# 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.Prabhjot Singh | 4. Sampreet Klair |
| 2.Dhruv Kakadiya | 5.Siya Khanna |
| 3.Prince Prince | 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** |
| **Prabhjot Singh** | **Completed reflection and scrum, helped with acceptance testing** |  |
| **Dhruv Kakadiya** | **Wrote Integration testing and document it** |  |
| **Siya Khanna** | **Tested Integration Tests and debugged** |  |
| **Sampreet Klair** | **Acceptance tests** |  |
| **Prince Prince** | **Updated function test matrix** |  |
|  |  |  |
|  |  |  |

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 |
| Integration Tests written | **Done** |  |
| Testing of integration tests | **Partially Done** |  |
| Acceptance Tests | **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 |
| Function implementation | Half Done |
| Integration tests | Need time |
| Acceptance Tests | Almost Done |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Prabhjot Singh | **Reflection, Scrum and Acceptance Tests help** | **30min** | **Yes** |
| Dhruv Kakadiya | **Discussed about Integration Testing** | **20 min** | **Yes** |
| Siya Khanna | **Started debugging** | **10 min** | **Had some issues and need correction** |
| Sampreet Klair | **Acceptance Tests** | **20 min** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Prabhjot Singh | Scrum, Reflection, tasks allocation and acceptance test helping |
| Dhruv Kakadiya | Management of Jira and final solution |
| Sampreet Klair | Completing the acceptance tests |
| Prince Prince | Helping with any other problems |
| Siya Khanna | Working on debugging |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Acceptance Tests | **Almost Done** |
| Integration Tests | **Done** |
| Debugging of integration Tests | **Pending** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Acceptance tests discussion | **Everybody tried to contribute** |
| Integration Tests discussion | **Everybody gave ideas** |
| Testing | **Teamwork** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Nothing | **Everything was good** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?  
     
   As we know that currently, we utilize GitHub Desktop to monitor test outcomes, identifying any failures reflected in testing files. We promptly address these issues, maintaining a robust and reliable code base

After resolving issues and implementing fixes, we proceed with the commit. We also acknowledge the significance of documenting progress, using commits to log significant advancements in files or the overall project, promoting transparency and collaboration.

While automated hooks offer efficiency, we're mindful of potential limitations on version control and progress tracking. Striking a balance between automation and developer flexibility remains a key focus.

In essence, our prompt messages and strategic use of GitHub Desktop underscore our commitment to code quality, thorough version control, and comprehensive progress tracking in our development process.

1. Explain why we are automating the testing process and what the advantages of this automation are.  
   We are automating the testing process because of its vast advantages .

Such as:

1. We can run our tests faster and quicker
2. Automated tests ensure consistent test execution, reducing the likelihood of human error.
3. While there is an initial investment in setting up automated tests, the long-term benefits include significant time and cost savings
4. We Can easily detect bugs
5. Automated tests can easily scale to accommodate the growing complexity of software projects.
6. 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?  
   Integration testing posed a greater challenge than white box testing due to the need for intricate connections between units and the creation of new testing code. Acceptance testing, particularly Alpha and Beta phases, brought further complexity, requiring careful user selection and feedback coordination.

Since we found it hard we still tried to maintain a balance between the integration and Blackbox tests.

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.

The reason is that :

For integration testing , we have the following advantages:

1. integration tests focus on verifying the interactions between different components or systems.
2. Integration tests are crucial for testing complex scenarios that involve the collaboration of multiple components.
3. Integration Tests help us to bring other software in demand to integrate in our software and still maintaining the proper interaction.

For Acceptances tests , we have the following advantages:

1. Acceptance tests validate the entire system from end to end, simulating real user interactions.
2. While black box tests focus on functionality, acceptance tests take a user-centric perspective, ensuring that the software meets user expectations and business requirements.
3. acceptance tests often involve testing the user interface, making sure that user interactions, workflows, and visual elements align with design specifications.