## Journal Entry 1, Mod 2 – 1/22/2025

Chapters 1 through 5

As I read through Chapter 1, I was reminded of all the difficulties I encountered while attempting to collaborate with others back in the early 1990’s. One of the key takeaways for me in the first chapter was the various options for saving work and collaboration and that git represents a safer and more efficient means of doing so. I can clearly remember collecting multiple diskettes and later USB drives and having to meet with other members of my project at odd hours. One of the difficult things about using git is that it is so far removed from what I first learned. This makes it hard to come up with analogies for what it is doing which is a habit I have formed in my learning. Also, some of the terminology is very new to me, specifically concepts like repositories, staging, working areas, and committing. Something that makes the concepts especially difficult is the fact that at least in the beginning, the Git Bash interface we are using, doesn’t have a visual representation of “where” things are. I am hoping perhaps there is something with more of a graphical representation as we proceed.

Chapter 2 was fairly straightforward to follow. I actually installed git on both a PC and a Mac. I have been more of a Mac user for the past ten years, but I recently purchased a PC for use in a IT Support class I am taking. The most troubling aspect of chapter 2 was that there were many details of installation that of necessity were glossed over. My experience working “under the hood” of a PC and a Mac is very limited, but I am not one that likes to just select options because I am told to do so. I like to understand every aspect of a process, but at this level in my education, there are too many aspects of PC and Mac operating systems that I have yet to been fully exposed to. One specific complaint I have about the text is that it was stated that we could default our Git Bash to use “master” or “main” as our original branch and that it “didn’t really matter” but GitHub uses “main” by default so in the early stages when I was relying on code lines from GitHub to set up my repository, I was having problems because I was using Master with one program and Main with another.

The processes we learned in Chapter 3 were somewhat challenging for me because I have had limited exposure to using terminal commands and at times it was tempting to do as much as I could in Windows with respect to file manipulation. With respect to the actual concepts of git, I feel the most important things to keep in mind are that we modify our files in the working directory, then we add them to staging area, then we commit them which is just creating a “snapshot” of the project. These concepts are very new to me as a method of saving work and collaborating. As I stated earlier, I try to learn by analogy, but I am trying to be cautious because this is very different from my initial experience when swapping physical media was one of the primary means of collaboration and backup, and it was common to have multiple distinct versions of a project saved in various places in their entirety.

Chapter 4 focused on the .gitignore feature. It was actually interesting to see commands put into place and the effect they had on which files were tracked. The most important part of the chapter from my perspective was the use of what I am told are called “wildcard” characters to allow multiple files and directories to be ignored with one line or to create very specific parameters if needed. This chapter also included the beginnings of seeing the history of our changes and even seeing what some of the changes were. It also introduced keyboard commands to allow navigation among the various commits.

Chapter 5 provided a helpful recap of the three possible states a file can be in: modified, staged, or committed. As stated, these concepts are very new to me, so repetition is extremely helpful. The chapter also tried to emphasize the fact the way git “saves” is different from my previous experience. An important concept is the idea of the “snapshot” that occurs during a commit. The text says that “the state of the entire project is saved, not just small changes.” It also clarifies that changed files are saved for the snapshot, while files that are unchanged are only referenced for the snapshot. This concept is still very difficult to grasp, and it will take time to understand the finer points of it. The last key takeaways for me with respect to Chapter 5 are perhaps the most useful processes which are the checkout, revert, and reset features. These were very difficult processes for me to implement perhaps because I have been using Windows and MacOS for so long that I need a special representation of what I am working on. Like my difficulties utilizing terminal commands in chapter 3, if I am working on a certain branch, or moving backwards or forwards through a series of commits, I want a visual representation of where I am “at”. As I have said before, I hope that once we understand the git basics, there are more sophisticated interfaces available.

## Journal Entry 2, Mod 3 – 1/27/2025

Chapters 7 through 9

Chapter 7 does not contain any specific commands or procedures that can be used, but it was important because it underlined the difference between Git and GitHub. It was interesting to revisit this idea, because prior to starting this course, I thought GitHub was just an online software development tool and did not realize that Git was a separate entity from GitHub or that Git resides on your computer. This chapter reminded me of how much I have already learned about what Git is and how many things I can do now in terms of backing up my work that I could not before.

The most interesting information I found in Chapter 8 was the introduction of Markup language. I had never heard of this before, but I am interested in learning more about its use since I document my edits excessively in the event I need to revisit something or explain my work to others. These utilities can help me keep my documentation organized if it gets too long. I was also interested to learn about the wiki resources available. Something I found helpful as I read was the idea that the wiki is actually a git itself. I had never considered this before, but wikis are something I am very familiar with so I may have found the analogy was looking for as I try to grasp what git really is.

Chapter 9 was the most important chapter I have encountered so far. Because we were asked to use git to submit our assignments, we had to set up and sync our GitHub repository with a local one early in the course. When I first started out, I was forced to make use of the code that GitHub provided and paste it into Git Bash because I didn’t know any other way to do it. I was glad to finally learn the commands that would allow me to manually link my local and remote repositories. After reading chapter 9, I finally understand what the code that GitHub provides does. It goes back to what I have talked about before when I say that I don’t just like to do things without knowing what’s going on under the hood. Instead of using pre-generated code supplied by Git Hub, I can now use commands like “remote” to link my repositories manually. Chapter 9 provided a very helpful explanation of the proper format of the git remote command and what each portion of the command meant. With what I have learned so far, when I start my SDEV and Computer Science classes again, I will approach backing up my work very differently. and I will have a better understanding of the project setup process. It is even interesting to note that as I save this Word document on my computer, the process is somewhat similar to what happens with git including the option to “merge” my changes.

## Journal Entry 3, Mod 5 – 2/4/2025

Chapters 10 through 14

With respect to chapter10, I feel the most important concept is the idea of issues. GitHub provides many utilities related to issues, and it is a tool that could be very useful for keeping teams organized. Users can create custom labels and tie them to specific commits. They can also use issues to create accountability and provide a sense where the team is at on the timeline using milestones. I do think it would take me a great deal of time before I was proficient enough in the use of issues before they would replace my current means of task management, however, leverage the new features available in the development world is why I took this class so it would definitely be worth the effort to learn. Chapter 11 is focused on branches. Prior to starting this class, I had no idea that the ability to branch off lines of development existed in any way other than by use of brute force methods I definitely did not distinguish between creating a branch to work on and traveling backwards and forwards along a branch. Any experimental or other changes that I made to my code were perceived to me to be fully linear with one version replacing anything version and overwriting it. It was difficult to grasp the idea that both version could still exist side by side. I found it interesting that the book and other sources suggested the use of branches to be able to focus on bug fixes and new features. Prior to this suggestion, the primary use I perceived for branches was so that large projects could be compartmentalized and divided among users. The problem that I will struggle with related to branches is that on GitBash itself there is no visual representation of a branch. I just noticed today after all this time, that if I checked out a new branch, my local repository immediately changed to match the branch. I didn’t realize that the names of files or presence or absence of file adjusted themselves on my local repository. I will definitely use branches all the time as I move forward. Important commands that are specific to branches include creating them, checking them out, deleting them, and seeing a list of them, including seeing which branch the user is currently on. While trying to work with pull requests (sometimes unsuccessfully), I received a great deal of unintended practice at deleting and creating new branches.

Chapter 12 discusses the topic of pull requests. This was an area where I really struggled, because some of the steps in the process seemed very similar to each other and there was a great deal of clicking even when there were no conflicts in versions. I never really felt like I knew when the process was “done”, and I also struggled with the idea that a push request in the terminal required a pull request on GitHub.

Chapters 13 and 14 cover the content which causes me the most difficulty. The main reason that I took this class is because while using GitHub for a computer science class I was unable to save my work on a large project, because I had inadvertently created a merge conflict. I had to spend a long time communicating with one of my instructors remotely in order to resolve the conflict and get back to a version that I wanted. Ironically enough, my struggle with these chapters was in trying to create a merge conflict so that I could practice addressing it. Merging branches will remain the most challenging concept for me to master for a long time. One important takeaway for me is to try to make small changes and frequent commits. I remember there were times when I was working with massive code changes for a large program I was working on and there were definitely times when I should have remembered the axiom “save early, save often”. In this case, I suppose it would be “commit and push early” instead of “save”. The chapters also point out, however, that even this will not prevent conflicts as there could be several users working on the same branch and trying to push changes at roughly the same time. Dealing with conflicts is challenging because it requires everything we already know about git including creating a branch, pulling from a branch, pushing to a branch, traversing among branches, communicating and collaborating with the team, and even making sure that we have correctly set up our remote GitHub repository in the first place. Also, merging from the command line felt very different to me that doing most of the work in GitHub and both experiences were different than using the GitHub app as I did in my programming class.

## Journal Entry 4, Mod 6 – 2/10/2025

As part of my continuing study of chapters 13 and 14, I went back and reviewed the entire text from the section starting with setting up git and moving forward. With a slower re-reading and redoing several of the exercises, I now feel more comfortable using git (although, I still have very few commands committed to memory). Now that I know how to use them, the ability to resolve conflicts and merge, as well as the ability to cherrypick and use git clean will be of great value to me moving forward. Based on a very stressful situation during my computer science class where I thought I had “broken” 2500 lines of code at the 11th hour, my experience with git had been one of extreme anxiety. Git remained a mystery to me, and it’s use seemed much less secure than just saving multiple files on physical media. After a great deal of effort and reworking the exercises, I can say that I left this chapter with a great sense of accomplishment. One of the most difficult situations I encountered with Git and GitHub was not being able to push updates because I couldn’t pull recent remote branch changes without overwriting my work. This was the number one activity that I wanted to be able to engage in, and after some effort, I was able to use git stash to proceed and address the issue that had caused me so much frustration. When I take Computer Science II in the near future, I will have an entirely new perception of git and the software development tools that we use to code. I’m sure I will face additional challenges moving forward, but cherrypicking, stashing, and merging are no longer foreign to me and are not things to avoid or leave to the instructor to help with. This module was a good beginning in my future with software development more so than anything else I have done so far.