# SCRUM Report and Reflections - 5

**Members Present**:

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

## Milestone 5 Tasks

* Write integration tests
* Implement and execute integration tests
* Determine if unit tests can be used for automated testing
* Manually check outputs when necessary
* Include test execution details in test documentation
* Update function-test matrix with brief descriptions of integration tests
* Write acceptance tests based on user requirements
* Perform manual testing for acceptance tests
* Show correct output for various inputs in acceptance tests

**SCRUM Report**

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

|  |  |  |
| --- | --- | --- |
| **Member** | **Tasks Completed** | **Tasks Delayed/Blocked** |
| **Prabhjot** | **Designed, wrote and documented integration tests.** |  |
| **Mukul** | **Worked primarily on debugging. All update and any prior unresolved bugs** |  |
| **Harsh** | **Implemented and executed tests, recorded results and observations, updated test files accordingly** |  |
| **Divya** | **Designed, wrote and documented acceptance tests.**  **Worked on the scrum report** |  |
|  |  |  |
|  |  |  |
|  |  |  |

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:**

|  |  |  |
| --- | --- | --- |
| Topic | Discussion Summary | Outcome |
| Planned, created, and documented integration tests. | Integration testing was the main topic of discussion, with a focus on careful planning and comprehensive documentation for seamless component interactions and enhanced software quality. | The result was a precise plan for effective integration testing that ensured seamless interactions between components and improved the overall performance of the product. |
| mostly focused on debugging. Updates and any lingering issues from the past | We emphasized the value of systematic debugging methods and team knowledge sharing. | As a result of our discussion, we developed a more sophisticated debugging strategy that makes use of modern tools, shares knowledge gained from previous mistakes, and works together to tackle problems that haven't been solved. |
| implemented and carried out tests, noted findings and observations, and updated test files as necessary. | We discussed the conduct and application of tests, highlighting the value of precise testing, careful observation, and detailed findings documentation. | We decided on a clear procedure for carrying out tests, recording findings, and updating test files as a result of our discussion. |
| created, written, and recorded acceptance tests.  Finished the scrum report. | Our conversation focused on finishing necessary tasks, such as creating, developing, and documenting acceptance tests. We also talked about the Scrum report's successful completion, emphasizing the value of accurate documentation and efficient communication. | Following our talk, we completed the Scrum report, encouraging open communication within the team and giving useful insights into our progress and future objectives, and successfully executed acceptance tests, validating the software's alignment with requirements. |
|  |  |  |
|  |  |  |
|  |  |  |

**Summary of Decisions Made:**

|  |  |
| --- | --- |
| Decision | Rationale |
| Integration testing will be created, planned, and documented by prabhjot. | By encouraging collaboration, knowledge sharing, and ownership, this strategy raises software quality through teamwork. |
| Mukul will be working on the debugging part of the code | This strategy maximizes resource use, encourages knowledge sharing, and ensures a thorough analysis of the code, ultimately improving the stability and performance of the product. |
| Harsh will be implementing and executing the tests, record results and observations and update test files accordingly. | This strategy fosters thorough observation, ensures current test files, and streamlines testing procedures, ultimately enhancing software quality through community effort. |
| Divya will design, write and document the acceptance tests and work on the scrum report. | This method improves software quality and efficient project management by utilizing a variety of expertise, encouraging collaborative documentation, and adding to a thorough testing and reporting procedure. |
|  |  |
|  |  |
|  |  |

**Tasks Attempted During Meeting:**

|  |  |  |  |
| --- | --- | --- | --- |
| Member | Task Attempted | Time Spent | Complete? |
| Mukul | Worked primarily on debugging. All update and any prior unresolved bugs | **2.5 hours** | **YES** |
| Harsh | Implemented and executed tests, recorded results and observations, updated test files accordingly | **3 hours** | **YES** |
| Prabhjot | Planned, created, and documented integration tests. | **3 hours** | **YES** |
| Divya | created, wrote, and recorded acceptance tests.  Finished the scrum report. | **2.5 hours** | **YES** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**SCRUM Tasks Selected for Next Week**:

|  |  |
| --- | --- |
| Group Member | Task Description |
| Prabhjot | Final testing report listing tests conducted, bugs fixed and the final test passed |
| Mukul | Update function-test matrix stored to the repository. |
| Harsh | Prepare the scrum report and reflections |
| Divya | Execute acceptance tests and debug |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**Major Outcomes of Meeting:**

|  |  |
| --- | --- |
| Outcome | Impact on Project |
| Acceptance test cases have been discussed | Clear understanding of system behavior is ensured by discussing acceptance and integration test cases, which also helps to eliminate ambiguities and direct development in the direction of requirement fulfilment. |
| Integration test cases have been discussed | Test cases that have been thoroughly explored increase software reliability by spotting problems early, simplifying efficient bug fixing, and eventually enhancing the project's overall quality. |
| Scrum report and reflections have been discussed | It enables continuous improvement and informed decision-making, positively impacting the project's progress and team dynamics. |
| Implementation of the function and debugging has been done by all the group members | The impact on the project likely resulted in a more reliable and efficient software development process with improved code quality and faster testing cycles. |
|  |  |
|  |  |
|  |  |

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

|  |  |
| --- | --- |
| Topic/Work Item | Reason for Success |
| Implementation of the functions have been done | efficient and accurate conversion of requirements into workable code that achieves the desired functionality objectives. |
| Acceptance tests and integration tests have been created | The careful planning, thorough examination of test scenarios, and cooperative efforts that ensure accurate validation of software functionality and fluid interactions between components are responsible for the effective creation of acceptance and integration tests. |
| Scrum report and reflections have been created | The success of Scrum reports and reflections is made possible due to the team's clear communication with each other. |
| Debugging of the code has been done | The methodical issue discovery, thorough investigation, and efficient teamwork that resulted in better code quality, expanded functionality, and a more dependable software system are all factors that contributed to the code's successful debugging. |
|  |  |
|  |  |
|  |  |

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

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

**Reflections**:

1. At this point, you are using the GIT hook to automate testing. Have you found that any of the tests failed and prevented you from pushing your code to the repository? If so, how did you handle the situation?  
     
   🡪 Right now, I've used the GIT hook to automate testing, which has helped me maintain the quality of my work. I came across instances during this procedure where certain tests did fail, serving as a check against submitting flawed code to the repository. I responded to these circumstances by carefully going over the test results that failed, figuring out the causes of the errors, and implementing the appropriate code modifications. The need of proactive testing and ongoing improvement in our development workflow was further underscored by this iterative approach, which not only made sure the code adhered to the necessary standards.
2. Explain why we are automating the testing process and what the advantages of this automation are.  
     
   🡪 Looking back on the development of our project, the choice to use automated testing has produced noteworthy outcomes. As a result of automated tests' ability to speed up iterations and facilitate problem resolution, our development process has been noticeably accelerated. The consistent execution of test cases has increased trust in the dependability of our program and reduced the variability of findings. Furthermore, thorough testing has uncovered complex edge cases and latent flaws that manual testing may have overlooked. The risk of introducing regressions is reduced by ensuring that code changes are properly examined before deployment by integrating automated tests into our CI/CD workflow. Even though the original setup required a lot of labor, the continuous benefits—improved quality, less physical labor, and easier development—far surpass the initial outlay.
3. Did you find the integration and acceptance tests more difficult to write than the black box and white box tests? If so, why were they harder to write? Did you write more white box and black box tests or more integration and acceptance tests?  
     
   🡪 When I think back on the various testing steps, I realize that writing integration and acceptance tests was a little more difficult than writing black box and white box tests. The requirement to mimic interactions between several components and guarantee end-to-end functionality led to this complexity. Integration and acceptability tests took more time because they needed to carefully analyze all possible scenarios and dependencies.

I put more time and effort into building acceptance and integration tests. These tests were crucial for ensuring that the software satisfied user needs and that various components interacted smoothly. The integration and acceptance tests ultimately played a crucial part in verifying the overall performance and user experience of the product, even though the black box and white box tests were crucial for ensuring that each function and piece of code worked as intended. As a result, prioritizing these tests contributed to a more comprehensive and robust testing approach.

1. Explain why it is necessary to write integration and acceptance tests given that all of the code has already passed black box and white box tests.

* Integration and acceptance tests are crucial to guarantee that the entire system works flawlessly and satisfies user expectations, even while black box and white box tests validate individual components and code pathways. Integration tests look at how well different parts interact with one another and look for any potential problems. On the other side, acceptance tests verify that the software is in line with user requirements and intended use cases, ensuring a positive user experience. These tests must be written because they evaluate the system as a whole and address any problems that might not be apparent until after components have been merged and realistic scenarios have been simulated. This thorough testing strategy reduces the possibility of systemic failures, improves software dependability, and eventually produces a polished and user-friendly result.