# **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. Isabela Jorge Bulla | 4. Devarsh Patel |
| 2. Ketia Teta | 5. |
| 3. Abdullah Al Mahfuz | 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** |
| **Isabela Jorge Bulla** | **Code implementation, reflection question 1 and scrum report** |  |
| **Ketia Teta** | **Integration tests document and acceptance tests cases sheet** |  |
| **Abdullah Al Mahfuz** | **Implementation of white box test cases, integration test cases and acceptance test cases and answer reflection question 3** |  |
| **Devarsh Patel** | **Update traceability matrix and answer reflection question 2** |  |
| **ALL** | **Hook files** |  |
|  |  |  |
|  |  |  |

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 |
| Tasks | **Discussion about how/when each task should be done** | **Clear understanding of project** |
| Hook files | **Discussion about how should the hook files be implemented, and definition of questions to ask the professor** | **Clear understanding of hook files** |
| Integration tests | **Discussion about how the integration tests should be implemented** | **Proper implementation of integration test cases** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Ask professor about hook files | Proper implementation of hook files. |
| Prioritize integration tests | Since acceptance tests are only due in week 6, we will prioritize integration tests |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Isabela Jorge Bulla | **Fill scrum report** | **10min** | **YES** |
| ALL | **Definition of questions to ask the professor about hook files** | **15min** | **YES** |
| ALL | **Discussion about integration and acceptance test cases** | **15min** | **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 |
| ALL | Agreed to divide tasks after the tasks have been discussed and understood in the class next week. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Understanding of hook files | **Completion of all requirements and deliverables** |
| Definition of tasks | **Everyone has a clear understanding of the tasks** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Participation | **Every member participated and was able to express their thoughts and ideas** |
| Timing of the meeting | **By having a meeting later in the project timeline we had more topics to discuss.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?  
     
   The main difference between manual and automated testing is in the time required for each. In manual testing, each data set must be entered individually, which is time-consuming. On the other hand, automated testing allows the computer to handle all data entry, improving efficiency and reducing the time required.

Automating the testing process not only improves efficiency, but also benefits the development and bug-fixing stages. By using automation tools such as hook files, we ensure that all unit tests are run before code is pushed to the Git repository. This not only accelerates the testing process but also prevents defective code from being added to the repository.

In conclusion, automation speeds up testing, simplifies the process, and reduces the chances of human error. As the application grows, manually executing all unit tests becomes impractical, making automation an essential tool for testing complex and big applications efficiently.

1. Why it is necessary to write integration tests given that the code has already passed blackbox and whitebox tests?   
     
   An integration test is important because it focuses on the interactions of different functions and checks how they work together, unlike unit tests which focus on individual functions of the system. It often includes external systems or services, like databases, API, etc. It helps verify that dependencies are correctly integrated and the system interacts with them as expected. Integration tests also assess the system's performance and stability under realistic conditions, unlike unit tests. It includes testing to test the interfaces between modules. This includes checking data passed is correctly handled, and whether APIs are called correctly. It ensures that the data flows correctly through the system, from the input to the output, among various layers and components. In summary, integration testing is important for validating the correct functioning of the system as a whole.
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.

Integration test: TEST\_METHOD(T012) – Green Line.

Purpose: This integration test verifies the chooseTruckLine function’s ability to correctly select

the most suitable truck based on a shipment’s weight and volume. The test checks if the

function correctly updates the truck's weight and volume after selecting the Green line truck.

Flow of Parameters and Function Interaction:

Setup:

• Map Initialization: populateMap() creates a Map structure representing the grid of the

city. This map will be used to determine valid destinations and routes for the trucks.

• Route Initialization: getBlueRoute(), getGreenRoute(), and getYellowRoute() retrieve

predefined routes for the blue, green, and yellow trucks, respectively. Each Route

includes a series of points representing the path of the truck.

• Truck Initialization: Three Truck structures are created, each associated with one of

the routes and initialized with specific weights and volumes.

Test Setup:

A Shipment is defined with a weight of 500, a volume of 5, and a destination at coordinates (2,

19) – these coordinates are for a valid green route defined in mapping.c

Function Under Test:

chooseTruckLine Function:

Parameters Passed:

• map: Provides the grid layout of the city for route calculations.

• shipment: Contains the details of the shipment to be delivered.

• trucks: Array of truck structures, each with its own route, weight, and volume.

• 3- the Number of trucks.

Function Behavior:

• Iterates through each truck to check if it can carry the shipment using canCarry.

• For each truck, calculates the route length from the current truck’s path to the shipment's

destination using shortestPath.

• Chooses the truck with the shortest diversion route. If two trucks have the same route

length, it chooses the one with a better space-to-weight ratio using compareTrucks.

Assertions:

Assert::AreEqual(1, chosenTruck):

• Verifies that the green line truck (index 1) is selected. This is based on the expected

logic of the chooseTruckLine function, which should select the green truck if it is the

most suitable option for the given shipment.