# **Milestone 3 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. Devarsh Patel | 4. Abdullah Al Mahfuz |
| 2. Ketia Teta | 5. |
| 3. Isabela Jorge Bulla | 6. |

## Milestone 3 Tasks

In this milestone you will create issues to design the functions, design all of the functions you need to complete the project and store the specifications in the repository. As soon as the specifications start to be produced, you can start to design the blackbox tests (what they test, how to perform them and test data). Once tests are written, they can be implemented and added to the repository and any team members not otherwise busy can start to implement the functions. You will also build a function-test matrix that shows the blackbox tests for each function. This will be maintained through the testing cycle as new tests are added.

**Deliverables due 4 days after your lab day:**

* A set of AT LEAST 4 function specifications added to a new header file and stored in the repository.
* A set of blackbox tests as test documents (in an Excel file) with test data for the functions you created. At least 4 sets of test data are required for each function. You must have test cases for at least 6 functions (including all your custom function). Stored in the repository.
* **Create and add a C++ testing project to your solution.**
* Start writing blackbox test code (for the functions above) and store in repository (at least 1 is required for this milestone).
* Start implementing the functions and store them in repository (optional).
* A requirements traceability matrix added to the repository and shows the mapping between the requirements and test cases.
* Updated Jira project to show activities and progress.
* Completed scrum report including reflection questions answered.

**Rubric:**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Function specifications (documented, complete, well-written, added to the project) | 10% |
| Blackbox test cases document (well-written, complete, good test data) | 10% |
| Blackbox test code (in the C++ project) well-designed and documented | 10% |
| Functions implementation (coded in the C project & well documented) | 15% |
| Requirements traceability matrix (complete and added to GitHub) | 15% |
| Git usage (used properly with good structure) | 10% |
| 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** | **Reflection question 1, scrum report and function specifications** |  |
| **Ketia Teta** | **Create black box test cases for 6 functions and 4 sets of test data for each test case, 24 in total, reflection question 1** |  |
| **Devarsh Patel** | **Reflection question 3 and traceability** |  |
| **Abdullah Al Mahfuz** | **Add C++ unit test project, share instructions on how to set up on everyone’s personal machine and develop at least 1 black box test case** |  |
|  |  |  |
|  |  |  |
|  |  |  |

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 and Jira | **Assignment of tasks and definition of main communication form through Jira** | **More participation and comments through Jira** |
| Understanding of the project | **Making sure everyone understands what each task is and what needs to be done** | **Clear understanding of tasks** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Make Jira main form of communication | When a task is completed, add to the comments a quick description of what was done. Post questions in the task’s comment section. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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** | **10 min** | **Yes** |
| All present | **Discussion about Jira** | **5min** | **Yes** |
| All present | **Discussion about tasks** | **5min** | **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 (“test automation, “hook files”) 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 |
| Clear understanding of tasks | **More efficient execution of 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 |
| Discussion about the tasks | **Everyone was clear about the tasks and what needed to be done.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Timing of the meeting | **The meeting could have happened later in the week, so the team would have had more things to discuss. Schedule a meeting later in the week.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 blackbox tests cases and blackbox test code? Explain how we use assertion in Visual Studio to execute tests.

Black box test cases are scenarios designed to test an application’s functionality not taking into consideration the structure or code inside the software. They are built around specifications and requirements, focusing on input test data and getting expected output.

Black box test code refers to the actual implementation of the black box test cases using a programming language. The process involves encoding the black box test cases into executable scripts and writing these test code may require knowledge to write and maintain. The test code can contain logic to cater different test scenarios, do some actions, and verify some results.

Assertions refer to statements within the test code that check if a condition is true. Otherwise, the test fails if it is false. In Visual Studio, some common assertions are made within unit tests to assert the expected behavior against the code being tested.

MSTest is a testing framework developed by Microsoft and bundled into Visual Studio. It provides us with the simplest way of writing and executing unit tests. Close integration with Visual Studio and Azure DevOps makes MSTest very popular and is used on most enterprise projects.

Here are some common assertions:

**Assert.AreEqual(expected, actual)**: Checks if two values are equal.

Example: AreEqual(value1, value2 [, "error message"] )

Compares value1 to value2 and throws an exception if they are not equal. If they are not equal, the optional error string will be displayed.

**Assert.AreNotEqual(notExpected, actual)**: Checks if two values are not equal.

Example: AreNotEqual(value1, value2 [, "error message"] )

Compares value1 to value2 and throws an exception if they are equal. If they are equal, the optional error string will be displayed.

**Assert.IsTrue(condition)**: Checks if a condition is true.

Example: IsTrue(b1 [, "error message"] )

If the Boolean b1 is not true it throws an exception. If not true, the optional error string will be displayed.

**Assert.IsFalse(condition)**: Checks if a condition is false.

Example: IsFalse(b1 [, "error message"] )

If the Boolean b1 is not false it throws an exception. If not false, the optional error string will be displayed.

1. How can a traceability matrix help in the testing process?  
     
   The traceability matrix is a tool that makes sure all the requirements are covered by the test cases. This ensures that all requirements have test cases, preventing any requirements from being overlooked. It also provides a clear understanding of requirements, test cases, and defects making it easier to track issues. If any requirements change during the process, it is easy to keep track of which test case should be updated or add a new test case, reducing the risk of regression errors. By linking requirements with test cases, it enables the tester to identify the root cause of the issue. It also assists in providing a clear status of testing progress and identifying early gaps in the process.
2. Write down two of the function prototypes you submitted. Why did do you need each one of them and how will each one help you achieve the project needs?

Two function prototypes:

int validDestination(struct Map map, struct Shipment\* shipment, int row, char col);

int chooseTruckLine(const struct Map\* map, struct Shipment\* shipment, struct Truck\* trucks, int numTrucks);

These functions play an essential role in ensuring the program operates effectively. The first function, validDestination, validates user input for the destination, guaranteeing that it is within the map boundaries, corresponds to a building, and is accessible via clear pathways. This validation ensures that the destination is reachable. The second function, chooseTruckLine, identifies the best truck route for delivering shipments. It considers factors such as shipment weight and volume, the current truck capacity, and the distance between points in the route and the destination. The function evaluates routes to minimize diversions and selects the most efficient option. Implementing these functions is crucial for meeting project requirements as they address two primary needs: validating user input and optimizing shipment routes for efficiency. This approach ensures the program correctly confirms valid destinations and identifies the optimal delivery routes.