# **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**: 6

**Members Present**:

|  |  |
| --- | --- |
| 1.Sean Li | 4. Dhanuth Chathurmika |
| 2. Sahan Vimukthi | 5. |
| 3. Ileperuma Achchige Dineth Damishka Gunarathna | 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 in the repository, ensuring it shows both passed (green) and failed (red) tests.
* 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** |
| **Sean Li** | **All functions in shipment.c, intended to complete the program, preliminary tests in old\_tests.c, and initial tests in the test project** | **Waiting for others to complete the tests/any remaining incomplete items from previous milestone** |
| **Dineth Damishka** | **Finalized the scrum report** | **None** |
| **Sahan Vimukthi** | **Created integration test cases** | **Yet to complete and add more test cases** |
| **Dhanuth Chathurmika** | **Updated traceability matrix file** | **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** | **Whitebox test cases** |
| **Reason for delay or block** | **Function dependency not yet pushed to repo** |
| **Impact on Project** | **Incomplete whitebox coverage** |
| **Solution or work-around** | **Wait for function push; pair programming suggested** |
|  |  |
| **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 Test Issues | **Difficulty in linking and parameter passing between modules** | **Assigned debugging and code review to Sahan and Dineth** |
| Test Matrix Updates | **Discussed need to show passed/failed status clearly** | **Standardized green/red color codes for status** |
| Jira Task Assignments | **Some tasks not linked to commits** | **Agreed to tag all future commits with Jira ticket IDs** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Use function-level test IDs in traceability matrix | Easier to track test coverage |
| Manual execution for acceptance tests | User input/output validation needed |
| Scope of integration/user acceptance test | Projecting based on our ability and past performance with milestones, what we can realistically do, as well as what we may not be able to accomplish by the due date |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Sahan | **Contributed to report, coded some integration test cases** | **2 hrs** | **Done** |
| Dhanuth | **Updated Traceability Excel sheet, coded some test cases** | **1 hr & 45 ms** | **Done** |
| Sean Li | **Updated Jira with results and finalized the source code** | **2 hrs** | **Done** |
| Dineth | **Contributed to report, coded some integration test cases** | **2 hrs** | **Done** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Dineth | Finalize and submit integration test documentation |
| Dhanuth | Refactor whitebox tests and rerun with edge cases |
| Sean Li | Complete traceability matrix updates and finalize Jira tasks |
| Sahan | Finalize and submit integration test documentation |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Team agreed on test documentation standards | **Prevents inconsistencies across deliverables** |
| Debugged critical whitebox issues | **Enabled tests to pass and improved coverage** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Jira Update and Coordination | **All members updated Jira with correct statuse** |
| Scrum Report Completion | **Team collaborated efficiently and met documentation goals** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Integration test consolidation | **No clear owner, delayed completion. Assign role earlier next time.** |
| Whitebox function dependencies | **Code was not committed on time. Implement deadline tracking.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 biggest challenges we encountered during integration testing was managing the flow of data between multiple components, especially when the output of one function had to be adapted to match the input requirements of another. In several cases, we had to refactor the return types and parameter lists to maintain consistency. Identifying these mismatches early allowed us to address subtle logic issues that wouldn't have been caught during unit testing. This process ultimately improved the overall robustness of our system, as it forced us to ensure that modules were not only functioning individually, but also collaboratively. By catching these integration issues, we reduced the chances of runtime errors and user-facing bugs.
2. 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 end-user requirements shifted our mindset from thinking technically to thinking functionally. Instead of simply verifying that the code executes without errors, we had to consider whether it produced outputs that aligned with what a user would expect under real-world conditions. This led us to create test cases based on user stories, actual usage scenarios, and boundary conditions derived from the requirements. For instance, instead of just checking that a file was read, we checked that it handled edge cases like empty files or invalid data formats. This perspective helped us bridge the gap between code correctness and user satisfaction, which is essential for ensuring the software meets its intended purpose.
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.

One integration test we created focused on the interaction between parseInput() and downstream validation logic such as validWeight(), validVolume(), and canCarryShipment(). The test used structured input lines (e.g., "100 0.5 A2") to verify that parsing produced correct data objects, which were then validated for weight, volume, and delivery eligibility. We tested with four variations, including malformed or edge-case data. This was critical in ensuring that parsing logic and business rules were correctly aligned, especially when interpreting destination formats and checking truck capacity constraints.