# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Agenor Dionizio da Silva Junior | 4. Renan De Alencar Queiroz |
| 2. Ashley Shin | 5. Thanh Dat Trinh |
| 3. Ian Hartog | 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** |
| Agenor Junior | • Integration tests  • Traceability matrix |  |
| Ashley Shin | • Integration tests  • Traceability matrix |  |
| Ian Hartog | • Integration tests  • Traceability matrix |  |
| Renan Queiroz | Scrum Report and Reflection |  |
| Thanh Dat Trinh | • Integration tests  • Traceability 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 |
| Division of work | Each member will implement the respective function created last milestone, Create and run the tests, as well to fill pertinent documentations (Excel Tables) | Agreed |
| Organization | Each person Will create and upload its own file to the GitHub. In the end, all the files would be joined to one main file. | Agreed |
| Scrum Report and Reflection | One person will fill the Scrum Report and another one will fill the Reflections. | Agreed |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Prioritization of tasks | Equal as possible amount of work assigned to each member of team. |
| Function implementation | Implement functions created in previous Milestone |
| WhiteBox Texting | Implement, execute and record. Also fill Excell Tables. |
|  |  |
|  |  |
|  |  |
|  |  |

**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 Members | Analysis, discussion of debugging black box testing that was done last week and discuss about white box implementation and execution | 1h | yes |
| All Members | Analysis and implementation of functions in program and discussed | 1:30 | yes |
| All Members | Jira and Github Project page updated and assigned | 00:30 | 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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Function implementation | It’s Done according to functions specs |
| Black Box Testing | Implemented and executed |
| White Box Testing | Implemented and executed |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

Based on what we saw on class, Manual testing involves a human tester manually executing test cases without the assistance of tools or scripts. This type of testing is useful for exploratory, ad-hoc, and usability testing where human intuition and experience are invaluable. However, it is time-consuming, prone to human error, and not easily scalable. Automated testing, on the other hand, uses scripts and tools to run tests on the software automatically. This approach is ideal for repetitive, regression, and performance testing. Automation ensures consistency, as the same tests can be run in the same manner repeatedly without variation. It is also faster and more efficient, especially for large projects with extensive test suites. Moreover, automated tests can be run at any time, often integrated into a continuous integration/continuous deployment (CI/CD) pipeline, allowing for rapid feedback on the code quality. The primary benefits of automation include increased test coverage, improved accuracy, faster execution, and the ability to run complex test cases. By automating the testing process, development teams can identify defects earlier in the development cycle, reduce the time and cost of testing, and ultimately deliver higher quality software more quickly.

1. Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?   
     
   Integration tests are essential even after code has passed blackbox and whitebox tests because they serve a different purpose. Blackbox tests focus on the functional requirements without considering the internal workings of the code, while whitebox tests evaluate the internal structures and logic. Both are typically limited to testing individual units or components in isolation. Integration tests, however, ensure that different components or modules of the application work together as expected. When individual units are combined, they may interact in unexpected ways, leading to integration issues that would not be identified through unit testing alone. These tests verify that the interfaces between modules are correct and that the integrated components produce the desired outcome.
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.

Taking this method as example:  
TEST\_METHOD(integration\_T001)

{

struct Truck truck = { 0 };

struct Package package;

struct Map basemap = populateMap();

truck.currentWeight = 100.0f;

truck.currentVolume = 97.0f;

package.weight = 2400.0f;

package.volume = 3.0f;

package.destination.row = 6;

package.destination.col = 4;

bool result = addPackage(&truck, package, basemap);

Assert::AreEqual(true, result);

}  
The truck and package are assigned specific weights and volumes to simulate a realistic scenario. The addPackage function is called to attempt adding the package to the truck. The addPackage function likely performs the following checks and operations:

* Weight Check: Ensures the package weight does not exceed the truck's capacity.
* Volume Check: Ensures the package volume fits within the remaining truck space.

If all checks pass, the package is added to the truck, and the function returns true. This ensures that under the given conditions, the addPackage function behaves correctly and the package is added as expected.