# **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**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_E\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Shovana Shrestha | 4. Tu Yin Hnit Aung |
| 2. Kusum Acharya | 5. Krish Sanjaybhai Patel |
| 3. Kemal Batu Turgut | 6. Roy Bryan D. Franck |

## 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** |
| Tu Yin Hnit Aung | Divided the task for this week  Table (task, meeting, decisions, task attempted, this class task) of the Scrum Report |  |
| Roy | Create data structures |  |
| Krish Patel | Create Test Plan |  |
| Kemal Batu | Create Test Plan |  |
| Shovana Shrestha | Table (outcomes, things went well, did not go well, reflect question 1) |  |
| Kusum Acharya | Reflect question 2 and 3 |  |
|  |  |  |

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 discussed in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Dividing Roles | **The tasks had to be divided into equal parts that each person can finish before the due date** | **The task was divided equally** |
| Importance of each task | **We discussed which part of each task were important and have to be done carefully and thoughtfully** | **We managed to figure out how the test plan is to be done and the code** |
| Deadlines and Due Dates | **The deadline was enforced and to be followed by all the group members** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Prioritize Code | The code is the most important part of the project so it should be done well and concisely. |
| Meetings should be prioritized | It is important to attend meetings and be on time so that everyone can manage their time well without having to reschedule anything |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Tu Yin Hnit Aung | * Divided the task for this week   Table (task, meeting, decisions, task attempted, this class task) of the Scrum Report | **1 hour** |  |
| Roy | Create data structures | **1 hour** |  |
| Krish Patel | Create Test Plan | **1 hour** |  |
| Kemal Batu | Create Test Plan | **1 hour** |  |
| Shovana Shrestha | Table (outcomes, things went well, did not go well, reflect question 1) | **1 hour** |  |
| Kusum Acharya | Reflect question 2 and 3 | **1 hour** |  |
|  |  |  |  |

**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 |
| Tu Yin Hnit Aung | * Divided the task for this week   Table (task, meeting, decisions, task attempted, this class task) of the Scrum Report |
| Roy | Create data structures |
| Krish Patel | Create Test Plan |
| Kemal Batu | Create Test Plan |
| Shovana Shrestha | Table (outcomes, things went well, did not go well, reflect question 1) |
| Kusum Acharya | Reflect question 2 and 3 |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Divided the group's work among the members. | **Ensures that tasks are assigned efficiently, fosters responsibility, and helps team members coordinate their work more easily.** |
| Understanding of Project Objectives- Jira, test plans and data structures. | **Helps create thorough test strategies, enhance data structures for algorithm execution, and integrate Jira more effectively.** |
| Set the agenda for the upcoming week's tasks. | **Setting priorities for our projects helps us focus on the most important things and make the most efficient use of our time and resources to complete the work efficiently and on schedule.** |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Division of the tasks/issues | **Clear communication and agreement on project objectives among team members.** |
| Organized timetable | **The meeting's agenda was clearly laid out, which helped targeted conversation and effective time management.** |
| Decision-making process | **Open discussion and reaching an agreement on important project decisions.** |
| Positive feedback | **Opportunities for improvement and feedback.** |
| Effective problem-solving | **Team members collaborated to address challenges and find solutions.** |
|  |  |
|  |  |

**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 before the meeting | **Prior to the meeting there was very little communication between the team members due to which we were not able to hold the meeting on the designated time. To resolve this, all the team members must maintain effective communication with each other for successful completion of the project.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

**ANS:** When working on a group project, figuring out a problem and setting up software functionality before actually writing the code can be more difficult than implementing the code straight away. This method requires everyone on the team to collaborate, agree on each other’s ideas, and communicate effectively to ensure that everyone understands the plan and its effects. Planning without coding allows us to think in general and solve problems, but it requires everyone to agree on the ideas and project direction. Despite the difficulty, this method of working allows everyone to fully understand the problem, identify drawbacks, and ultimately come up with an effective solution. That's why it's ideal for collaborative projects where everyone collaborates to finish the project efficiently.

* 1. Describe two advantages of developing software in this manner rather than just moving on to writing the functions without writing specifications first.  
       
     There are various benefits to developing software by first designing functions and understanding the problem rather than just writing the code, compared to writing the code without any specifications. Any two advantages of doing that are as follows:

1. Clarity and Understanding: Members of the team working on the project can better understand the needs and goals of the project by dedicating time to problem analysis and software design. Everyone involved gains a clear understanding of the project's objectives and scope through the thorough description and explanation of the logic and interactions between different components of the program. This shared perspective minimizes the possibility of miscommunications between the team members throughout the development process, resulting in more productive teamwork and, eventually, a better final product.

2. Resource optimization: The development team working on the project can allocate resources more effectively if software functionalities are designed before code is written. Teams can more efficiently manage resources throughout the development lifecycle by focusing up front to brainstorm the solution and predict potential issues. By addressing problems early on, when they are less expensive and time-consuming to resolve, this method reduces the possibility of expensive rework or redesign later in the process. Therefore, the group can maximize its time, energy, and assets, leading to a more proficient and economical software delivery process.

1. Why is it a good idea to create a test plan? Describe at least 3 advantages of test plans.  
     
   A software testing plan is the most essential document for software testing and quality assurance process. It provides an overview of how the software testing should be conducted.

It is a good idea to create a test plan because it provides a **structural approach**   
by defining the objectives, parameters, resources, and timeline for testing. This methodology makes it possible to guarantee that every aspect of the software, from performance and functionality to security and usability is thoroughly tested. A test plan helps the team to stay organized by specifying testing goals and procedures.

Test plans help to identify the risks and vulnerabilities in the software. This is called **risk identification and mitigation**. Before the product is made available to the users the team can identify potential issues by defining test scenarios, test cases and expected outcomes. This allows the developers to address the issue and reduce the risk of serious malfunctions, failures, or security breaches in the production environment.

Test plans work as comprehensive documentation of testing requirements, procedures, and outcomes of the test. They provide a reference point for everyone involved in the project guaranteeing that everyone is aware of the testing objectives and expectations. Test plans facilitate communication across various teams and departments by specifying roles, responsibilities, milestones, and deliverable. Through **documentation and communication,** teams can track the progress of testing activities and identify areas of improvement for the next iteration.

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

To analyze and understand the existing software the following steps were done based on the provided code:

1.**Understanding the code structure:** First it is important to see the structure of the code, identify the components and their roles in the code. We have three module files, so we need to identify the roles of each file in the program.

**2.Function Overview:** Going through each function in the code and understanding the functionality. Review function call in main.c file and its purpose.

**3.** **Data Structures:** Reviewed the data structures and how they are used for representing the maps, points, and routes.

**4. Function Calls:** To understand how the functions are called it is important to trace their flow and how the control flows through the code. It was also important to identify the sequence of function calls and analyze them.

**5.** **Understanding how functions are created and manipulated for mapping:** In the code there are functions like for example getGreenRoute, getYellowRoute for creating routes. Similarly, there are functions like ‘shortestPath’ to calculate the shortest path. Moreover, there are functions for map representation. These are important functions that needed to be analyzed thoroughly for understanding the software.

**6.Comments and Documentation:** The comments were the most important part for analyzing the code. The comments provided insights to how the functions were being implemented and the purpose of functions and data structures.

**7.Creating Test plan:** Although the test plan does not necessarily help but it provides us with a foundation for code testing scenarios and test cases, which gives us insights into the software. Like for example, requirements understanding, functional understanding and error handling.