**SOURCE CODE MANAGEMENT**

**LABFILE**

**Step 1: Downloading Git**

1. Open your web browser and navigate to the official Git website: <https://git-scm.com>.
2. On the homepage, you will see a "**Download**" button that automatically detects your OS. Click on the "Download" button to download the appropriate installer for your operating system (Windows, macOS, or Linux).
3. Alternatively, you can manually select your OS from the website to download a specific version.

A screenshot of a website

AI-generated content may be incorrect.

**Step 2: Running the Git Installer**

Locate the downloaded Git.exe file and double-click to run it.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: License (Terms and Conditions)**

Read the **GNU** General Public License’s terms and conditions and click on **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4: Choose Installation Location**

Choose the installation location (default is **C:\Program Files\Git**) and click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 5: Select the Components**

Select the components you want (default options are fine) and click **Next**.

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 6: Select Start Menu Folder**

Choose the Start Menu folder where Git shortcuts will be placed. By default, the folder is named **"Git"**. Keep the default name and click **Next** to Proceed.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 7: Choose the Text Editor**

Choose a default text editor (select **Vim**) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 8: Adjusting Initial Branch Name**

Choose the default name for the first branch when initializing a new Git repository. Go with **‘Let Git Decide’** option setting the branch as **Master** branch and proceed with **Next**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Step 9: Adjusting PATH Environment**

Select **Git from the command line and also from third-party software** (recommended). Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 10: Choosing the SSH Executable**

Select "**Use bundled OpenSSH**" for better compatibility and Click on **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 11: Choosing the HTTP Transport Background**

Choose Use the **OpenSSL library** (default) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 12: Configuring Line Ending Configs**

Select **Checkout Windows-style, commit Unix-style line endings** (recommended) and Click **Next**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Step 13: Configuring the Terminal Emulator**

Select **Use MinTTY** **(default terminal for MSYS2**) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 14: Choosing the Default Behaviour**

Select **Fast-forward or Merge** (recommended) option and click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 15: Choosing a Credential Helper**

Select **Git Credential Manager** (recommended) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 16: Configuring Extra Options**

Select **Enable file system caching** (recommended) and Click on **Install**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 17: Installation Overview**

A progress bar (**green bar**) will appear, indicating that Git is being installed. Wait for the installation to complete. This may take a few minutes.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 18: Completing the Git Set - Up Wizard**

Once the installation is complete, **"Completing the Git Setup Wizard"** screen appears. Check the ‘**Launch Git bash’** option and Click on **Finish**.

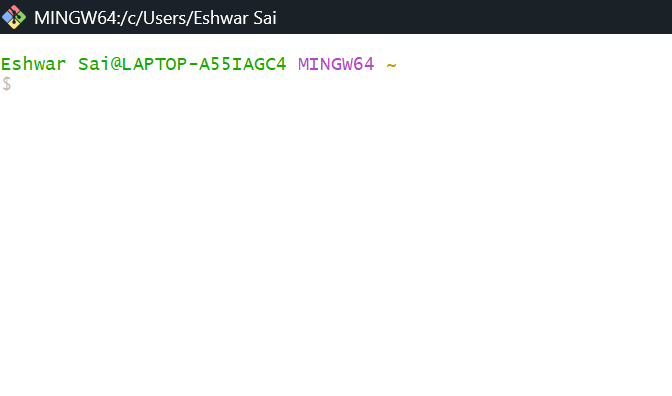
A screenshot of a computer

AI-generated content may be incorrect.

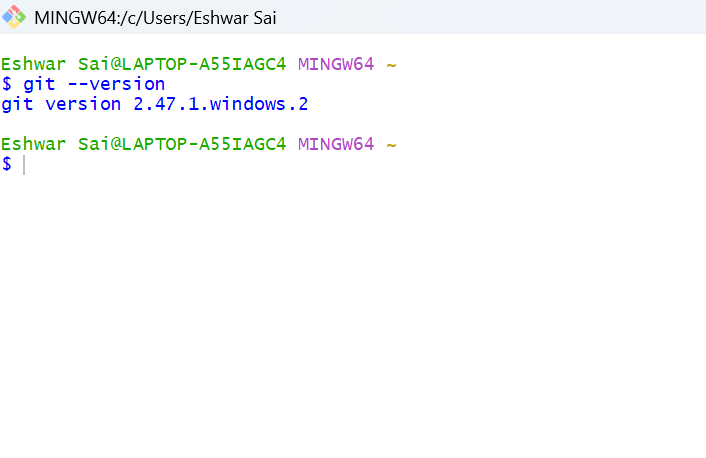
**LAB REPORT – 2**

**Step 1: Open Git Bash**

Open **Git Bash** from the Start menu or by searching for it.



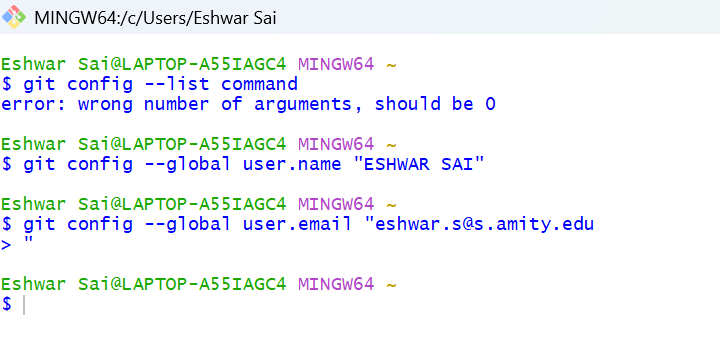
**Step 2: Check Git Version**

To verify that Git is installed correctly, run: **git –version** 

**Step 3: Configure Git**

Set up your Git username and email (required for commits):

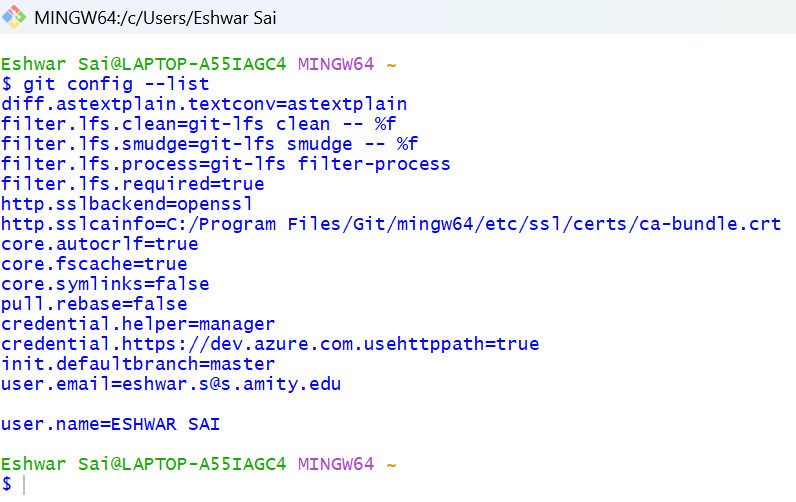
* **git config --global user.name "Your Name"**
* **git config --global user.email “**[**your-email@example.com**](mailto:your-email@example.com)**”**

****

**Step 4: Verify Git Configurations**

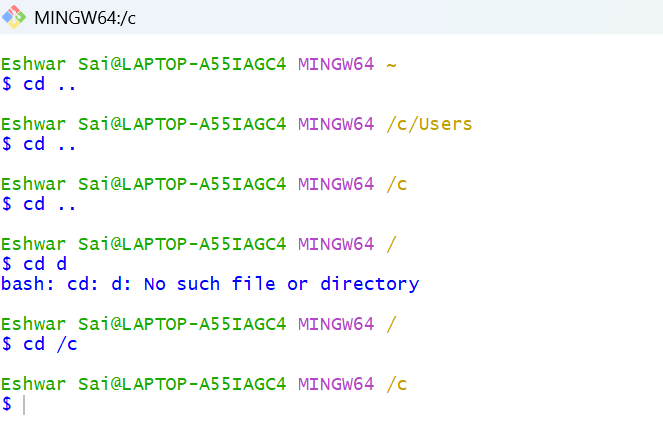
To check if the configurations were set correctly, run:

* **git config -–list**



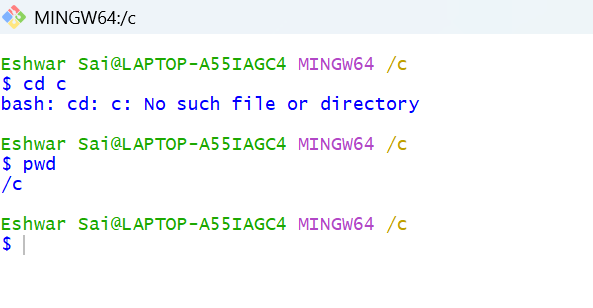
**Step 5: Change Directory**

Change directory (cd) to your preferred location using the **‘cd’** command.



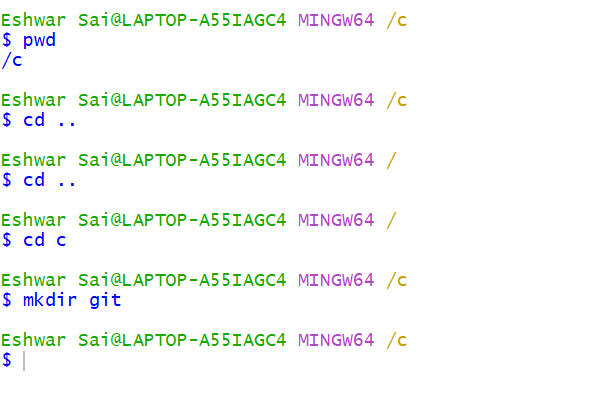
**Step 6: Print the Current Directory**

To print the full path of your current Directory use the ‘**pwd’** command.



**Step 7: Create a New Folder**

To Create a new folder in the Directory, use the command: **mkdir** folder-name.



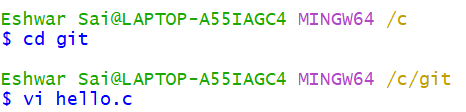
**Step 8: Listing the Files and Folders**

To Display the list of all files and folders in the current directory use the **‘ls’** command.



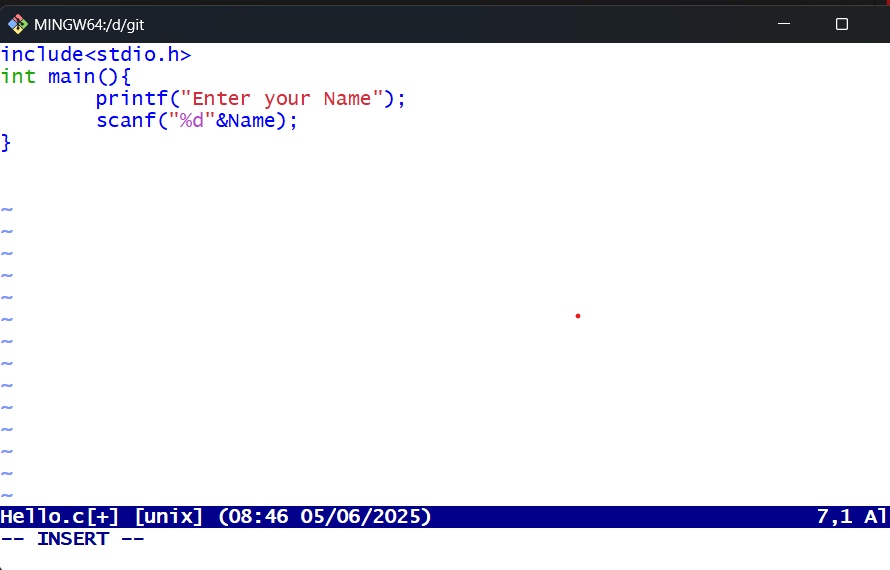
**Step 9: Creating a File Inside the Folder**

To create a C++ File inside the **Git** Folder, move inside the folder using the ‘**cd**’ command and then use **‘vi’** command to create a file.



**Step 10: Inside the VI Editor**

Once typed Git opens the **‘vi’** editor to create or edit a file named **Hello.cpp.** Press **i** to enter **INSERT** mode. Now start typing your code in the **vi** Editor.

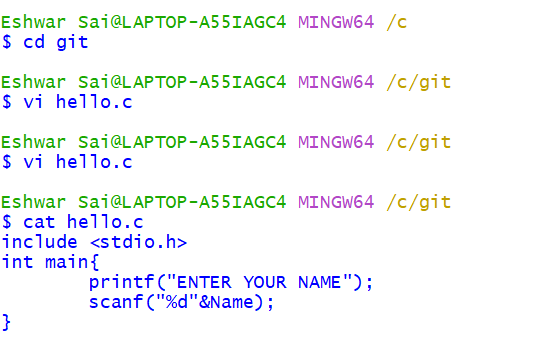


**Step 11: Exiting the VI Editor**

Once done with the code Press **ESC** to exit **INSERT** mode and type **:wq** and press **Enter** to save and exit.

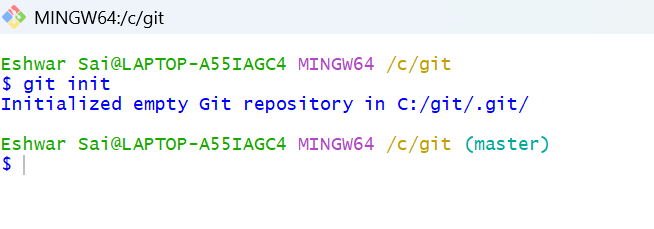
**Step 12: Display File Contents**

To Display the contents of the CPP File use the **cat** Command as: **cat** filename.extension.



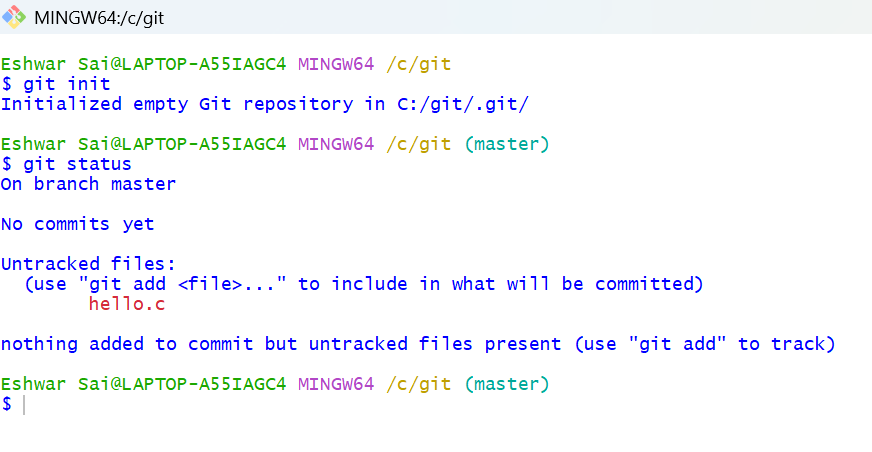
**Step 13: Initialize Git in Directory**

To turn the directory into a Git repository, run: **git init**



**Step 14: Check Git Status**

The **git status** command is used to check for **untracked files**, along with other changes in the repository. You should see Hello.cpp as an **untracked file**.

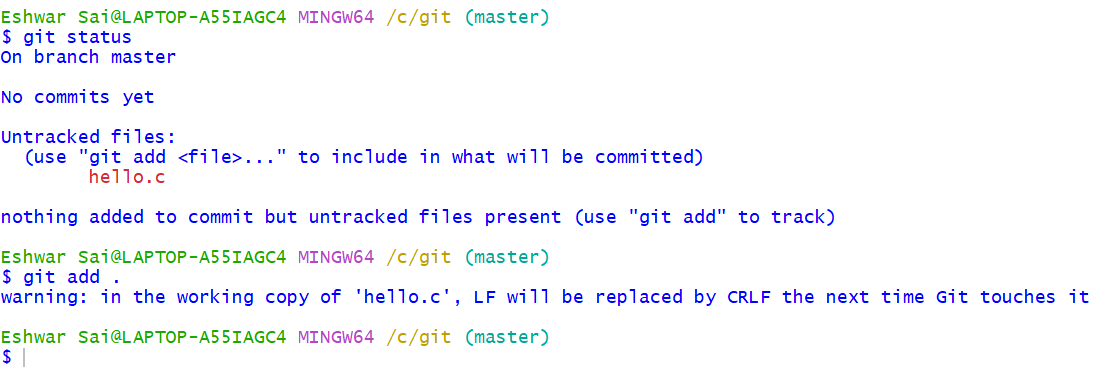


**Step 15: Add Files to Staging Area**

To stage all newly created and modified files use the command: **git add .**

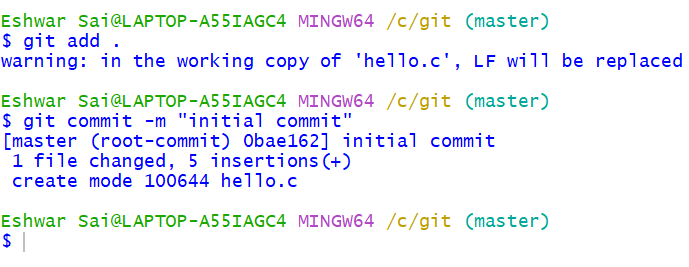
To confirm, check the status again using the command: **git status**

Now, all tracked files will appear as **staged**.



**Step 16: Commit the File**

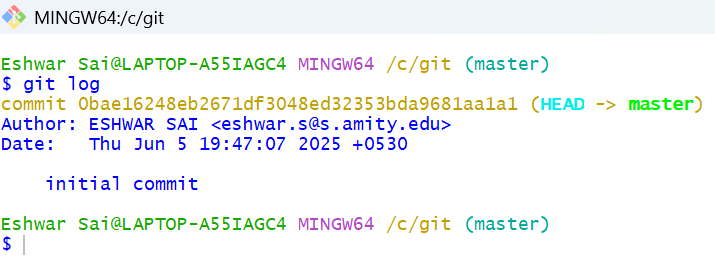
To save the changes in Git, commit the file with a message**: git commit -m "Initial commit: Added main.cpp"**

****

**LAB REPORT – 3**

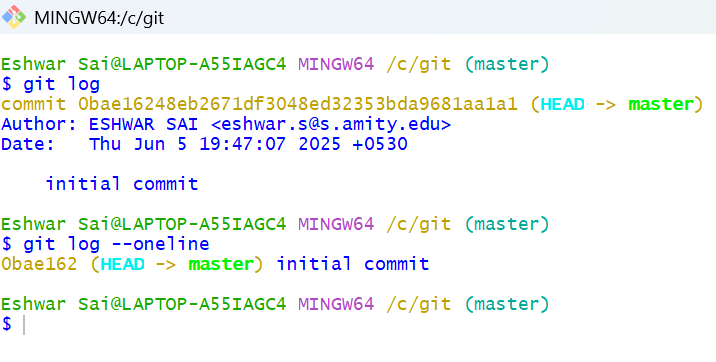
**Step 1: Check Git Commit History**

* The **git log** command displays the commit history in detail.
* It shows the commit hash, author, date, and commit message.



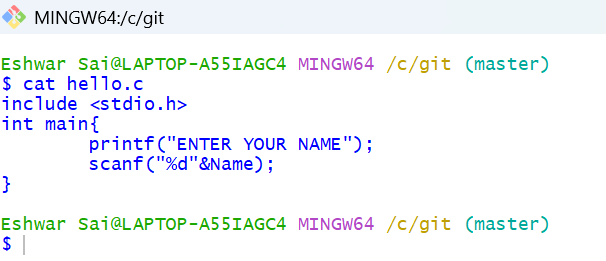
**Step 2: View Git Log in One Line Format**

* The **git log --oneline** command displays a compact version of the commit history.
* It only shows the commit hash and the commit message.



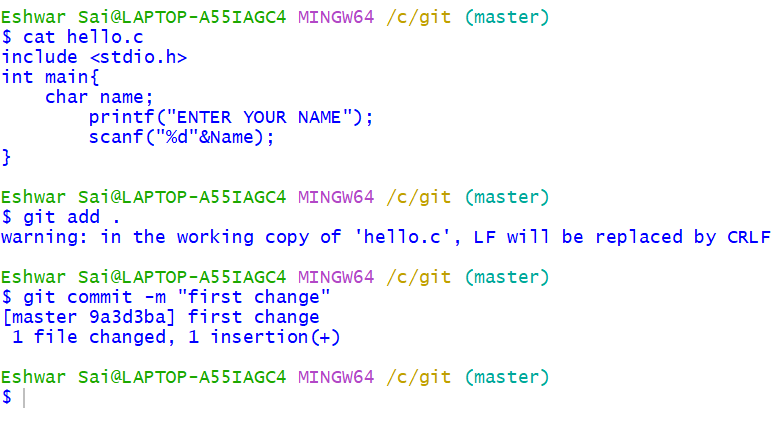
**Step 3: Modify the Hello.c File (First Change)**

* Open the Hello.cpp file in a text editor using the **vi** command.
* Make a small change (e.g., add a new function or modify a print statement).
* Save the file and display it using the **cat** command.



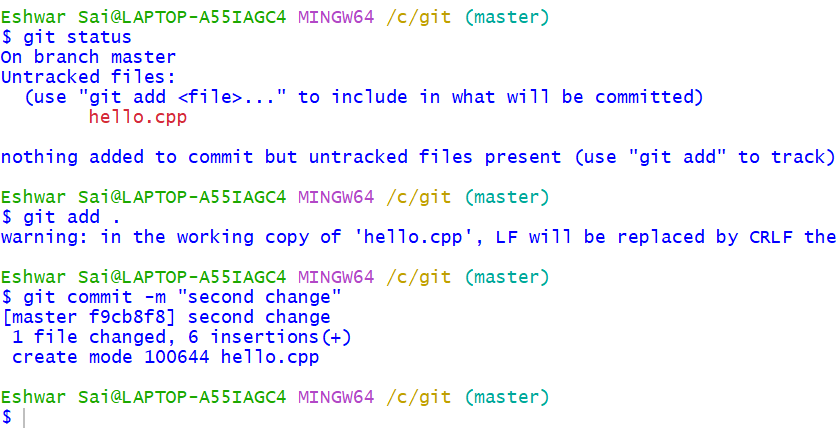
**Step 4: Stage and Commit the First Change**

Use **git add .** command to stage the modified file for commit **and git commit -m** to create a commit with a message describing the change.



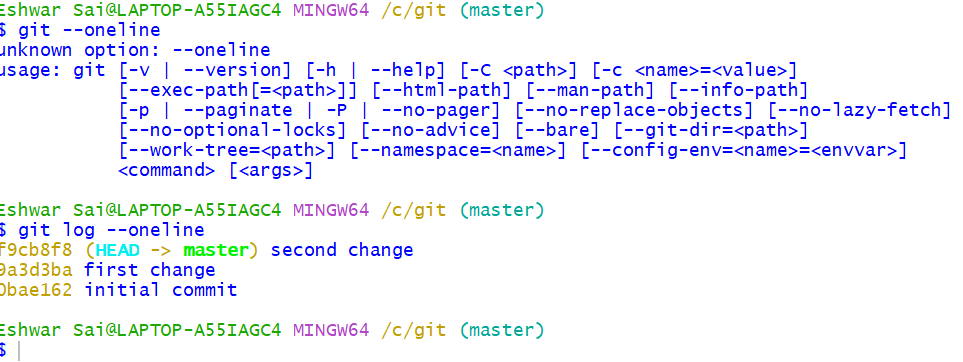
**Step 5: Modify the Hello.cpp File Again (Second Change)**

* Make another change in the same **Hello.cpp** file.
* Example: Modify a different function or add a new comment.
* Save the file and commit it.



**Step 6: View Git Log Again in One Line Format**

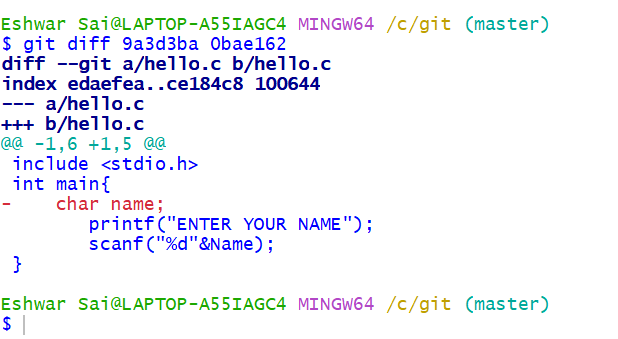
This will now show the latest two commits along with previous commits.



**Step 7: View Differences Between Commits**

The **git diff** command shows the exact lines changed between each commits. You can compare between multiple commits. Example: First commit and Second commit or Second commit and Third commit or even multiple commits.

This shows changes between the First commit and Second commit.



This shows changes between the Second commit and Third commit.

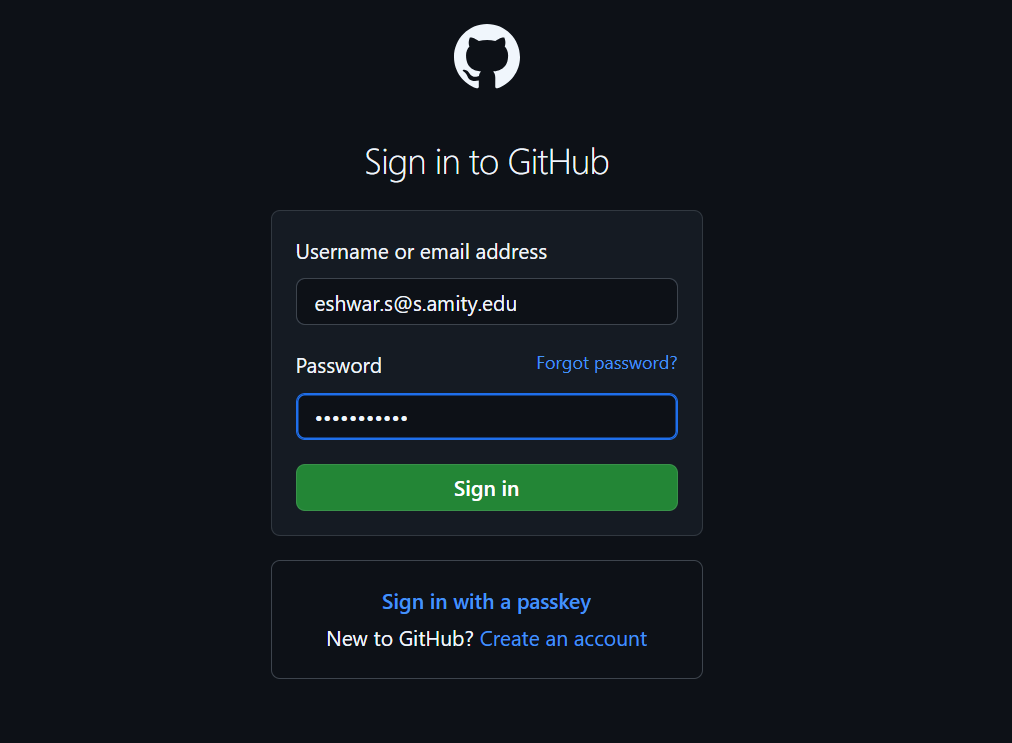
**LAB REPORT – 4**

**Step 1: Sign in to GitHub**

Open a web browser and go to [github.com](https://github.com/)

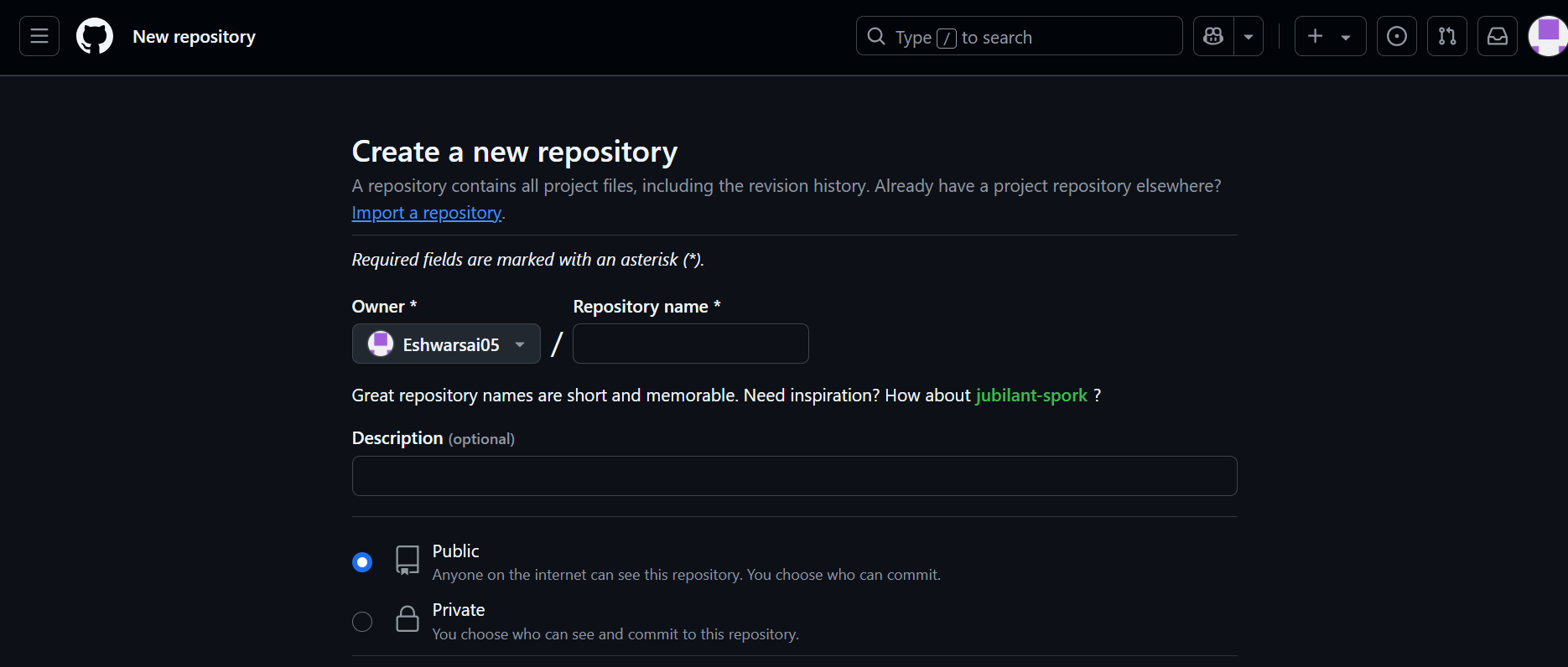
A screenshot of a computer

AI-generated content may be incorrect. Click Sign in and enter your credentials.



**Step 2: Creating a Repository**

Click on the **"+"** icon (top-right corner) and select "**New repository**".



In the **Repository name** field, enter the same name as your local folder. Select Public. **Do not** check "**Initialize this repository with a README**". Click **Create repository**

**Step 3: Connect Local Repository to GitHub**

On the next page, copy the **HTTPS URL** under **"Quick setup"** it looks like (**https://github.com/yourusername/repositoryname.git**).

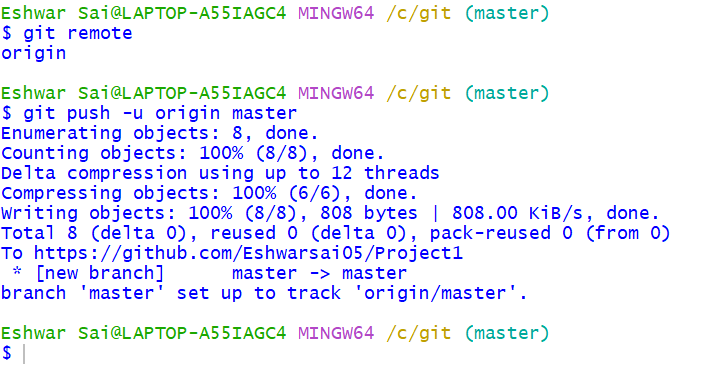
Add the GitHub repository as a remote:

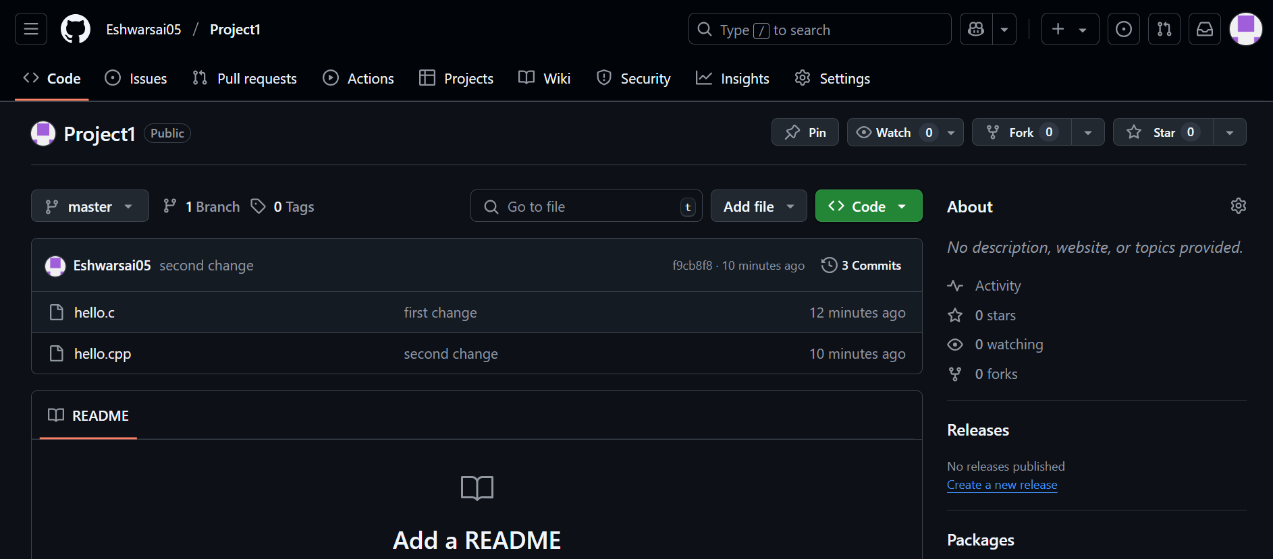
* **git remote**
* **git remote add origin <repository-URL>**



**Step 4: Push Code To GitHub**

Push the committed files to GitHub using the command: **git push -u origin master**





**LAB REPORT – 5**

**Step 1: Create a New Branch**

Use the following command to create a new branch named **dev** and switch to it:

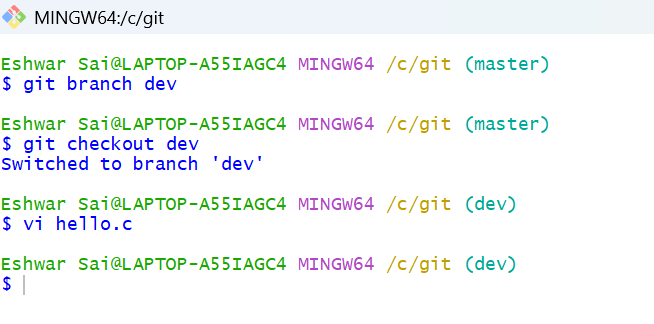
**Git branch dev**

**git checkout -b dev**

****

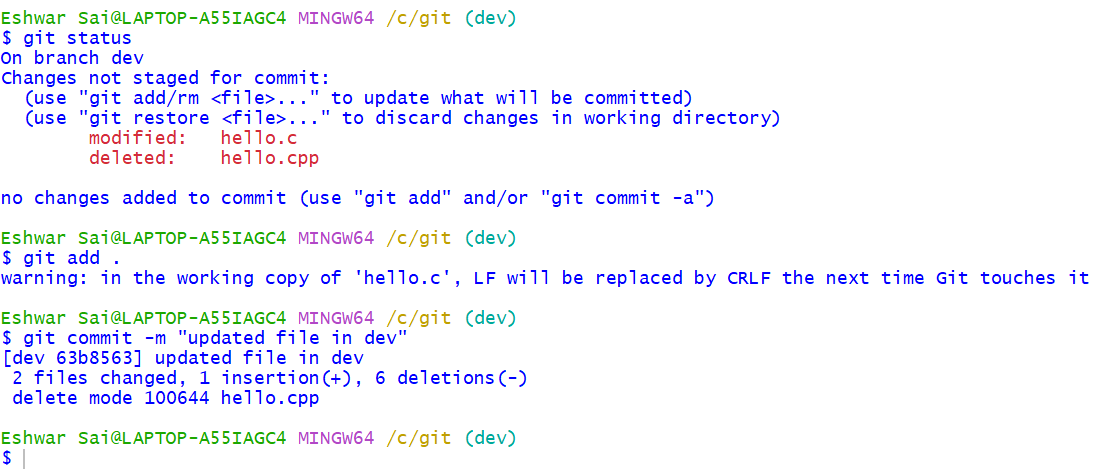
**Step 2: Make Changes in the dev Branch**

**Open the hello.cpp file and make some changes.**

****

**Step 3: Stage and Commit Changes**

* **git add .**
* **git commit -m "Added a new file in dev branch"**

****

**Figure – 3**

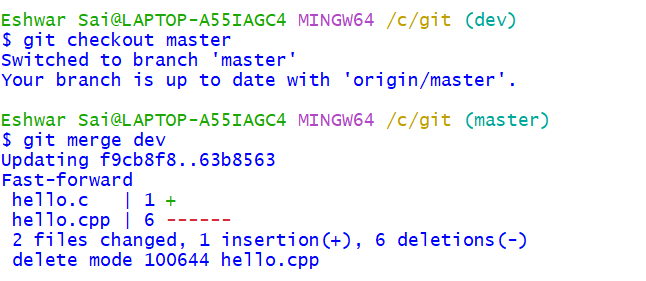
**Step 4:** Switch Back to master Branch

**git checkout master**

**Step 5: Merge dev into master**

If there are no conflicts, this will merge the changes from the dev branch into master.

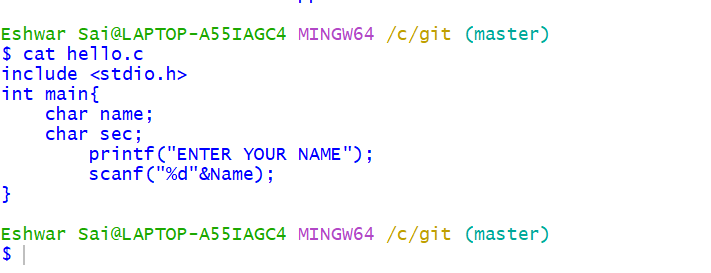
**git merge dev**

****

**Step 6: Verify the Merge**

Use cat command to check is the files are merged.

**cat hello.cpp**

****

**Step 7: Run the Git Merge Tool**

Use git mergetool to open the conflict screen. Close it using **escape :wqa**

**git mergetool**

