# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |
| --- |
| Kate De Leon |
| Jarod Jian Kang Hery Chen |
| Ronak Jung Rayamajhi |
| Carson Ji |
| Kemono Onomek |

## 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** |
| Kate De Leon | * Integration test case * Finishes Blackbox and Whitebox cases | * None |
| Jarod Jian Kang Hery Chen | * Updated the requirements traceability matrix * Integration test code * Acceptance tests | * None |
| Ronak Jung Rayamajhi | * Completed the main * Acceptance tests * Updated Jira and GitHub repository control * Debugged some bugs in the code * Scrum report | * None |
| Carson Ji | * Integration test code * Test execution * Debugged some bugs in the code | * None |
| Kemono Onomek | * Integration test case * Finishes Blackbox and Whitebox cases | * 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** |  |
| **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 Testing | Discussed about Integration testing and possible issues | Completed |
| Acceptance Tests | Discussed about Integration testing and possible issues | Completed |
| Scrum report | Discussed the report and reflection | Completed |
| Jira | All tasks were updated | Completed |
| GitHub | Updated the repository to current progress | Completed |
|  |  |  |
|  |  |  |

**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 |
| Task Priority | All the members were assigned with equal amount of work. |
| Acceptance tests | All the requirements were clear to everyone. |
| Implementation and main file | Everyone had clear idea implementing the functions and compiling everything. |
|  |  |
|  |  |

**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? |
| Integration Testing | Discussed about the integration testing, how it works and why it is necessary. | 1 hour | Yes |
| Acceptance Testing | Discussion about the acceptance test, its purpose and its working method. | 1 hour | Yes |
| Scrum report | Some question for the reflection was discussed | 20 minutes | 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 |
| Ronak Jung Rayamajhi | Jira and GitHub repository management |
| Everyone | Final test execution |
| Everyone | Attend final meeting |
| Everyone | Review test matrix and test report |
| Everyone | Scrum report and reflection |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Proper division of work among the group members | Everyone was able to do their task in time. |
| Acceptance testing and Integration testing | All the members had better idea about this topic after the meeting |
| Issues solved | Member who had some issues in their code were able to solve them after bringing it up in the meeting |
|  |  |

**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 |
| Creation and documentation of Integration Test cases | Contribution from all the members |
| Creation and documentation of Acceptance testing | Contribution from all the members |
| Update traceability matrix | Contribution from all the members |
| Main file implementation | Proper discussion and planning |

**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**:

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 is when people test software by using it and checking if it works as expected. It's important for trying out new ideas, making sure it's easy to use, and doing random checks. But it takes a lot of time, can have mistakes, and isn't good for doing the same test many times. It costs less at first but gets expensive because it needs more people over time.

Automated testing uses tools and scripts to test software automatically. It's great for checking if old features still work, testing performance, and doing the same tests repeatedly. It gives faster and more accurate results. Although it costs more at the start, it saves money later because it needs less human work. Automated testing can test more things quickly, making it good for big software projects.

We’re automating testing because it makes things more efficient. Automated tests run more often, helping us find mistakes in the code quickly. They also reduce the chance of human errors that can happen with manual testing, so we get more reliable results. Plus, automated tests can run in different environments at the same time, which saves a lot of time compared to having multiple people test manually.

1. Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?

Even after passing black box and white box testing, the software still needs to go through integration and acceptance testing before it's ready for release.

Integration Testing: This checks if different parts of the software work together properly. While individual components might work fine on their own, integration tests make sure they function well as a whole. This helps find issues that might only show up when the components interact with each other.

Acceptance Testing: This type of testing ensures the software meets the user’s needs and expectations. It simulates real-world use to confirm the software behaves correctly from the user’s perspective. Passing black box and white box tests doesn't guarantee users will be satisfied, so acceptance tests are crucial to ensure the software works as intended for its users.

In summary, integration and acceptance tests are essential for confirming that the software works well as a complete system, meets user requirements, and behaves correctly in real-life scenarios. They are the final check before the software is deployed, ensuring it is of high quality and usable.

1. 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.

The integration test Test1BlueDeliveryNoDiversion verifies the selection and parameter for a Blue truck in a delivery system. Initially, three trucks (Blue, Green, Yellow) are reset to zero weight and volume. The test sets a shipment's weight to 20, size to 1, and destination to "12L". It then calls the findBestTruckShipment function with these parameters. This function processes the trucks to determine the most suitable one for the delivery. The test expects the Blue truck to be selected, so it asserts that the returned truck's symbol matches the Blue truck's symbol, and that its weight and volume are updated correctly to 20 and 1, respectively. This ensures that the integration between the truck data and shipment assignment functions operates correctly, validating the system's ability to select the appropriate truck and update its parameters accurately.