# 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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Irish Banga | 4. Gulpreet Kaur |
| 2. In Tae Chung | 5. |
| 3. Cesca Dela Cruz | 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 at end of Lab:**

* Completed SCRUM report and reflections

**Deliverables Due at 23:59 12 Days after Lab:**

* integration tests written and stored in repository,
* integration tests written (store in repo), executed (results in Jira and in test documents) and debugged.
* acceptance tests written and stored in repository.
* Updated function-integration-requirements-test matrix stored to the repository.

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
| Teamwork | 10% |
| SCRUM Report and reflections | 15% |
| Group | integration tests (well-designed, written and documented) | 20% |
| acceptance tests (well-designed, written and documented) | 20% |
| Test Execution (performed, results recorded, issues created) | 15% |
| Debugging (Bugs fixed, documented, Jira updated) | 5% |
| Function-test matrix updated | 5% |
| Git Usage (used properly with good structure) | 5% |
| Jira Usage (creates issues, tracks progress) | 5% |
| Meets Deadlines | 5% |
| SCRUM Report and Reflections | 20% |

**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** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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

**Summary of Meeting:**

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

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

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?
2. Explain why we are automating the testing process and what the advantages of this automation are.

By employing computers to run tests automatically, software testing is automated. This strategy was chosen primarily because it has several advantages over manual testing. First off, automated tests run significantly more quickly, giving instant feedback on the software's quality, and quickening the development cycle. They also increase dependability since they operate more reliably and less frequently make mistakes due to human error. Additionally, automation enables a broad number of test scenarios, providing complete software testing and identifying any problems.

Secondly, automated tests are incredibly economical. Once written, test scripts can be utilized again, saving time and labor during software updates, and negating the demand for a sizable testing staff. Continuous integration, which smoothly incorporates tests into the development process and finds issues early on, is supported by automation as well. It offers thorough and beneficial reports for problem-solving, assisting in quickly identifying and resolving problems. Overall, automating testing promotes higher software quality and improves the effectiveness and productivity of the development process.

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?
2. 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.

Even if earlier tests pass, integration and acceptance tests are vital in software testing. Integration tests concentrate on the interactions between various software components. It guarantees that even if individual parts perform admirably on their own, they perform admirably together as a whole system. These tests ensure that data flows between the various components seamlessly and that they operate in harmony by identifying any potential problems that may develop during integration.

Acceptance tests, on the other hand, focus on fulfilling the demands and expectations of users. They determine if the software achieves its goals and provides an enjoyable user experience. Acceptance tests examine the software from a user's perspective, ensuring it behaves as intended, whereas other tests may check technical features. By doing these tests, developers can be sure that the software will function flawlessly in actual scenarios and will satisfy all necessary specifications, resulting in a high-quality and user-friendly final product.