# 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.

**GROUP: 1**

**Members Present**:

|  |  |
| --- | --- |
| 1. Hoi Kit Cheung | 4. Sau Ching Yuki Wong |
| 2. Gyeongrok oh | 5. Yonghun Won |
| 3. Lap Chi Wong | 6. Pui Wai Tse |

## Milestone 6 Tasks

This is the final milestone where you will run the acceptance tests and fix any remaining bugs found. In addition, you will produce a testing report which lists all the tests conducted, the results and whether the bugs were fixed, and the final test passed. You will also review the test matrix to ensure every test has been performed and passed. You can change the colour of the test in the matrix to show it was run and passed. At the end, all tests in the matrix should have been passed.

The final test report can be tabular like this:

|  |  |  |  |
| --- | --- | --- | --- |
| Function/acceptance/requirement | Test Run | Bugs Fixed | Passed |
| Distance | TF001 | Did not handle negative coordinates | 🗹 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Deliverables Due at end of Lab:**

* SCRUM Report and reflections

**Deliverables Due at 23:59 4 Days after Lab:**

* Execute acceptance tests(results in Jira), and debug.
* Updated function-test matrix stored to the repository.
* Final Testing report listing tests conducted, bugs fixed and the final test passed.

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
|  | Teamwork | 10% |
|  | SCRUM Report & reflections | 15% |
| Group | Updated test matrix | 20% |
|  | Final test report | 20% |
|  | Test Execution (performed, results recorded, issues created) | 10% |
|  | 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 & reflections | 30% |

**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** |
| Hoi Kit Cheung | * Prepare test report template and example * Prepare test reports   + Unit Test – dispatchTruck   + Acceptance Test * Manpower statistics * Prepare reflection Questions | None |
| Gyeongrok oh | * Prepare test reports   + Unit Test – convertStrToPoint   + Integration Test 2 * Prepare reflection Questions | None |
| Lap Chi Wong | * Double-check function-test matrix * Perform bug fixing upon any bugs discovered in acceptance test * Prepare reflection Questions * Prepare scrum report | None |
| Sau Ching Yuki Wong | * Prepare test reports   + Unit Test – getShipmentFromInput   + Integration Test 1 * Prepare reflection Questions | None |
| Yonghun Won | * Prepare test reports   + Unit Test – convertPointToStr   + Integration Test 3 * Prepare reflection Questions | None |
| Pui Wai Tse | * Prepare test reports   + Unit Test – listShortestDiversion   + Integration Test 1 * Prepare reflection Questions | 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 |
| Workload distribution | To distribute the tasks for milestone 6 to team members so that each member has a fair share of workload | The workload distribution is agreed in the meeting |
| Discussed project timeline | Defined project timeline and deadlines. Adjusted the timeline based on everyone's needs and ensured that all team members agreed on the deadlines. | Project timeline finalized with adjustments based on team consensus. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Testing Strategy Enhancement | It was decided to focus on improving the testing strategy to ensure all implemented functions meet the required quality standards. This decision was prompted by the identification of a failed test and the recognition that all tests should pass seamlessly to ensure reliable software. |
| Increased Documentation and Collaboration | The team agreed to enhance collaboration and communication among team members by incorporating more comments in Jira tickets. This decision arose from the feedback that greater transparency and insight into individual contributions are essential for tracking progress effectively. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Gyeongrok, Oh | Reflections, code review, function review | 30 mins | done |
| Hoi Kit, Cheung | Reflections, code review, function review | 30 mins | done |
| Lap Chi, Wong | Reflections, code review, function review | 30 mins | done |
| Pui Wai, Tse | Reflections, code review, function review | 30 mins | done |
| Sau Ching Yuki, Wong | Reflections, code review, function review | 30 mins | done |
| Yonghun Won | Reflections, code review, function review | 30 mins | 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 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Defined Project Scope and Objectives | By establishing a well-defined project scope and objectives, the team can now focus their efforts on delivering the essential features within the specified boundaries. This clarity will prevent scope creep and ensure a more efficient use of resources. |
| Assigned Roles and Responsibilities | The meeting resulted in the assignment of roles and responsibilities to each team member based on their expertise and interests. Clear role assignments enhance accountability and productivity. Each team member now knows their specific tasks and contributions, reducing overlaps. |
| Planned Task Prioritization and Scheduling | The team collectively prioritized project tasks and developed a timeline for their completion. Task prioritization and scheduling allow the team to allocate resources effectively, focusing on critical components first and meeting project milestones in a timely manner. |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Workload distribution | Requirements of Test Case and code are fully understood by all team members to create reasonable sub-tasks |
| Project Timeline | Realistic and achievable milestones set |
| Responsibilities | Clear understanding of individual tasks and roles |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Nil |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

1. Although we wrote a report on the testing that shows which tests were run and passed or failed, we also updated the function test matrix. What are the advantages of updating the function test matrix in addition to writing the test report?  
     
   The function test matrix is like a simplified map that shows the status of all the functions we tested. It's like a traffic light – green for functions that worked well, red for those that had problems, and yellow for those in progress. This visual summary is very convenient because it gives everyone a quick idea of which parts of our project are doing well and which ones might need some extra attention.  
     
   The matrix doesn't just show red or green; it also tells us which testing has gone wrong. It's like having a magnifying glass that lets us zoom in and see which part of the program is malfunctioning. If a function is marked as having issues (red), the matrix helps us to quickly identify that function. This detail is essential because it allows us to find exactly which functions need more work or fixing. Instead of searching through a big pile of code, we can go directly to the functions that need our help. This way, we can put our energy where it's most needed, making our testing efforts efficient and effective.

As mentioned, the function test matrix is like a map that shows how the testing was done. Sometimes, the people who do the testing might change or new people join the team. This map helps everyone understand how things were tested before, so they can learn from it. It's helps newcomer understand the status of testing and easier way to pick up the ongoing task in the team.

On the other hand, the matrix also helps on better test planning, before we start work on the testing, matrix provide a picture of a test plan that we are going to do. It gives a chance to review and reassess the testing plan, whether the coverage on testing is adequate or not. And to achieve the more comprehensive result and effective testing process.

1. Teamwork on a project like this is vital to success. How well did your team work? If it worked well, what contributed to its success? If it did not work well, what contributed to the problems?

Our team works well in terms of role assignment based on the expertise of every individual member. Apart from each member performing the same task that would cause overlapping of duty and failure to safeguard the quality of the deliverable of the specific role, we take ownership on the specific roles by function including development team, QA team, project management.

Project manager defines the executable tasks, plans the working schedule and distributes workloads to suitable members.

QA team defends on the test results of each function in the system without being involved in uncontrolled code modification.

Development team develops and updates the whole code base according to the request of QA team to prevent fluctuation of testing standard to fit in what they coded.

A best practice to work on the same project is to isolate roles by job nature not only to allocate right people in right role, but to maintain the standard of the specific roles so that those standards will not be loosen just for compromising the deadline coming close.

1. In every milestone you were asked what worked and did not work along the way. Were you able to incorporate what you learned to improving your team’s performance on the next milestone? Did your team learn from its mistakes and improve? If so, why? If not, why?  
     
   While the team worked well in general, we still had areas that needed improvement as we went through the journey of software development. At times, team members were too focused on their own duties and may sometimes overlooked what other team members were working on or obstacles there were facing. This could hinder the performance of the group as the success of software development not only relies on individual brilliance, but also team collaboration.  
     
   The group reflected on this issue and learnt a lesson. We decided to improve communication by encouraging team members to speak out promptly when encountering problems, reporting promptly on the progress of the tasks they were assigned, and giving comments on each other’s work.  
     
   Eventually, the team improved as any problems faced by individual team members are promptly attended by his/her teammates, the progress of individual task were clear to the whole team throughout the project, and team members were able to adjust their work according to what others have produced or working on during the project. The team has improved as a whole after the reflection and implementation of new measures.
2. Did you end up testing the code to the point where you were convinced it worked correctly? Were there any tests that had not passed at the end? If so, what was the impact of this on the project?

Yes, we have ended up testing the code to the point that we were convinced that the program would work correctly.

Thorough testing is a fundamental pillar of the development process, essential to ensuring the code meets expectations and is devoid of critical issues. In this project, the team went the extra mile in testing, meticulously covering various test cases and real-world scenarios the code might encounter. We have tested all combinations of trucks’ status and package data, e.g. trucks having same minimum diversion distance but with different package loading, some trucks are not ready to load package. We have also tested edge cases, e.g. destination at the edge of the map, package that is near to the maximum loading capacities of the trucks, null inputs to different variables, etc.

By pushing the testing process to the point where our confidence in the code's correctness was unwavering, we achieved a robust validation of core functionality, performance, and security. This rigorous testing approach instilled a high certainty that the code could gracefully handle diverse situations, minimizing the potential for unexpected hiccups in the production environment.

While all tests are passed and the program can handle all the cases within the project scope, there may be a few cases that the program cannot handle, for example, having 4 trucks for delivery. Although this is already out of the project scope, we still tried to cover this scenario by introducing a global variable to store the number of trucks. This allows the program to be easily modified should the user requirements change in future and enables faster software development.

The absence of failed tests in the end indicated that our development approach had been thorough and meticulously effective, culminating in a solution that was not only reliable but also efficient. The fact that we didn't encounter any failed tests meant that we could adhere to the project timeline without the need for exhaustive debugging or last-minute frantic adjustments.