Sprint 3 Report

Lost & Found
Team Rocket
23 July 2018

Actions To Stop Doing

- Source Control: Stop merging with the master branch unless you have tested that your feature still works, and no old features have been broken.
- Team Meeting: Stop working on individual code when the team allocates time to work together (ex: visually inspecting code)

Actions To Start Doing

- Source Control: Start merging the master branch with your individual branch asd
- Source Control: Start making backup copies of branches so that it is easier to recover older working versions of code if something breaks.
- Sprint planning: Tasks need to be included for testing and refactoring code, because these actions are necessary and take up significant time.

Actions To Keep Doing

- Individual work: Team members should continue being as productive when working by themselves, outside of the group meetings. This keeps the overall group productivity high.
- Individual work: Team members should continue taking the initiatives to work on tasks that aren't explicitly assigned to them, including fixing code, editing design code, and working on stories from the product backlog. This also keeps the overall group productivity high, and allows the team to progress even better than expected on the project.
- Meetings: Team members should continue attending all team meetings, being (mostly) on time to team meetings, and being flexible with meeting times and locations. This upholds the mutual respect amongst team members (by not wasting each other's time) and again keeps the group productivity high.
- Communication: Team members should continue the high levels of communication and participation through Slack when discussing team logistics (e.g. discussing what still needs to be done, scheduling meeting times). This keeps the group productivity high.
- Assisting others: Team members should continue helping other members who need assistance with their task and teaching other members how to use various technologies. This helps team members learn the necessary technologies more quickly, thereby keeping group productivity high.
- Communication: Team members should keep communicating about which features they are working on. This helps to avoid overlap and redundancy in the work being done.
- Communication: Team members should keep using the appropriate channels to discuss the various aspects of the project (e.g. backend, frontend). This keeps communication lines clear and allows members to find specific information that has been discussed more easily.

- Code commenting: Team members should continue commenting their code to clarify obscure sections or to help other team members identify the purpose of a code block.
- Refactoring: The team did a good job refactoring code into separate components. Code was separated to follow the single responsibility principle, which not only adheres to the SOLID design principles, but made the code more readable as well.
- Code styling: Team members should keep using the ESLint extension so that every team member's code follows the same style guidelines.
- Source Control: We did a great job communicating when each team member would merge their GitHub branch with the master. There were no overlaps, and any errors that resulted from merging could be traced back to the last merge.
- Unit testing: Keep writing unit tests so that we can validate that certain components/functions are working as intended.

Work Completed

- User Story 2: Task 1, 2, 3
- User Story 4: Task 1
- User Story 5: Task 1, 2
- User Story 6: Task 1, 2
- User Story 7: Task 1, 2
- User Story 8: Task 1, 2
- User Story 9: Task 1, 2
- User Story 10: Task 1
- User Story 11: Task 1
- User Story 12: Task 1, 2, 3
- User Story 13: Task 1
- User Story 15: Task 1, 2

Work Not Completed

- User Story 1: Task 1, 2 was attempted, but not finished; some team members were unable to figure out how to write a unit test, and on the whole not enough unit tests were written.
- User Story 3: Task 1, 2 was put into the backlog after implementation difficulties and a team discussion about effort vs. priority.
- User Story 14: Task 1 was started but not completed at the end of this sprint.

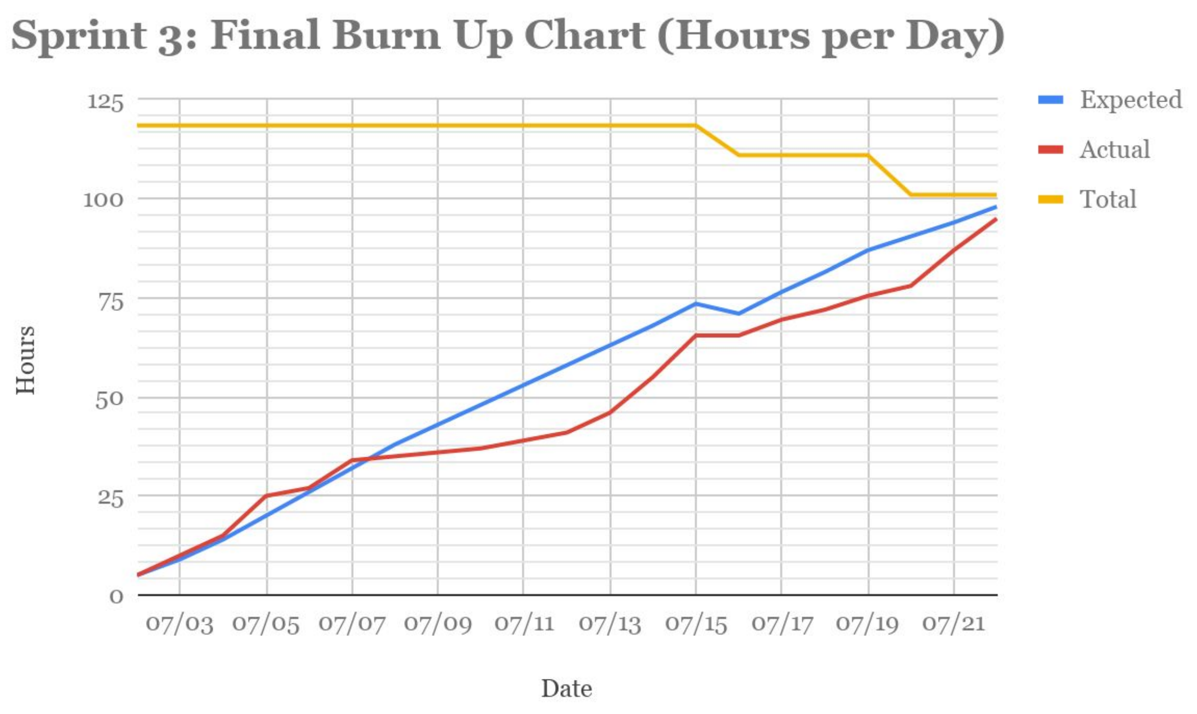
Work Completion Rate

Sprint 3	
Number of User Stories Completed	12 stories
Estimated Ideal Work Hours Completed	32 hours
Working Days	7 days
Average User Stories Per Day	12/7 = 1.71 user stories per day
Average Ideal Work Hours Per Day	32/7 = 4.57 hours per day

Sprint 1 + Sprint 2 + Sprint 3	
Total Number of User Stories Completed	28 stories
Total Estimated Ideal Work Hours Completed	98.5 hours
Total Working Days	21 days
Average User Stories Per Day	28/21 = 1.33 user stories per day
Average Ideal Work Hours Per Day	98.5/21 = 4.69 hours per day

Burnup Chart

The burn up chart following the completion of Sprint 3:


Sprint 3: Final Burn Up Chart (Hours per Day)
Hours
07/03
07/05
07/07
07/09
07/11
07/13
07/15
07/17
07/19
07/21
Date
Expected
Actual
Total