# **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.Shuja Lashkari | 4. Cristian David Vargas Marin |
| 2.Phuong Bac Nguyen | 5. |
| 3. Maryam Jawed | 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** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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** | **Test cases** |
| **Reason for delay or block** | **One of our team members had a family emergency** |
| **Impact on Project** | **It delayed the project submission by couple days** |
| **Solution or work-around** | **Team members helped him with his task to complete it** |
|  |  |
| **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 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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

**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? |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Success | **Being able to work effectively on a task that needs to be completed, if one person is having issues, the rest of the team member will try to help resolve it as soon as possible** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Issues with the test cases not working | **One of our team members encountered an issue with a test case where the results weren’t coming out correctly. He shared the problem in detail in our WhatsApp group chat, which helped us quickly understand the situation and resolve it as a team.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Time management | **At times, there was a significant delay in communication due to our different schedules, which made it harder to quickly resolve issues when someone needed help. One improvement we can make is to set a time where everyone is available, so we can collaborate more effectively and support each other in solving problems.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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.

The difficulties I encountered when examining how integration testing interacted with Mismatches in unit assumptions were discovered during integration testing when the calculateDistance() function—which is utilized for diversion paths—was verified according to the checkTruckCapacity() method. For instance: When integrated, the findNearestTruckRoute() method occasionally passed incorrect coordinates (such as negative values) because of map boundary problems, contrary to Test Case AT-R001-01's assumption that Euclidean distance operated independently. Fixing & improving: To constrain coordinates to the 25x25 grid, findNearestTruckRoute() now has input validation. Test Case AT-R006-01 has been updated to incorporate edge-case destinations (such as "0A" and "25Y"), guaranteeing condition. The algorithm is now more reliable in real-world situations (such as delivery close to map borders) by handling edge cases correctly.

1. 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.

Impact on Test Design: User requirements were the only source of information used to create the acceptance tests (such as AT-R003-01, AT-R004-01). For instance: The weight validation test case AT-R003-01 passes the condition that "Trucks must reject shipments outside 1–5000 kg." Actions to take: Weights of input: 500, 0, 6000 Check the reasoning for the denial. Result: made sure the system complies with corporate regulations to avoid overloading. Why It Is Important Gaps such as missing error messages for incorrect inputs (such as "Invalid weight" for 0) were discovered by concentrating on requirements. More clarity for warehouse workers while entering shipments.

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.  
     
   Test Case: Verify the Assignment of Shipments to the Best Truck Parts Integrated: findNearestTruckRoute() (which makes advantage of the Euclidean distance logic of AT-R002-01). Verifies against AT-R003-01/AT-R004-01 using checkTruckCapacity(). the A\* algorithm's calculateDiversionPath(). Flow: Entry: Shipment (size=2m³, weight=2000kg, destination="8Y"). Step 1: Distances are calculated by findNearestTruckRoute(): Blue Truck: 11K is the closest point (5.83). Green Truck: 6T is the closest point (4.47). CalculateDiversionPath() is the second step. It calculates the path from 6T to 8Y, such as 6T - 7T - 7U - 7V - 7W - 7X - 7Y - 8Y. Step 3: checkTruckCapacity() confirms that there is room for Green Truck (e.g., 60% capacity left). As a result, Green Truck was given the shipment along with a printed diversion course. The Reason It Works combines AT-R006-01 (destination validation) with AT-R002-01 (distance). Verifies reasoning in the actual world: Choose the closest truck with available capacity.