# **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**: **2**

**Members Present**:

|  |  |
| --- | --- |
| 1.Luca Novello | 4. Philip Ayomide Tijani |
| 2.Eric Yakimoff | 5. Tyler Kay |
| 3.Karishma Singh Mahender | 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** |
| Eric | Integration test code, Integration Test cases, Acceptance Tests, Acceptance test cases. | **n/a** |
| Luca | Integration test code, Integration Test cases, Acceptance Tests. | **n/a** |
| Karishma | Acceptance Tests, scrum report, Acceptance test cases, Traceability matrix. | **n/a** |
| Philip | Helped with Acceptance Tests, scrum report | **n/a** |
| Tyler | Helped with Integration test code, Integration Test | **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** |  |
| **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 |
| Dividing Work | Discussed the tasks that needed to be completed and the once that needed to be revised. | Updated Jira and scrum report accordingly. |
| Integration tests | Discussed the integration test code and cases. | Integration tests added to repo |
| Acceptance tests | Discussed the acceptance test code and cases. | Acceptance tests added to repo |
| Test Documents | Requirement analysis | Traceability matrix |
|  |  |  |
|  |  |  |
|  |  |  |

**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 distribution | Tasks were dispersed based on comfort with the skillsets needed as well as an emphasis on splitting the work equally. |
| Integration Test code | Function Implementation was discussed along with the test cases for revision |
| Acceptance Test code | Requirements were analyzed to create test methods. |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Eric | Task distribution, Integration test methods, Acceptance test code | **10 min** | **some** |
| Luca | Task distribution, Integration test methods, Integration test cases, Acceptance test methods | **10 min** | **some** |
| Karishma | Task distribution, Acceptance test code, Acceptance test cases | **10 min** | **some** |
| Philip | Task distribution, code review, Integration test cases | **10 min** | **some** |
| Tyler | Task distribution, code review, Integration test cases | **10 min** | **some** |
|  |  |  |  |
|  |  |  |  |

**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 |
| Eric | **Review milestone deliverables, distribute tasks, scrum report, reflections** |
| Luca | **Review milestone deliverables, distribute tasks, scrum report, reflections** |
| Karishma | **Review milestone deliverables, distribute tasks, scrum report, reflections** |
| Philip | **Review milestone deliverables, distribute tasks, scrum report, reflections** |
| Tyler | **Review milestone deliverables, distribute tasks, scrum report, reflections** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Preliminary prioritization of tasks | Allows members to specialize and get comfortable with the work they prefer doing |
| Preliminary development of Integration test code | Provided a base to create test cases. |
| Requirement Analysis | development of Acceptance test code |
|  |  |
|  |  |
|  |  |
|  |  |

**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 meeting | Meeting was productive in establishing a starting point for the week’s tasks |
| Communication | Clear and honest discourse around the parts of the project we were all comfortable with ensured that work would be completed |
| Git | Allows us to share work easier and show our contributions |
| Jira | Allows us to communicate tasks and responsibilities easier |
|  |  |
|  |  |
|  |  |

**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 | **n/a** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

**Manual Testing:**

* Involves a human tester executing test cases without automation tools.
* Requires manually checking the functionalities of an application by following predefined steps.
* Tester records results and notes any discrepancies from expected behavior.
* Useful for exploratory, usability, and ad-hoc testing where human insight is valuable.

**Automated Testing:**

* Involves using software tools to execute test cases automatically.
* Ideal for repetitive tests, regression tests, and load testing.

**Benefits of Automation:**

* **Speed:** Accelerates the testing process.
* **Accuracy:** Reduces human error.
* **Consistency:** Ensures uniformity in testing results.
* **Continuous Testing:** Allows for ongoing testing processes.
* **Efficient Feedback:** Provides faster feedback cycles.
* **Complexity Handling:** Can efficiently manage complex test cases.

Automation increases test coverage, ensures reliable execution, and allows tests to run unattended, saving time and resources in the long run.

1. Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?   
     
     
   A. Integration tests are crucial even after code has successfully passed blackbox and whitebox testing because they evaluate how different modules interact with one another within the application. Blackbox testing focuses on validating the application's functionality from an external user's perspective, while whitebox testing examines the internal logic and structure of the code. However, neither approach thoroughly addresses the challenges that can emerge when different components of the system must work together or in integration. Integration testing ensures that modules, which might function flawlessly in isolation, operate correctly when combined with other parts of the system. This type of testing helps identify issues such as mismatched data formats, incorrect API interactions, and communication errors between modules. Integration tests are essential for verifying that the system functions as one entity, rather than merely as a collection of individual parts. By creating integration tests, developers can gain confidence that the entire application behaves as expected, reducing the risk of failures when the components interact.
2. 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.

A. **Integration Test: INT\_FTFS001**

**Test Purpose:** The INT\_FTFS001 test validates the interaction between the isValidPoint function and the findTruckForShipment function. This test ensures that a valid point for a package destination is correctly processed and used in the findTruckForShipment function to determine if there is a suitable truck available for the shipment.

**Test Description:** The test checks that when a valid destination point is provided, the findTruckForShipment function can successfully find a suitable truck for the package.

**Test Steps:**

1. **Create Map and Points:** A map is populated using the populateMap function. Two trucks are defined with starting points (0, 0) and (20, 20), respectively. The destination point for the package is set to (0, 0).
2. **Route Initialization:** Each truck is assigned a simple route (path) to follow, which includes several points. The route for each truck is defined to simulate real-world scenarios where trucks have predefined paths.
3. **Validate Destination Point:** The isValidPoint function is called with the destination point (0, 0) to ensure it's a valid location on the map. This step ensures that the test scenario uses a realistic and valid package destination.
4. **Find Suitable Truck:** With a valid point confirmed, the findTruckForShipment function is invoked. It uses the map, array of trucks, the number of trucks, and the package details (including the valid destination point) to determine which truck can handle the package.
5. **Assertions:** The test asserts that the index of the suitable truck found is 0, indicating the first truck in the array is capable of handling the package.

**Flow of Parameters:**

* **Step 3:** The destination point is passed to the isValidPoint function, which checks if the point is within the boundaries of the map and returns a boolean value indicating validity.
* **Step 4:** The same valid point, along with the map and truck details, is passed to findTruckForShipment. This function internally utilizes the point to assess whether the truck can deliver the package to the specified location without any issues.
* **Step 5:** If the point is valid and the truck's capacity can accommodate the package, findTruckForShipment returns the index of the truck, confirming successful integration between checking point validity and shipment feasibility.

**Explanation of Integration:**

This integration test demonstrates how functions collaborate to achieve a common goal—efficient and valid shipment of a package. By validating the destination point first, the system ensures that only feasible shipment scenarios are processed further. The flow from isValidPoint to findTruckForShipment ensures that the data integrity of the destination point is maintained and reused, illustrating a smooth handover of parameters between functions.

This test case highlights the importance of integration testing in verifying the interactions and dependencies between different system components, ensuring they work together as expected. It demonstrates that a valid point recognized by isValidPoint leads to successful truck allocation in findTruckForShipment, confirming the seamless flow of data across these components.