# **Milestone 5 Scrum Report**

All students are expected to attend the scrum meetings and to participate. Failure to do so will result in greatly reduced grades.

**GROUP**: **Team4**

**Members Present**:

|  |  |
| --- | --- |
| 1. Seulgi Lee | 4. Alejandro Mercado |
| 2. Khassan Suleimanov | 5. |
| 3. Philip Grahamm | 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 11 days after your lab day:**

* Integration tests document (for the new functions you added) stored in repository with at least 4 sets of distinct 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.
* One acceptance test case for each requirement added to the test cases excel sheet.
* All acceptance tests implemented and added to the testing C++ project.
* Updated requirements traceability matrix stored in 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) | 5% |
| Acceptance tests (well-designed, documented, and implemented) | 15% |
| 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) | 15% |
| Scrum report & reflections | 15% |
| **Deadline** | 20% deduction for each day you are late |  |

**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** |
| Seulgi Lee | 1. Setup milestone of Jira issues. 2. Exchanged opinions on the answers to the reflection questions. 3. Assigned roles and completed Jira issues and assignments. 4. Completed scrum report. 5. Wrote the Reflection. 6. Made timeline and board about Milestone4 using Jira and assign team members. 7. Updated to Requirements Traceability Matrix. 8. Completed of Hook Files for Test Automation. 9. Updated Jira with the work performed and planned. | None |
| Khassan Suleimanov | 1. Exchanged opinions on the answers to the reflection questions. 2. Wrote the Reflection. 3. Wrote test-description document and stored in a repository. 4. Completed of Hook Files for Test Automation. 5. Updated Jira with the work performed and planned. | None |
| Philip Grahamm | 1. Exchanged opinions on the answers to the reflection questions. 2. Wrote the Reflection. 3. Wrote test-description document and stored in a repository. 4. Completed of Hook Files for Test Automation. 5. Updated Jira with the work performed and planned. | None |
| Alejandro Mercado | 1. Exchanged opinions on the answers to the reflection questions. 2. Finished implementing/coding the functions. 3. Finished implementing/coding tests. Store in repo, executed, results in Jira (and on corresponding test documents, and debugged. 4. Completed of Hook Files for Test Automation. 5. Updated Jira with the work performed and planned. | None |

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** | None |
| **Reason for delay or block** | None |
| **Impact on Project** | None |
| **Solution or work-around** | None |

**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 | Discussed the integration tests required. Team member was assigned to write, implement, and execute integration tests for specific modules. | Add at least 4 sets of distinct test cases to the repository. Tests will be executed and results documented in Jira. We agreed to review each other's tests for completeness and accuracy. |
| Acceptance Tests | Talked about the importance of acceptance tests to ensure that our outputs meet user requirements. The need for manual testing for acceptance tests was highlighted. | Contribute one acceptance test case per requirement, to be included in the test cases excel sheet. All acceptance tests will be implemented in the testing C++ project. |
| Whitebox Tests | The status of whitebox tests was reviewed. We discussed the remaining tasks to complete these tests. | Finish implementing, executing, and debugging whitebox tests, and store the results in the repository and Jira. |
| Update to Requirements Traceability Matrix | Reviewed the existing project requirements and discussed how to effectively update them. | The requirements traceability matrix will be kept up-to-date and stored in the repository. |
| Jira Project Update | Discussed updating the Jira project to show activities and progress. | Agreed to share progress transparently among team members via Jira and to perform regular updates. |
| Completed Scrum Report | Discussed writing a completed scrum report, including reflection questions. | Decided to use the scrum report to facilitate team reflection, gather feedback, and continuously improve project processes. |
| Deadline Awareness | Emphasized the importance of meeting deadlines and reminded them that there is a 20% penalty for being late. | All team members agreed to honor the deadlines set via Jira and to notify the team leader of any changes. |

**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 |
| Completion of Integration Test documentation and Coding | Integration tests were set as the highest priority, with Khassan and Alejandro responsible for creating and adding 4 sets of unique integration test cases to the repository. |
| Completion of Acceptance Test documentation and Coding | After completing integration tests, acceptance tests were set as the second priority. Seulgi and Philip will each create one acceptance test case per requirement and add them to the test case in a documentation. |
| Updating Requirements Traceability Matrix | After completing acceptance tests, updating the requirements traceability matrix was set as the third priority, with Seulgi responsible for updating and storing it in the repository. |
| Completing Whitebox Tests | Philip will complete the whitebox tests and store the results in the repository and Jira. |
| Writing Scrum Report and Reflections | The scrum report and reflections will be written and reviewed in the next meeting. |
| Updating to Jira Project | To keep all team members transparently informed about the project's progress and ensure everyone is up-to-date, active use and regular updating of Jira were agreed upon. |
| Adhering to Deadlines | All team members will adhere to deadlines set via Jira and notify the team leader of any changes. |

**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? |
| Seulgi Lee,  Khassan Suleimanov,  Philip Grahamm,  Alejandro Mercado | Drafting the scrum report. | 30min | Yes |
| MS4 setup of Jira issues. | 10min | Yes |
| Exchanged opinions on the answers to the reflection questions. | 10min | Yes |
| Assigned roles and completed Jira issues and assignments. | 10min | Yes |
| Discussed and shared format and understanding of Integration Tests | 10min | Yes |
| Collaborated on format and understanding of Acceptance Tests | 10min | 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 |
| Seulgi Lee | Complete scrum report. |
|  | Make timeline and board about Milestone4 using Jira and assign team members.. |
|  | All acceptance tests implemented and added to the testing C++ project |
|  | Update requirements traceability matrix stored in the repository. |
|  | Update Jira with the work performed and planned. |
| Khassan Suleimanov | Answer reflection question 2. (about wrote integration test) |
|  | Update function-description document. |
|  | Integration tests document (for the new functions you added) stored in repository with at least 4 sets of distinct test cases (each case must have at least 4 distinct test data). |
|  | Update Jira with the work performed and planned. |
| Philip Grahamm | Finish coding blackbox and whitebox cases (well-designed, written, and documented) Store in repo, executed, results in Jira (and on corresponding test documents, and debugged. |
|  | One acceptance test case for each requirement added to the test cases document. |
|  | Answer reflection question 1. (about difference between manual and automated testing) |
|  | Update Jira with the work performed and planned. |
| Alejandro Mercado | Integration tests coded (store in repo), executed (results in Jira and in test documents) and debugged. |
|  | Finish coding all functions and main (well-designed, written). |
|  | Answer reflection question 3. (about integration test coding) |
|  | Update Jira with the work performed and planned. |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Finalized Integration Test Plan | Ensures all team members have a clear understanding of their responsibilities for writing, implementing, and executing integration tests. This will lead to a comprehensive and systematic approach to integration testing, improving the overall quality and robustness of the code. |
| Defined Acceptance Test Procedures | Establishes a clear process for acceptance testing, ensuring that all outputs meet user requirements. This will help in demonstrating the software’s functionality and reliability to stakeholders, increasing their confidence in the project. |
| Updated Requirements Traceability Matrix | Keeps the project aligned with initial requirements and tracks the fulfillment of these requirements through to the final product. This provides a clear documentation trail and ensures no requirement is overlooked. |
| Jira Project Update | Agree to share progress transparently among team members via Jira and to perform regular updates. |
| Completed Scrum Report | Decide to use the scrum report to facilitate team reflection, gather feedback, and continuously improve project processes. |
| Adherence to Deadlines | Ensures timely completion of tasks, which is critical for maintaining project momentum and avoiding penalties. This will help in delivering the project within the stipulated time frame, meeting all deadlines effectively. |

**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 |
| Held scrum meeting after class | Held a Scrum meeting immediately after class. It was time-efficient, allowed everyone to meet in person, and we could ask the professor questions directly if needed. |
| Discussion on Reflection | We discussed the overall reflection and decided that each person responsible for a particular task would write their part. This approach ensures that the reflection is more accurate and effective. |
| Discussion of Integration Test Plan | Team members understood the importance of integration tests and clearly defined their roles and responsibilities. There was active exchange of opinions and feedback, facilitating smooth collaboration. |
| Discussion of Acceptance Test Procedures | The team recognized the importance of acceptance tests and the need for manual testing. By discussing test cases for each requirement, a clear procedure was established. |
| Discussion of Updating Requirements Traceability Matrix | The team had a clear understanding of project requirements and discussed the need to update the traceability matrix. Continuous collaboration and communication among team members supported this process. |
| Task Sharing via Jira | The team lead shared tasks on Jira, increasing clarity about the tasks and helping team members understand their roles better. |
| Prioritizing and Handling Tasks | By prioritizing and handling tasks in order, we ensured that critical tasks were completed on time, enhancing the overall efficiency of the team. |
| Discussion of Adhering to Deadlines | All team members recognized the importance of meeting deadlines and discussed how to collaborate effectively to ensure this. The team leader emphasized setting clear deadlines and reminding team members regularly. |

**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 |
| none | none |

**Reflections**:

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

1. **What is the difference between manual and automated testing? Why are we automating the testing process and what benefits does automation offer?**  
     
   Manual testing involves human testers manually executing test cases without using tools or scripts, making it ideal for exploratory, usability, and ad-hoc testing scenarios. In contrast, automated testing uses scripts and tools to run tests automatically, providing efficiency, consistency, and scalability. This approach is particularly useful for repetitive tasks like regression, load, and performance testing. Automatic tests also increase speed and improves cost efficiency. It is also beneficial for most repetitive tasks like regression, load, and performance testing. By doing this, we also limit the possibility of human errors which produce more reliable results.
2. **Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?**   
     
   Integration tests are really crucial even after whitebox and blackbox test cases. Firstly, they ensure different parts of the code work well together, not just on their own. Also, they check that data is passed correctly between parts of the system. Integration tests heps us to find problems that only appear when parts interact. Moreover, they simulate real use cases to see how the system behaves. Overall, integration tests make sure new changes don’t break existing interactions and they catch issues that unit tests might miss, like how different parts work together. By ensuring that all components work together as expected, integration tests provide an additional layer of confidence in the system's overall functionality and reliability.
3. **List and describe one of the integration tests you created. Provide a thorough explanation of how the integration operates, detailing the flow of parameters from one function to another. Use one of your integration tests to support your answer.**

I created integration test for ‘validateShipment’ function, this test verifies that the ‘validateShipment’ function correctly integrates the ‘validateBox’, ‘validateWeight’, and ‘validateDest’ functions to ensure a shipment is valid. The test uses various Shipment structs with different sizes, weights, and destinations to check if the function returns the expected validation results. This test verifies that the ‘validateShipment’ function correctly integrates the individual validation functions ‘validateBox’, ‘validateWeight’, and ‘validateDest’ to determine the validity of a shipment.

Flow of Parameters:

‘validateShipment’ is called with a Shipment struct.

‘validateShipment’ first calls ‘validateBox’ with the size from the Shipment.

If the size is valid, it proceeds; otherwise, it prints "Invalid size!" and returns 0.

If the size is valid, ‘validateShipment’ then calls ‘validateWeight’ with the weight from the Shipment.

If the weight is valid, it proceeds; otherwise, it prints "Invalid weight! (must be 1-2500 Kg)" and returns 0.

If the weight is valid, ‘validateShipment’ calls ‘validateDest’ with the dest from the Shipment.

If the destination is valid, it returns 1; otherwise, it prints "Invalid destination!" and returns 0.

This integration test ensures that ‘validateShipment’ functions correctly by combining and utilizing the results of the individual validation functions. Each test case validates a different aspect of the shipment, ensuring comprehensive coverage of all possible scenarios.