# 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**: C

**Members Present**:

|  |  |
| --- | --- |
| 1.Manav Zadafiya | 4.Fenil Soni |
| 2.Sunny Vavadiya | 5. |
| 3.Ashraf Bharot | 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** |
| **Manav** | **Integration test execution and managing deliverables and tracking workflow** |  |
| **Sunny** | **Integration test execution and Updating Function Test matrix** |  |
| **Ashraf** | **Integration and Acceptance Test documentation** |  |
| **Fenil** | **Integration and Acceptance Test Documentation** |  |
| **ALL** | **SCRUM and reflection** |  |
|  |  |  |
|  |  |  |

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** |  |
| **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 |
| Milestone 5 Planning and Deliverables | **Discussed tasks and deliverables for Milestone 5, including integration tests and acceptance tests.** | **Get familiar with Tasks and deadlines for milestone 5** |
| Integration Testing | **Integration testing discussed and documented** | **Integration test execution started** |
| Acceptance Testing | **Acceptance testing discussed and documented** | **Acceptance tests documented** |
| Traceability Matrix updated to business requirements | **The team reviewed the function-integration-requirements-test matrix.** | **The matrix will be updated and stored in the repository to track integration test coverage.** |
| Git | **Git update to each branch (Debug tickets on Git project)** | **Completed** |
| Jira | **Task Schedule setup in Jira** | **Completed** |
| SCRUM | **SCRUM Completed** | **Completed** |
| Milestone 6 | **The team discussed and review milestone 6 requirements** | **Team members get familiar with ms6** |

**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 |
| Integration and Acceptance Tests | To ensure multiple functions work together without bugs and demonstrate customer requirements. |
| Test Execution and Documentation | Storing all tests in the repository and recording results in Jira and corresponding documents. |
| Updating traceability matrix | Updating the matrix to track new integration tests and ensure comprehensive test coverage. |
| Timely Completion and Collaboration | Emphasizing the importance of meeting deadlines and effective teamwork for project success. |
| GitHub Repository | Updating Github repo and source code for smoother workflow and easy to track progress |
| Weekly Meetings | Holding regular meetings and using Jira to create, track, and update work items and progress. |
| Review Next Milestone | To get familiar with tasks and prepared for upcoming challenges helps us to boost productivity |

**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? |
| ALL | **Scrum Report and Reflection** | **35 min** | **Yes** |
| Manav | **Github Repository Updated** | **15min** | **Yes** |
| ALL | **Test Planning and Tasks distribution** | **10 min** | **Yes** |
| ALL | **Issue Creation** | **30 min** | **Yes** |
| Ashraf, Fenil | **Writing Integration and acceptance Tests** | **1hr** | **Yes** |
| sunny | **Updating Test Documents** | **1hr** | **Yes** |
| ALL | **Discussed Next milestone tasks and requirements** | **15 min** | **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 |
| Sunny | JIRA Control, updating Traceability matrix |
| Manav | Git and Github and repository management for final deliverables |
| Fenil, Ashraf | Writing and storing integration and acceptance test |
| ALL | Ms6 scrum report and reflection |
| ALL | Meeting on Monday, 4th December |
| ALL | Final Test report |
| ALL | Finalize All project documentation and repository organization |
| Manav, sunny | Implementing test and storing result in repository |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Detailed improvement on functionality of program | **With multiple Black box and white box testing and debugging, our program is ready for integration and acceptance testing and then final touch.** |
| Task Assignment & Collaboration | **Each team member was assigned specific tasks, and collaboration was encouraged.** |
| Standardization of Documentation | **The team decided on a template and format for documentation, ensuring consistency.** |
| Task Distribution | **Team members select their own task for milestone and committed to due dates** |
|  |  |
|  |  |
|  |  |

**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 |
| SCRUM | **All contributed.** |
| Git | **Useful for version control and keeping track of changes** |
| Meeting | **All attended meeting.** |
| Tests Documents | **Everyone contributed** |
| JIRA | **Everyone used Jira to update and track progress** |
|  |  |

**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?

It is obvious that after implementing the GIT hook for automated testing, we were prevented from pushing code when some tests failed. In this case, we followed a systematic approach by determining the exact reasons of the failures, we first thoroughly analyzed the test outcomes. This helped us in determining the exact cause of the errors and if they were caused by coding mistakes or modifications in the project's implementation.

After determining the reasons, we set out to solve the issues. We checked the affected code sections for any coding flaws and fixed any faults that were causing the tests to fail. Testing failed due to requirements changes; thus, we had to review and update our code implementation. The updated requirements indicate the need of new feature or restore to previous one. After fixing the issues, we ran the program on our local machine to ensure that they had been fixed. We could confidently push our code to the repository after the tests had passed properly and help us to ensure bug-free push.

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

There are numerous advantages of automating testing which can greatly increase the productivity and dependability of software development. We automate testing for the following reasons, along with a few benefits:

* Efficiency: When it comes to time and effort, automated testing is more efficient than manual testing. It allows us to quickly execute a large number of test cases, which is incredibly helpful for intricate projects that require frequent changes.
* Early Bug Detection: Automated tests can be run immediately after integrating new code, helping in the early discovery of errors. As a result, errors can be addressed right away rather than getting worse and more difficult to correct later in the development cycle.
* Reusability: Automated test scripts can be used repeatedly for related projects or versions, reducing the need to create new tests from scratch.
* Enhanced Test Coverage: Testing edge scenarios and conditions that may be challenging to tackle with manual testing is made easier by automation, increasing the scope of tests overall.
* Scalability: Automated tests have been able to be scaled easily to address a variety of situations, including multiple platforms, browsers, and configurations. This facilitates thorough test coverage.
* Cost-Effectiveness: While installing automation could need a small initial expenditure, the long-term advantages surpass the costs. In the long run, cost benefits are achieved since automated testing decreases the demand for manual testing resources.
* Automated testing is an essential part of CI/CD pipelines, which stand for Continuous Integration (CI) and Continuous Deployment (CD). It makes it possible to integrate, test, and deploy code frequently ensuring that updates are immediately verified.
* Regression testing: By rerunning test suites following a code change, we can easily perform regression testing using automation. This makes it easier to spot any regressions and guarantees that new updates won't affect the functioning of older versions.
* Consistency: Automation makes certain that the same tests are conducted consistently each time, removing the possibility of human errors or modifications to the testing 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?

In comparison to the black box and white box tests, the integration and acceptance tests were, in fact, more difficult to create. The nature of these tests and their scope were mainly the reason for their difficulty.

Writing integration tests needed an extensive understanding of how various components or modules interacted with one another throughout the entire system, which made them more difficult to create. It required thorough planning and test case design to make sure that all the combined components operated smoothly as well as find any potential compatibility problems.

In a similar way, acceptance tests were more challenging since they focused on validating the operation of the application from the perspective of the end user. We had to accurately represent real-world scenarios and user interactions for these tests in order for them to closely match the defined project requirements.

We included more white box and black box tests for different parts and functionality during the testing process. Due to their focus on specific elements and more concentrated testing, these tests were easier to implement. Nevertheless, we noticed the importance of integration and acceptance tests in generating a more thorough analysis of the entire system. We made sure to devote enough time to writing integration and acceptance tests to make sure complete test coverage, and to ensure the overall quality of the project implementation.

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.

Writing integration and acceptance tests is still required to create a thorough and reliable testing approach, even though all of the code has successfully passed black box and white box tests. The following are the reasons:

* Future modifications and Regressions: Writing acceptance and integration tests provide a safeguard against upcoming code updates and modifications. Future changes may bring about unexpected results or regressions even though the current code passes all tests. These more complex tests help in identifying any problems that might come up as the software develops.
* Comprehensive Test Coverage: A more thorough test coverage is attained by integrating the black box, white box, integration, and acceptance tests. Each kind of test focuses on a different part of the software in order to completely evaluate all important areas.
* Testing the overall behavior of the system: Both black box and white box tests concentrate on distinct units or components. However, it is essential to confirm how these components work together as a whole and interact with one another. Integration tests support in identifying potential problems that can occur during the integration of various components, guaranteeing a smooth interaction between them.
* Testing of Real-World Scenarios: End-to-end testing is provided through integration and acceptance tests, which replicate real-world user interactions and scenarios. These tests make sure that the program functions how a user would anticipate and that all parts interact smoothly.
* Testing the overall behavior of the system: Both black box and white box tests concentrate on distinct units or components. However, it is essential to confirm how these components work together as a whole and interact with one another. Integration tests support in identifying potential problems that can occur during the integration of various components, guaranteeing a smooth interaction between them.
* Handling advanced Logic and Flows: Acceptance tests concentrate on assessing complex application logic and business requirements. Acceptance tests make sure that the program satisfies the defined business goals and user expectations, whereas black box and white box tests focus on unit-level functionalities.
* Compatibility and External Dependencies: Integration tests verify how well different modules and external dependencies interact with one another. Even if individual components are working as intended, combining them with other components of the system or outside resources may cause unanticipated issues.