# **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**: \_\_\_\_\_\_\_\_\_\_B\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Members Present**:

|  |  |
| --- | --- |
| 1. Jeet Patel | 4. Arthav Patel |
| 2. Yash Shah | 5. Samarth Shah |
| 3. Sahil khatri | 6. Jeetkumar Patel |

## 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** |
| **Jeet Patel** | **Analyse suggestions for test plan and modify as per agreement of other members** |  |
| **Other members** | **Suggest test plan and data structure for the project and how many header files to be made.** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

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** | **Files not uploading on github via tortoisegit** |
| **Reason for delay or block** | **Student vpn** |
| **Impact on Project** | **Delay to upload** |
| **Solution or work-around** | **Always connect to student vpn** |
|  |  |
| **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 |
| Analyse the code logic | **During the meeting, discuss the code working and think of possible corrections that can be made and what structures to define.** | **Different opinions on the topic concluding a final decision on structures.** |
| Jira working | **Questions regarding how Jira works and how work will be assigned and to be completed.** | **All understood how Jira works .** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**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 |
| Header files | As per the plan , 2 header files are needed for now, one for shipment and the other for trucks. These will include the structs declaration. Definition will be made later as it is not to be done in this milestone. |
| Tests execution to be discussed in milestone 3. | As per the test plan , each member will be assigned testing tasks in each milestone on Jira. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Jeet Patel | Check all testing parts of individuals and applt them into the code. |
| Jeetkumar and Arthav | Blackbox testing. |
| Samarth , Sahil,Yash | Document the tests for blackbox. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Test plan | **Different viewpoints from members helped come up with better plan that all agreed upon.** |
| Jira | **Everybody now fully understand how Jira works as Arthav had some daubs regarding which led to a detailed conversation on Jira and we did research and all understood perfectly.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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?

The process of problem analysis and software design entails effectively managing complexity, encompassing considerations such as performance, scalability, maintainability, and usability. This task can prove more intricate than simply writing code to address a specific task.

Engaging in software analysis and design necessitates abstract thinking and problem-solving abilities, which pose a greater challenge compared to adhering to predefined structures and syntax during coding. Software analysis and design frequently entail collaboration and the effective communication of ideas with others, introducing an additional layer of complexity to the process.

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

**Minimized Rework:** By thoroughly analyzing the problem and designing specifications beforehand, developers can identify potential issues, requirements, and constraints early in the process. This proactive approach helps in anticipating challenges and addressing them in the initial design phase, reducing the need for extensive rework or refactoring later. By having a clear roadmap laid out through specifications, developers can build the software with greater confidence, minimizing the likelihood of significant changes or revisions during the coding phase.

**Increased Client Satisfaction:** Engaging in problem analysis and specification design before coding allows for better alignment with clients’ expectations and requirements. Through collaborative discussions and feedback sessions during the specification phase, developers can ensure that the proposed solution meets the needs and priorities of end-users and clients. By incorporating their input early on, developers can build trust, foster transparency, and ultimately deliver a software solution that better satisfies the clients’ needs and objectives. This approach can lead to higher client satisfaction and increased likelihood of project success.

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

Below are three key advantages of having a test plan, which is essential for ensuring the quality and reliability of a software project:

**Enhanced Test Coverage:** A test plan ensures thorough coverage by delineating various scenarios, edge cases, and functional requirements for testing. It offers a structured method for validating the software's functionality and behavior across diverse situations. By adhering to the test plan, developers can detect potential issues and bugs early on, mitigating the likelihood of critical failures in the production environment.

**Streamlined Testing Efficiency:** A well-defined test plan optimizes testing efforts by enabling testers to prioritize test cases based on their criticality and potential impact on the software's performance. By concentrating on crucial areas and essential functionalities, testers can efficiently allocate resources and conduct targeted testing. This approach minimizes time and effort by avoiding redundant tests, ensuring that testing efforts focus on pivotal aspects.

**Ensured Consistency and Reproducibility:** A test plan furnishes clear guidelines and instructions for test execution, ensuring consistency and reproducibility. Testers can adhere to predefined procedures and test data, facilitating consistent testing across different team members or testing cycles. Consistent testing aids in the easier identification of issues and yields more reliable results. Additionally, in the event of a discovered bug, a well-documented test plan allows developers to replicate the issue, comprehend its root cause, and resolve it effectively.

To recap ,a properly designed test plan serves as the foundation for organized and structured testing, leading to comprehensive coverage, efficient efforts, and consistent results. It contributes to higher quality, enhanced reliability, and expedited issue resolution, thereby bolstering customer satisfaction and project success.

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

To analyze and understand existing software, we start with an initial review of the code to grasp its overall structure. Identify key functionalities and features, tracing the program flow to understand its sequence and dependencies. Analyze the data structures and algorithms employed, documenting assumptions and limitations found in the code or accompanying documentation. Debug and test the software to observe its behavior under different scenarios. Consult additional resources like documentation and user manuals for further insights. This systematic approach provides a comprehensive understanding of the software's architecture, functionality, and constraints, laying the groundwork for any necessary modifications or improvements.