# Scrum Events

If you’re unfamiliar with Scrum, first check out the ‘Scrum Basics’ document!

Sprints will be two weeks in length, starting on Tuesday morning and ending during class on Mondays. Event times are as follows:

* Daily Scrum: Monday/Wednesday/Friday: Status on Discord by 2pm
* Sprint Planning: Every second Monday, after class
* Backlog Grooming: Every Wednesday
* Sprint Retrospective: Every second Monday, after class
* Sprint Review: 2pm every second Monday, in class

# Development Workflow

All development on the product should be related to a backlog item (bug or user story). This ensures we stay on track towards completing sprints, and that all work being completed is logged and documented.

During the sprint, we work items in the sprint backlog. Choose a sprint backlog item, assign it to yourself and mark it as active. While you’re working on that backlog item, update the statuses on related sub-tasks as you see fit. Feel free to add tasks as needed.

After work on a backlog item is complete and all code has been merged with develop, the backlog item should be marked as resolved. Resolved backlog items are ready for testing by someone else on the team. See ‘Testing Workflow’ for details on the testing process.

For each work item, here’s a general workflow with respect to Git and Azure DevOps:

1. Change the status of the work item to “Active”
2. Branch off develop, naming the new branch *feature/product-backlog-id* (Example: feature/32, or bug/35)
3. Make all commits for your work item to this branch
4. When you’re done with the work item, push the branch to DevOps.
5. Create new pull request in DevOps to merge your branch into develop
6. In order for a pull request to be completed, at least one other team member needs to approve it. This peer review will help prevent mistakes or bugs from reaching the develop branch.
7. After approval, the pull request can be completed. Your branch will then be merged with develop. Don’t forget to checkout the develop branch and pull new commits after completing the pull request!
8. Change status of work item to “Resolved”
9. Another team member will test the work item to ensure it passes acceptance:
   1. For User Stories: testing that all parts of acceptance criteria are implemented
   2. For Bugs: testing that the repro steps no longer produce the undesired behavior
10. After tests pass, mark the bug/user story as ‘Closed’

# Git

## Configuration

To properly configure origins after cloning the git repo (regardless of which origin you used), run the following git commands:

git remote remove origin

git remote add origin https://<your\_devops\_username>@dev.azure.com/96ethanh/Balto/\_git/WebTeamClient

git remote add csus http://pc2.ecs.csus.edu/git/WebTeamClient.git

For more details on working with multiple remotes, see this page: <https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes>

## Branches

* **Master:** Stable/Release branch
* **Develop:** The active branch where developers merge code into. Relatively stable.

## Branch Maintenance

As the developers work, they will branch off of develop and use pull requests to merge their changes back into develop.

After the sprint ends, develop will be merged into master. After branch master contains all the commits from develop, someone will need to push changes to the CSUS origin. Run the following commands to do this:

git checkout master

git pull

git push csus master

# Testing Workflow

Work items (bugs and user stories) that are marked ‘Resolved’ are ready for testing by someone other than the team member who worked on them.

To begin testing an item, assign the item to yourself. Then…

### For User Stories:

Check that functionality described in the Acceptance Criteria is present and complete in full. In the event that bugs relating to this story are discovered but the Acceptance Criteria passes, file a bug and link it to this story as related.

If the story passes Acceptance Criteria: set status to ‘Closed’

If the story doesn’t pass Acceptance Criteria: assign it back to the team member who worked on it and set the status back to ‘Active’. (If you don’t remember who worked on the item, this can be seen in the history tab on the story)

### For bugs

Follow the repro steps to ensure that the undesired functionality is no longer reproducible. If the bug has been fixed, mark it as ‘Closed’. Otherwise, assign it back to the person who originally worked on it, and change the status to ‘Active’.

# Team Communications

* Discord is the primary form of communication;
* Only the Product Owner (Josie) should communicate with the Client directly;
* Clarifications on work items should go thru Product Owner. Relevant details should be saved as comments on the related work item;

# Pointing System

Stories and bugs should be pointed using a fibonacci scale. Ideally, a single work item shouldn’t usually be pointed above a 8. If this happens, the story should be broken into multiple smaller stories. Below is a basic outline for reference on how much work is involved for each point:

* 1 = Less than 15 minutes (quick fixing of something, changing a text label for example)
* 2 = Roughly between 15 minutes and 3 hours (working on implementation of a small feature addition or bugfix; adding a new rest endpoint for example)
* 3 = Roughly a day of work
* 5 = Roughly three days of work
* 8 = Roughly a week of work
* 13 = Roughly two weeks of work