# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |
| --- |
| 1. Kate De Leon |
| 1. Jarod Jian Kang Hery Chen |
| 1. Ronak Jung Rayamajhi |
| 1. Carson Ji |
| 1. Kemono Onomek |

## 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** |
| Kate De Leon | * Function specifications and its documentation. * Function implementation and documentation. * Scrum meetings. | * None |
| Jarod Jian Kang Hery Chen | * Function specifications and its documentation. * Blackbox test cases and its documentation * Requirements traceability matrix. * Scrum meetings. | * None |
| Ronak Jung Rayamajhi | * Update the Jira page after distributing the task within the group members. * Update the GitHub repository with current progress. * Scrum meetings. | * None |
| Carson Ji | * Blackbox test code documentation. * Scrum meetings. | * None |
| Kemono Onomek | * Function implementation and documentation. * Blackbox test code documentation. * Scrum meetings. | * None |

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 |
| Function Specification | Everyone contributed on documenting new functions | Function specification done |
| Test cases documentation | Worked on creating test cases and documenting it properly | Test case documentation done |
| Test Functions implementation | Testing of the functions | Functions implementation done. |
| Scrum report | About completing the report | Scrum completed |
| Jira | All tasks were updated | Done |
| GitHub | Updated the repositor to current progress | Done |
|  |  |  |

**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 Priority | All the members were assigned with equal amount of work. |
| Function specs | Added the function specs in the finder.h file and documentation were done |
| Test cases documentation | Test cases for the functions were documented. |
| Test functions implementation | Blackbox tests to test the functions |
|  |  |

**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? |
| Everyone | Discussion for testing the program | 1 hour | Yes |
| Everyone | Analysis of function specs | 1 hour | Yes |
| Everyone | Discussion for testing the functions | 30 minutes | Yes |
| Everyone | Scrum report | 20 minutes | Yes |
| Everyone | Jira and GitHub, assign and update | 20 minutes | 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 |
| Ronak Jung Rayamajhi | Jira and GitHub repository management |
| Everyone | Meeting for the next milestone |
| Everyone | Scrum report and reflection |
| Everyone | Debugging |
| Everyone | Document Whitebox test cases |
| Everyone | Complete current Blackbox testing and the upcoming Whitebox testing |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Function specification | Functions will be used in the project and will be tested. |
| Test cases documentation | All the test cases were documented properly. |
| Testing the codes | Blackbox testing was done as per the discussion. |
|  |  |

**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 |
| Scum report | Everyone contributed. |
| Meeting | Everyone discussed and solved any problems or misunderstanding related to the tasks. |
| Documentation | Proper discussion and distribution of task. |
|  |  |

**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 |
| None | None |
|  |  |
|  |  |

**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.

Blackbox test cases are designed to validate the functionality of a software application based on its requirements and specifications without any knowledge of the internal code structure. These test cases focus on the inputs and expected outputs, ensuring that the software behaves correctly under various conditions. For example, a Blackbox test case might involve verifying that a user can successfully log into an application by providing the correct username and password. The tester designs these cases based on what the software is supposed to do based on the requirements in the documents and does not mind how the code achieves these functionalities internally.

In contrast, Blackbox test code refers to the actual implementation of these test cases in a programming or scripting language, using a testing framework or tool to automate the process. This code executes the scenarios defined in the test cases, inputs the specified data, and checks the outputs against the expected results.

In Visual Studio, assertions are used to verify that certain conditions hold true during the execution of a program. When an assertion fails, it indicates that there is a bug in the code, as the expected condition was not met. This helps in identifying issues early in the development process.

To use assertions in Visual Studio, you typically include the #include <assert> header for C++ programs. You then use the assert(expression) function, where expression is the condition, you expect to be true. If the expression is evaluated to be false, the program will terminate, and Visual Studio will highlight the location of the failed assertion. This allows you to quickly find and fix the issue.

1. How can a traceability matrix help in the testing process?

A traceability matrix helps in testing by linking requirements to test cases, making sure nothing is missed. It shows which test cases need updates when requirements change, improving team communication. It also helps manage defects by connecting them to specific requirements and test cases, and it provides proof of thorough testing for audits. Additionally, it makes test planning easier by showing how requirements and test cases relate, helping to prioritize tests and allocate resources better.

1. Write down two of the function prototypes you submitted. Why did you need each one of them and how will each one help you achieve the project needs?

Among the various function prototypes we've submitted, each one is essential for the success of this project, serving its own specific purpose. Here are two examples and their reasons:

**int isObstacle(­­­­­­­const struct Map\* map, struct Point pt);**

**Why Needed:** This function checks if a specific point on the map is an obstacle.

**How It Helps Achieve Project Needs:** The isObstacle function is required for making sure that the path planned for shipments or trucks does not pass through any obstacles. By checking each point, it maintains a safe route, preventing any navigation issues caused by blocked paths. It validates the shipment destinations and helps in route planning by avoiding obstacles.

**int validInput(struct Shipment shipment, const struct Map\* map);**

**Why Needed:** This function validates the user input for shipment details, including weight, size, and destination.

**How It Helps Achieve Project Needs:** The validInput function makes sure that all shipment details entered by the user are within acceptable ranges and constraints, such as weight limits, size limits, and valid map coordinates. It also checks for obstacles at the destination point. By validating inputs, it avoids impossible shipment details from being processed and improves the reliability and accuracy of the data used in the project.