# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_**D**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Manas Gandotra | 4. Gurmehak Kaur Uppal |
| 2. Ajaypartap Singh Maan | 5. |
| 3. Arshnoor Kaur | 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** |
| **AJAYPARTAP SINGH MAAN** | **Create integration test document (well-written, complete, good test data) + Complete scrum report & reflections** | **NONE** |
| **ARSHNOOR KAUR** | **Finish coding integration tests (well-designed, written, and documented) + Test execution (performed, results recorded, issues created) + Jira Updation** | **NONE** |
| **MANAS GANDOTRA** | **Implement integration tests (well-designed, and documented) + Debugging (bugs fixed, documented, Jira updated)** | **NONE** |
| **GURMEHAK KAUR UPPAL** | **Update requirements traceability matrix + Whitebox test code (well designed and documented)** | **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 tests document | **Discussed the test cases that can be formatted while performing integration testing.** | **The integration test cases were formatted timely so that the following code could progress.** |
| Whitebox tests coding and execution | **Discussed the coding involved in white box test cases and putting results in Jira and docx.** | **The code ran properly with no failures in uploading on Jira.** |
| Integration tests coding and execution, putting results in Jira and docx. | **Discussed the integration test cases, their implementation, and corresponding documentation** | **Integration test cases are documented, implemented, and stored in the repository** |
| Debugging errors raised in white box testing and integration testing | **Everybody discussed debugging the errors raised while running integration test cases.** | **Bugs fixed, documented, and Jira updated** |
| Scrum report and reflection | **Reflection questions were discussed and updated as per each member’s opinion.** | **Completed Reflection Questions** |
| Update requirements traceability matrix | **Updated the matrix with new requirements and test cases** | **The matrix is up to date and stored in the repository** |
| Acceptance Test Cases for Each business requirement | **Thoroughly discussed the acceptance test case with the help of the PDF given** | **Completed acceptance test cases on time.** |
|  |  |  |

**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 |
|  |  |

|  |  |
| --- | --- |
| All deliverables should be ready 6 hours prior to the deadline | The submission of deliverables prior to the deadline ensures there is enough time for error checking and helping each other with remaining tasks. |
| Any possible delays in submission or inability to join the meeting should be reported in advance | **Effective communication between group members is crucial. Reporting tasks in progress on Jira gives time for other team members to come up with a solution or work together on the problem as soon as possible.** |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| AJAYPARTAP SINGH MAAN | **Overviewed the milestone 5 pdf thoroughly, acceptance testing discussed** | **30 MIN** | **YES** |
| ARSHNOOR KAUR | **Overviewed the milestone 5 pdf thoroughly, integration testing deliverables discussed** | **30 MIN** | **YES** |
| MANAS GANDOTRA | **Overviewed the milestone 5 pdf thoroughly, Scrum Report discussed along with reflection questions.** | **30 MIN** | **YES** |
| GURMEHAK KAUR UPPAL | **Overviewed the milestone 5 pdf thoroughly, Jira management discussed along with traceability matrix.** | **30 MIN** | **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 |
| AJAYPARTAP SINGH MAAN | **Scrum report & reflections + Debugging** |
| ARSHNOOR KAUR | **Test execution (performed, results recorded, issues created)** |
| MANAS GANDOTRA | **Final Test Report + JIRA management** |
| GURMEHAK KAUR UPPAL | **Update requirements traceability matrix + Acceptance test code solution (well-designed and documented)** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
|  |  |
| Milestone 5 towards completion | |  | | --- | | **The deliverables for MS5 are on track to be successfully submitted.** |  |  | | --- | |  | |
| Milestone 5 tasks assigned to each member | **Milestone 5 will be completed before the deadline with all deliverables documented and tested.** |
| The project requirements understood in detail | **Each team member is aware of their assigned tasks and the project requirements for smooth execution and timely submission.** |
| Integration and Acceptance Testing understood in detail | **Each member reviewed the integration and acceptance test cases lectures along with PDF to understand the project effectively.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
| |  | | --- | | **Scrum Report** |  |  | | --- | |  | | **Everyone participated efficiently and answered reflection questions thoroughly.** |
| Jira Task Assignment | **Members chose their roles effectively and collaborated well on task progress.** |
| Integration Testing | **The integration test cases were thoroughly reviewed, and the results were accurately documented and executed.** |
| Acceptance Testing | |  | | --- | | **The test cases were well-documented and implemented with good test data, leading to effective verification of functions.** |  |  | | --- | |  | |
| Traceability Matrix | **The matrix was updated promptly and accurately, reflecting the current requirements and test cases.** |
| Team Collaboration | **Team members worked together cohesively, helping each other complete tasks and address any issues promptly.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
|  |  |
|  |  |
| Everything went well as far as milestone 5 is concerned. | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

Test cases are completed manually by human testers without the use of automated technology. This approach is often needed for assessing user interfaces, carrying out exploratory testing, or handling complex and erratic test scenarios. Testers can utilize their intuition and experience to spot issues that automated testing might miss since manual testing is flexible. It can be less efficient, take longer, and be more prone to human mistake for repetitive tasks.

In contrast, automated testing uses scripts and software tools to run test cases automatically. When it comes to laborious and repetitive jobs like regression testing and performance testing, where the same test cases must be run again, automation is especially helpful. The testing process can be automated, which has various benefits, such as:

- Efficiency: Compared to manual testing, tests can be run more quickly and often.   
- Consistency: Automated tests yield consistent results while lowering the possibility of human error.   
- Coverage: Comprehensive testing of various scenarios, including edge cases, which may be impossible to cover manually, is made possible by automation.   
- Cost-effectiveness: Although automated testing requires a large initial setup, it ultimately saves time and money by lowering the amount of manual labor needed.

Teams can increase test accuracy, get quicker feedback on code changes, and devote more time to exploratory and higher-level testing techniques by automating the testing process.

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

Blackbox and whitebox tests do not completely address how various modules or functions interact with one another to achieve a task; this is why integration tests are so important. Whitebox testing looks at the internal logic and structure of the code, whereas blackbox testing concentrates on confirming a system's functionality based on its requirements without taking into account the internal workings. Validating individual components and their accuracy requires both kinds of tests.

Integration tests, however, are required to guarantee that a system's many parts function together as intended. They examine the points of integration between several modules and can identify problems that result from inter-component interactions that may not show up in isolated tests. Integration tests make sure the function connects with other components of the system correctly, whereas blackbox and whitebox tests, for instance, might check that a function generates the right output and that its internal logic is sound.

However, Integration tests essentially check the system's overall operation and interactions, whereas blackbox and whitebox tests concentrate on the accuracy of particular components. To make that the entire system satisfies the necessary criteria and performs as intended, they assist in identifying problems with data flow, communication between modules, and integration with other systems.

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 is: T5

{

// Truck with more remaining weight capacity than volume

Truck truck1 = { 1, 1500.0, 25.0, 0, 0, {'B', {}}, nullptr }; // 1500 kg weight, 25 m³ volume

Truck truck2 = { 2, 1200.0, 30.0, 0, 0, {'Y', {}}, nullptr }; // 1200 kg weight, 30 m³ volume

// Determine limiting factors

char factor1 = limitingFactor(&truck1);

char factor2 = limitingFactor(&truck2);

// Check limiting factor for truck1

Assert::AreEqual('W', factor1, L"Truck1 limiting factor should be weight");

// Check limiting factor for truck2

Assert::AreEqual('V', factor2, L"Truck2 limiting factor should be volume");

// Compare remaining capacities

int comparisonResult = compareRemaining(&truck1, &truck2);

Assert::AreEqual(-1, comparisonResult, L"No comparison");

}

In TEST\_METHOD(T5), I set up an integration test to check how the limitingFactor and compareRemaining functions work together. I started by creating two truck structures: truck1 has a weight capacity of 1500 kg and a volume capacity of 25 m³, while truck2 has a weight capacity of 1200 kg and a volume capacity of 30 m³.

When I called the limitingFactor function for both trucks, it determined their primary constraints. For truck1, the weight percentage was higher than the volume percentage, so the limiting factor turned out to be weight ('W'). In contrast, truck2 had a greater volume percentage, making its limiting factor volume ('V').

Next, I passed both trucks to the compareRemaining function to check their remaining capacities. Since truck1 is limited by weight and truck2 by volume, the function returned -1, indicating that a direct comparison can't be done. This integration test checks the flow of values between functions and makes sure that the program accurately identifies and compares truck capacities based on their limiting factors.