# **Milestone 3 Scrum Report**

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

**GROUP**: **\_\_\_\_\_1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Members Present**:

|  |  |
| --- | --- |
| 1. Mahmadsahil Mahmadrafik Shah | 4. Ying Wang |
| 2. Pouya Rad | 5. Dil Humyra Sultana Borna |
| 3. Ryaan Farrukh | 6. |

## Milestone 3 Tasks

In this milestone you will create issues to design the functions, design all of the functions you need to complete the project and store the specifications in the repository. As soon as the specifications start to be produced, you can start to design the blackbox tests (what they test, how to perform them and test data). Once tests are written, they can be implemented and added to the repository and any team members not otherwise busy can start to implement the functions. You will also build a function-test matrix that shows the blackbox tests for each function. This will be maintained through the testing cycle as new tests are added.

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

* A set of AT LEAST 4 function specifications added to a new header file and stored in the repository.
* A set of blackbox tests as test documents (in an Excel file) with test data for the functions you created. At least 4 sets of test data are required for each function. You must have test cases for at least 6 functions (including all your custom function). Stored in the repository.
* **Create and add a C++ testing project to your solution.**
* Start writing blackbox test code (for the functions above) and store in repository (at least 1 is required for this milestone).
* Start implementing the functions and store them in repository (optional).
* A requirements traceability matrix added to the repository and shows the mapping between the requirements and test cases.
* Updated Jira project to show activities and progress.
* Completed scrum report including reflection questions answered.

**Rubric:**

|  |  |  |
| --- | --- | --- |
| **Individual** | Group participation (includes GitHub commits and Jira usage) | 80% |
| Teamwork | 20% |
| **Group** | Function specifications (documented, complete, well-written, added to the project) | 10% |
| Blackbox test cases document (well-written, complete, good test data) | 10% |
| Blackbox test code (in the C++ project) well-designed and documented | 10% |
| Functions implementation (coded in the C project & well documented) | 15% |
| Requirements traceability matrix (complete and added to GitHub) | 15% |
| Git usage (used properly with good structure) | 10% |
| Jira usage (creates issues, tracks progress) | 15% |
| Scrum report & reflections | 15% |
| **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 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** |
| Mahmadsahil Mahmadrafik Shah | Blackbox test code (in the C++ project) well-designed and documented, Git usage (used properly with good structure), worked on function too | **n/a** |
| Pouya Rad | Function specifications (documented, complete, well-written, added to the project), Blackbox test cases document (well-written, complete, good test data) | **n/a** |
| Ryaan Farrukh | Requirements traceability matrix (complete and added to GitHub), Jira usage (creates issues, tracks progress), worked on function and Blackbox testing too | **n/a** |
| Ying Wang | Functions implementation (coded in the C project & well documented), Scrum report & reflections, worked on function and black box testing too | **n/a** |
| Dil Humyra Sultana Borna | Blackbox test cases document (well-written, good test data), worked on function and Blackbox testing too, Scrum report & reflections, Functions implementation | **n/a** |
|  |  |  |
|  |  |  |

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** | **n/a** |
| **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 discusses in the meeting and the outcomes of the discussions.

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Distribution of the work | According to the MS requirements, the tasks were divided among the group members. Assigning creating functions, test cases, and SCRUM report fairly between all members. Everyone was given the choice to select their tasks and work on them as discussed to make it easy for their schedules | Members were given the flexibility to select tasks based on their preferences, allowing them to maintain a sense of a fair amount of work. |
| Test plan | The four functions were divided among the group members, and they were asked to do the test cases according to their function. Discussing this ensured how functions and their test cases would work so they can be done accurately. | Four members worked on different functions and test cases according to their functions. |
| Reflect Questions | To answer the reflection questions effectively, a comprehensive examination and research effort into both the project and relevant theoretical subjects are essential. Collaboratively, members engaged in discussions, sharing their perspectives and offering constructive suggestions to enhance the overall understanding. | 2 members were responsible for the 3 reflection questions addressing their needs through research and understanding, drawing insights from the discussions held during the meeting. |
| Filling up the Scrum Report | This milestone presents a more extensive set of components compared to the previous one, emphasizing the crucial need for accurate completion of the SCRUM report. In the collaborative discussion, members strategized on the distribution of tables among the team. It was decided that each member would independently fill the tables related to their individual work. | Members are responsible for individually completing all tables associated with their specific tasks. For the remaining tables, a collaborative approach was adopted. This ensures that the SCRUM report is accurately and efficiently compiled. |
| Function & Blackbox testing | As this included coding, every member was responsible for this part, so that we can help each other and completed the task on time, this also increase our knowledge of coding and testing by group working. | It was done by everyone’s participation and whenever we faced any problem, solved it immediately without wasting any time. |
| Deadlines | To prevent last-minute panic and rushed work, the team made decisions regarding individual deadlines, emphasizing the importance of thoughtful planning and timely completion of tasks. | The team collectively agreed that each member is required to finalize their individual tasks at least 6 hours before the deadline, specifically by 6 PM on Tuesday. This decision was made to ensure ample time for review, adjustments, and the overall avoidance of last-minute rush. |
| JIRA Collaboration | Recognizing the integral role of each member's utilization of JIRA in determining their grade, the team engaged in discussions to establish measures that would ensure active team involvement in this aspect. | It is mandatory for each team member to provide comments under their assigned tasks and consistently update the status of their tasks. This practice ensures clear communication, accountability, and transparency within the team. |

**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 |
| The deadline for each individual task was set to 9 PM Tuesday | This made sure everyone got enough time to discuss and edit the group submission and gave enough time to review and consider the suggestions of each group member |
| SCRUM report must be checked by individuals after the completion of each task | This would help create an accurate log of all the activities carried out by each member. it is important because it makes sure every member is informed about the progress made. |
| Collaborating and communicating on JIRA is important and comments must be made under every completed or in-progress task. | To utilize the project management software JIRA, it is important to use it as a means of communication by using it to comment, add descriptions and tag other members. |
|  |  |
|  |  |
|  |  |
|  |  |

**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? |
| Mahmadsahil Mahmadrafik Shah | Worked on functions and made test cases for it. Checked the individual SCRUM report. | 7 | completed |
| Pouya Rad | Worked on functions and made test cases for it. Checked the individual SCRUM report. | 9 | completed |
| Ryaan Farrukh | Worked on functions and made test cases for it. Checked the individual SCRUM report. | 8 | completed |
| Ying Wang | Worked on functions and made test cases for it. Checked the individual SCRUM report. | 10 | completed |
| Dil Humyra Sultana Borna | Worked on functions and made test cases for it. Checked the individual SCRUM report. | 9 | completed |
|  |  |  |  |
|  |  |  |  |

**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 |
| Mahmadsahil Mahmadrafik Shah | Implement the functions and Whitebox testing documents with test data for these functions. Checking SCRUM report. |
| Pouya Rad | Implement the functions and Whitebox testing Checking SCRUM report. (Mainly the coding part) |
| Ryaan Farrukh | Implement the functions and Whitebox testing Checking SCRUM report. . (Mainly the coding part). |
| Ying Wang | Implement the functions and Whitebox testing documents with test data for these functions. Checking SCRUM report. |
| Dil Humyra Sultana Borna | Implement the functions and Whitebox testing documents with test data for these functions. Checking SCRUM report. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

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

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| New functions | Understanding how the code works, and coming up with functions that benefit the code. |
| Testing cases of functions | Making sure we have an organized test case plan (black box) for each function so that it benefits the code. |
| Tracking purpose | Making sure we also fill out a traceability matrix sheet to keep track of functions purpose for the program. |
| Keeping in contact | Using JIRA to keep in touch, as well as committing to Github to collaborate. |
|  |  |
|  |  |
|  |  |

**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 |
| Participation | Everyone participated in the meeting and did their part. |
| Collaboration | We collaborated to understand the whole milestone and discussed all the requirements. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**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 |
| Assigning Tasks | **Before we completed each part separately but, in this milestone, we have worked together for each task so that everyone can participate in the coding and testing part. As this is the first time, we all worked together for each task, there was a bit confusion in the beginning but then we figured it out by discussing it with each other.** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

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

1. What is the difference between blackbox tests cases and blackbox test code? Explain how we use assertion in Visual Studio to execute tests.

**1 Answer:**

**BlackBox:** Blackbox testing is a method of software testing where the internal structure, design, or implementation of the item being tested is not known to the tester. There are two main aspects to consider in Blackbox testing: test cases and test code.

1. **Test cases:** Blackbox test cases are specific inputs, execution conditions, and expected outcomes developed to verify that a software application behaves as expected. These test cases are designed without any knowledge of the internal workings of the software. They focus on the functional requirements and are used to check the overall behavior of the software, ensuring it meets specified criteria. For example, a test case might check if a login function correctly authenticates a user with valid credentials and denies access with invalid ones.
2. **Test code:** On the other hand, Blackbox test code refers to the actual scripts or programs written to automate the execution of these test cases. This code interacts with the software's external interfaces to provide inputs and verify outputs, but it does not require access to the software's internal source code. The test code uses assertions to validate the expected outcomes of the test cases.

**Using Assertion in VS:** In Visual Studio, assertions are used within the test code to automatically verify that the software behaves as expected. An assertion is a statement that checks if a particular condition is true. If the condition is true, the test passes; if it is false, the test fails, and an error is reported. Visual Studio provides a framework for writing and running these automated tests. For example, in MSTest, an assertion might look like this: Assert.AreEqual(expected, actual); where expected is the expected result, and actual is the result produced by the software. If the actual value does not match the expected value, the assertion fails, indicating a problem in the software.

Using assertions in Visual Studio helps streamline the testing process, making it more efficient and reliable. It allows testers to quickly identify and address issues, ensuring the software meets its functional requirements without needing to manually verify each test case.

1. How can a traceability matrix help in the testing process?

**2 Answer:**

**Traceability Matrix:** A traceability matrix is a powerful tool in the testing process that helps ensure comprehensive test coverage and maintains the integrity of the project requirements. It is a document that maps and traces user requirements with the test cases designed to verify those requirements. This matrix ensures that all requirements are covered by test cases and helps identify any gaps in the testing process.

**Traceability Matrix help in the testing process:**

By using a traceability matrix, testers can easily track the progress of testing efforts, verify that all specified requirements have been tested, and ensure that any changes in requirements are addressed in the corresponding test cases. This enhances the overall quality of the software, as it ensures that all functionalities are tested and validated against the initial requirements. Furthermore, it facilitates communication between project stakeholders, as it provides a clear and concise overview of the testing status and any potential issues. Ultimately, a traceability matrix improves the efficiency and effectiveness of the testing process, leading to a more reliable and robust software product.

1. Write down two of the function prototypes you submitted. Why did do you need each one of them and how will each one help you achieve the project needs?  
   int getBestRoute(struct Route\* routes[MAX\_ROUTE], struct Shipment shipment, int size);

**Purpose and Need:**

* **Purpose**: This function calculates the best route among the given valid routes to reach the shipment's destination.
* **Need**: In a delivery system, it's crucial to find the shortest and most efficient route to ensure timely deliveries and optimize resource usage. This function helps identify which route to take by comparing the distances of multiple valid routes.

**How it Helps**:

* **Efficiency**: By determining the shortest route, it minimizes the travel time and fuel consumption.
* **Resource Optimization**: Ensures that the delivery system uses the least amount of resources to complete the shipment.
* **Decision Making**: Provides a clear decision on which route to select when multiple options are available.

int validateDestination(int destination1, char destination2);

**Purpose and Need:**

* **Purpose**: This function checks if the provided destination coordinates are within the valid range of the delivery area.
* **Need**: Validating input is a fundamental step to ensure the system processes only correct and expected data. Incorrect or out-of-range destinations could lead to errors or inefficiencies in the delivery process.

**How it Helps**:

* **Data Integrity**: Ensures that the destination coordinates entered by the user are valid and within the allowed limits, preventing erroneous data from being processed.
* **Error Prevention**: Helps in avoiding errors during the delivery route calculation by filtering out invalid destinations at an early stage.
* **User Feedback**: Provides immediate feedback to the user if an invalid destination is entered, improving the user experience and guiding them to input correct data.