**Lesson 1: Practicing Git Basics**

| **Title of Unit** | Foundations | **Grade Level** | 11-12 |
| --- | --- | --- | --- |
| **Subject** | Mobile App Development | **Time Frame** |  |
| | **Description** | Before you can understand how to program the web, you need a more rigorous understanding of the web itself than you likely have now. These concepts provide a more holistic understanding of the ecosystem in which you will be working and will enable you to talk intelligently with other developers about your work.  Git is like a really epic save button for your files and directories. Officially, Git is a version control system.  A save in a text editor records all of the words in a document as a single file. You are only ever given one record of the file, such as essay.doc, unless you make duplicate copies (which is difficult to remember to do and keep track of):  essay-draft1.doc, essay-draft2.doc, essay-final.doc  However, a save in Git records differences in the files and folders AND keeps a historical record of each save. This feature is a game changer. As an individual developer, Git enables you to review how your project grows and to easily look at or restore file states from the past. Once connected to a network, Git allows you to push your project to GitHub for sharing and collaborating with other developers. | | --- | --- | | | | |
| **Stage 1 - Identify Desired Results** | | | |
| **Learning Outcomes**  What relevant goals will this lesson address? | | | |
| Computer and Information Sciences, General.  **CIP#**: 11.0101  Pathway Competencies   * **Networks and the Internet**: * **Computing Systems**: An understanding of the physical components and software that make up a computing system which communicate and process information in digital form, along with practices and methodology for troubleshooting issues in those systems. | | | |
| **Enduring Understandings** | | **Essential Questions** | |
| *Students will understand…*   * *How version control is important to programming* * *The basics of the Internet and how it functions* * *How git functions and is used to create version control* * *Installation and setup of virtual environments for coding* | | *Content specific….*   * *What kind of program is Git?* * *What are the differences between Git and a text editor in terms of what they save and their record keeping?* * *Does Git work at a local or remote level?* * *Does GitHub work at a local or remote level?* * *Why is Git useful for an individual developer?* * *Why are Git and GitHub useful for a team of developers?* * *What is a network?* * *What is the internet?* * *What is an IP address?* * *What is a router?* * *What is an ISP?* * *What are packets and how are they used to transfer data?* * *What is a client?* * *What is a server?* * *What is a web page?* * *What is a web server?* * *What is a web browser?* * *What is a search engine?* * *What is a DNS request?* * *What browser are you currently using?* * *What is the Git command used to get a full copy of an existing Git repository from GitHub?* * *What is the Git command used to check the status of your files?* * *What is the Git command used to track files with Git?* * *What is the Git command used to remove tracked files with Git?* * *What is the Git command used to commit files?* * *What is the Git command used to view your commit history?* * *What is the Git command used to upload projects into GitHub?* * *Explain the two-stage system that Git uses to save files.* * *Explain what origin is in git push origin main.* * *Explain what main is in git push origin main.* | |
| **Knowledge:** | | **Skills:** | |
| *Students will know how to...*   * *Describe what the internet is.* * *Describe what packets are and how they are used to transfer data.* * *Understand the differences between a web page, web server, web browser and search engine.* * *Briefly explain what a client is.* * *Briefly explain what a server is.* * *Explain what IP addresses are.* * *Explain what DNS servers are.* * *Explain what Git and GitHub are and the differences between the two.* * *Describe the differences between Git and a text editor in terms of what they save and their record keeping.* * *Describe why Git is useful for an individual developer and a team of developers.* * *Describe how to copy an existing repository from GitHub onto your local machine.* * *Explain the two-stage system that Git uses to save files.* * *Describe how to upload your work to GitHub using Git.* * *Describe how to check the status of your files and how to view your commit history.* * *Describe what the command line is.* * *Describe the difference between staging and committing changes.* * *Describe the difference between committing your changes and pushing them onto GitHub.* * *Explain how to check the status of your current repository from the command line.* * *Explain how to look at the history of your previous commits from the command line.* | | *Students will be able to…*   * *Access github and git* * *Upload and modify files using git and github* * *Actively create files using version control* * *Open the command line on your computer.* * *Use the command line to navigate directories and display directory contents.* * *Use the command line to create a new directory and a new file.* * *Use the command line to rename or destroy a directory and a file.* * *Use the command line to open a file or folder in a program.* * *Create a Git repository on GitHub and copy it onto your local machine.* | |

| **Stage 2 – Assessment Evidence** | | |
| --- | --- | --- |
| **Performance Task** | | |
| **PROJECT: PRACTICING GIT BASICS****Introduction** This short project will demonstrate how to use Git to manage and track your project folders.   1. You will set up a remote repository on GitHub and then copy it onto your local machine. 2. Once this repo is set up locally, you will be able to use Git like a save button for your files and folders. 3. When you have finalized your saves, you can then push your local repo up onto GitHub to share with everyone!  **Learning Outcomes** By the end of this lesson, you should be able to do the following:   * Create a Git repository on GitHub and copy it onto your local machine. * Describe the difference between staging and committing changes. * Describe the difference between committing your changes and pushing them onto GitHub. * Explain how to check the status of your current repository from the command line. * Explain how to look at the history of your previous commits from the command line. | | |
| **Other Evidence** | | **Student Self-Assessment** |
| * Exercise * Correct installation * Creating setup and login of git and github | | * Project Reflection * Knowledge checks (Exit Tickets) |

| **Stage 3 – Learning Plan** | | | | |
| --- | --- | --- | --- | --- |
| Activity 1 - How Does The Web Work?  Activity 2 - Command Line Basics  Activity 3 - Setting Up Git  Activity 4 - Introduction to Git  Activity 5 - Git Basics  Project 1 - Practicing Git Basics | | | | |
| **How will you engage students at the beginning of the unit? (motivational set)** | | | | |
| **HOW DOES THE WEB WORK?** Before you can understand how to program the web, you need a more rigorous understanding of the web itself than you likely have now. These concepts provide a more holistic understanding of the ecosystem in which you will be working and will enable you to talk intelligently with other developers about your work. | | | | |
| **#** | **Lesson Title** | **Lesson Activities** | **CCCs** | **Resources** |
| 1 | How Does The Web Work? | 1. Watch this [BBC short](https://www.youtube.com/watch?v=eHp1l73ztB8) for an overview of how the internet works. 2. Read this [article](https://developer.mozilla.org/en-US/Learn/Common_questions/How_does_the_Internet_work) from Mozilla on “How does the Internet work?”. 3. Watch [How the Internet Works in 5 Minutes](https://youtu.be/7_LPdttKXPc?t=46s). 4. Read up on the [differences](https://developer.mozilla.org/en-US/Learn/Common_questions/Pages_sites_servers_and_search_engines) between a web page, a web server, and a search engine. 5. Watch this [Google short](https://youtu.be/BrXPcaRlBqo) explaining what a web browser is. Then, find out what web browser you are using right [now](https://www.whatsmybrowser.org/). 6. Read about how one part of the web [interacts with another](https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/How_the_Web_works#Clients_and_servers) and [read about](https://developer.mozilla.org/en-US/Learn/Common_questions/What_is_a_domain_name#How_does_a_DNS_request_work) or [watch](https://www.youtube.com/watch?v=72snZctFFtA&feature=youtu.be&t=45s) a DNS request in action. |  |  |
| 2 | Command Line Basics | 1. Before diving into the command line lesson, you’ll want to know how to create a file. You can do so with the touch command. Open your terminal and enter ls (the l is a lowercase L). ls will show you the files and folders in the current directory (or will show nothing if the current directory is empty). Create a file called test.txt by entering this in your terminal: touch test.txt. Now enter ls once again. You should see test.txt listed in the output. You can also create more than one file at once using the touch command. Enter touch index.html script.js style.css and press the enter. Then enter ls once more. You should see the files in the output. Here is a small way that the terminal reveals its power. How long would it have taken to create all three of those files with your mouse? Thanks, terminal. 2. Read through [chapter 1 of Conquering the Command Line](http://conqueringthecommandline.com/book/basics).  **Exercise** In this exercise, you will practice creating files and directories and deleting them. You’ll need to enter the commands for this exercise in your terminal. If you can’t recall how to open a terminal, scroll up for a reminder.   1. Create a new directory in your home directory with the name test. 2. Navigate to the test directory. 3. Create a new file called test.txt. *Hint: use the touch or echo command.* 4. Open your newly created file in VSCode and make some changes, save the file, and close it. 5. Navigate back out of the test directory. 6. Delete the test directory.   That’s it–you’re done with command line basics! If you commit to doing most things from the command line from here on out, these commands will become second nature to you. Moving and copying files is much more efficiently done through the command line, even if it feels like more of a hassle at this point. |  |  |
| 3 | Setting Up Git | Step 1: Install Git  Step 2: Configure Git and GitHub  Step 3: Let us know how it went! |  |  |
| 4 | Introduction to Git | 1. Read Chapter 1.1 through 1.4 in [this book about version control](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control) to learn the differences between local, centralized, and distributed version control systems. 2. Watch [this video](https://www.youtube.com/watch?v=8oRjP8yj2Wo) about how Git can improve the workflow of both an individual and a team of developers. 3. Watch [this video](https://www.youtube.com/watch?v=1h9_cB9mPT8&feature=youtu.be&t=13s) for some history on Git and GitHub, and make sure you know the difference between the two. Git is a technology used in the command line while GitHub is a [website](https://github.com/) you can visit. 4. If you haven’t yet installed Git, visit the Setting Up Git lesson. |  |  |
| 5 | Git Basics | 1. Watch [this video](https://www.youtube.com/watch?v=HVsySz-h9r4) by Corey Schafer for a great overview of some basic Git commands. 2. Cheatsheet |  |  |
| P1 | Project 1: Practicing Git Basics | The main take away from the past few lessons is how to use Git and GitHub for your projects. You now have this very powerful skill that will help you immensely when we get into the coding projects. You will be able to share your work with others for code reviews and to get help with your code if you’re stuck.  In later Git lessons, we will cover some of the more advanced Git features, such as branches, which will further expand your abilities and make you more productive.  For now, concentrate on using the basics of Git that you’ve learned here with all of your projects from now on. You will soon know each of the basic Git commands from memory! **Reflection** This section contains questions for you to check your understanding of this project. If you’re having trouble answering the questions below on your own, review the material above to find the answer. Share your github repo with Mack also.   * How do you create a new repository on GitHub? * How do you copy a repository onto your local machine? * What is the default name of your remote connection? * How do you check the status of your current repository? * How do you add files to the staging area in git? * How do you commit the files to the staging area and add a descriptive message? * How do you push your changes to your repository on GitHub? * How do you look at the history of your previous commits? |  |  |

| **Stage 4 - Assess and Reflect** | |
| --- | --- |
| **Considerations** | **Comments** |
| **Is there alignment between outcomes, performance assessment and learning experiences?** |  |
| **Have I made purposeful adjustments to the curriculum content (not outcomes), instructional practices, and/or the learning environment to meet the learning needs and diversities of all my students?** | For struggling students:                    For students who need a challenge: |
| **Do I use a variety of teacher directed and student centered instructional approaches?** |  |
| **Do the students have access to various resources on an ongoing basis?** |  |
| **Have I nurtured and promoted diversity while honoring each child’s identity?** |  |