the tests for each requirement.

Acceptance tests are the final tests and are largely aimed at showing the customer that the correct output is produced for different inputs. This will largely require manual testing.

**Deliverables due 4 days after your lab day:**

* Integration tests document stored in repository with at least 4 sets of distance test cases (each case must have at least 4 distinct test data).
* Integration tests coded (store in repo), executed (results in Jira and in test documents) and debugged.
* Finish implementing/coding whitebox tests. Store in repo, executed, results in Jira (and on corresponding test documents, and debugged.
* Acceptance tests written and stored in repository.
* Updated requirements traceability matrix stored to the repository.
* Completed scrum report including reflection questions answered.

**Rubric:**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Integration test case document (well written, complete, good test data) | 10% |
| Integration test code (well designed and documented) | 10% |
| Finish coding all functions and main (well-designed, written, and documented) | 10% |
| Finish coding blackbox and whitebox cases (well-designed, written, and documented) | 10% |
| Acceptance tests (well-designed, written and documented) | 5% |
| Requirements traceability matrix updated | 5% |
| Test execution (performed, results recorded, issues created) | 5% |
| Debugging (bugs fixed, documented, Jira updated) | 5% |
| Git usage (used properly with good structure) | 5% |
| Jira usage (creates issues, tracks progress) | 10% |
| Scrum report & reflections | 20% |
| Meets deadlines | 5% |

**SCRUM Report**

**Summary of Tasks Completed or Delayed in the last week:**

Here you can list all of the tasks completed in the last week along with any tasks which could not be completed with a reason why they could not be completed.

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| Asem M | **White Box Testing** | **N/A** |
| Thomas Lewis | **Hook file** | **N/A** |
| Hla Myint Myat | **Scrum & Reflection** | **N/A** |
| Kiki | **Black Box Testing** | **N/A** |
|  |  |  |
|  |  |  |
|  |  |  |

For every task delayed or blocked, describe the reason for the delay or block, how it impacts the project and the proposed solution or workaround**.**

|  |  |
| --- | --- |
| **Delayed or Blocked Task** | **N/A** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |
|  |  |
| **Delayed or Blocked Task** | **N/A** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |

**Summary of Meeting:**

A summary of the main points discusses in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Integration Testing | **written and discussed** | **Done** |
| Acceptance Testing | **written and discussed** | **Done** |
| Test execution | **executed** | **Done** |
| SCRUM | **Completed** | **Done** |
|  |  |  |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

This will include major architecture and design decisions, testing decisions, prioritization of tasks, dealing with problems encountered and other major outcomes from the meeting.

|  |  |
| --- | --- |
| Decision | Rationale |
| Breaking down the task | Equal amount of works assigned to each member of team. |
| Acceptance Testing | Choosing Alpha and beta testing as process |
| Integration Testing | Choosing Bottom – up testing as process |
| Testing execution | executed and recorded |
|  |  |
|  |  |
|  |  |

**Tasks Attempted During Meeting:**

Each member is assumed to participate in the SCRUM meeting and contribute to the completion of the SCRUM report and reflections. Since the SCRUM meeting will not take more than 20-30 minutes, there is lots of time left to undertake some of the actual work tasks. In the table below, each member should list what they did to complete the SCRUM report, the reflections, and 1-4 other tasks they completed during the class period. If a task could not be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Kiki | **Test execution** | **1hr** | **Yes** |
| Thomas Lewis/Asem M | **debugging and assigning new tickets to members** | **1hr** | **Yes** |
| Thomas Lewis | **Github updated** | **15mins** | **Yes** |
| Asem M | **Jira assigned** | **15mins** | **Yes** |
| Hla Myint Myat | **Scrum report** | **1hr** | **Yes** |
|  |  |  |  |
|  |  |  |  |

**SCRUM Tasks Selected for Next Week**:

The tasks each member has selected to pursue for this class or the next week.

|  |  |
| --- | --- |
| Group Member | Task Description |
|  |  |
| Thomas Lewis | Debuggings |
| Asem M | Test Execution() |
| Kiki | Final test report |
| Hla Myint Myat | SCRUM , Reflection |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

This is where you should highlight the major accomplishments of the class.

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Confirmation on Acceptance testing | **listed out Debugs for matrix and ticketed to members** |
| Confirmation on Integration Testing | **listed out Debugs for matrix and ticketed to members** |
| Functionality improvement | **achieved a certain level of perfection on our new implemented Functions with data structure within the existing program.** |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**

Here you can highlight things which worked well. This indicates that the way you worked on these items is working and should be continued.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Integration testing | **tests passed** |
| Acceptance testing | **Well documented** |
| SCRUM | All contributed. |
| Git | **Useful for version control** |
|  |  |
|  |  |
|  |  |

**Things That Did NOT go Well in This Meeting:**

This is where you can list things which did not go well in the class. You should analyze why this happened and suggest how you can improve it next time. This will lead to the goal of *continuous process improvement*.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Problem and How to do Better |
| N/A | N/A |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

1. At this point, you are using the GIT hook to automate testing. Have you found that any of the tests failed and prevented you from pushing your code to the repository? If so, how did you handle the situation?  
     
   We've made improvements to our development workflow, such as adding helpful prompt messages. With the help of these prompts, developers may ensure that changes are carefully considered before being committed in the Bash terminal.

At the moment, we track test results using GitHub Desktop, tracking any errors that show up in test files. We take quick action to fix these problems, keeping the code base strong and secure.

We commit after addressing problems and putting solutions in place. We also recognize the importance of tracking progress, encouraging communication and collaboration by using commits to record significant modifications made to files or the project overall.

Although automated hooks are efficient, we are aware that version control and progress tracking may be limited. One of the main priorities is still finding a balance between developer flexibility and automation. Essentially, our rapid communication and insightful use of GitHub Desktop highlight the attention we give to challenging version control, high-quality code, and detailed progress monitoring throughout our development process.

1. Explain why we are automating the testing process and what the advantages of this automation are.  
     
   Testing process automation benefits software development in several way.

Efficiency: Automated tests improve the feedback loop by quickly identifying errors and regressions.

Consistency: Accurate execution of test scripts removes human mistake and guarantees reliable results.

Coverage: Edge cases and setups are included in the thorough coverage provided by automated tests.

Regularity: By integrating into CI/CD and agile workflows, testing may be done frequently to identify problems later on.

Scalability: Managing increased test cases effectively is beneficial for developing codebases.

Collaboration: Effective communication among members is made possible by objective tests.

Cost Savings: Long-term reductions in manual efforts results in cost benefits, even after the initial setup.

1. Did you find the integration and acceptance tests more difficult to write than the black box and white box tests? If so, why were they harder to write? Did you write more white box and black box tests or more integration and acceptance tests?  
     
     
   If complex relationships between units and additional testing code were required, integration testing was more difficult than white box testing. Acceptance testing introduced additional complexity by demanding certain user selection and feedback integration, especially during the Alpha and Beta phases.

It was difficult for our team to choose the right Beta testers from among ourselves while taking into account constructive criticism and replication of real-world usage.

We increased our attention on integration testing codes in response to complexity. We all put extra effort into improving our testing processes because we were not satisfied with the results of the original black and white box testing, especially in MS5.

This effort was in line with our commitment to a reliable final product and aimed to fully comprehend the behaviour of our software, guarantee compatibility, and actively fix any potential problems. Our testing process proved our commitment to improving strategies and producing highly qualified software.

1. Explain why it is necessary to write integration and acceptance tests given that all of the code has already passed black box and white box tests.

While black box and white box tests are essential for testing individual software components, acceptance and integration testing provide special benefits that improve these phases.

Integration tests verify that various components that passed previous tests function together without any problems. They reveal unnoticed fails, inconsistencies, or dependencies that arise from component interactions. This ensures flawless performance in practical situations.

Users in reality participate in acceptance tests, which, like alpha and beta testing, expose information beyond technical accuracy. They detect speed problems, UI bugs, and usability problems that previous tests might have missed. Including users who use it ensures that the program satisfies their needs, improving quality and confidence for this sector. To sum up, acceptance and integration tests go beyond white- and black-box validations. While acceptance tests verify usability and user satisfaction, reducing errors and increasing software reliability, integration tests guarantee flawless phase interaction.