**Git & Github**

Objectives: Work with version control, git, and github.

1. “Git” is a version control system.
2. “Github” is on the cloud.
3. Version control allows you to take “snapshots” of your project to which you can back up the project.
   1. Provides safety by allowing you to archive and revert working code.
   2. Version control is a key tool in collaborative coding approaches.
   3. Git is the most common version control software.
4. Terminologies:
   1. Repository – source folder
   2. .git folder – the head references denote changes and are stored in this hidden folder.
   3. **Commit** – snapshots of the working code. Commits are saved as a linked list, where each commit is a node referencing the previous commit.
   4. .gitignore – used to exclude certain iles from git tracking.
5. File system commands
   1. Mkdir – make directory
   2. Cd – change directory
   3. Ls – list the stored things.
   4. Ls -a – list the stored things… **all** of them.
   5. Touch – creates an empty text file
   6. Echo – created a file with text in it (echo “hi” > foo.txt) 🡪 creates a text with “hi” in it.
   7. Cp – **copy** a file. Syntax (cp [oldname] [newname])
6. Common commands
   1. Git init -> initializes a folder as git repo
   2. Git status -> prints status of files in repo
   3. Git add -> adds file to version control
      1. In order to add “all” files, you type:
      2. Git add .
         1. “.” Means “current dir”
         2. “..” means “previous dir”
   4. Git commit –> commits changes
   5. Git log –> logs recent operations
   6. Git checkout -> creates or switches to “branch” (what’s that?)
      1. There’s a “main” branch and “feature” branch –
         1. The “Feature” branch is where you’d work on “side features” of an app…
   7. Git merge -> merges one branch with another.
7. As far as working on branches…
   1. You can use “git checkout -b “name”
   2. To go back to main, use git checkout again.
   3. And the branches can be used to work on certain features.
8. Git infrastructure:
   1. Github remote repo is the “origin”
   2. The computer has a local “repository” to which the repo is cloned from the origin.
   3. Before you merge changes back into the main branch,
   4. You would make a “pull request.”
   5. Git “pull” downloads changes from the remote repo.
      1. “git fetch” + “git merge”
   6. Forking –
      1. This is **not** what we do at a company workflow.
      2. Forking is like making a copy of someone else’s google doc.
      3. This is now “mine.”
   7. Git push <remote> <branch>
      1. Git push origin main
         1. “origin” is the “remote” location.
         2. “main” is the branch we are pushing (we are pushing our “main” branch to the origin remote.)
   8. We can have our github repo pointing to multiple remotes.
      1. Git remote add upstream <url>
      2. We are saying that “this other url repository is another upstream source”
   9. “Origin” usually refers to your forked repository.
      1. “git pull upstream main” -> in this context, “main” is “we pull into main.”