**CMSC203**

**Lab #1 - Instructor Instructions**

**GitHub – To Do List**

**GitHub** is a web-based [hosting service](https://en.wikipedia.org/wiki/Internet_hosting_service) for [version control](https://en.wikipedia.org/wiki/Version_control) using [Git](https://en.wikipedia.org/wiki/Git). It is mostly used for [computer code](https://en.wikipedia.org/wiki/Source_code). It offers all of the [distributed version control](https://en.wikipedia.org/wiki/Distributed_version_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality of **Git** as well as adding its own features.

If you have ever edited several versions of a file, you may have added a date onto the file name so you don’t erase the earlier versions. Git was created to manage these versions transparently to you.

You might need to use Git or GitHub on a software development team in industry or government in the future. Therefore, this lab will introduce you to GitHub.

Octocat is the mascot of GitHub. You can see many versions of Octocat at <https://octodex.github.com/>.

* Using Git and GitHub

**Concepts tested by this assignment**

* + Creating an account in GitHub
  + Forking a repository
  + Editing a file
  + Merging a file

**Lab Description – (for students)**

In this lab, we will use GitHub to create a to-do list for a week, and update it as you complete the tasks. In the process, we will learn how to use GitHub, a collaboration tool used by many projects in industry and government.

**Task #1 – Create a GitHub Account**

To begin, create an account in GitHub at <https://github.com/>. The general account in GitHub is free, although it only allows you to create public repositories. Check your email and confirm that you signed up for GitHub (be sure you are logged in to GitHub when you check your email, and check your trash and spam folders).

**Task #2 – Fork Your Instructor’s Repository**

Then email the instructor with your GitHub username, so they can invite you to collaborate. Check your email for another invitation about collaboration, which you will need to accept.

When you accept an invitation from your instructor, you will have access to their repository. GitHub will send you to your account, with the instructor’s repo shown. DO NOT navigate away from this before you do the next step, or you may have trouble getting back to it.

Select your instructor’s repository, and then select fork. This will create your own version of the upstream repository. Once you have your own version of the instructor’s master branch, you can edit your version without causing issues with the master branch.

**Task #3 – Upload a To-Do List**

Then create a text file with your list of to-do’s for the week. Name it as follows: LastName\_FirstName\_ToDoList.txt. Upload the text file to your repo and click commit.

Optional: branch your master by clicking on the down-arrow to the right of branch:master and entering a branch name; edit your to-do list by adding DONE to tasks you have finished.

Optional: continue to branch for each day of the week

**Task #4 – Merge your To-Do List**

Then merge your repo (with your to-do list) to the instructor’s upstream repo. First, select new pull request next to your branch name. Then select the green button labeled Create pull request. You will be shown a comment page – enter a subject, and optionally a comment, then select the Create pull request button (again…☹). If your branch has no conflicts with the upstream repo (which it should not, since you are not changing anyone else’s files), you can select Merge pull request and Confirm merge. Now you have added your file to the instructor’s upstream repo.

By the way, the reason for the pull request is to allow discussion about your potential merge. Since Git is often used with large projects, your changes might or might not be accepted into the main project thread.

**Deliverables**

Your deliverables will be:

* your forked repo from the instructor’s upstream master branch.
* Your repo merged with the instructor’s upstream master branch, including the text to-do list file.

**Instructor Instructions**

You will need to sign up in GitHub (follow the directions for students above).

Check your email to confirm your account. I found that my confirmation went to my Trash folder. Check your Spam folder as well.

Sign up for an educator account so you can label your repo as “private”. To do this, select “Help” from your main menu (far upper right drop-down box), then type “educator account” in the text-box. Select the first choice, “About GitHub Education for educators and researchers”. The select “About GitHub Education” and “Join GitHub Education”. Select the Teacher and Individual Account radio buttons. Enter your information, including your MC email (…@montgomerycollege.edu). Check your email again to insure that you have educator status.

Create a repo with Lab1.docx from Blackboard. Select the “private” radio button. (If you don’t have a choice of public or private, the previous step may not have gone through yet.) You may want to include the Readme.md with some short comment about the assignment.

When your students have sent you their GitHub user names, you can invite them as “collaborators”. To do this, **select your repo**, then select “settings”, and “collaborators”. Enter their GitHub user name, select it, and select “Add collaborator”.

Then to grade the assignment, there are two parts.

* First, each student needs to “fork” your repo so they have a personal version. Their version of the repo will also be private, even though their accounts may only let them create public repos. You can check that they have successfully signed up for GitHub and successfully forked your repo by clicking on the number next to the “Fork” button. This will bring up the list of everyone who has forked your repo.
* Second, each student needs to create a To-Do List text file and upload it. They can create new branches to upload the revised file as they accomplish the tasks. At the end of the week (or whatever time period you prescribe), they should create a pull-request, followed by a merge that merges their repo with your master branch. You can check this by seeing what files have been added to your master branch.

Questions? See me (Rob Alexander) for a hands-on demo.