# SFT221 SCRUM Report and Reflections

This report should be completed in the class and submitted at the end of class. Late submissions cannot be accepted without prior approval of the instructor. All students are expected to attend the in-class SCRUM meetings and to participate. Failure to do so will result in greatly reduced grades.

**GROUP**: **SFT221 NEE group1**

**Members Present**:

|  |  |
| --- | --- |
| 1. Kim Ming Chau | 4. Gordon Tan |
| 2. Hak Kan Poon | 5. Yuhong Fan |
| 3. Elvin Karikari | 6. |

## Milestone 4 Tasks

**Deliverables Due at end of Lab:**

* Completed SCRUM report and reflections

**Deliverables Due at 23:59 6 Days after Lab:**

* Implemented Functions
* Implemented blackbox tests (store in repo), executed (results in Jira and on corresponding test documents) and debugged,
* whitebox tests written and stored in repository.
* whitebox tests implemented (store in repo), executed (results in Jira and on corresponding test documents) and debugged.
* Updated function-test matrix stored in the repository.
* Completed hook for test automation

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
| Teamwork | 5% |
| SCRUM Report | 10% |
| Automation Hook | 10% |
| Group | Implemented Functions (well-designed, written and documented) | 20% |
| Whitebox tests (well-designed, written and documented) | 20% |
| Test Execution (performed, results recorded, issues created) | 20% |
| Debugging (Bugs fixed, documented, Jira updated) | 5% |
| Git Usage (used properly with good structure) | 5% |
| Jira Usage (creates issues, tracks progress) | 5% |
| Meets Deadlines | 5% |
| SCRUM Report and Reflections | 20% |

**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** |
| **Gordon Tan** | **Prepare all the function in the header file & TB11** |  |
| **HK Poon** | **Test case and code for TB01-TB03** |  |
| **KM Chau** | **Test case and code for TB04-TB07** |  |
| **Yuhong Fan** | **Test case and code for TB08-TB10** |  |
| **Elvin Karikari** | **Scrum report and reflection** |  |
|  |  |  |
|  |  |  |

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 implementation | Add and remove some functions after MS3 | RemoveisOnOriginalPath, findIndexOfShortestPath and calcMinDistance functions. Add alShortPathDist and printDiverPath function. |
| Task allocation | Divided the task for blackbox and whitebox | Each member would take part in blackbox testing and whitebox execution. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| After discussing the requirements of the milestone. | We designed and prioritized the task. We decided on how we would distribute the workload for blackbox and whitebox testing. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Gordon Tan | **Implemented the change in function** | **3 hr** | **Complete** |
| HK Poon | **White box test case – TB01 & TB02 and partial scrum report** | **2 hr** | **Complete** |
| Kim Ming Chau | **White box test case – TB04, TB05, TB06** | **2 hr** | **Complete** |
| Yuhong Fan | **White box test case – TB08, TB09, TB10** | **2 hr** | **Complete** |
| Elvin Karikari | **Scrum report and reflection** | **2 hr** | **Complete** |
|  |  |  |  |
|  |  |  |  |

**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 |
| Gordon Tan | Integration tests (well-designed, written and documented) |
| Kim Ming Chau | Acceptance tests (well-designed, written and documented) |
| HK Poon | Test Execution (performed, results recorded, issues created) |
| Elvin Karikari | Debugging (Bugs fixed, documented, Jira updated) |
| Yuhong Fan | Function-test matrix updated |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Completed whitebox and blackbox testing | **Divided the work evenly amongst us all, to make our efforts efficient** |
| Timeline to complete this Milestone | **Completing the timeline was imperative to ensure the project was completed effectively and without any headaches** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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

**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 |
| Miscommunication | **Due to an unprecedented workload, Elvin was not as communicative for this milestone. There was a delay in completing his portion of the test cases and we had to pivot. He stated that this won’t happen again and make greater efforts into be more interactive.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

1. **After you run your blackbox and whitebox tests you are asked to record the results in both the original test document as well as in Jira. Explain why it is a good idea to record the results in both places.**

It Is essential to record results in both the original test document and Jira because of these six reasons:

1. Keeping Everything Organized:
   1. By documenting the results in the original test document, we have a neat and tidy record of all the testing outcomes in one central place. It's super handy for tracking progress, identifying trends, and referring to previous results whenever needed. Plus, it helps maintain traceability between test cases, requirements, and the final outcomes.
2. Getting the Big Picture:
   1. The original test document usually gives a solid overview of the testing efforts, including objectives and methodologies. Adding the results here lets anyone checking it out get a quick grasp of how the testing went down and what the overall outcomes were.
3. Syncing with Test Planning:
   1. When we include the results in the original test document, we could see if we met the initial test planning's objectives. It's a way to evaluate how successful our testing was and if there's anything we need to tweak or improve for future rounds.
4. Making Issue Tracking Easy:
   1. Jira is a popular tool for issue tracking and bug management. When we record the test results in Jira, it creates a direct link between the outcomes and any issues or defects we spotted. This smooths out the process of assigning and tracking problems, making it simple for the dev and testing teams to work together to fix things.
5. Team Collaboration and Transparency:
   1. Jira's like a virtual hangout where everyone can check out the test results in real-time. These fosters open communication among team members, so we all stay in the loop about testing progress and any snags we hit. And it's not just us—stakeholders can easily peek into Jira to see where things stand.
6. Useful Reports and Metrics:
   1. Recording results in Jira lets us generate all sorts of cool test reports and metrics. We get valuable insights into testing progress, defect trends, and the overall app quality. These reports help with decision-making, so project managers and stakeholders can keep a finger on the project's pulse and make informed choices based on solid data.

1. **Why did we wait until the fourth milestone to write the whitebox tests?**

Here some reasons why we decided to do that:

1. Prioritizing Functionality: In the initial milestones, we focused on getting the truck route management system's core functionalities up and running. Blackbox testing was a great way to check if everything works from the user's perspective. Once we had the essential features stable and functional, it made sense to move on to whitebox testing.

2. Code Stability: Whitebox testing involves digging into the code's nitty-gritty, so it's important to make sure the codebase is pretty stable and ready for that level of scrutiny. By waiting until the fourth milestone, we ensured that the code had reached a certain level of maturity.

3. Incremental Testing: We followed an incremental testing approach, where testing happens in phases alongside development. In the beginning, blackbox testing helps validate high-level functionality. As we progress and gain more confidence in the system, whitebox testing comes into play for a deeper analysis.

4. Resource and Time Constraints: Whitebox testing can be more time-consuming and requires specific skills for code analysis. We managed our resources and time wisely by prioritizing other testing activities and development tasks first.

5. Test Prioritization: Our test plan likely focused on blackbox testing initially to address critical functionality and user experience issues early on. Once we tackled those, we dug into whitebox testing to optimize the internal workings of the system.

6. Learning from Feedback: Early testing cycles often involve gathering user feedback and identifying major issues through blackbox testing. Fixing those issues before diving into whitebox testing would create a more stable codebase and make the whitebox testing more effective.

1. **For a given function did you produce more blackbox or whitebox tests? Explain why your answer (more blackbox or more whitebox) happens for most functions.**

For most functions, we have prioritized creating more blackbox tests over whitebox tests. The reason behind this choice is that blackbox testing allows us to focus on validating the external behavior of the function without delving into its internal implementation details. It helps us ensure that the function behaves correctly based on its specified requirements and produces the expected outputs for different input scenarios.

Blackbox testing is particularly valuable when dealing with complex functions or those with multiple interactions within the system. It enables us to verify the function's intended behavior and its seamless integration with the overall application.

On the other hand, whitebox testing involves scrutinizing the function's logic. While essential for achieving higher code coverage and verifying the correctness of the implementation, whitebox testing can be more resource-intensive and time-consuming. It requires in-depth knowledge of the codebase and careful consideration of various code paths.

Given the emphasis on functionality validation and the efficiency of blackbox testing, we have chosen this approach for the majority of functions. Nevertheless, we acknowledge the significance of whitebox testing in uncovering potential edge cases and code-related issues that might not be apparent through external testing alone.

To strike a comprehensive testing balance, we aim for a well-rounded approach by incorporating both blackbox and whitebox tests. This ensures thorough validation and enhances the overall software quality.

1. **Explain the purpose of the automation hook for GIT and explain how it can improve the quality of the software in the project.**

The automation hook for GIT is like a super cool feature that automates stuff in our development and testing workflow. It's basically a customizable script that runs automatically when we do specific things in GIT, like making code changes or pushing our code to the repository.

It improves the software quality of the project in several ways.

1. Continuous Integration (CI):

With the automation hook, we can integrate our code changes super quickly into the shared repository. It triggers automatic builds and tests every time we push new code. This way, we catch any integration issues early on and squash those bugs before they become a big headache.

2. Automated Testing:

We can set up the hook to run automated tests whenever we commit code. That means we get instant feedback on our changes, catch any issues, and fix them on the spot. It's like having our own personal testing assistant!

3. Code Reviews and Static Analysis:

The automation hook helps us stick to good coding practices. It can enforce code reviews before merging code, which is super helpful for maintaining code quality and spotting potential vulnerabilities through static code analysis.

4. Deployment Automation:

We can automate deployments to test environments or staging servers whenever we merge specific branches. This makes testing easier and ensures we're working in an environment that's similar to the real deal.

5. Error Prevention and Early Detection:

The hook can even stop us from making silly mistakes. We can set it up to run checks before we commit, like code linting and unit test coverage thresholds. It's like having our own coding guardian angel.

6. Version Control Enforcement:

With the automation hook, we can enforce version control best practices, like proper commit messages and branch names. It keeps things neat and tidy in our codebase, making it easier to collaborate with others.