# **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**: **B**

**Members Present**:

|  |  |
| --- | --- |
| 1. Aung Moe Thwe | 4. Thiri Aung |
| 2. Jhonatan Lopez Olguin | 5. |
| 3. Kashish Verma | 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** |
| **Aung Moe Thwe** | Function implementation and testing | **N/A** |
| **Kashish Verma** | Developing blackbox test cases | **N/A** |
| **Thiri Aung** | Implementing and documenting whitebox tests | **N/A** |
| **Jhonathan** **Lopez Olguin** | Updating requirements traceability matrix | **N/A** |
| **ALL** | Reflection | **N/A** |
|  |  |  |
|  |  |  |

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** | **N/A** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |
|  |  |
| **Delayed or Blocked Task** | **N/A** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |

**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 | **How to implement and setting a blueprint for what to do** | **Decided to implement it in another c file** |
| Acceptance Testing | **What is it and how it outlines with the business requirements** | **Made plans to add it to the unit test project** |
| Hooks | **Implementation in everyone’s environments** | **Delayed** |
| debugs | **Review of the bugs found, including all new and old ones.** | **Found some new bugs and fixed some old ones.** |
| Updating jira and github | **Setting deadlines and to better utilize them** | **N/A** |
| Plan outlines | **Discussed plans to outlines what to add for testing implementation** | **Modified excel files** |
|  |  |  |

**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 |
| Change source code | Ensuring all parts of the codes work as expected. Without affecting the already implemented and successful test cases. |
| Fix traceability matrix | Update it to more efficiently reflect what we have to implement |
| Better Use of Jira and GitHub | Ensure it’s more readable and concise so third-party or other new members could properly read and understand |
| Expand Test cases | Update Acceptance and Integration testings to better integrate and automate testings. |
|  |  |
|  |  |
|  |  |

**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? |
| Kashish Verma | Integration test case document | **35** | **Yes** |
| Aung Moe Thwe | Integration test code | 30 | **Yes** |
| Thiri Aung | Implementing whitebox test cases | **25** | **Yes** |
| Jhonathan Lopez Olguin | Updating requirements traceability matrix | **25** | **Yes** |
| All | The scrum report and updating assigned functions | **20** | **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 |
| Johanathan | Traceability Matrix Update |
| Kashish Verma | Test Report Finalization |
| Aung Moe Thwe | Source Code Implementation |
| Aung Moe Thwe | Complete Solution |
| Thiri | Debugging of the remaining bugs |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Deadlines Update | **We can predict when we can finish the project.** |
| Finalized solution | **This makes it easier to focus on the end-user perspective side of the code. Especially the acceptance and fulfillment business requirements** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Source Code Implementation | **This went well as we finished the program. But it is not bug-free. We started this early on way before it was required to do so.** |
| Unit Test Framework | **We had set up templates for this, and this made it easier to just implement them** |
| Testing Easier | **Finished up frameworks that could make future tests much easier.** |
| Majority of the Code implemented | **Most of the main sections of the program is done and implemented. Mainly need to find and fix bugs** |
|  |  |
|  |  |
|  |  |

**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 |
| N/A | **Everything went smoothly in this meeting** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

The difference between them, as the name suggests, is that one is done by a human “manually”, whereas the other is done by itself with the help of either a script or some form of tool. Manual testing is ideal when used in situations where human judgement is crucial and scenarios where we need to explore and adapt to different applications. This takes more time but is not useful in repetitive cases. This is when automated testing comes in. It outshines manual methods in repetitive, regressive and performance testing. They execute way faster than manual testing, especially in large test suites, with greater accuracy and reliable results (because of the decrease in human error). However, they require an initial investment in the utilization tools and time to write the scripts. But in situations where human judgement is needed, manual testing is the better option.

1. Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?   
     
   It is necessary to write the integration tests given that the code has already passed blackbox and whitebox tests because they ensure that various system components work together in harmony, even after code has passed blackbox and whitebox testing. Moreover, integration tests primarily target the interaction between modules, whereas whitebox tests look at the internal logic and structure and blackbox tests verify the system's exterior behavior. Therefore, these tests assist in locating flaws that individual unit tests could overlook, such as unexpected interactions, overlapping interfaces, and problems with data flow. Furthermore, integration tests can also identify system-wide problems and performance issues that may affect the overall functionality of the application.

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.

An example of the integration testing we did was simulating the delivery with a valid input and no diversion. This case tests a delivery with a shipment weight of 20 units and a volume of 5 units to the target location 12L.   
First, we initialize the main functions (populateMap and initializeTrucks). Then we simulated the function calls using the simulateDelivery function. It contains the parameters that the user would need to input to test the overall program. In this case, 20, 3, 12L, initialized map and arrays of trucks and number of trucks. Inside the function, Shipment structure is created and populated with the provided weight and volume. The target location is parsed using scanf and processPackage function was called to handle the logic of determining which truck to use for the shipment and checking the truck's current capacity, routes, and positions. Let's have a look at the flow of parameters. From main to simulateDelivery, we passed the parameters weight, volume, target, map, trucks array, and the number of trucks. Then we use the parameters of the simulateDelivery function to create the structure of the shipment and determine the target coordinates. Then we go to the processPackage function that uses the parameters from simulateDelivery to decide how to handle shipment, which trucks to assign, and how to update the truck’s route and position.