## SCRUM Report and Reflections – 3

**Members Present**:

|  |  |
| --- | --- |
| 1. Prabhjot Singh Longia | 4. Divya Devendrasinh Rana |
| 2. Harsh Pahurkar | 5. |
| 3. Mukul Sharma | 6. |

## Milestone 3 Tasks

* Create issues for function design
* Design all required functions
* Store function specifications in the repository
* Begin designing blackbox tests
* Write and implement blackbox tests
* Add implemented tests to the repository
* Assign function implementation to available team members
* Build and maintain a function-test matrix

**SCRUM Report:**

**Summary of Tasks Completed or Delayed in the last week:**

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| **Prabhjot Singh** | Formulated function specifications and declarations and incorporated them into the repository. |  |
| **Mukul Sharma** | Generated black-box test cases and crafted a test document covering both the creation of new test cases and the utilization of existing ones. |  |
| **Harsh Pahurkar** | Collected and organized the necessary test data, and then constructed a comprehensive functions test matrix. Finally, added both the test data and the test matrix to the repository for future reference and use. |  |
| **Divya Rana** | Wrapped up the scrum report and answered the reflection questions. |  |
|  |  |  |
|  |  |  |
|  |  |  |

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 |
| Formulate function specifications and declarations. | The team carefully analyzed the project requirements and identified the necessary functions that need to be implemented. Later we collaborated to ensure clarity and completeness in defining each function's purpose, inputs, outputs, and expected behavior. | As a result of the discussion, the team successfully created comprehensive function specifications and declarations. Each function's purpose and requirements were clearly defined, providing a detailed roadmap for the development phase. |
| Develop black box test cases and a comprehensive test document for both new and existing test cases. | We carefully analyzed the requirements for both new and existing test cases and crafted test scenarios to evaluate the software's external functionality. | As a result of the discussion, the team successfully developed a set of black box test cases covering both new and existing test scenarios. The comprehensive test document captures each test case's purpose, inputs, expected outputs, and testing criteria. |
| Organize test data and construct a functions test matrix. | We carefully analyzed the testing requirements and identified the necessary test data to effectively evaluate the software's functionality. | As a result of the discussion, the team successfully organized the test data and constructed a functions test matrix. |
| Ensure the readiness of Scrum reports and reflection questions. | We reviewed the progress made during the sprint and gathered relevant information to compile the Scrum reports. | Reflection questions were answered, allowing team to have a thorough understanding of the milestone. |
|  |  |  |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

|  |  |
| --- | --- |
| Decision | Rationale |
| Creating scrum report and answering reflect questions, the team will double check it once it is done. | The team has collectively reviewed and analyzed the Reflect questions to gain a deeper understanding of the content covered in the milestone. Everybody contributes and adds to the SCRUM reports after it has been made. |
| Function specifications will be created by Prabhjot. | Prabhjot is assigned to create the function specifications, ensuring accurate and clear definitions for the functions, which is crucial for a solid development foundation. |
| Black box test cases will be created by Mukul, then double checked by the team later. | The black box test cases are implemented into the code to evaluate the software's external functionality without knowledge of its internal code |
| Test data will be prepared by Harsh and will be double checked by the team later. | The team will gather and organize the necessary test data efficiently and then double-check the test data afterward ensures its accuracy and completeness, reducing the likelihood of errors and ensuring a more reliable testing phase. |
|  |  |
|  |  |
|  |  |

**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? |
| Prabhjot | Created function specifications | 2 hours | YES |
| Harsh | Created test data | 1.5 hours | YES |
| Mukul | Make black box test cases | 2 hours | YES |
| Divya | Worked on scrum report and reflection | 1.5 hours | YES |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**SCRUM Tasks Selected for Next Week**:

|  |  |
| --- | --- |
| Group Member | Task Description |
| Prabhjot | Creating and finalizing the hook for automation of tests |
| Harsh | Preparing test plan |
| Mukul | Preparing white box test cases and implementing them |
| Divya | Scrum report and reflections |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Creating function specifications was discussed | provides clarity, reduces risks, improves collaboration, and streamlines development and testing processes. |
| Scrum report and reflections have been discussed | It enables continuous improvement and informed decision-making, positively impacting the project's progress and team dynamics. |
| Black box test cases have been discussed | ensures software quality, requirement validation, bug detection, and customer satisfaction. |
| Test data was discussed | It enhances testing efficiency and accuracy, leading to more reliable results and a better-quality software product. |
|  |  |
|  |  |
|  |  |

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

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Function specifications have been created | Enthusiasm and a genuine interest in understanding the problem statement are key factors in successfully creating it. |
| Test data is ready | By scrutinizing the test data in detail, the team aimed to identify and address any potential loopholes or gaps, leaving no room for uncertainties during testing. |
| Scrum report and reflections have been done | The success of Scrum reports and reflections is made possible due to the team's clear communication with each other. |
| Black box test cases have been ready | The team's collective effort ensured that all critical functionalities are thoroughly tested, enhancing the overall quality of the testing process. |
|  |  |
|  |  |
|  |  |

**Things That Did NOT go Well in This Meeting:**

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Problem and How to do Better |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Reflections**:

1. In this milestone, we write the blackbox tests but not the whitebox tests. Explain why we can write the blackbox tests but not the whitebox tests.   
     
   🡪 We are required to do Blackbox tests for this particular milestone but not Whitebox tests. This strategy was chosen because of the nature of the testing techniques:

**Blackbox testing:** Blackbox testing treats the software as a "black box" and concentrates on its external functionality without considering its internal implementation.

Blackbox tests are created by testers using the requirements, specifications, and anticipated behavior of the software.

Without requiring knowledge of the core code, the tests' inputs and outputs are analyzed to see if the software functions as expected.  
**why we can write Blackbox tests:**

Blackbox tests only consider the software's user interface and intended features.

We can create black box tests to assess the behavior of the offered software and validate that it complies with the criteria because we have access to the software's specs and design.

Since Blackbox tests are independent of the software's implementation, they may be created without having to delve into the core code.

**Whitebox testing** entails comprehending the internal logic and code structure of the software.

To look at the internal paths, branches, and circumstances followed during code execution, testers create Whitebox tests.

To construct tests based on the internal structure of the code, this sort of testing needs access to the source code.

Why we can't write Whitebox tests at this milestone:

The offered software has not been tested, and neither its internal implementation nor the caliber of the code is known to us.

Blackbox tests allow us to properly test and validate the functionality of the software, but writing Whitebox tests necessitates a better comprehension of the code's subtleties.

Since Whitebox tests depend on the code, attempting to develop them before confirming the software's fundamental operation through black box testing may result in tests that are based on false assumptions or produce incorrect results.

Finally, given that Blackbox tests are the only ones that depend on the software's external behavior and specifications at this point, we may confidently design them. However, we defer building Whitebox tests until after we've used Blackbox tests to test the functionality of the product and have a better grasp of how it operates inside.

1. Explain why we need the function-test matrix and why it is important in a large project.

* In a large project, the function-test matrix is crucial because it offers a systematic mapping of functions and the accompanying test cases. It guarantees thorough test coverage, making it simpler to keep track of which features have been tested and which want more focus. The matrix supports effective resource allocation by indicating the scope and complexity of testing efforts for each function. It makes bug discovery easier because testers may easily identify the pertinent test cases for a broken function. By acting as a central point of reference, the matrix encourages team members' participation and communication. It offers efficient regression testing to make sure that updates or new code don't adversely affect working functions. The matrix verifies if each function serves its intended purpose by connecting test cases to requirements. Overall, the function-test matrix contributes to the project's continued success, dependability, and high level of quality.

1. Other life cycle models left team members idle while waiting for parts of the project to be completed. Describe how an agile model, like the one we are using, avoids this problem and keeps the whole team busy all the time. Does this make managing the project simpler or more complex and why?

* Iterative development is followed by shorter, more targeted sprints in an agile paradigm like Scrum, keeping team members from being idle. Cross-functional teams make ensuring that work is done even when some tasks are delayed. Daily stand-up meetings provide ongoing collaboration and early issue support. Adjustments can be made with adaptive planning to meet shifting needs. The agile product backlog guarantees that the team will always have work to do. This facilitates early issue resolution and incremental progress, which simplifies project management. However, due to the requirement for active participation from every team member, sustaining constant discussion, and balancing priorities within the backlog, managing an agile project can be more difficult. Despite these difficulties, agile is a recommended method for effectively managing projects due to its productivity and adaptability advantages.