CS673 Software Engineering

Lab 1 - Set up Git

## Description:

The purpose of this lab is to make sure every student gets their git/github set up for their project and can perform basic git operations.

## Instructions:

1. Set up a GitHub account if you don’t have one already. Go to GitHub.com and follow the instructions to set up your account. Then accept the Github invitation.
2. Install and set up Git on your local machine. Git is usually preinstalled on Mac and Linux. But if you are using Windows, or want a GUI client, you will need to install it separately. Git comes with built-in GUI tools <https://git-scm.com/downloads>. There are also some third party tools. If you are not familiar with the command line interface, a popular option is to use Github Desktop since we will use Github for our projects (<https://desktop.github.com/>).
3. If you are not familiar with git, please go through some git tutorials. There are many online tutorials, such as: <https://git-scm.com/docs/gittutorial> (this one is from the git official site), <https://try.github.io/> (this one is from github) and <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf> (a cheat sheet of git commands). Here are the tutorials about github desktop. <https://docs.github.com/en/desktop/installing-and-configuring-github-desktop>, <https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop>.

**Take some screenshots of the following steps and include them into your lab report.**

1. Clone your team github repository into your local machine. You can use either the git command (git clone) or any git GUI app.
2. Check out the lab1 branch using either the git command (git checkout lab1) or any git GUI app.
3. Change the team.md file in the lab1 branch to add a paragraph about yourself and commit the change to your local repository using either the git command (git commit) or any git GUI app. The team.md file is a simple markdown file. You can find more information about basic markdown syntax here: <https://www.markdownguide.org/basic-syntax/>.
4. Push your commit to your team remote GitHub repository using either the git command (git push) or any git GUI app. If there is any conflict, you need to manually merge and solve the conflict.
5. Go to the remote GitHub repository webpage, and check if your change is successful.
6. After everyone in your group has updated the file team.md and committed the change to the branch lab1, and the project leader has reviewed the file, if you are in charge of your team GitHub repository, you shall merge the lab1 branch into the master branch, and then delete the lab1 branch afterwards.
7. Submit a lab report on Blackboard to include the following sections in your report:
   1. Your github **project URL and your github username.**
   2. Screenshots to show how you use git to modify the file in the repository.
   3. A summary paragraph about your experience with git/GitHub and any issues you encountered.
   4. List at least three other commands or functionalities from git/github that you may use later, and provide a brief explanation for each.

Aidan Duffy

METCS 673

Lab Report 1

GitHub Project: <https://github.com/BUMETCS673/BUMETCS673OLF21P1>

GitHub Username: @AidanDuffy

Git Screenshots:

1. This one shows my cloning the repository, stepping into it, and moving to the lab1 branch.

Text

Description automatically generated

2. This one shows my committing and push my changes to the lab1 branch.

Text

Description automatically generated

I personally did not encounter any issues with git or GitHub. I have a fair bit of experience developing utilizing version control software like git, paired with platforms like GitHub, GitLab, and BitBucket. I use git on a daily basis in my current position, so I am quite familiar with it.

In terms of commands I will use in the future:

1. “git branch”: I will use this to create a new branch for any new functionality I would like to implement, but this would be prior to pushing it out on the main, public branch. This would allow for testing.
2. “git status”: This is a command I generally like to use frequently as it informs me the list of files that I’ve changed since my most recent commit.
3. “git merge [branch]”: I, or someone on my team, will have to use git merge when combining branches, especially when adding new, tested functionalities from a secondary branch to the main branch.
4. “git log”: This is incredibly helpful on group projects as it provides me the chance to review previous implementations of the project’s code prior to major revisions. This can also be achieved by looking through all of the commits on GitHub.