# SENG3011: Using GitHub

## Why GitHub

Version control systems have become common practice amongst most software companies and utilising it within a team setting is a key skill to have. In this workshop teams will be required to use git for version control and github as their remote repository service. Submissions will be done using this service. Teams will be required to add their mentors into their repository as soon as possible.

## Account creation

Students without a github account, can create one by going to:

https://github.com/join?source=header-home

One individual from each team will have to create a private github repository with the name **seng3011-<team-name>**, adding all team members and mentor as contributors to the repository.

## Suggested Git workflow

Using git effectively and efficiently vastly improves a team's productivity. Teams in many software engineering firm will form git workflows to improve team collaboration and to prevent conflicts before they can occur.

Nominate one person from your group to manage the github repository. They will be responsible for creating the github repo, adding the start code and be the only one with push access to master.

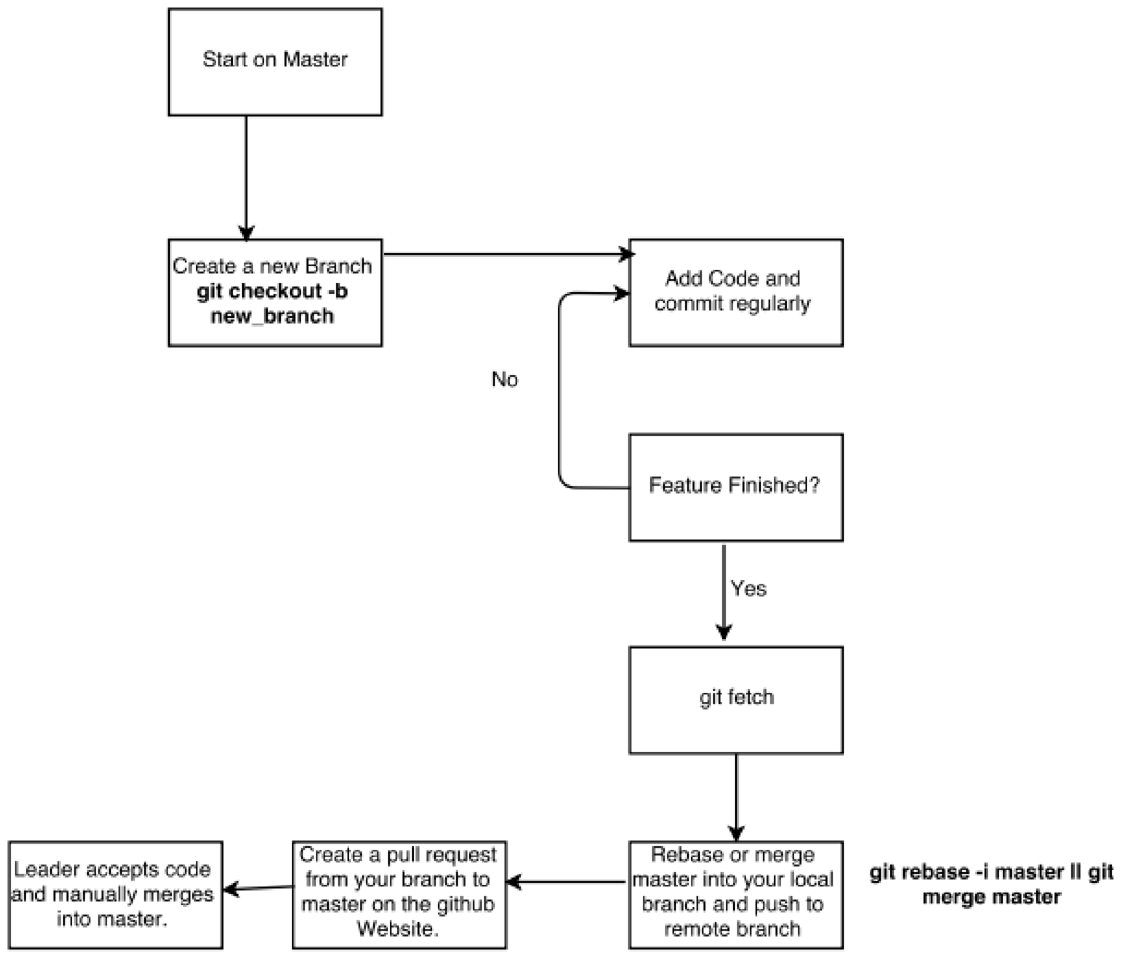
The workflow is as followed:

1. Create a separate branch from master (perhaps for a new feature to implement or a bug fix). Ensure your master is up to date before doing this.
2. Work on the feature or bug fix as normal and commit regularly to your new branch. Push this branch to the remote repository consistently (**git pull origin new_feature**)
3. When you have completed the feature or bug fix, perform a **git fetch** and **git merge master,** to ensure your branch is up to date with any new changes on the master branch. Ensure you resolve any merge conflicts before proceeding. Push your changes to the remote branch.
4. Go to your project page on github and create a pull request for your branch to be merged into master.
5. The leader will review the pull request and manually merge the feature branch into master. The branch used to work on the feature/bug fix may be deleted.

There are many ways to expand upon it such as including code reviews and unit tests that each feature must pass etc. The developer may also choose to rebase their branch before submitting a pull request for a cleaner commit history.

It is highly recommended that each team invests time in designing a workflow to ensure a smooth development process and less code and team conflicts.

A flow chart for the workflow is presented on the next page.


Start on Master
Create a new Branch **git checkout -b new_branch**
Add Code and commit regularly
No
Feature Finished?
Yes
git fetch
Rebase or merge master into your local branch and push to remote branch
**git rebase -i master || git merge master**
Create a pull request from your branch to master on the github Website.
Leader accepts code and manually merges into master.

## Account monitoring

We will facilitate submissions by having mentors check a team's repository. They will view the last commit timestamp before the set deadline for a given deliverable. This includes both prototypes and report submissions.

A small amount of marks can be gained for the final prototype based on a good usage of the repository between team members. Mentors will monitor progress in mentoring sessions by checking a team's repository and providing feedback on their usage.

## Resources for learning git:

https://learngitbranching.js.org/

http://jameschambers.co/writing/git-team-workflow-cheatsheet/

https://www.sitepoint.com/getting-started-git-team-environment/

https://www.atlassian.com/git/tutorials/comparing-workflows

https://www.atlassian.com/git/tutorials/merging-vs-rebasing