# **MILESTONE 2** -- SFT221 Scrum Report and Reflection

All students are expected to attend the SCRUM meetings and to participate. Failure to do so will result in greatly reduced grades.

**GROUP**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Seliya Marahatta | 4. Sameer Bal Tamang |
| 2. Janice Kang | 5. Daniel Kim |
| 3. Potpourri Fajilagot | 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) | 25% |
| Test plan (complete, well-written) | 25% |
| Git usage (used properly with good structure) | 10% |
| Jira usage (creates issues, tracks progress) | 20% |
| Scrum report & reflections | 20% |
| **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 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** |
| Seliya Marahatta | Function creation, upload each on github |  |
| Janice Kang | Function creation, upload each on github | Uploading on git |
| Potpourri Fajilagot | Function creation, upload each on github | Uploading on git |
| Sameer Bal Tamang | Function creation, upload each on github |  |
| Daniel Kim | Function creation, upload each on github |  |
| **all** | Reflection questions and review to modify |  |
|  |  |  |

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** | Uploading the modified files which is in my local computer to github |
| **Reason for delay or block** | updated files were the same file that I tried to upload, it occurs file confliction |
| **Impact on Project** | Not much |
| **Solution or work-around** | Figured out the reason for it and updated local repo and github all together by ignoring the confliction |
|  |  |
| **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 discussed in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| How do we create data structures | We first understood the task to do for this milestone by reviewing the provided codes and asked to professor with a sample example what we made | We made a outline to do it. |
| Tasks Assignment | We decided that everyone would generate function definitions and upload them to GitHub. | Proper distribution of workload and Synchronous creation of functions that avoided function repetition. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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

**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? |
| All members | Analysis of the Problem and Assignment of tasks in Jira | 30 min | complete |
| Seliya Marahatta |  |  |  |
| Janice Kang | Creating 2 function in header file and upload in the GitHub | 20mins | **complete** |
| Potpourri Fajilagot | Creating 2 functions in header file and committing in the GitHub | 20mins | **complete** |
| Sameer Bal Tamang |  |  |  |
| Daniel Kim |  |  |  |
| all | Scrum report, reflection questions, Jira | 2 hours | **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 |
| all | Function selection what we made last week. (at least 6) |
| Seliya Marahatta | 4 test data for one function |
| Janice Kang | 4 test data for one function |
| Potpourri Fajilagot | 4 test data for one function |
| Sameer Bal Tamang | 4 test data for one function |
| Daniel Kim | 4 test data for two funtions |
| Janice Kang | New header file creation |
| Seliya Marahatta | Create and add c++ testing project |
| Potpourri Fajilagot | 1 black box test case implementation |
| all | Scrum report and reflection questions |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Divided tasks again | Proper distribution of workload to all members |
| Analyzing a problem and designing software functions are more difficult than just writing the software | Answer the reflection question 1-a |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Things That Went Well in This Meeting:**

Here you can highlight things that worked well. This indicates that the way you worked on these items is working and should be continued.

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Creating functions | We found out the outline for this together and then we separated them for each group members |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

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

**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?

Analyzing a problem and designing software functions without actually writing the software are both challenging more than just directly writing the software. This is because designing software without writing code means we must think about how the software should work without typing out the instructions. This could be hard because we must imagine all the different situations the software might encounter and come up with the best ways to handle them. We also need to understand the problem the software is trying to solve, including any limits or rules it needs to follow. Talking about our ideas as a group, especially if we are not experts, can be tough too. It is important to be able to explain our ideas clearly so other members understand what we are trying to do.

* 1. Describe two advantages of developing software in this manner rather than just moving on to writing the functions without writing specifications first.

First, we can improve the clarity and understanding of the software. When we spend time thinking about the problem and planning how the software should work before we start writing code, it helps us understand exactly what needs to be done. By making clear plans and describing what each part of the software should do, everyone knows what to expect. This reduces confusion and mistakes while we’re working on it, making the whole process smoother and faster. It’s like having a map before a trip. This would help us know where we’re going and how to get there without getting lost.   
  
Second, we can have enhanced flexibility and adaptability. When we think about how the software should work before writing the code, it’s like planning a recipe before cooking. We can come up with different ways to design the software, test the code blocks out in our minds, and decide which one is the best. We can also ask others for their thoughts and make some adjustments based on their feedback. In this way, we finally start coding, we are more prepared and can handle any surprises that could come up. This is a helpful practice to make the process smoother and adapt if something unexpected happens.

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

Creating a test plan is an essential component of software development, with several benefits that enhance the project’s quality and success. A test plan, first and foremost, makes sure that all tests are thorough by carefully outlining the goals and parameters of the testing process. Through the specification of requirements, testing strategy, and expected output, developers can methodically investigate every aspect of the software, including specific features to the general behavior. By taking a comprehensive approach, the risk of undetected faults is reduced, leading to a software product that is more dependable.

Second, a test strategy also makes it easier to maintain consistency and repeatability during the testing process. A test plan makes sure that tests are carried out consistently by organizing testing practices among team members and across various steps of development. This allows tests to be reliably duplicated in addition to simplifying the process of comparing results and identifying inconsistencies. We may confirm and verify software changes consistently over time with documented testing techniques described in the plan. That will ensure the product continues its quality as it changes.

Lastly, a test plan increases efficiency by offering an organized framework for setting up the testing to prioritize. We can manage resources and concentrate on software areas that have a risk or problems by specifying test objectives, resources, and schedules in detail. Discovered problems and resolutions are made possible by this preventative strategy in advance. This also helps to avoid delays in the development lifecycle. A test plan helps make sure that testing matches the project’s goals and schedule by deciding which tests are most important based on deadlines and possible issues.

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

We successfully understand the existing software through proper collaboration among team members regarding how we will create our test plan. We have understood that the test plan is crucial because it provides a clear outline of the testing goals and scope, which helped our team define our roles and how we will efficiently work together towards achieving desired outcomes. With the help of the following processes:

First, to analyze and understand the existing software, we reviewed the documentation provided, such as source codes, project-student pdf files, and relevant information about the software from the milestone document.

Second, we took time to interact with the software ourselves, exploring its features and functionalities firsthand to get a practical understanding of how it works.

Next, we examined the codebase of the software, to understand the underlying structure and logic of the software. This involved reviewing the code files, identifying key modules and functions, and understanding how they interact with each other.

Overall, by combining document analysis, hands-on exploration, and code review. We were able to analyze and understand the existing software comprehensively to inform our approach to designing software functions for the solution.