# **MILESTONE 2** -- 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**: \_\_\_\_\_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Anton Samofalov | 4.Basil Tariq |
| 2. Seyam Chowdhury | 5. |
| 3. Anuraj Singh Osan | 6. |

## Milestone 2 Tasks

Some of the software for the project has already been written for you and is available on Blackboard. You must use this in your project and every team should add it to the source code for their repository. Anything in the main function is simply for demonstration purposes and can be replaced. The software you are being given has not been tested and you will need to test it.

You need to study the problem and the code provided for you and then:

* Add any new data structures you will require This will require a thorough analysis of the problem and the existing software. This should be done by creating a new header file in the directory where the rest of the source code has been placed. You do not want to go back and modify it later if you can avoid it as it will slow the project.
* Create a test plan for the project by replacing the text in the supplied test plan template with your test plan.

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

* An analysis of the problem (no written artifacts produced).
* A series of data structures created as header files and stored in the repository.
* A test plan stored in the repository.
* Completed scrum report including reflection questions answered.

**Rubric**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Data structures (complete, correct, and well-designed, & project updated) | 20% |
| Test plan (complete, well-written) | 20% |
| Git usage (used properly with good structure) | 10% |
| Jira usage (creates issues, tracks progress) | 15% |
| Scrum report & reflections | 25% |
| Meets deadlines | 10% |

**Scrum Report**

**Summary of Tasks Completed or Delayed in the last week:**

Here you can list all 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** |
| Anton Samofalov | **Focused on coding part** | **No Task Delayed/Blocked** |
| Seyam Chowdhury | **Test planning** | **No Task Delayed/Blocked** |
| Anuraj Singh Osan | **Worked on completing the scrum report** | **No Task Delayed/Blocked** |
| Basil Tariq | **Worked on completing the scrum report** | **No Task Delayed/Blocked** |
|  |  |  |
|  |  |  |
|  |  |  |

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** | **No Task Delayed OR Blocked** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |
|  |  |
| **Delayed or Blocked Task** | **N/A** |
| **Reason for delay or block** | **N/A** |
| **Impact on Project** | **N/A** |
| **Solution or work-around** | **N/A** |

**Summary of Meeting:**

A summary of the main points discussed in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Task Distribution | **Tasks were assigned as follows:** |  |
|  | Anton Samofalov: worked on the coding parts | **All members understood their tasks and started working respectively** |
|  | Seyam Chowdhury: Test Planning |  |
|  | Anuraj Singh Osan & Basil Tariq: Worked on the scrum reports and reflections |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Set testing goals based on edge cases | Testing focuses on edge cases, such as obstacles blocking paths or routes overlapping, to ensure robustness of the navigation functions. |
| Task Distribution | To make it clear that everyone focuses on their specific assigned task |
| Documentation and creation of the scrum report | Having one member focus on documentation ensures that all team activities and decisions are well-recorded for reference and future improvements. |
|  |  |
|  |  |
|  |  |
|  |  |

**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 cannot be completed, the student should indicate why this was not possible.

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Anton Samofalov | **Described the coding parts** | **30 min** | **In progress** |
| Seyam Chowdhury | **Discussed the test plan** | **30 min** | **In progress** |
| Anuraj Singh Osan | **Srum report drafting and reflections** | **20 min** | **In progress** |
| Basil Tariq | **Srum report drafting and reflections** | **20 min** | **In progress** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Anton Samofalov | Work on the coding part |
| Seyam Chowdhury | Work on the test cases |
| Anuraj Singh Osan | Work on the scrum report and reflections |
| Basil Tariq | Work on the scrum report and reflections |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Task Distribution | **Everyone was clear about their roles during the project** |
| Test plan discussion | **Agreed on specific test cases and edge cases to cover, which ensures thorough testing of all functionalities and improves overall project reliability.** |
| Code Review & Analysis | **We reviewed the provided code together, which helped us spot potential challenges early and decide on any changes we might need.** |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**No problems encountered completing this section.

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 |
| Task Distribution | **Everyone agreed on their roles quickly, which helped set a clear direction for the project.** |
| Code Review & Analysis | **We worked together to understand the existing code, which helped us spot potential issues early and ensured we’re all on the same page.** |
| Test plan discussion | **We had a productive discussion about test cases, making sure everyone knows what to test and why.** |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Did NOT Go Well in This Meeting:**No problems encountered completing this section.

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 |
| Communication With each other | **Due to different time schedules our communication with each other did not go well as we planned** |
| Time management | **Some tasks took longer than expected** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflection Questions:**

Answer the following questions using your own words. Make sure that each answer comprises a minimum of 100 words.

1. In this milestone you have been asked to analyze a problem and design software (functions) to complete the solution without writing the software.
   1. Is this process more difficult than just writing the software to complete the project? If so, why is it more difficult? If not, why is it easier than just writing the software?

Answer: Thinking through a problem and planning out how to solve it can actually feel harder than just jumping in and coding right away. When we take time to analyze and design, we have to think carefully about what the software needs to do and how each part should work. This can feel like extra work because we’re not building anything yet; instead, we're creating a sort of blueprint.

If we skip straight to writing code, it might seem faster, but we often end up with messy code that doesn’t fully solve the problem or needs fixing later. Planning first helps us avoid big mistakes and keeps things organized, so even though it takes more effort upfront, it usually saves time in the end and leads to better, more reliable software.

* 1. Describe two advantages of developing software in this manner rather than just moving on to writing the functions without writing specifications first.  
       
     Answer: Planning out software before coding has two big benefits.

First, it helps us get a clear idea of what we’re building. When we take time to plan, we understand exactly what each part of the software needs to do, which helps us avoid mistakes or confusion later. It’s like planning a road trip before you start driving—when you know the route, you’re less likely to get lost or make wrong turns.

Second, planning saves time in the long run. When we already know how everything should connect, we’re less likely to face big issues that force us to go back and rewrite code. This way, we can build the software more smoothly, with fewer surprises, making testing and improving it easier as we go.

1. Why is it a good idea to create a test plan? Describe at least 3 advantages of test plans.

Answer: Creating a test plan is really helpful because it ensures our software works as expected. Here are three key reasons why a test plan is so useful:

Knowing What to Check: With a test plan, we have a clear list of things to test, so we don’t miss any important features or functions. It’s like having a checklist that keeps us on track.

Catching Problems Early: Testing each part with a plan helps us spot issues early, making it easier and quicker to fix them before they become big problems.

Saving Time and Effort: A test plan keeps testing organized and efficient. Instead of guessing what to test or repeating steps, we know exactly what to do, making the whole process faster and smoother as the software gets more complex.

1. Describe the process you used to analyze and understand the existing software.

Answer: To understand the software, I took a simple, step-by-step approach:

Reading the Code Carefully: I went through the code line by line to get familiar with how everything worked. This helped me understand the structure and see what each part was supposed to do.

Spotting Key Parts: I identified the main functions and core components that are central to the software’s function. Understanding these important parts helped me see how they work together to make everything run smoothly.

Following the Flow: I paid attention to how the logic flowed from one part to another, looking for any patterns or repeated steps. This gave me a big-picture view of how the software operates.

Marking Confusing Spots: If I found areas that were unclear, I made a note to come back to them. This way, I could focus on understanding everything without getting stuck.