**STATE UNIVERSITY OF INTELLECTUAL TECHNOLOGIES AND TELECOMMUNICATIONS**

**Department of software engineering**

**Laboratory Work 2**

**Exploring GIT**

Prepared by

Student of S.E-3.01

Rodion Karakonstantin

Odesa 2024

# Objectives:

# ● To master the basics of working with the version control system Git.● Create and conﬁgure a project using the React and TypeScript template.

# ● Learn to use various tools for working with Git, including the command line and the graphical user interface of the IDE.

# Tasks:

# 1. Tutorial Review:

# Study the Git tutorial available at Git How To. Pay special attention to the ﬁrst 21 steps, which cover the basic operations with Git.

# 2. Choice of Development Tool:

# For this assignment, you may use one of the following Integrated Development Environments (IDE): WebStorm or Visual Studio Code. Ensure that the IDE is installed and conﬁgured to work with Git.

# 3. Using the Command Line:

# Use the command line to execute the Git operations described in the tutorial. This will help better understand the version control system process.

# 4. Using the IDE's Graphical Interface:

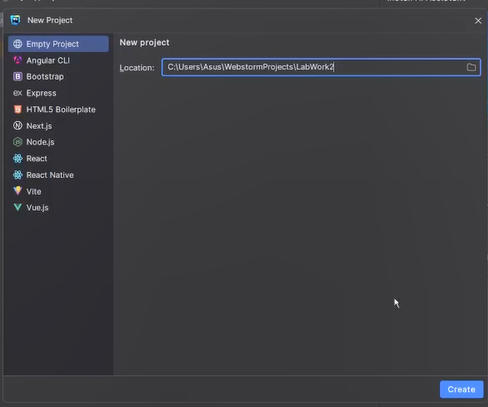
# Use the built-in features of the graphical interface of the chosen IDE to track changes in the repository. Demonstrate visually the results of the commands performed in the IDE for each step of the tutorial.

# 5. Visual Conﬁrmation of Changes:

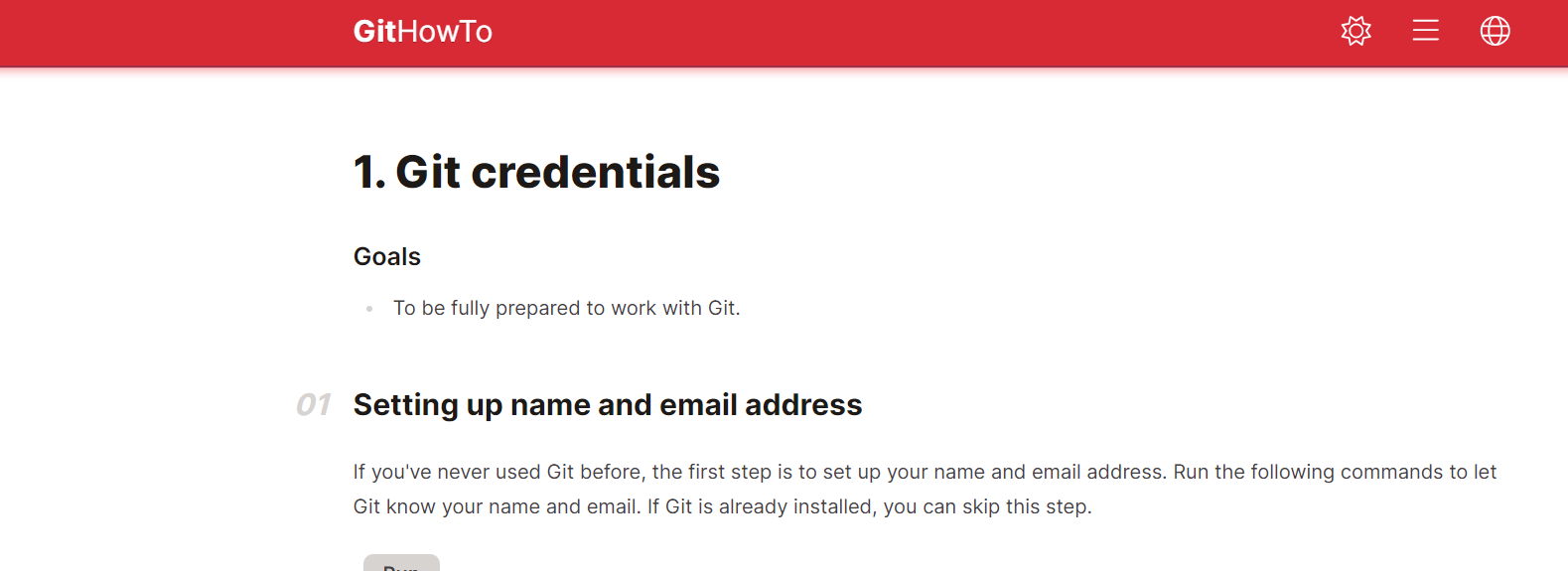
# Provide screenshots or other visual conﬁrmations of the repository changes made for each of the ﬁrst 21 steps of the tutorial, using the graphical interface of the IDE.

**SOLUTION:**

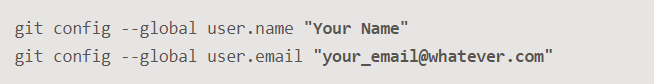
1) So firstly, we create a new project specifically for this lab task



2) Then we go to the site of GitHowTo, and follow steps to get familiar with it



3) Then we open the terminal go step by step and input commands





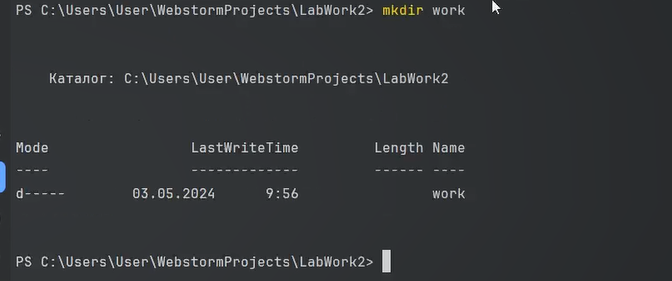
4) We get the default braunch name



5) Line ending treatments



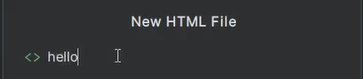
6) We get to the 2n STEP and now create a subdirectory named “work”



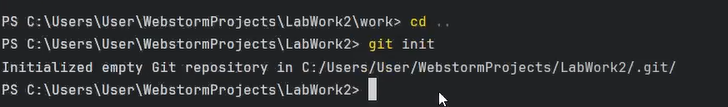
7) Then we move to file work



8) Now we create a file



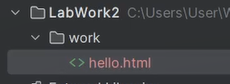
9) Now we move to main directory and create an empty repository



10) We have a hidden file .git

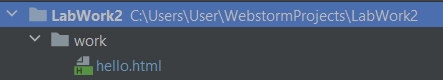


11) We can see, that the file is red



But if we add the file under GIT, then it stands as green





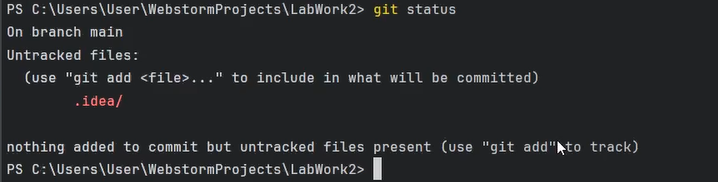
IDEA says that it is under the git

12) Committing changes

-m about messages

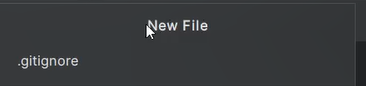


13) We check the status of git

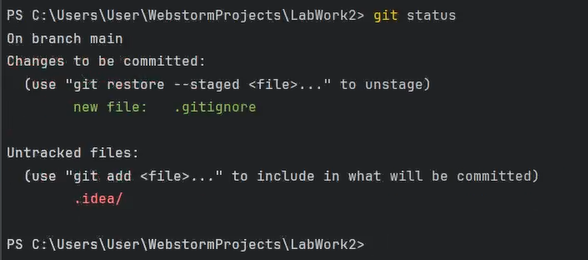


It says it has hidden file. Such files should not be committed because it is about configuration in your local environment. These file should be ignored.

14) Create a new file

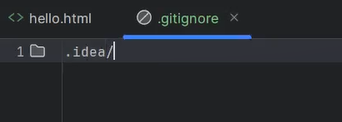


Check again the git status

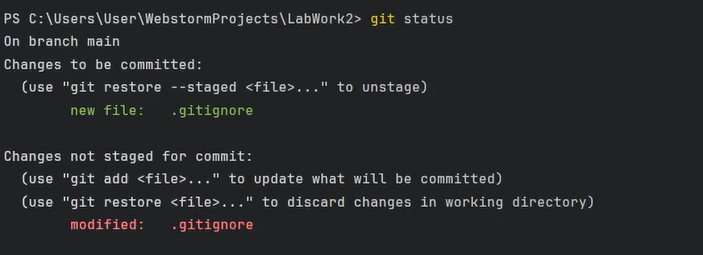


We have need file there

15) We add a row in this file with folder name “.git”

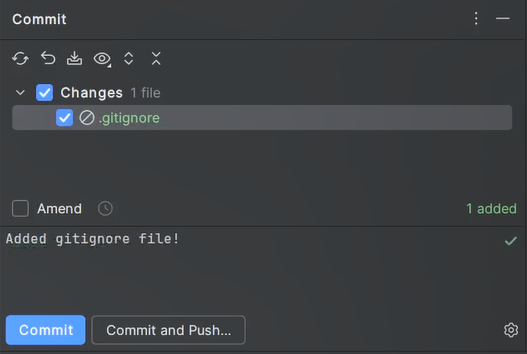


We again check the status

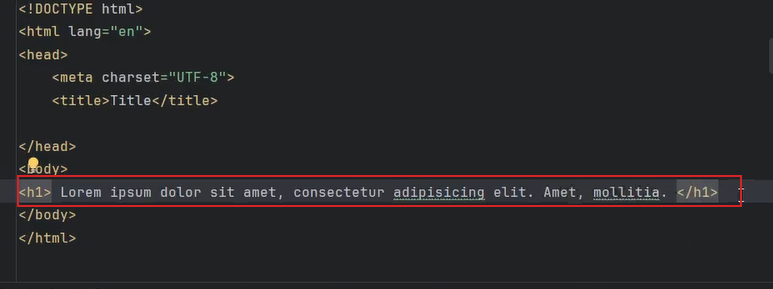


This file is ignored by GIT

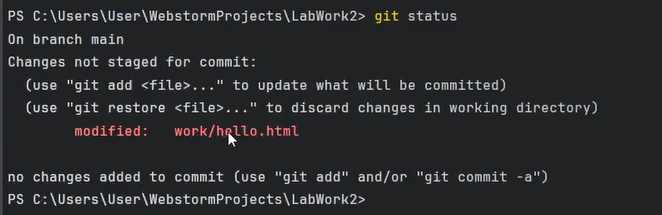
16) Then we commit this file



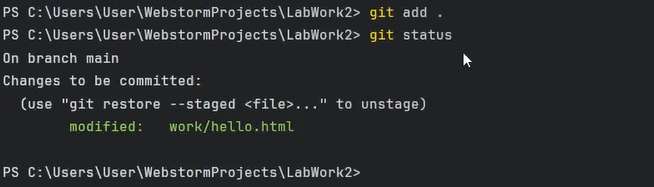
17) Then we change the file hello.html by adding some tags

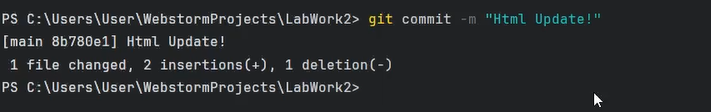


We run the git status, and see, that the file is modified

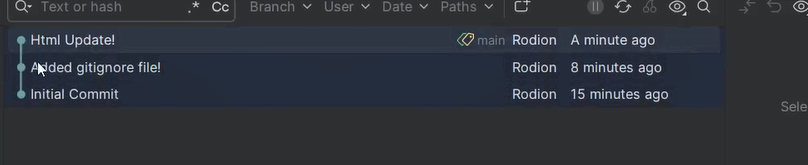


Then we commit the changes

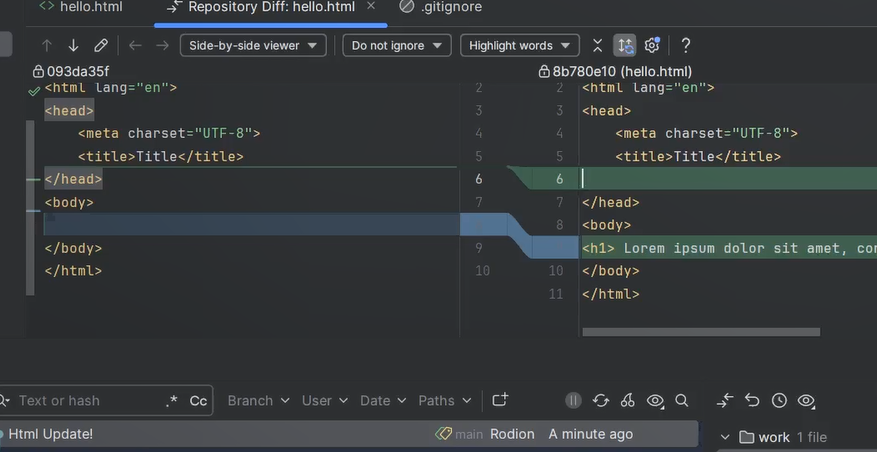




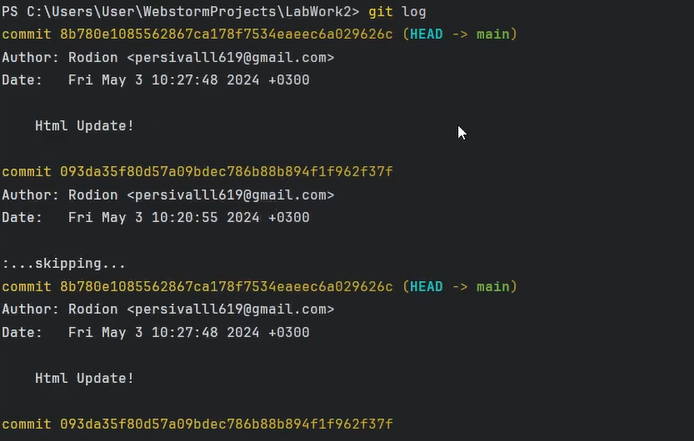
Here we have a git history



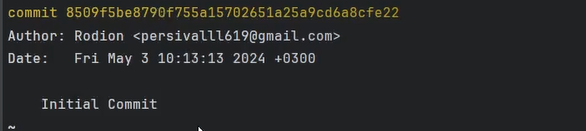
By double-click we see the changes we have inputted in the file

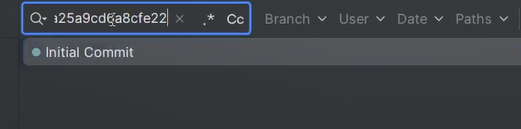


18) We can see the list of commit, we have already done. Previously we did it as UI, but now with command

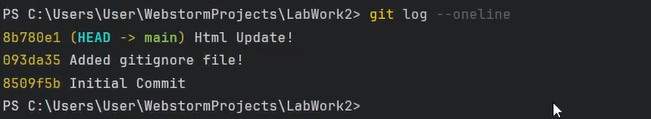


We can copy this commit hash, enter it in UI of GIT, and find the specific commit

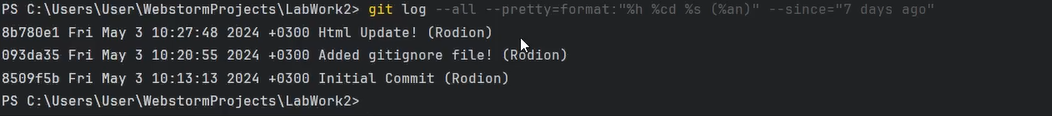




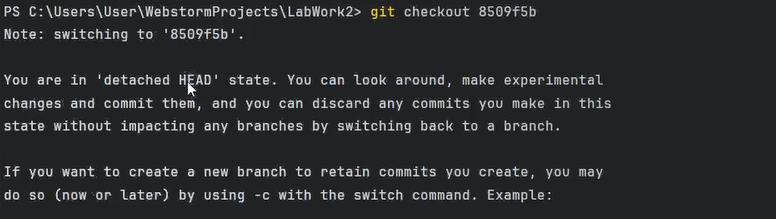
19) We can see the one line history of commits



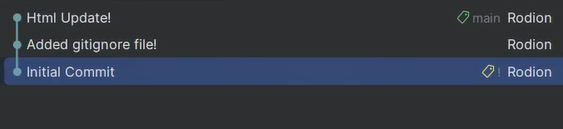
19) We check the changes made within the last week for example



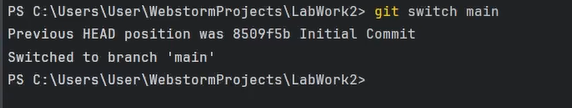
20) We input the following command. It is about changing current git repository state to special commit

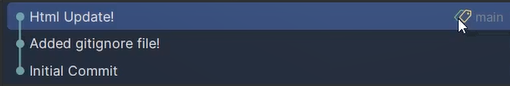


We go to the UI GIT, we can see yellow index. It is head. Current state of repository

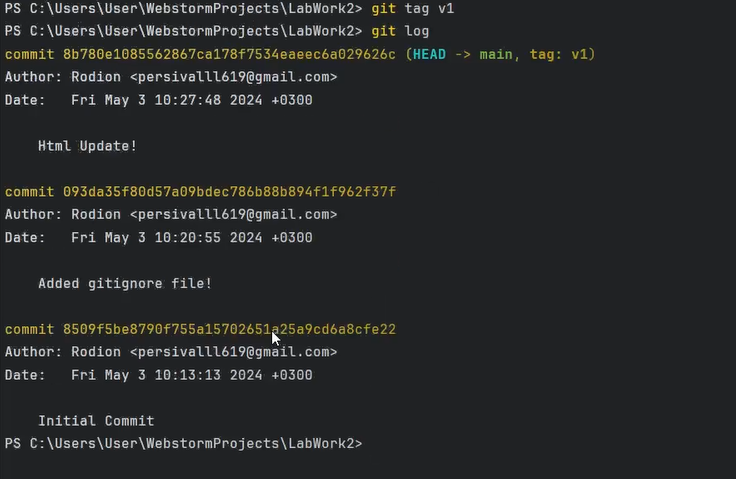


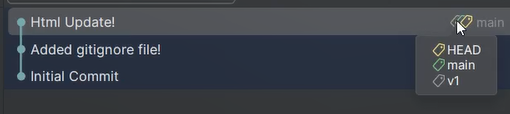
21) We now can switch between branches





22) We can add tags. This time for like 1st version



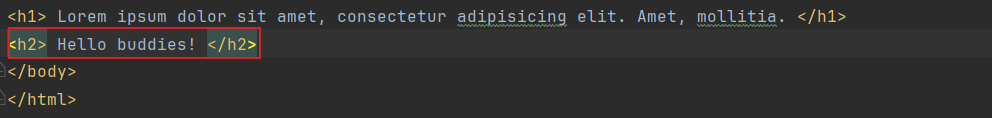


23) We can see, what tags we generally we have



(v0 was created as well, but manually in UI GIT)

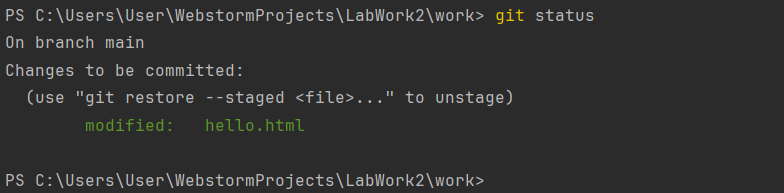
24) We get some changes in hello.html file



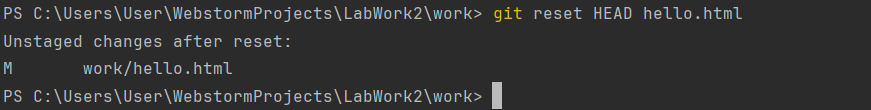
We stage the modified file



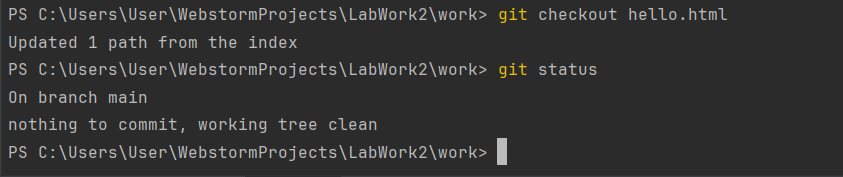
Then we check git status



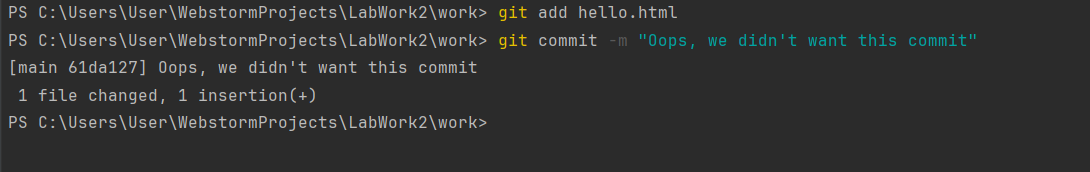
25) Then we reset the staging area to HEAD. This clears the staging area from the changes that we have just staged.



26) We switch to commit version

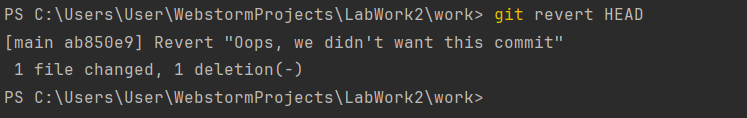


27) We add some changes to file



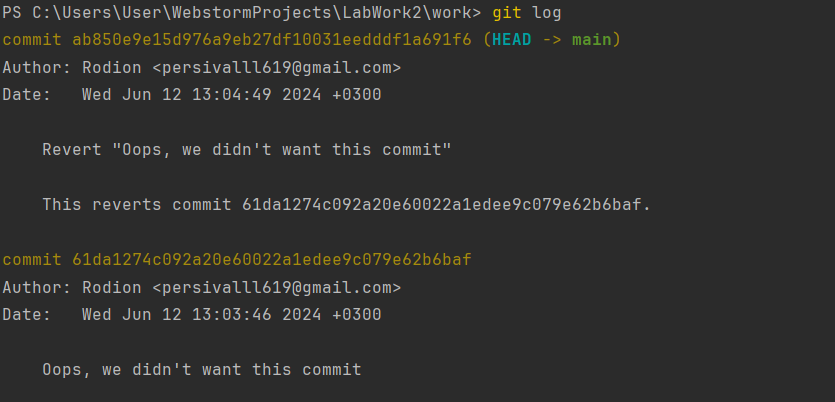
Now we cancel commit. We save the file in editor and see what happens next



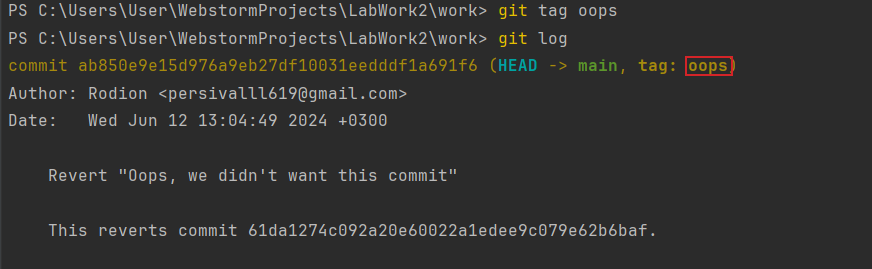


Now we do not have any changed in the file.

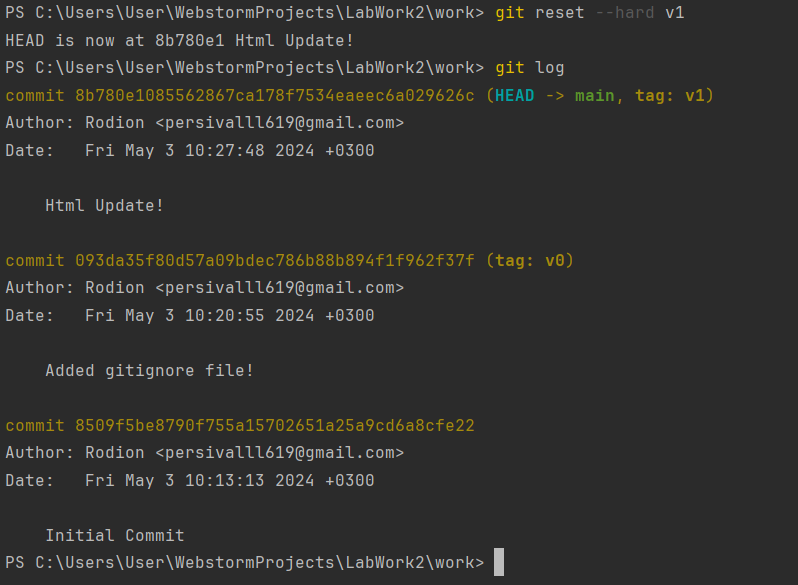
28) We check the log. It shows the unwanted cancellations and commits in our repository.



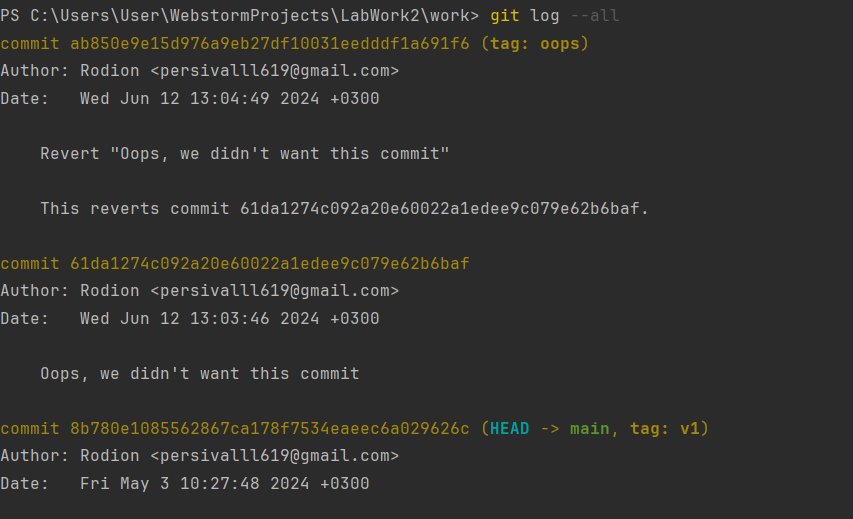
29) We mark last commit with tag, so we can find it after removing a commit(s).



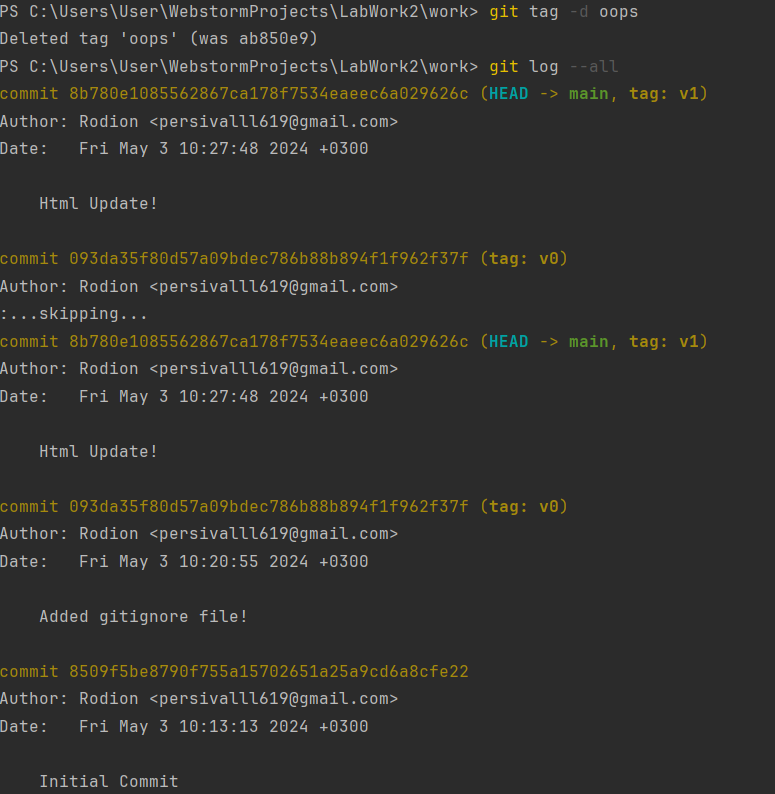
30) In the history log above, the commit tagged v1 is before the "Oops" and "Revert Oops" commits. Let us reset the branch to that point. As the branch has a tag, we can use the tag name in the reset command



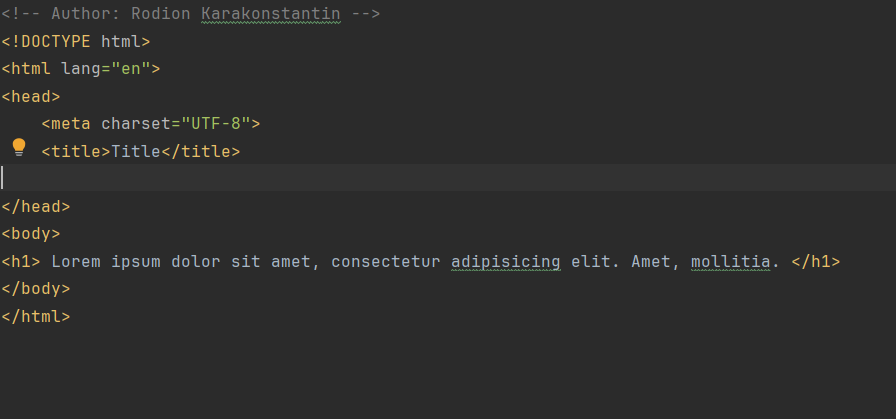
31) We can see all the commits. Even the wrong ones, they are not gone.



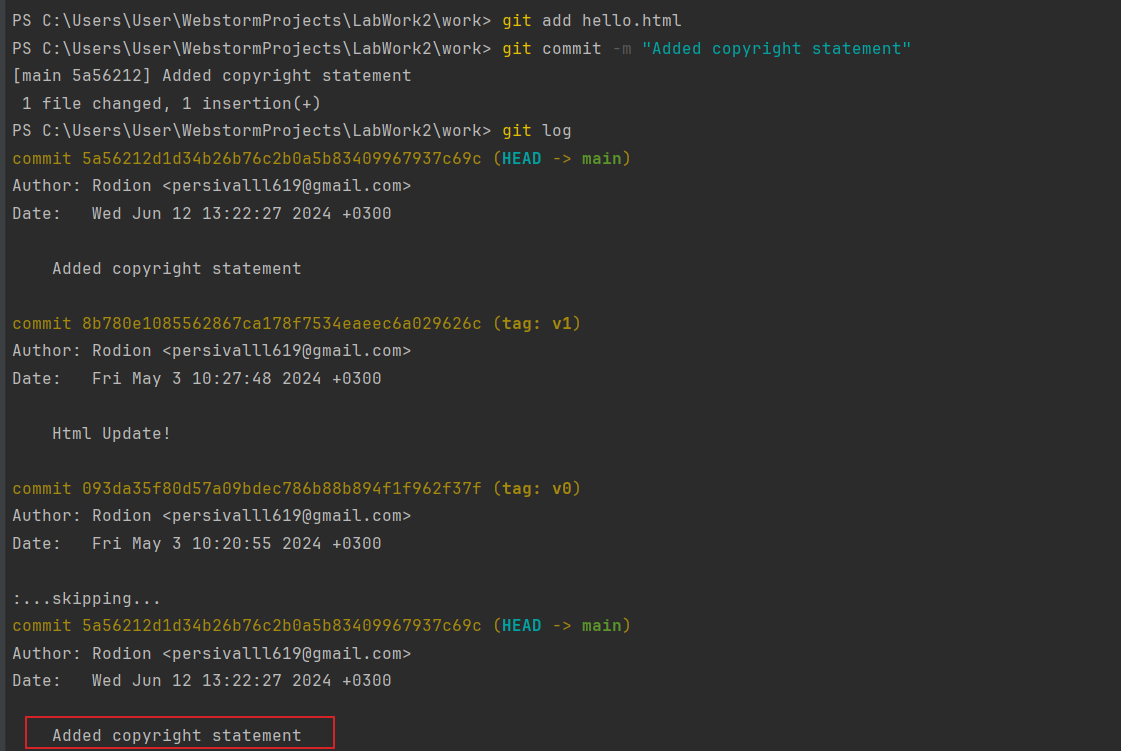
32) Now we remove the oops tag.



33) We put the author in our file



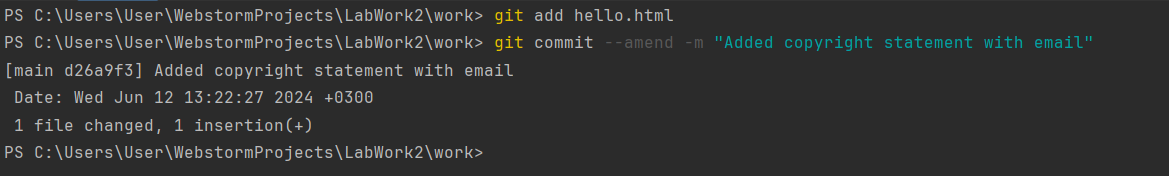
We commit it and check all the commits

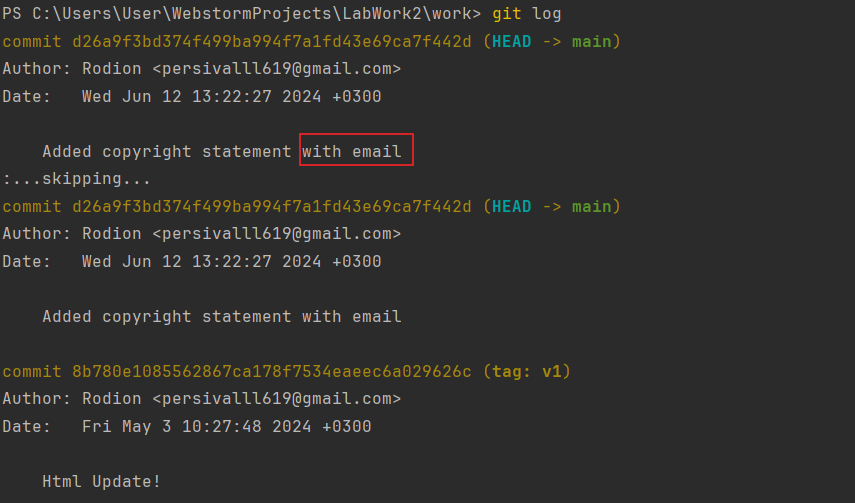


34) After making the commit you understand that every good comment should include the author's email. Edit the hello.html page to provide an email.

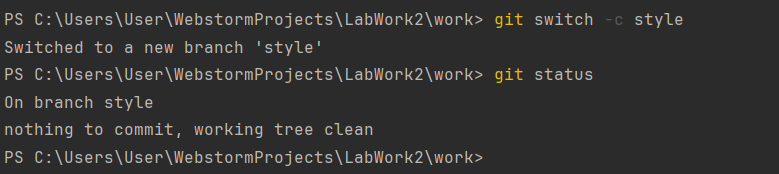


Now we change the previous commit

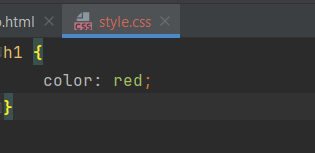


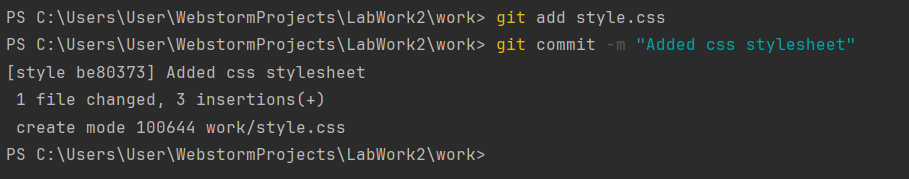


35) We change the branch

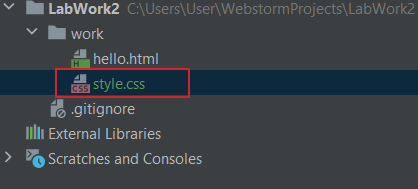


36) We add the stylesheet





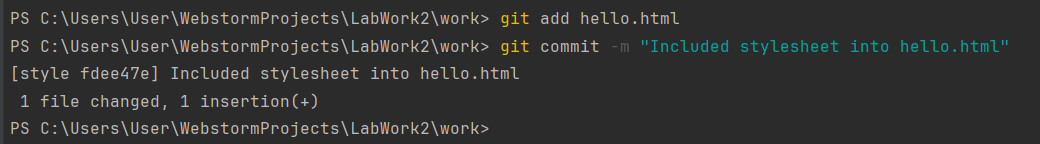
Now it is green, because we added to git

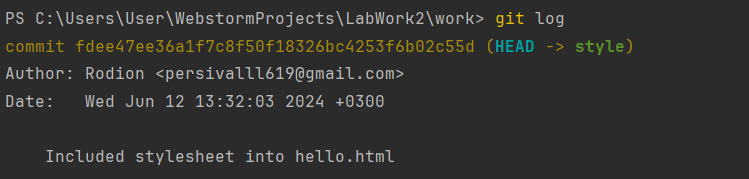


37) We change hello.html to add style.css

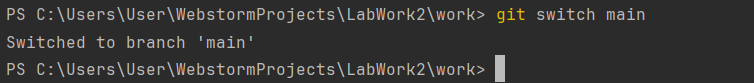


Now we add commit.

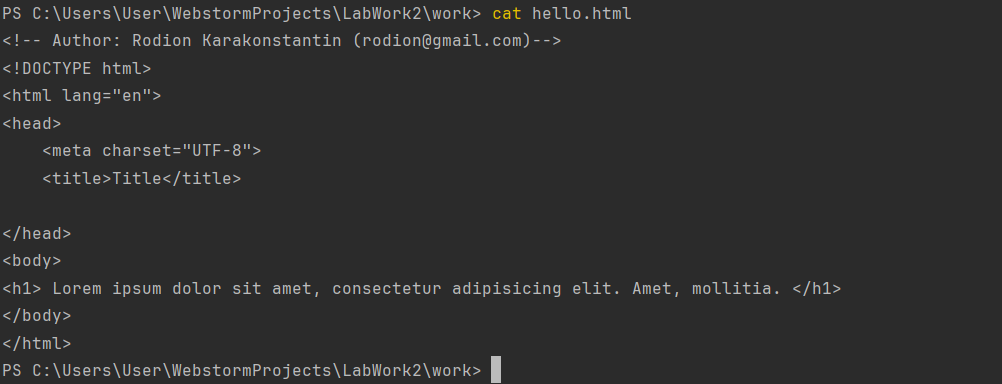




38) We switch between branches



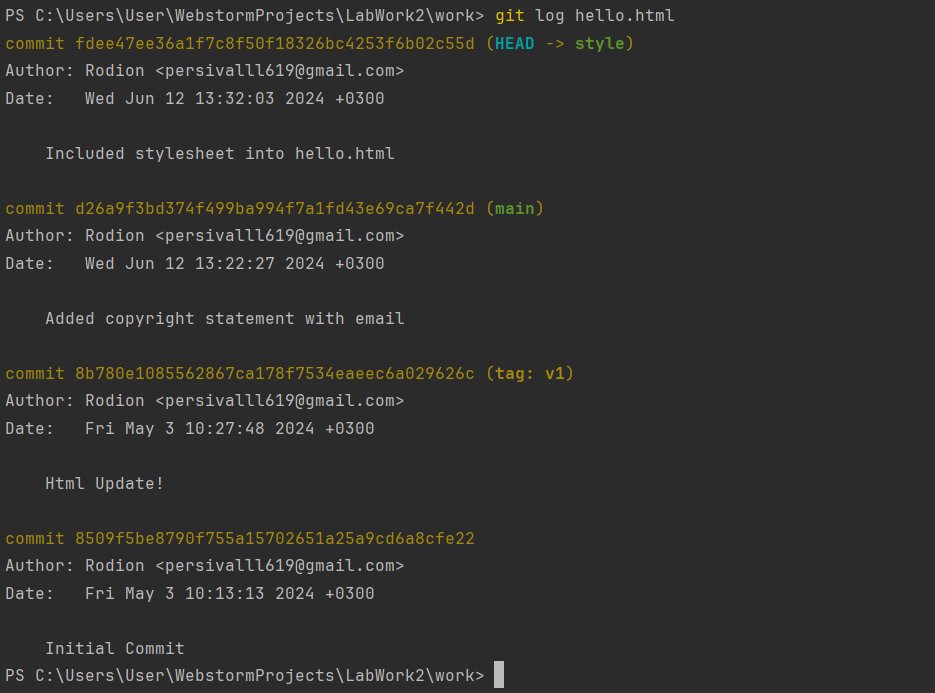
We can see, what is in the file

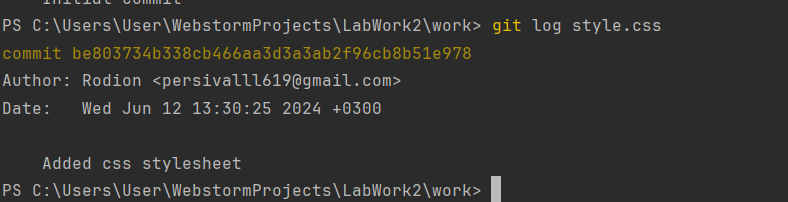


39) We switch back to style branch and check what is the file hello.html

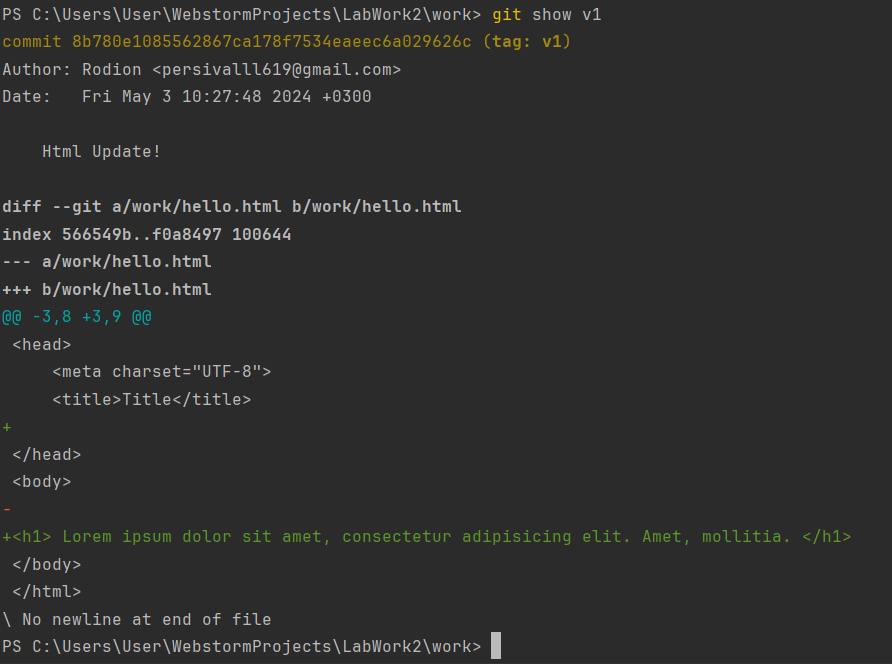


40) We have a look at the change of log.

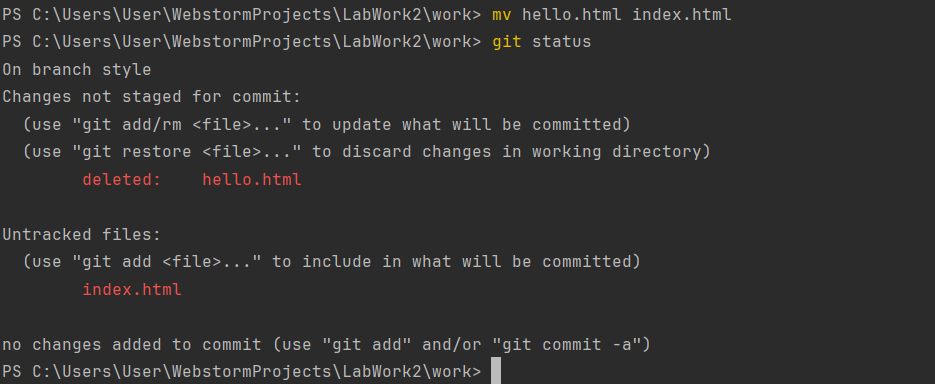




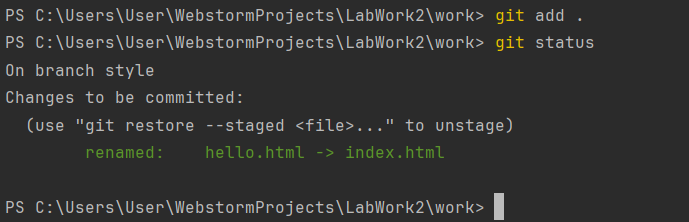
41) We display the changes in specific commit (v1 here)



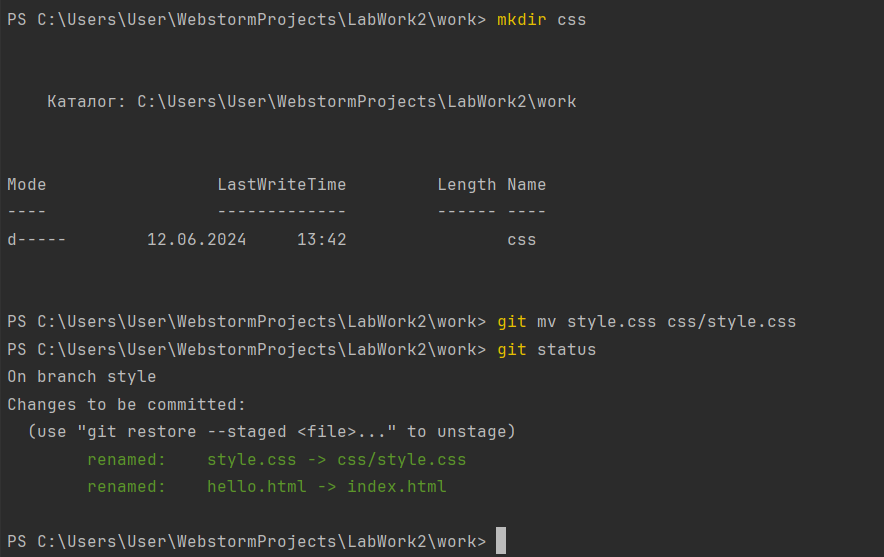
42) We change the game of our hello.html



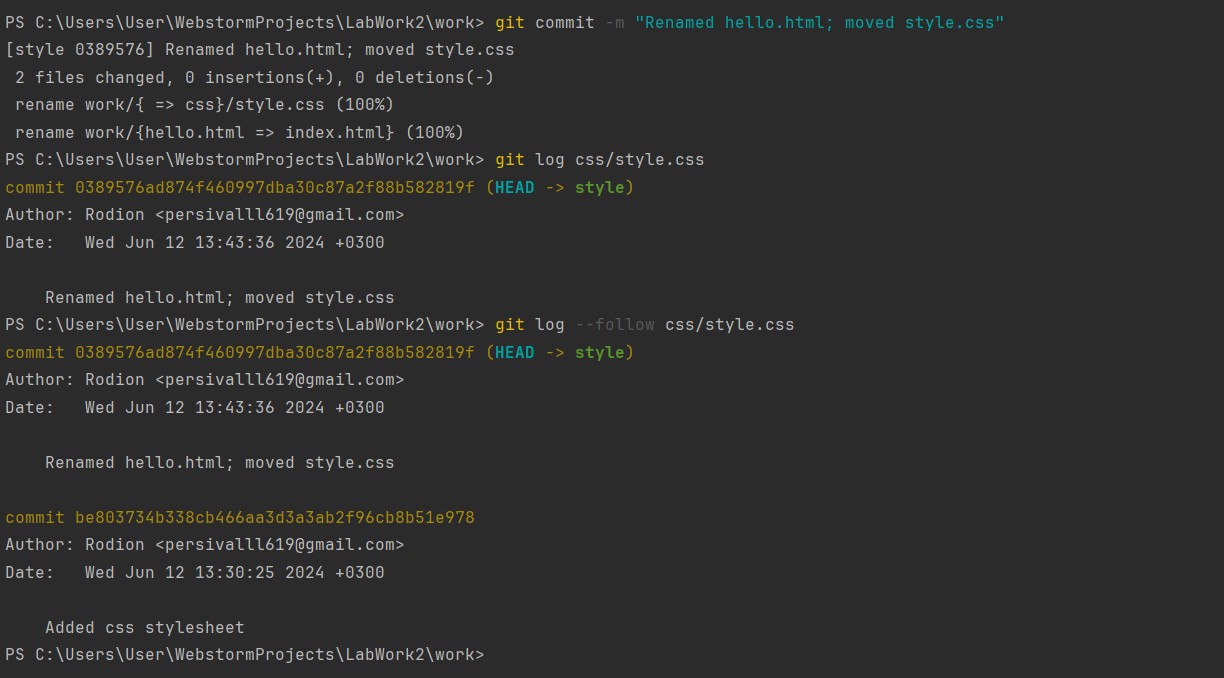
We add the commit and check status



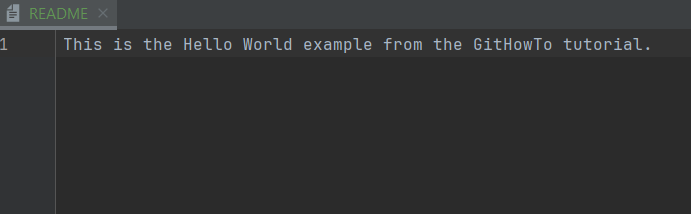
43) We change the name to stylesheet



44) We commit our changes and examine the change history of the css/style.css file. To see the file's history prior to its relocation, we'll need to include the --follow option.



45) We create a file README



We switch to main and commit



I have inspected 21 steps.