# **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**: \_\_\_\_\_\_\_4\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Heet Hiteshbhai Patel | 4. Sumit Kumar |
| 2. Ankush Ankush | 5. Priyangan Chandrathayan |
| 3. Kashyap Hasmukbhai Patel | 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 at End of Lab**

* Completed SCRUM report & reflections

**Deliverables Due within 48 hours of lab**

* 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.

**Rubric**

|  |  |  |
| --- | --- | --- |
| Individual | Group Participation | 75% |
| Teamwork | 10% |
| SCRUM Report | 15% |
| Group | Data structures (complete, correct and well-designed) | 20% |
| Test Plan (complete, well-written) | 20% |
| Git Usage (used properly with good structure) | 10% |
| Jira Usage (creates issues, tracks progress) | 10% |
| Meets Deadlines | 15% |
| SCRUM Report and Reflections | 25% |

**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** |
| 1. Ankush Ankush | Test Planning | No |
| 2. Sumit Kumar | Problem Analysis and assisting other members | No |
| 3. Heet Patel | Scrum Report | No |
| 4. Kashyap Patel | Reflection | No |
| 5. Priyangan Chandrathayan | Coding | No |
|  |  |  |
|  |  |  |

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 |
| Understanding MS 2 deliverables | We discussed what this milestone requires | This helped us to assign work all the members |
| Distributing tasks to each team member | We discussed our strengths and weakness to take on tasks | Successfully assigned tasks to each team member |
| Deadline | Decided the deadline for each team member to submit their work | Decided the deadline for submission |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Code should be completed as soon as possible | Coding being the most challenging part of this milestone we decided to get it done as fast as possible. |
| Group effort for test planning | Test planning is quiet long and tedious to do it alone, so the team decided to do it together |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Kashyap Patel | Reflections | 45 min | Yes |
| Heet Patel | Scrum report | 35 min | Yes |
| Priyangan Chandrathayan | Coding the header file | 20 min | No |
| Ankush and Sumit | Test Planning | 10 min | No |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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 |
| Heet Patel | Completing Scrum report and track progress |
| Ankush | Test Planner and Tester |
| Kashyap Patel | Test Planner and Tester |
| Sumit Kumar | Coder |
| Priyangan Chandrathayan | Coder |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| MS 2 Successfully Planned | This allowed us to distribute tasks among team members |
| Tasks allotted to team members | Avoids any confusion during the project |
| Reflection and Scrum Report completed | Reduces the burden on all team members |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Dividing work | Good understanind and commucation among the team |
| Completed Reflection and Scrum Report | Good team work and because of contribution of all team 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 |
| Test Planning | Lack of knowledge. This could be easily fixed by studing more about test planning. |
| Coding | Lack of time. Next meeting should be planned in such a way that each team member has ample of time. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

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?

-> Creating a blueprint for a building before constructing it is analogous to designing software before programming it. It is not inherently more difficult than writing the software, but it is an important and slightly different stage.  
During the design phase, you plan and sketch out how the software will function, what it will perform, and how its many components will fit together. This step helps to avoid mistakes and makes coding easier.

Writing the software is like building the actual structure based on the blueprint. It can be more straightforward because you already know what to do, thanks to the design.

In easy words we can say that so, design makes the writing part easier and reduces the chances of costly errors. It's an important step, not necessarily more difficult, but essential for a successful project.

* 1. Writing software is analogous to constructing a physical structure based on a plan. It may be simpler because you already know what to do because of the design.

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

-> Developing software with proper specifications and design offers several advantages -over immediately moving on to writing functions without planning.

First and foremost, it Reduced Errors and Rework. When we begin by developing specs and designing software, you are effectively building a development plan. This preliminary planning aids in the identification of potential challenges, ambiguities, or conflicts in the requirements or design. By addressing these issues early on, you can avoid code errors and misconceptions. This decreases the chances of having to redo or remodel substantial chunks of the software later in the development process, which may be time-consuming and expensive. Another advantage is Improved Collaboration and Communication Creating software with specifications and design encourages greater communication and collaboration among members of the development team as well as stakeholders. Specifications and design documents provide a consistent reference point for all parties involved in understanding the project's goals, architecture, and intended functioning. This clarity ensures that everyone is on the same page and can contribute to the project productively.

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 a crucial aspect of the software development process, as it provides several advantages that contribute to the overall quality and reliability of the software. A test plan outlines a structured and systematic approach to testing the software. It defines the scope of testing, including what features or components will be tested, the test environments to be used, and the testing methodologies and tools. This structured approach helps ensure that all aspects of the software are thoroughly tested, reducing the risk of overlooking critical issues.

Early in the development process, test plans can assist detect and assess potential risks and concerns. You can reveal gaps in requirements, potential areas of vulnerability, and functionality that may not fit with user expectations by establishing test cases, test objectives, and acceptance criteria. This proactive strategy allows the development team to identify and minimize risks before they become more expensive and difficult to handle later in the project's life cycle.

A well-defined test strategy allows for efficient resource allocation, including time, persons, and equipment. It assists project managers and stakeholders in understanding the testing timetable, required staff, and hardware and software needs. This effective resource allocation ensures that testing progresses smoothly and within budget restrictions, avoiding overruns and unnecessary delays.

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

* To examine and comprehend existing software:
* Collect and review documentation, including design documents, specs, and source code.
* Interview stakeholders to learn about the history, purpose, and issues of the software.
* Examine source code to determine architecture, dependencies, and logic.
* The program should be tested for functionality, performance, and security.
* Examine the schema and data in the database.
* Examine the system and its surroundings, including any third-party dependencies.
* Investigate data flow, integration, and security issues.
* Document your findings and share them with others for comment and collaboration.