**SOURCE CODE MANAGEMENT**

**LAB FILE**

**COURSE CODE: CSE 2015**

**LAB SLOT: L14-L15**



Name: Shaun Sausman

Sen No: A86605224104

Faculty: Dr. Monit Kapoor

**INDEX**

**Lab Session 1: Git Fundamentals**

**Computer**

A computer is any device capable of performing calculations, whether they are

logical or mathematical.

**Program/Code**

A program (or code) is a set of instructions, often organised as an algorithm, that

directs a computer to perform a specific task.

**Need for Managing Source Code**

Modern applications, such as Spotify, consist of multiple programs working

together on both the frontend and backend to deliver a smooth user experience.

Regular updates are essential for:

• **Fixing Bugs:** Quickly resolving errors that may occur.

• **Improving UI/UX**: Enhancing the user interface and overall experience.

• **Optimising Performance**: Addressing and refining issues for better

performance.

For programmers, effective management of source code is crucial because:

• It ensures that all files remain in context throughout the lifecycle of the

program.

• It facilitates collaboration, allowing multiple developers to work together on a

shared codebase.

**Tools for Source Code Management**

1. Git:

A version control system that runs locally on your computer. Git helps track

changes and manage versions of your project.

2. GitHub:

A global, cloud-based platform that hosts Git repositories, enabling

developers to share, collaborate, and contribute to projects from anywhere in

the world.

**Version**

A version in version control represents a snapshot of your project at a specific

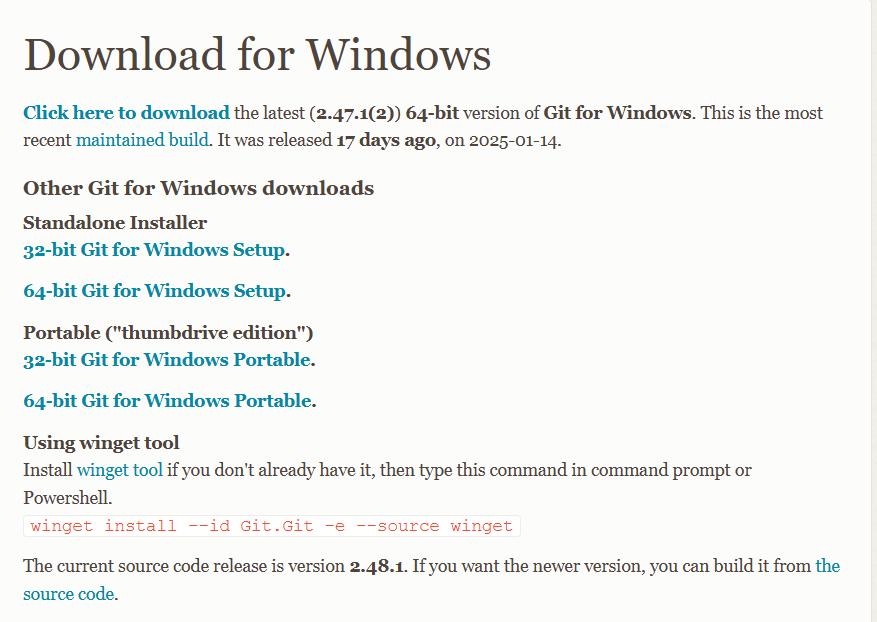
moment in time. This snapshot allows you to review, revert, or compare changes

made throughout the development process.

Lab Practical: 1

1. Installing Git

Step 1: Visit section 1.5 of pro git document and navigate to macOS section

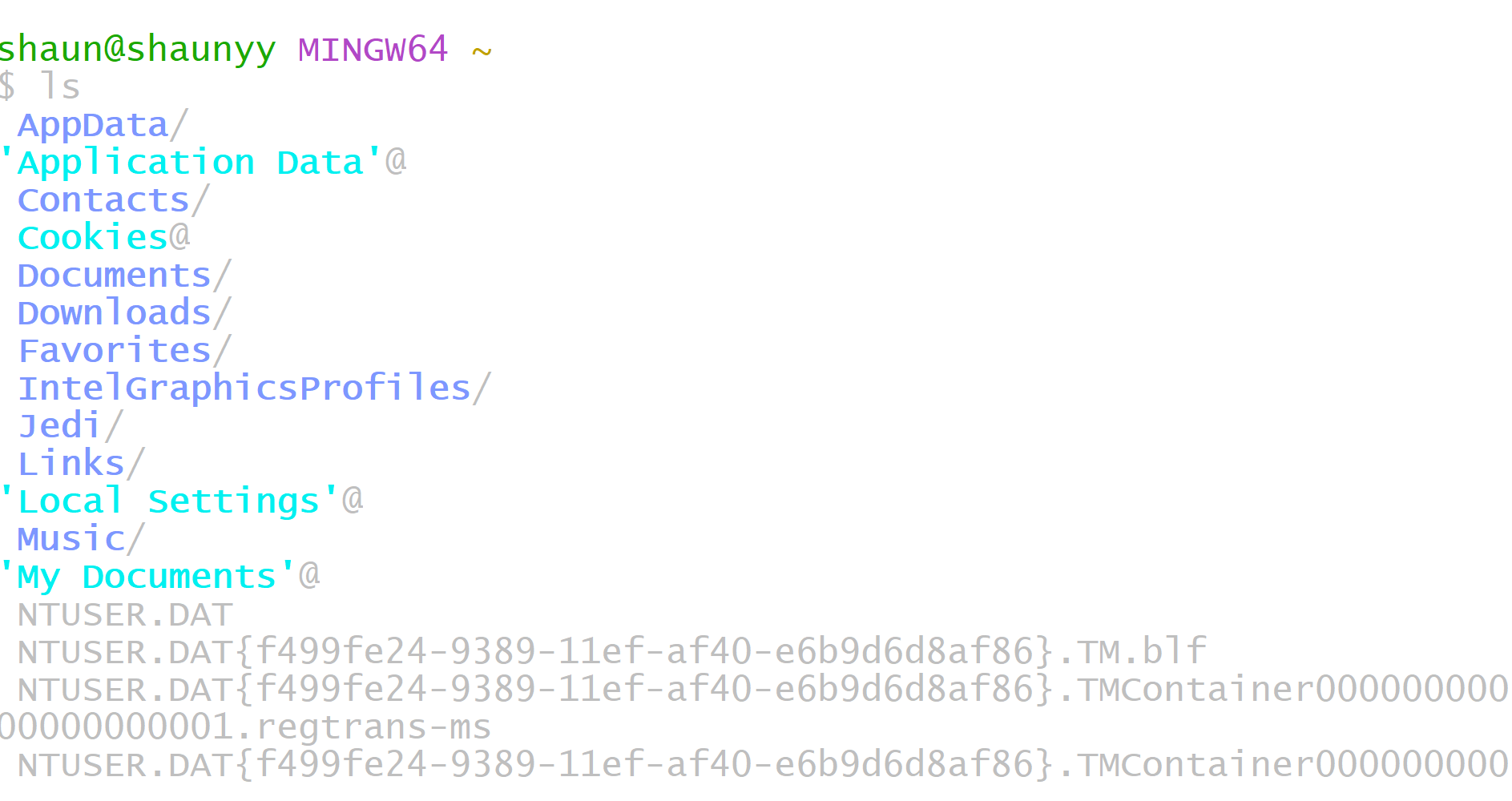


Step 2: let Git Bash install in your device. A screenshot of a computer

AI-generated content may be incorrect.

1. Basic CLI Commands
2. Command: ls

Description: Lists all files and directories in the current directory



1. Command: date

Description: shows the current date and time in a standard format.

A screenshot of a computer

AI-generated content may be incorrect.

1. Command: clear

Description: The clear command in the CLI is used to clear all the current text and output displayed in the terminal window.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

4)Command: time

Description: The time command in the CLI is used to measure the execution time of a command or program.

A white background with black text

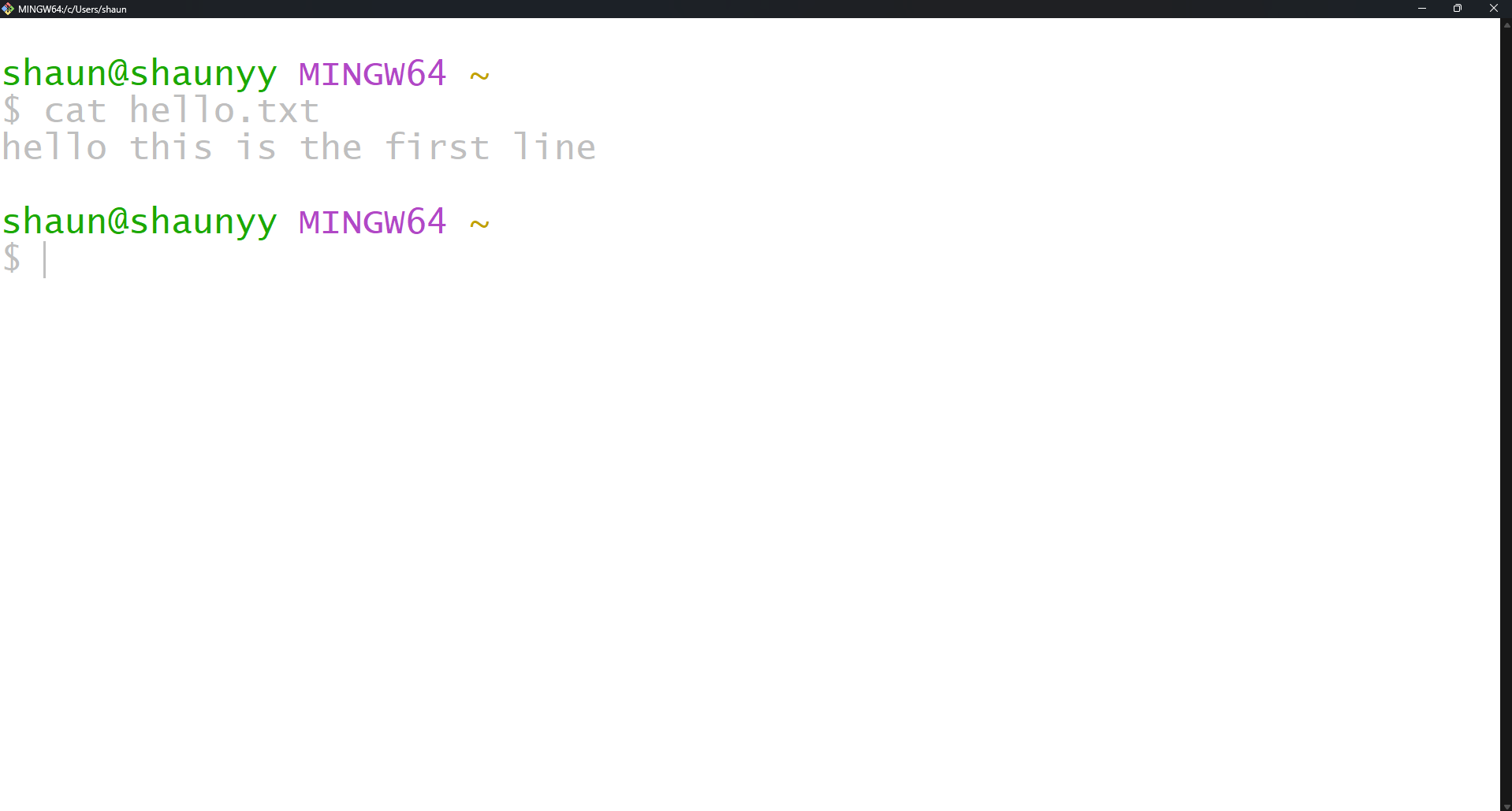
AI-generated content may be incorrect.

5)Command: rm hello.txt

Description: Removes the file hello.txt from the current directory.

6)Command: cat hello.txt

Description: The cat command (short for concatenate) is used to display the contents of a file.

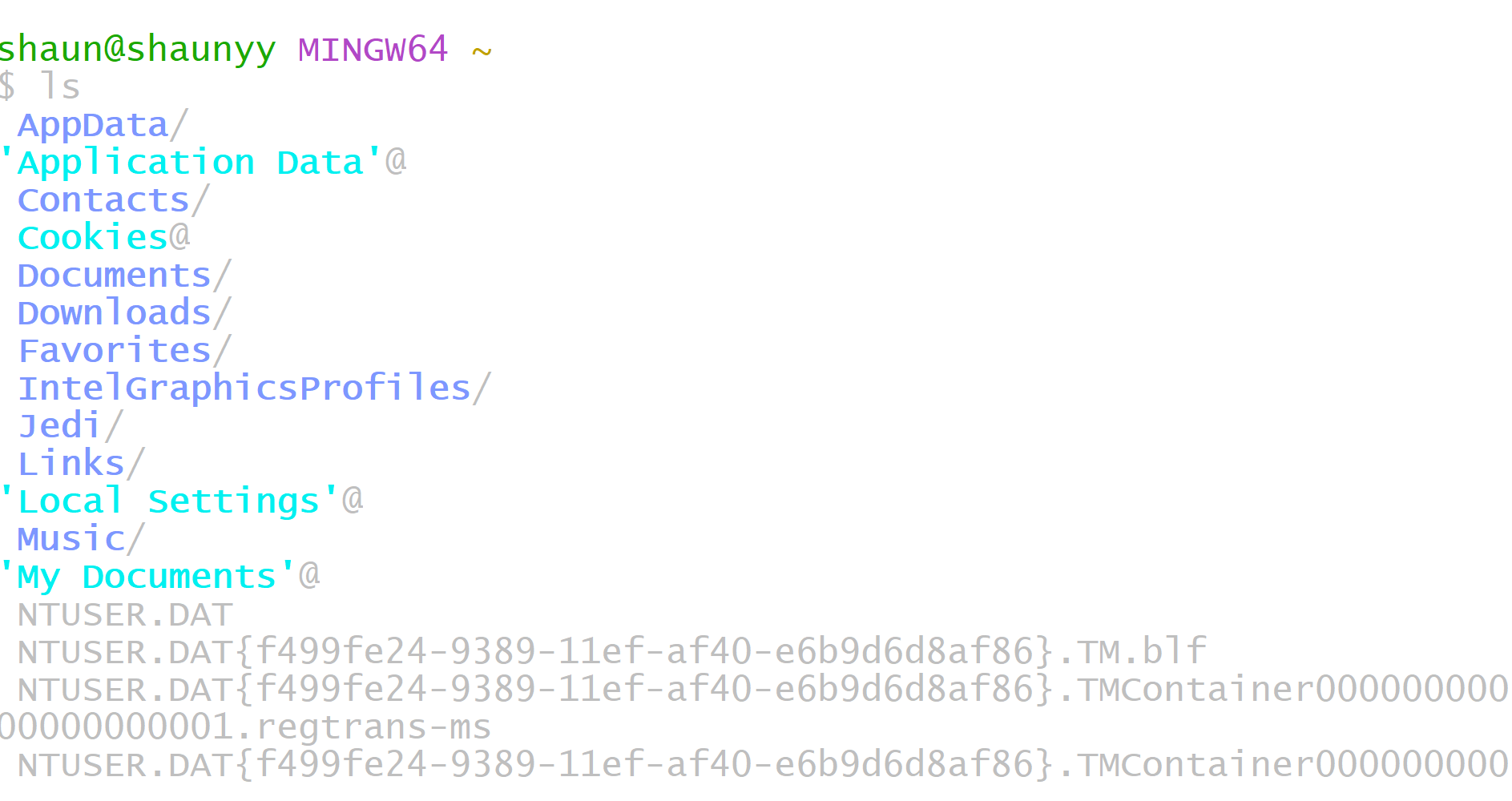


7)Command: cd Desktop

Description: Changes the current working directory to the Desktop directory.

8)Command: ls

Description: Lists all files and directories in the current directory.



1. **Vim Text Editor**

1)Command: vi hi.txt

Description: Opens (or creates) the file hi.txt in the Vim text editor.

A screenshot of a computer

AI-generated content may be incorrect.

2)Command: i (Insert Mode)

Description: Enters insert mode in Vim to allow text input.

A screenshot of a computer

AI-generated content may be incorrect.

3)Command: esc

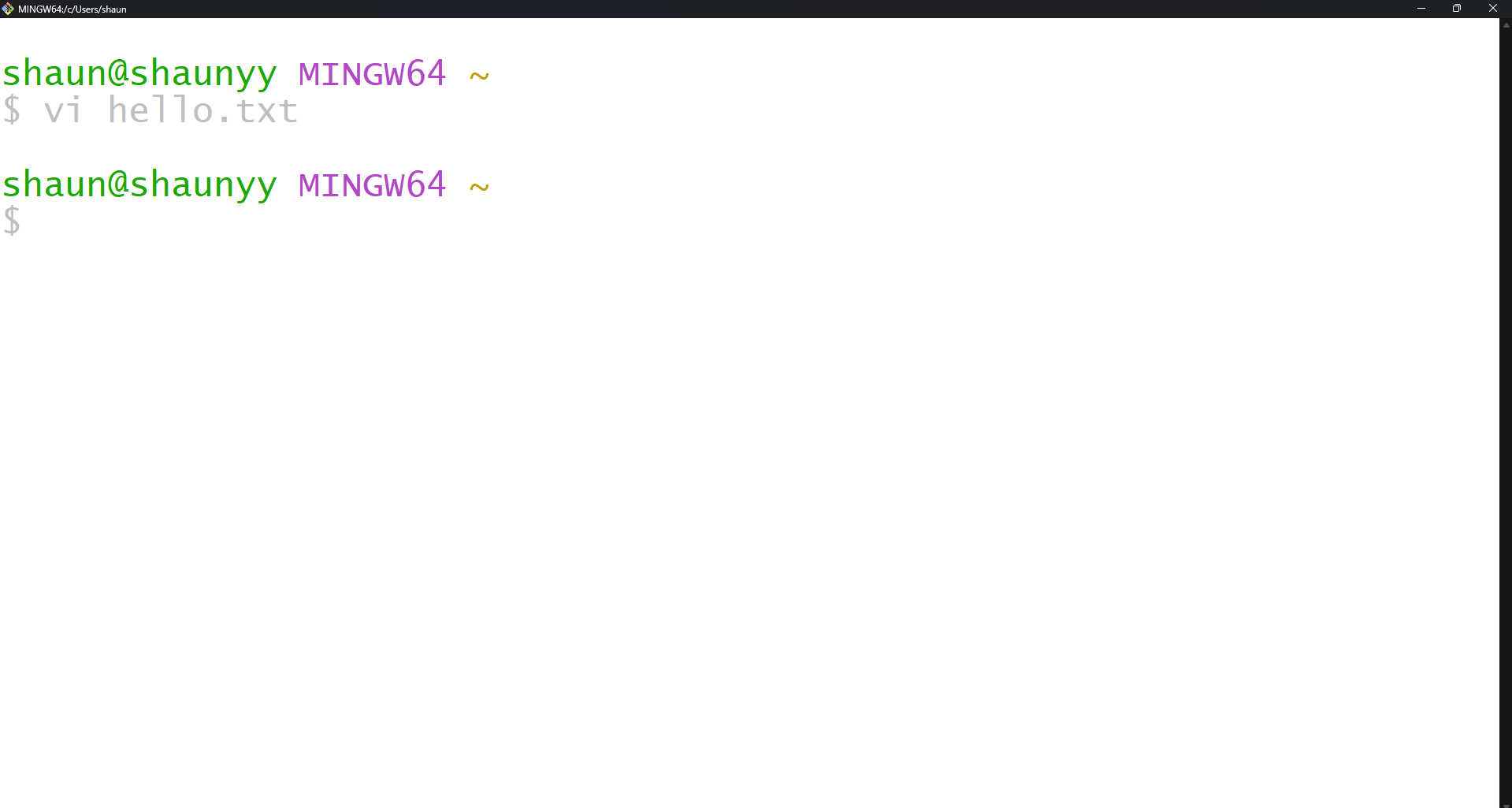
Description: Used to exit insert mode



4)Command: wq

Description: Saves the changes and exits the Vim editor.





1. **Git Commands**

1.Command: git - - version

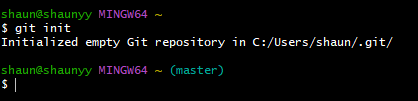
Description: The git --version command is used to check the installed version of Git on

your system.



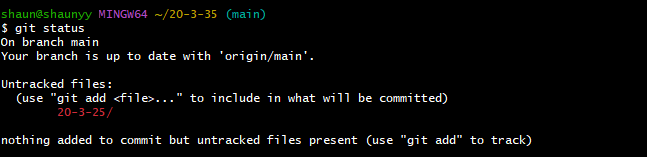
2. Command: git init

Description: Initializes a new Git repository in the current directory.



3.Command: git status

Description: Displays the current status of the working directory and staging area.



1. Command: git add hi.txt

Description: Adds testone.txt to the staging area in preparation for a commit.



5.Command: git commit -m "add file one"

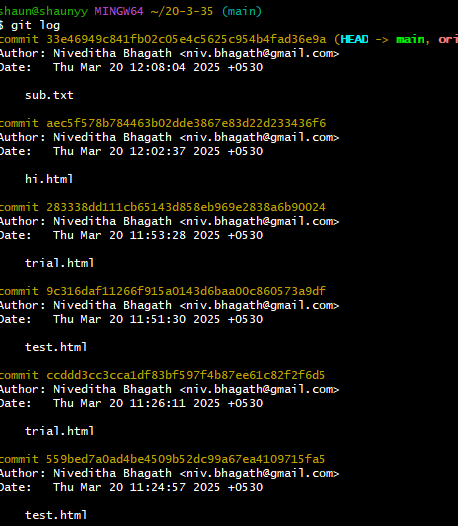
Description: Commits the staged changes with the message "add file one”.

A screenshot of a computer program

AI-generated content may be incorrect.

6.Command: git log

Description: Displays the commit history of the repository.



7.Command: git clone

A screenshot of a computer program

AI-generated content may be incorrect.

8.Command git config --global user.name “Shaun Sausman”

Description: used to set up user name which will be linked to future commits.



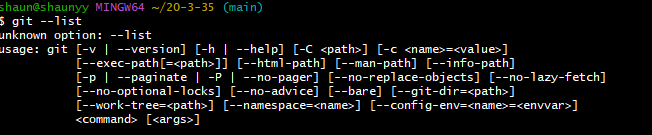
9.Command git config --global email.id “**shaun.sausman@s.amity.edu** “

Description: used to set up email Id which will be linked to future commits.



10.Command git config - - list

Description: used to view all the settings.



**Lab Session 2: Git Commands**

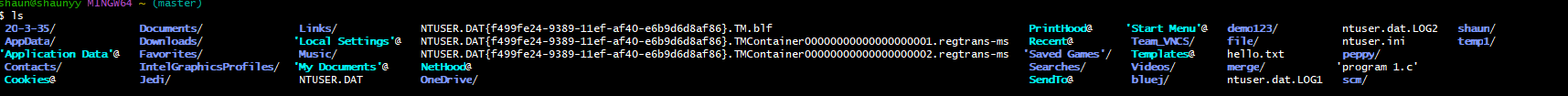
1. Command pwd

Description: returns the present working directory.



2.Command ls

Description: lists all files and folders in the current directory.



3.Command mkdir

Description: used to to make new directory/folder.



4.Command rmdir

Description: used to remove a directory.



5.Command cd

Description: used to change current directory.



6.Command cd ..

Description: used to exit the current sub directory.



**Git commit**

Step 1: create a file in the present working directory and add content.

Step 2: using git init initialise a hidden git repository for

tracking the files.

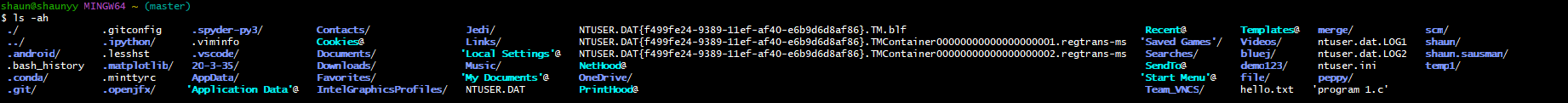
Step 3: using git add move the file to staging area.

Step 4: check git status for confirmation.

Step 5: commit the file to a local repository

Command ls-ah

Description: used to check hidden files in a directory.



Command git rm --cached <file>

Description : used to remove file from staging area.



**Lab Session 3: Git Diff**

**Mount Point**

Point from where we can access the desired folder directly

/Users/shaunsausman/Library/Mobile Documents/com~apple~CloudDocs/Amity/Git\_Amity/Lab\_2

In the above file path mount point of Lab\_2 is Git\_Amity

**Command touch <file\_name>**

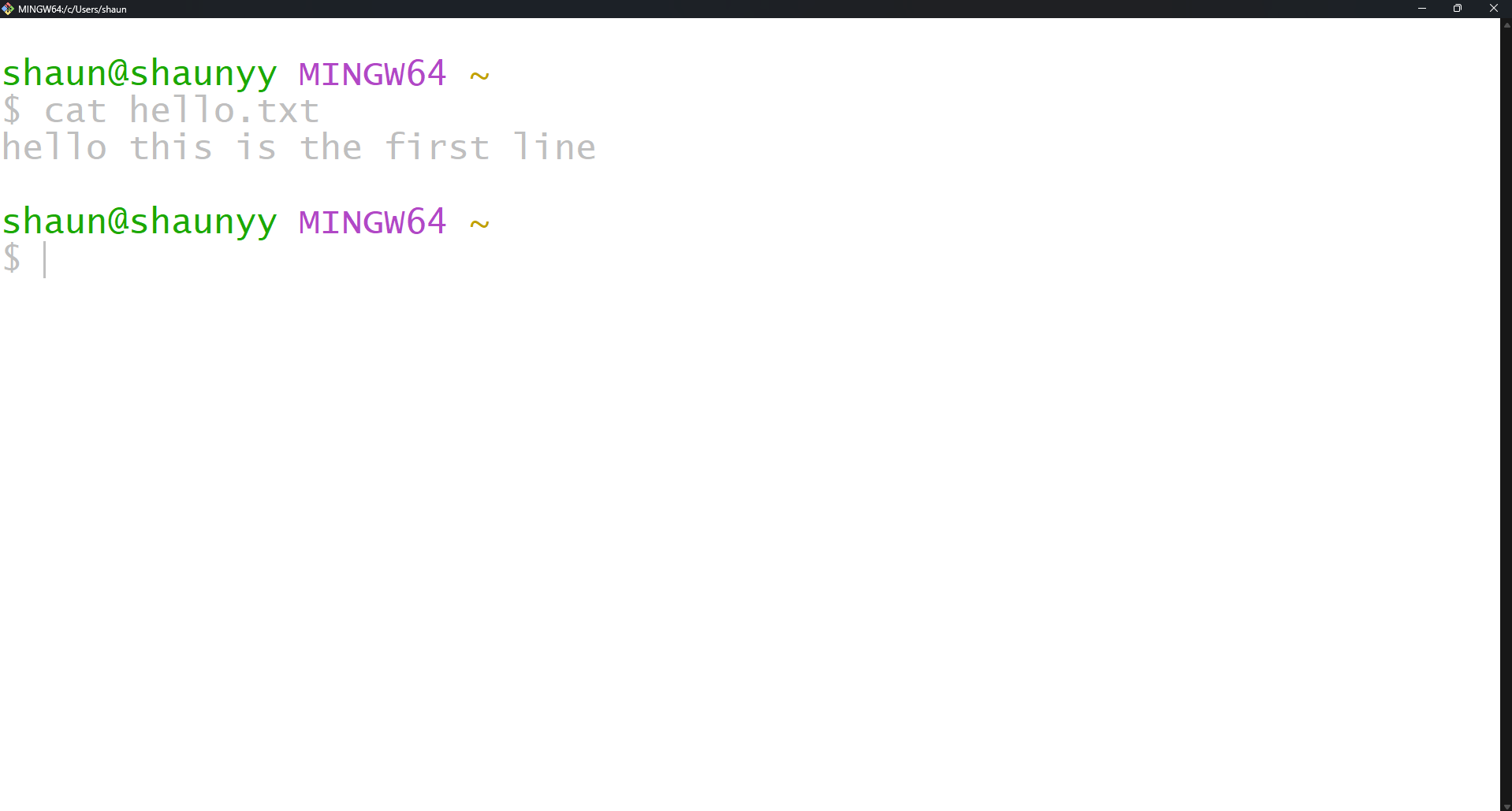
Used to create a file without any content

**Command git rm -rf <file\_name>**

Used to remove a file from git tracking

**Task 1: make two commits in a directory**

**Step 1: create two files with content in a directory**

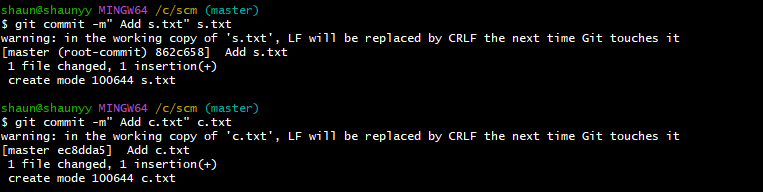


**Step 2: use git add. Command to add both files in the staging area.**

****

**Step 3: use git commit -m to commit both files to local**

**Repository**

****

**Step 4: use git log command to verify the commit history**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

**Step 5: use git log- - oneline for generating shorter commit id**

**A black background with white text

AI-generated content may be incorrect.**

**Task 2: compare two commits in a directory**

**Use git diff along with the commit id generated from git log - -online.**

**A computer screen with white text

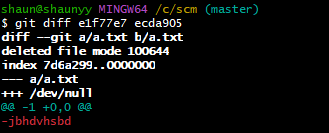
AI-generated content may be incorrect.**

**Lab Session 4 : git remote**

**Make 4 commits and compare them using git dif**

**A computer screen shot of text

AI-generated content may be incorrect.**

****

**Use git remote command to establish a connection between local Git repository and a remote repository.**

****

**Use The git push command to transfer commits from a local Git repository to a remote repository.**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Confirming the remote connection with git remote.**

****

**Checking the commits made on GitHub account**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Lab Session 5 : Working with remotes**

**Pointer to a commit Pointer: connects two memory address where at least one variable must have an active memory address Head: branch on which the last commit is made Git branch command used to view the existing branches in the git repository**

****

**Git checkout command used to switch the currently active branch to another branch. Here we want to create a new branch from a particular commit**

**A black background with white text

AI-generated content may be incorrect.**

**Git branch test\_1: used to create branch with name test\_1 Confirm the created branch by using git branch command to view all the branches.**

****

Use git\_checkout command to pivot to that particular branch

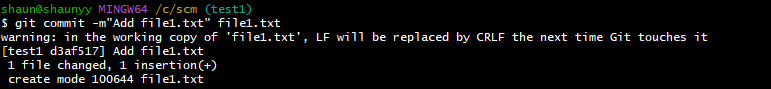
A screenshot of a computer program

AI-generated content may be incorrect.

Make commits in test\_1 branch

A black screen with white text

AI-generated content may be incorrect.



Viewing the commits on a particular branch

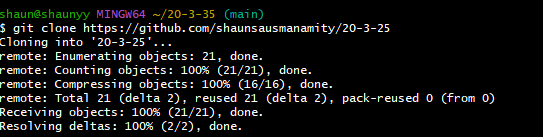
A screenshot of a computer code

AI-generated content may be incorrect.

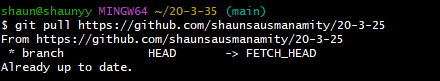
A computer screen with white text

AI-generated content may be incorrect.

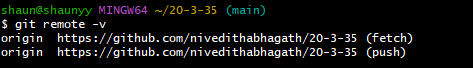
Git clone command used to create copy of an existing git repository



Git pull is used to fetch and integrate changes which are in the remote repository to local repository



Git remote-v: used to view all remote repositories in a directory



LAB SESSION 6: git branching

Branch: pointer to a commit

Pointer: connects two memory address where at least one variable must have an active memory address

Head: branch on which the last commit is made Git branch command used to view the existing branches in the git repository



Git checkout command used to switch the currently active branch to another branch. Here we want to create a new branch from a particular commit

A screenshot of a computer program

AI-generated content may be incorrect.

Git branch test\_1: used to create branch with name test\_1 Confirm the created branch by using git branch command to view all the branches

A screenshot of a computer

AI-generated content may be incorrect.

Use git\_checkout command to pivot to that particular branch

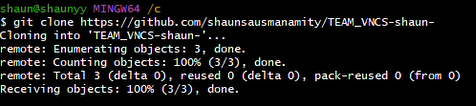
A screenshot of a computer screen

AI-generated content may be incorrect.

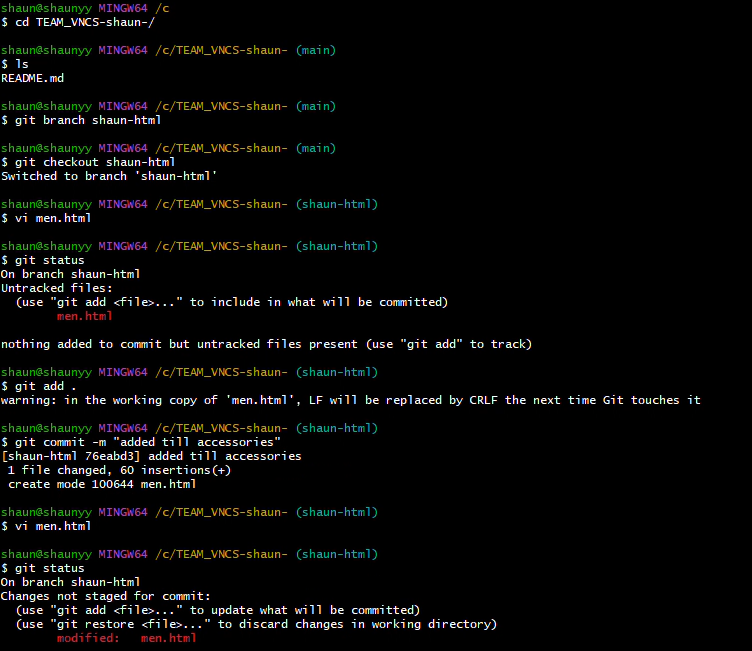
SCM PROJECT

The project was to make a repository in GitHub, make 3 branches and merge it with the main branch and access all 4 team-mate’s repositories, fork it, clone it, make some changes and merge them.

1.First, make your own repositories and clone it in your local file



2.Now create a branch and make commits in the branch



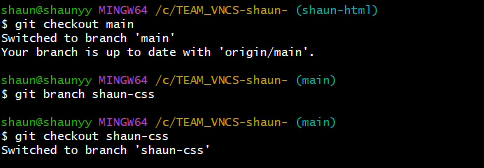
Use git log to check the commits done and checkout your main branch

A computer screen shot of a black screen

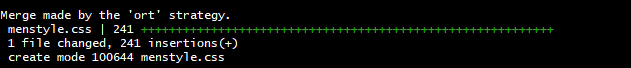
AI-generated content may be incorrect.

Now repeat this process for 2 more branches

Now checkout the main branch and merge the remaining 3 branches individually



Now merge the files



After the merges create a branch DEV and the push all the branches

1. Now create a forked repository of your first teammates repository.

A screenshot of a computer

AI-generated content may be incorrect.

1. Now clone the forked repo and make a new branch

A computer screen with white text

AI-generated content may be incorrect.

1. Now make changes in the new repo and commit the changes.

A screen shot of a computer

AI-generated content may be incorrect.

1. Now push that branch

A screen shot of a computer

AI-generated content may be incorrect.

1. Now go to Git Hub and create a pull request

A screenshot of a computer

AI-generated content may be incorrect.