# **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**: 5

**Members Present**:

|  |  |
| --- | --- |
| 1. Jubin Verma | 4. Sandhya Timsina |
| 2. Photswat Boonmee | 5. Udav Tamyal |
| 3. Susinta Bastola | 6. Parshav Nileshbhai Gandhi |

## 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** |
| **Jubin Verma** | **Finish WhiteBox and BlackBox Test Cases, and User Acceptance Testing, Test Execution** | **No delays or blocks** |
| **Photswat Boonmee** | **Jira Management, ScrumReport, Reflection Questions** | **No delays or blocks** |
| **Susinta Bastola** | **Update Traceability Matrix** | **No delays or blocks** |
| **Sandhya Timsina** | **Integration test code document.** | **No delays or blocks** |
| **Udhav Tamyal** | **Finish Coding all Function and Main, Test Execution** | **No delays or blocks** |
| **Parshav Nileshbhai Gadhi** | **Code Debugging, Test Execution** | **No delays or blocks** |
|  |  |  |

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 |
| Distribution and assignment of tasks | **There are certain changes to the assignment that still fit their abilities, but some team members need to learn more about other processes.** | **Tasks are assigned efficiently, taking into account each member's talents and weaknesses.** |
| Reflection questions | **What professor expectation in this MS5** | **Deep learning the complete MS5 project to satisfy the Reflection Questions requirement** |
| Treaceability Matrix | **What should be Update in Treaceability matrix** | **Update Traceability Matrix to meet all requirements** |
| MS6 Tasks assignment | **Talking on who should do each duty** | **Another meeting is required to clarify which tasks must be completed by whom.** |
| WhiteBox and BlackBox Test Cases | **What should we have to finish WhiteBox and** **BlackBox Test Cases in MS5** | **Completed understanding task and assigned this task to Jubin** |
| Test Execution | **Who can give a good result of recording and well-performed in Testing** | **Assigned this task to Parshav** |
| Integration test and Document | **Discussing on what we need to know and do on this task** | **Assigned this task to Sandhya** |

**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 |
| Task Assignment | A few roles have been reorganized in this MS5 to allow for greater flexibility for each team member, and some team members have learned new things. As a result, certain team members will require more time to become proficient in their tasks. |
| Coding Part | In every MS coding part should be finished first before doing other tasks |
| Reflection Part | We assigned the Reflection Part to be completed last because the Reflection Questions are connected to every assignment in MS5. |

**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? |
| Photswat Boonmee | **Jira Management, ScrumReport, Reflection Quesitons,** | **5 hrs** | **Completed** |
| Jubin Verma | **Finish WhiteBox and BlackBox Test Cases, and User Acceptance Testing, Integration Test, Integration test code document.** | **4 hr** | **Completed** |
| Susinta Bastola | **Updated Treaceability Matrix** | **1.30 hr** | **Completed** |
| Sandhya Timsina | **Finish Coding all Function and Main** | **3 hr** | **Completed** |
| Udhav Tamyal | **Finish Coding all Function and Main** | **3 hr** | **Completed** |
| Parshv Nileshbhai Gandhi | **Code Debugging and Test Execution** | **2hr** | **Completed** |
|  |  |  |  |

**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 |
| Photswat Boonmee | **Jira Management, Scrum Report, next meeting arrangement,** |
| Jubin Verma | **Overall Review before each member submits, Test Execution, Final Report** |
| Susinta Bastola | **Updated traceability matrix, Final Report.** |
| Sandhya Timsina | **Complete solution code running and executing successfully**  **Test Execution** |
| Udhav Tamyal | **Complete solution code running and executing successfully**  **Test Execution** |
| Parshv Nileshbhai Gandhi | **Debugging, Text execution** |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Gaining an improved understanding of MS coding concepts | **Faster learning and more effective Deep learning and project completion** |
| Traceability Matrix | **Faster and more efficient Completion of project** |
| Better understanding of MS5 Testing | **Faster learning and It is more difficult but more successful.** |
| Improved comprehension of the project | **Prepare our team for more work and be prepared to ensure that other assignments fulfill all standards.** |
|  |  |

**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 |
| Coding part | **Well communication, Good team management, Eager to work** |
| Test cases and unit testing | **Well communication, Good team management, Eager to work** |
| Reflection questions | **Well communication, Good team management, Eager to work** |
| Assignment of tasks | **Well communication, Good team management, Eager to work** |
| Traceability matrix | **Well communication, Good team management, Eager to work** |

**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?

When it comes to testing C language programs, the main areas of difference between automated and manual testing are execution and methodology. Manual testing entails a human tester carrying out test cases by hand, such as confirming C function outputs, utilizing tools like Valgrind to check for memory leaks, and confirming program behaviour through step-by-step interaction. When tests need to be repeated, this method can be laborious and prone to human mistake, but it is useful for exploratory testing where human intuition is essential. In contrast, automated testing in C runs test cases using scripts and tools. By integrating tests into a continuous integration pipeline, automation provides uniform repeatability, faster execution, and the ability to identify problems as soon as code changes. Automating the testing process guarantees effectiveness and dependability, lowers the possibility of error, and permits thorough and regression testing without requiring a lot of manual labour, all of which eventually result in C programs of superior quality.

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

Because integration tests concentrate on confirming how various modules or components of the program function as a cohesive whole, they are required even after passing blackbox and whitebox tests. Whitebox testing evaluates internal structures and logic, while blackbox testing looks at the product from the viewpoint of an external user. However, neither method by itself guarantees that individual components work together properly or manage dependencies. Integration tests assist in identifying problems that standalone testing frequently misses, such as mismatched data exchanges, improper API calls, or unexpected behaviour during component interactions. By ensuring that all linked components work as a cohesive unit, this testing layer raises trust in the application's general behaviour and lowers the possibility of problems when it is used in real-world situations.

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.

**-Integrated two functions hasroomforshipment and comparetrucksforshipemnt**

**-Pass data to fucntion and capture th e combined behaviour**

**"struct Map map = populateMap();**

**struct Shipment shipment = { { 5, 5 }, 200, 10 };**

**struct Truck trucks[] = {**

**{ getBlueRoute(), 300, 20 },**

**{ getGreenRoute(), 850, 5 },**

**{ getYellowRoute(), 300, 15 }**

**};**

**int numOfTrucks = 3;"**  
  
To guarantee smooth processing of shipments and truck capacity, one of my integration tests includes the hasRoomForShipment and compareTrucksForShipment services. In order to define a Shipment object with particular location, weight, and volume values, the test first creates a map using populateMap(). Three Truck instances are then initialized, each with its own routes and weight and volume capacities. In order to determine whether a vehicle can fit the shipment given its capacity, the integration test first provides the shipment and each truck to hasRoomForShipment, which returns a boolean answer. The information enters compareTrucksForShipment to identify the most appropriate truck based on factors like load distribution and route efficiency if a truck satisfies the capacity requirements. By capturing the combined behaviour of both functions, this integrated test makes sure that goods are assigned to vehicles that maximize their appropriateness for transportation while meeting capacity requirements. This methodical methodology verifies how modules interact with one another, identifying any possible problems that can arise from mismatched parameters or unexpected integration behaviour.