## **GIT GUIDELINE**

## **Steps we need to follow:**

## Clone project from github to your local machine

## Create your own branch in IntelliJ by typing -> git branch<branchname> (branch name will be user story related)

## Now you will work on your own version

## Please take below 6 steps before you start your working on your branch everyday :

## Before doing any changes on your branch in IntelliJ:

## **git checkout master** (switch to master branch)

## **git pull** (pull all updates from remote master branch to local master branch)

## **git checkout <branch name>** (switch to your own branch)

## **git merge master** (merge updates from master branch to your own branch)

## if you have any conflict, solve it (by right clicking on conflicted class, select git => click on resolve conflict)

## Continue your coding

## Before you push your updates to your remote branch, use these 3 commands:

## Be sure that you are working on your own branch on IntelliJ

## **git add .**

## **git commit -m "Your message here"**

## **git push**

## Then go to **Github** and make **pull request**

## Git Dos

* Create a Git repository for every new project.
* Always create a new branch for every new feature and bug.
* Regularly commit and push changes to the [remote](https://blog.axosoft.com/remote-repository/) branch to avoid loss of work.
* Include a [gitignore](https://git-scm.com/docs/gitignore) file in your project to avoid unwanted files being committed.
* Always commit changes with a concise and useful commit message.
* Utilize [git-submodule](https://blog.axosoft.com/learn-git-submodule/) for large projects.
* Keep your branch up to date with development branches.
* Follow a workflow like Gitflow. There are many workflows available, so choose the one that best suits your needs.
* Always create a [pull request](https://blog.axosoft.com/pull-requests-gitflow/) for merging changes from one branch to another.
* Always create one pull request addressing one issue.
* Always review your code once by yourself before creating a pull request.
* Have more than one person review a pull request. It’s not necessary, but is a best practice.
* Enforce standards by using pull request templates and adding continuous integrations.
* Merge changes from the release branch to master after each release.
* Delete branches if a feature or bug fix is merged to its intended branches and the branch is no longer required.
* Include read/write permission access control to repositories to prevent unauthorized access.
* Add protection for special branches like master and development to safeguard against accidental deletion.

## Git Don’ts

* Don’t commit directly to the master or development branches.
* Don’t hold up work by not committing local branch changes to remote branches.
* Never commit application secrets in public repositories.
* Don’t commit large files in the repository. This will increase the size of the repository. Use [Git LFS](https://blog.axosoft.com/learning-git-git-lfs/) for large files.
* Don’t create one pull request addressing multiple issues.
* Don’t work on multiple issues in the same branch. If a feature is dropped, it will be difficult to revert changes.
* Don’t reset a branch without committing/stashing your changes. If you do so, your changes will be lost.
* Don’t do a force push until you’re extremely comfortable performing this action.
* Don’t modify or delete public history.

For Our Projects:

* Do not merge your pull request.
* Do not make changes to Utilities or other team members’ classes.

Versions for our project:

Java SE 14.0.1

<https://www.oracle.com/java/technologies/javase-jdk14-downloads.html>

IntelliJ IDEA 2020

<https://www.jetbrains.com/idea/download/?_ga=2.130526543.1344273135.1587363497-1032979133.1586868262#section=windows>