# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Raiz Hossain | 4. Judd Niemi |
| 1. Mustafa Siddiqui | 5. |
| 1. Abdiwali Warsame | 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 newly added functions) must be stored in the repository. It should include at least 4 distinct test cases, with each test case containing a minimum of 3 unique test data inputs. Ensure the document clearly specifies the names of the integrated functions being tested.
* 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 in the repository, ensuring it shows both passed (green) and failed (red) tests.
* Completed scrum report including reflection questions answered.

Note: Your professor will **only grade** the **master** or **main** branch, unless you indicate otherwise.

**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** |
| Riaz Hossain | Integration test case |  |
| Mustafa Siddiqui | Functions |  |
| Abdiwali Warsame | Black and white box tests |  |
| Judd Niemi | Scrum, Jira and 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 |
| Debugging | **Function was crashing** | **Fixed the function** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Riaz Hossain | Test cases and matrix |
| Mustafa Siddiqui | Coding |
| Abdiwali Warsame | White box tests |
| Judd Niemi | Black box tests |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| debugging | **Everyone worked together** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 challenges did you encounter when testing the interactions between different components during integration testing? Reflect on how identifying and resolving these issues improved the overall functionality of the system.  
     
     
   One of the main challenges we faced during integration testing was getting different functions to work together smoothly. Sometimes a function would work fine on its own but would not behave as expected when used with others. For example, when the function that calculates distance was used with the one that assigns the truck, it gave incorrect results at first because of how data was passed. We had to trace the problem step by step to see where things were going wrong. Fixing these bugs made the system more accurate and reliable. It also helped us understand how important clear data flow and proper function integration are. Once we resolved the issues, the overall program worked better and more consistently. We learned that it’s not enough for functions to work alone—they need to work together properly to solve the bigger problem.

1. How did focusing on end-user requirements during acceptance testing influence your approach to creating test cases? Reflect on how this perspective helps ensure the software meets its intended purpose.  
     
   Focusing on the end-user during acceptance testing helped us think differently. Instead of just making sure the code works, we started thinking about what the customer expects the program to do. For example, the customer wants the system to reject packages that are too heavy. We made sure we had a test that gave a package over 5000kg to see if the system rejected it properly. Looking at it from a user’s point of view helped us catch things we might’ve missed. It tested us for real-life situations, not just technical ones. This way, we felt more confident that the program would work the way the users expect it to and be helpful to them.
2. 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.

One of the integration tests we made checked how the system assigns a truck to a delivery. In this test, we passed the weight, box size, and destination to the selectTruck() function. That function calls findClosestTruck() to figure out which truck is closest and can carry the load. After that, it uses calculateDiversion() to see if the truck needs to take a different route. All of these smaller functions work together in one big process. The test showed that a package for address 8Y, with 2000kg and 2m³, was correctly assigned to the GREEN line with a diversion. This test proved that the functions were passing the right data to each other and making the correct decision in the end.