# **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**: \_6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1.Hansol Nam | 4.Chia-yu Chien |
| 2.Wai Bong Yung | 5. |
| 3.Fang Lin | 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** |
| **Hansol Nam** | **Reflection and Unit test document, traceability matrix after black box (3 functions)** | **none** |
| **Chia-Yu Chien** | **Black box testing, function description, unit test, codes** | **none** |
| **Fang Lin** | **Black box testing, traceability matrix, unit test** | **none** |
| **Wai Bong Yung** | **Black box testing with blackbox test code (3functions)** | **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** | **N/A** |
| **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 |
| Assigning work | **Assign works for each member** | **MS 3 planned, work assigned** |
| Check Tasks Progress | **Decide the due time for tasks** | **Decided** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Assigned work | 2 members test 3 functions each with documenting |
| Deadline within MS3 | Each tasks deadline decided within MS 3 |
| Scheduling meeting | Scheduled meeting online |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Hansol Nam | **Reflection, review MS3, update Jira as needed,** | **4hr - reflections** | **done** |
| Chia-Yu Chien | **Update Jira** |  | **done** |
| Fang Lin | **Update Jira** |  | **done** |
| Wai Bong Yung | **Update Jira** |  | **done** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Wai Bong Yong | Reflection |
| Other members | Scrum reports, update Jira |
| Chien Yu | Reflection review and update Jira |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Assigned work | **Each member handle assigned task with responsibility** |
| Scheduling meeting | **Meeting scheduled to share the process of work** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Assigning work | **Every member agrees with assigned tasks** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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

**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 testing is a method that doesn’t require code inspection but instead focuses on comparing inputs and expected outputs to verify the program's behavior and functionality. Blackbox test cases are designed to validate the functional aspects of an application based on its requirements and specifications, while blackbox test code refers to the actual implementation of these test cases in a programming language or testing framework. This test code automates the execution of test cases, providing inputs to the application, capturing outputs, and comparing them against expected outcomes, thus counting as automated tests. In Visual Studio, tests can be executed by creating a unit test project, writing test methods with assertions using the ‘Test Method’ attribute, and running the tests to check coverage. Visual Studio provides tools to measure and report test coverage, ensuring all parts of the application are tested.

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

The traceability matrix is essential in the testing process, ensuring all requirements are covered by test cases. It tracks changes, identifies coverage gaps, and facilitates impact analysis. By logging test results and completing the matrix, testing progress becomes traceable, enhancing program accuracy. When requirements change, the matrix helps assess the impact on existing test cases, identifying necessary modifications or additions. This systematic approach improves software quality and reliability, ensuring thorough testing and validation of all specified requirements. Overall, the traceability matrix is a key tool for managing requirements and maintaining the integrity of the testing process.

1. 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?

• int isValidWeight(struct Shipment \*shipment);

This function is required to validate the weight of the shipment. The shipments weight should be 2500kg or under. By validating the weight, we can avoid processing shipments that are too heavy, which could cause issues in delivery to destinations. This function ensures that only shipments with valid weights are processed, thus preventing potential errors and delays.

• int isValidBoxSize(struct Shipment \*shipment);

This function is necessary to validate the size of the shipment, ensuring it is 100 cubic meters or less. By verifying the size, we can ensure that the shipment fits within the designated storage and transportation spaces, avoiding issues with packing and shipping logistics. This validation, along with weight validation, ensures that only shipments meeting all criteria are processed, thus preventing potential issues due to excess size during the delivery process. This function is essential for optimizing space usage and ensuring smooth operations throughout the shipping process.

These functions are crucial for validating shipment attributes, ensuring that only shipments meeting specific criteria are processed. They support the project's goals of maintaining accuracy, efficiency, and compliance in the shipping process. By using these validation functions, we can meet requirements and enhance the reliability of delivery.