Dang Bao Nghi Nguyen

COURSE: SOFTWARE DEVELOPMENT PROJECT

ASSIGNMENT 4

GIT REPOSITORY

2020



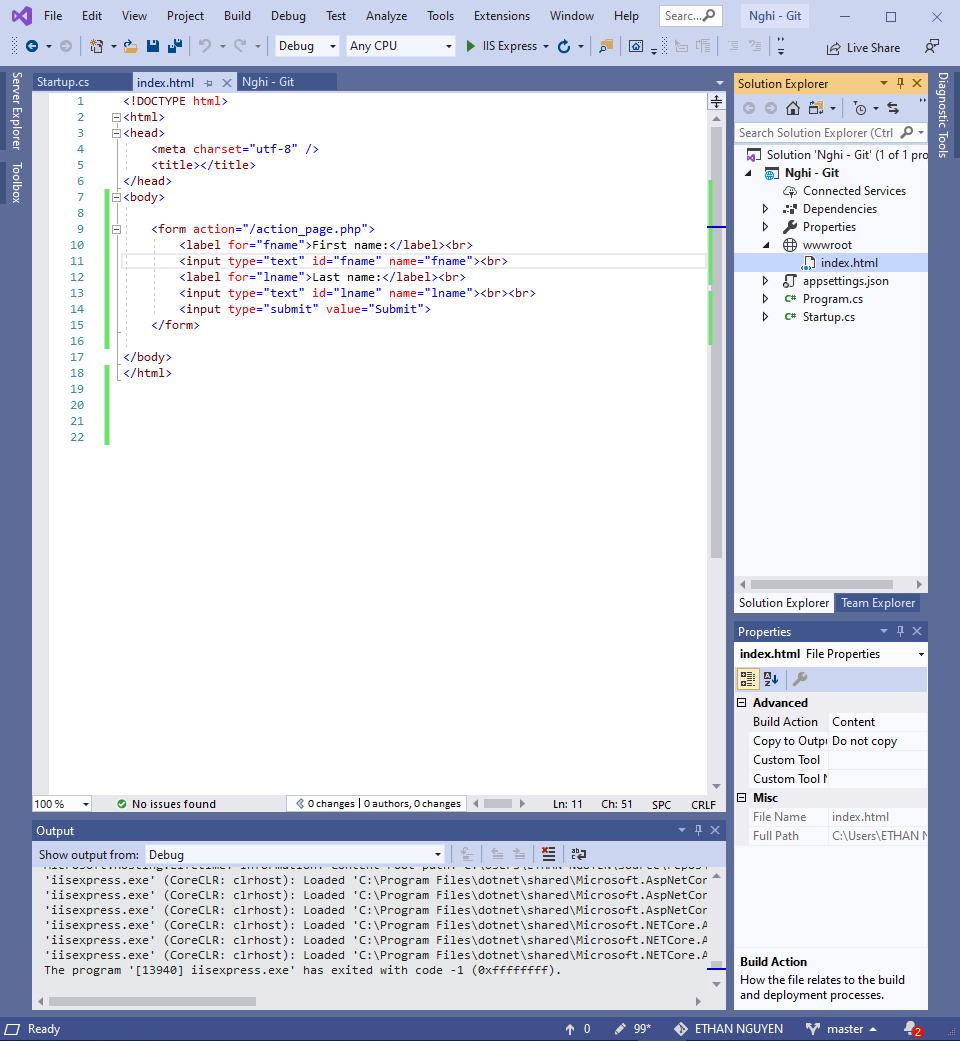
**CONTENTS**

1. **Procedure 3**
   1. **Create the Project 3**
   2. **Adding Project to Git Source Control 4**
   3. **Modify the Project 4**
   4. **Commit to Git 5**
   5. **Modify the Project Again 5**
   6. **Commit to Git 6**
   7. **Add new Branch 7**
   8. **Modify new Branch 7**
   9. **Merge to Master 8**

**1.10. Git Conflict 8**

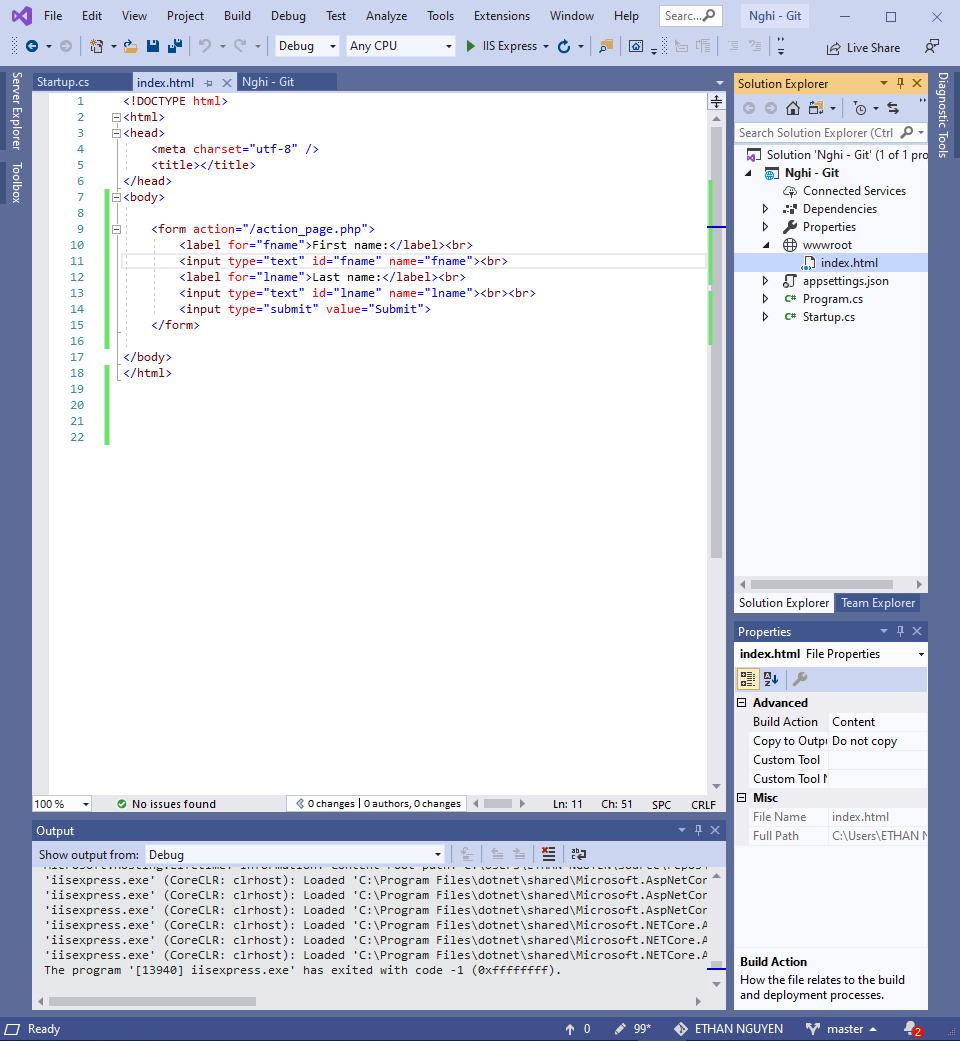
1. **Procedure**
   1. **Create the Project**

There are various ways of demonstrating **Git Repository**, in this case we will be using **Visual Studio 2019**. We first open **Visual Studio 2019** and create any type of web application we want. Something like this will do:



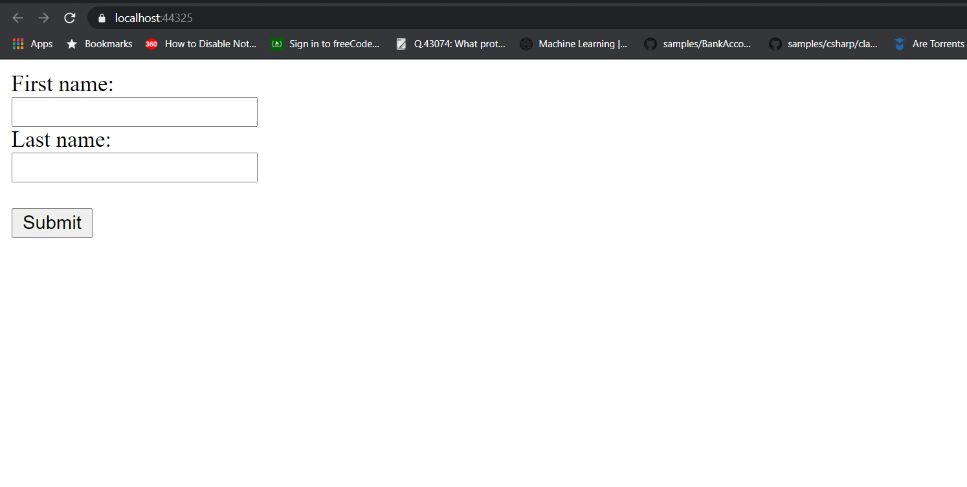
* 1. **Turn into Git Repo**

After we have successfully created and modify the project into our desire, we can now add the project to **Git Repo.** To be able to add the project into the **Git Repo**, look at the bottom right corner, we can easily connect our project with **Git Repo.**

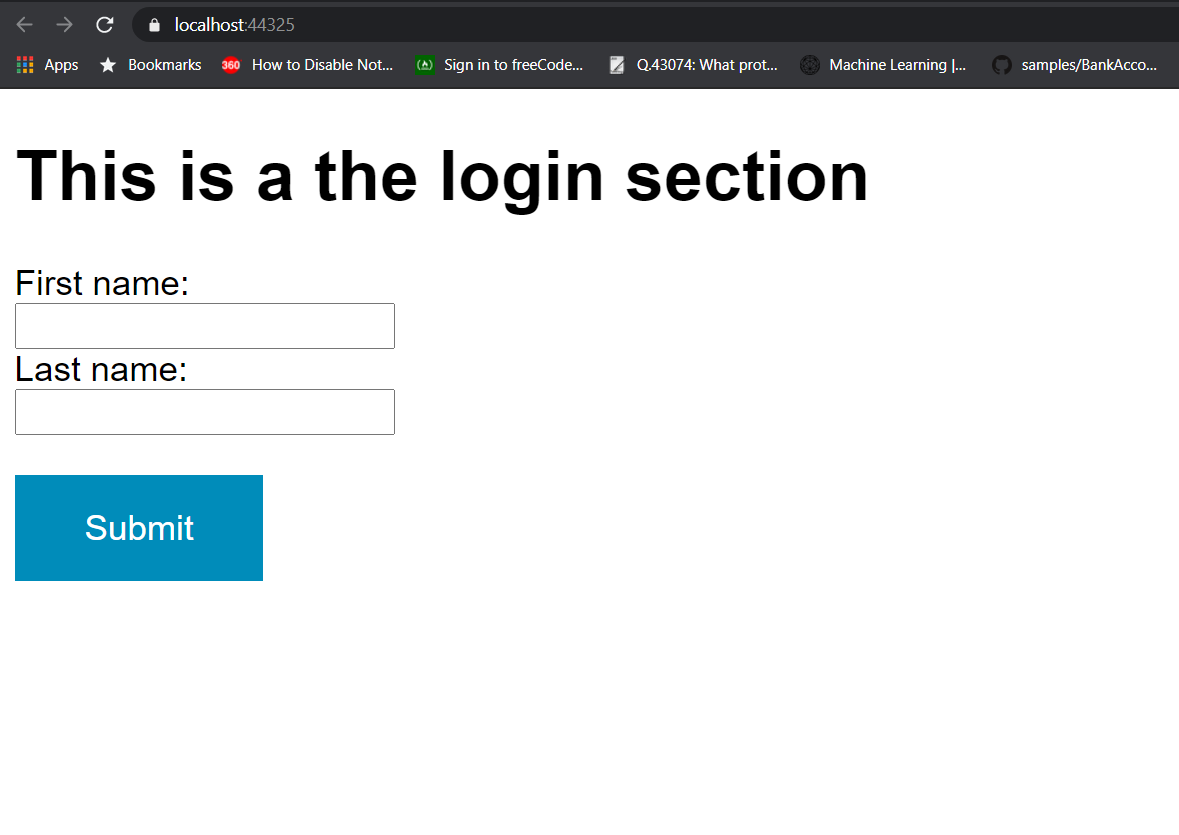


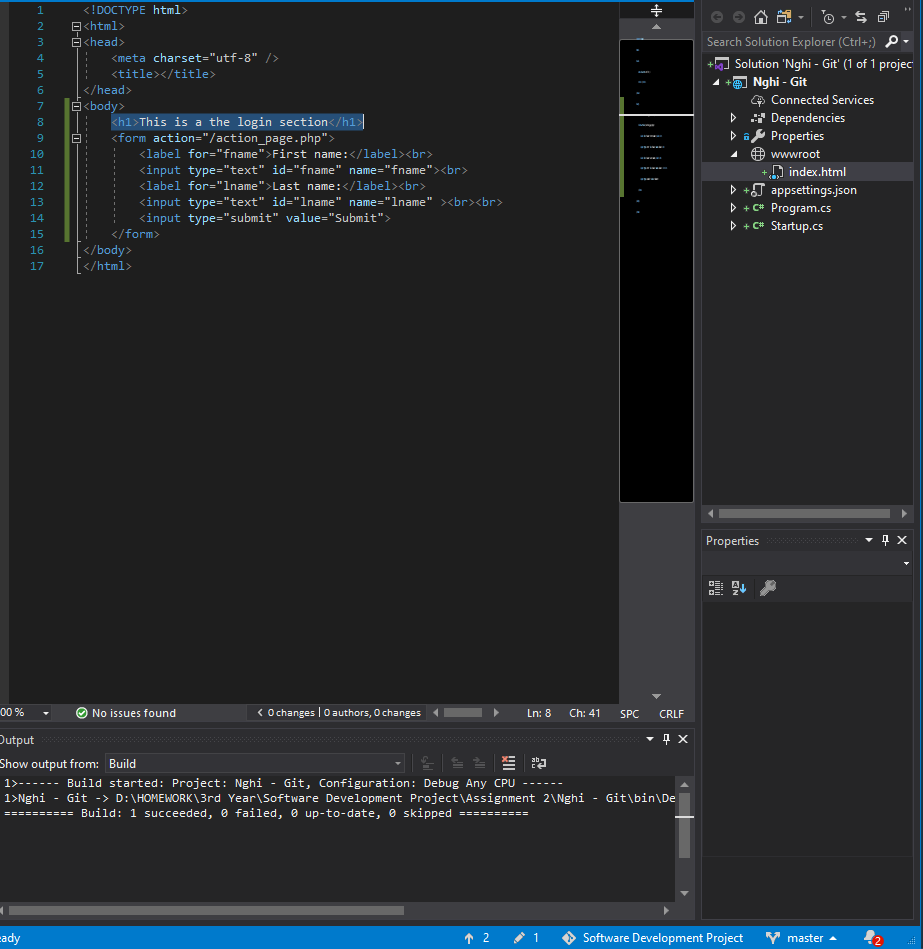
* 1. **Make Changes to the Project**

Let us first test out our new project to see if its work or not, the result should be expected to be working properly.



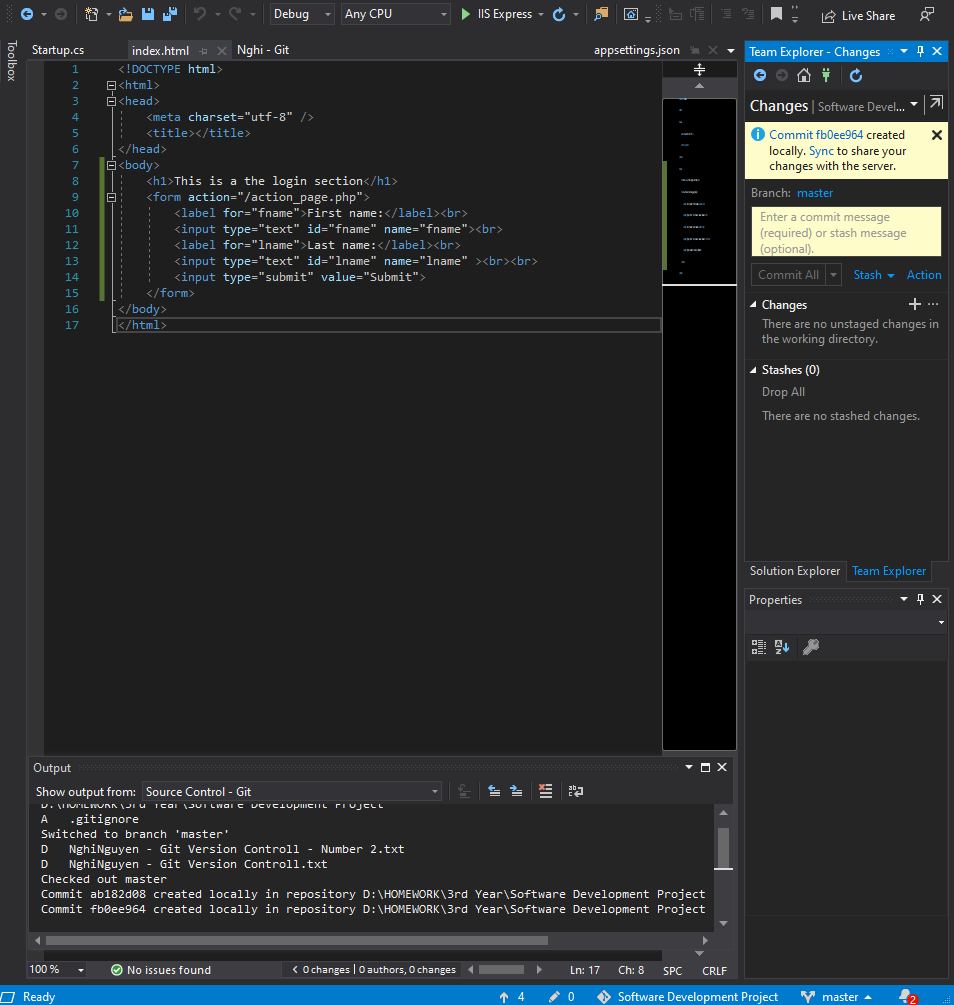
Now we try to modify the project by adding some new context, something like, in this case will be a new header, says “**This is the login section”**. The result should be expected like this:

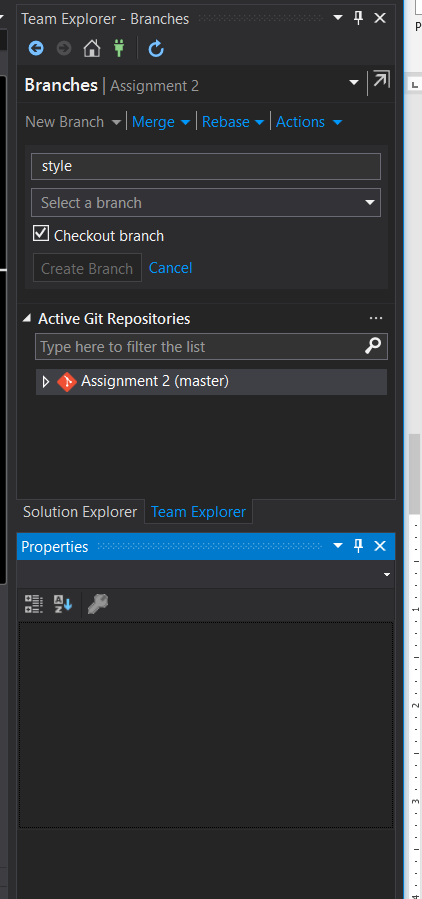


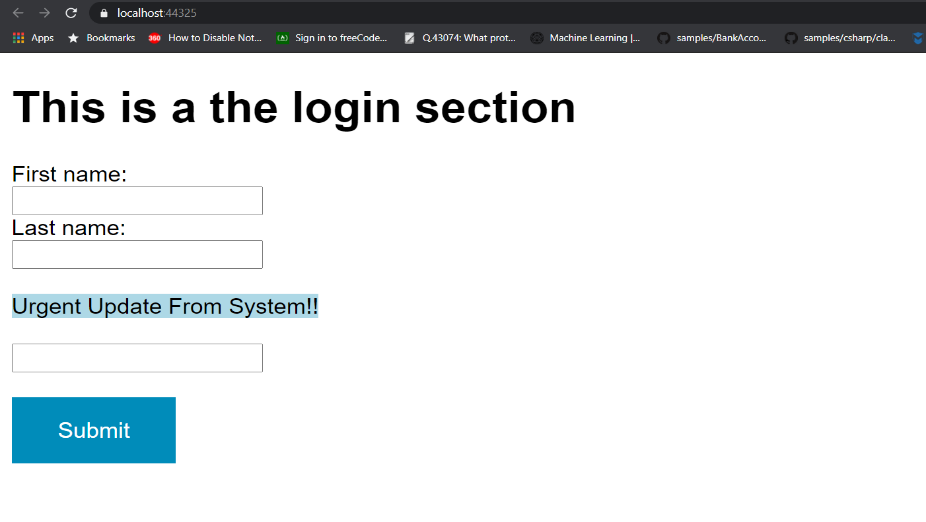


* 1. **Commit to Git**

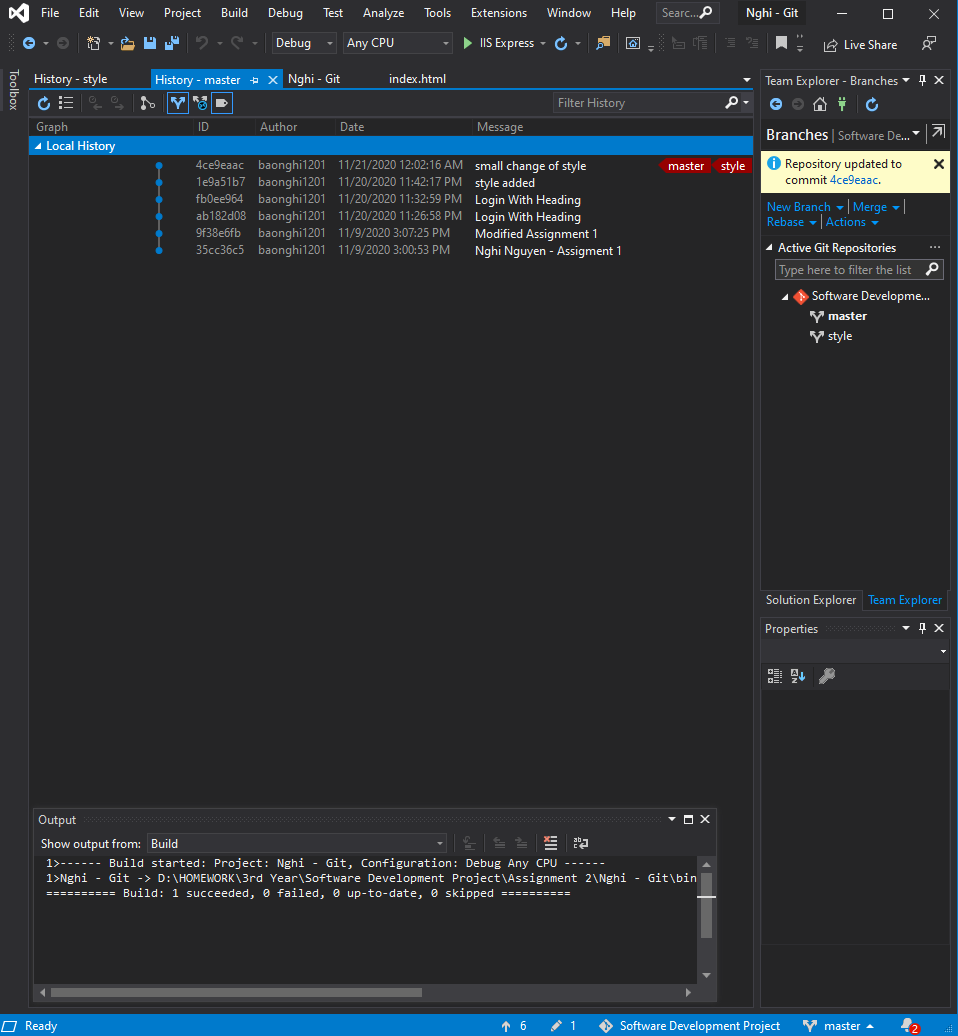
Now we do a **Git commit** to apply the project into **Git Repository**.



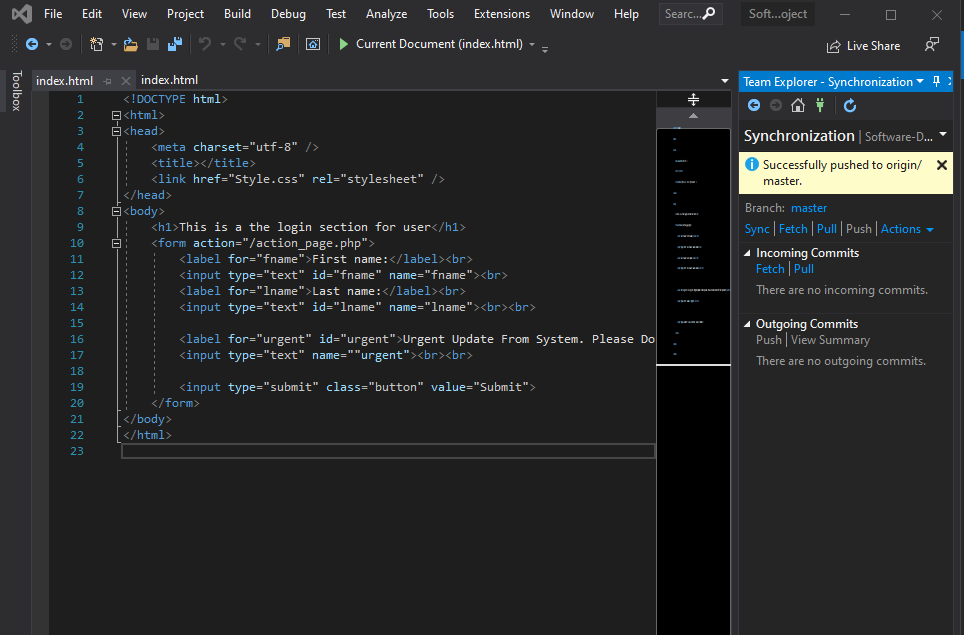
* 1.  **Add New Branch**
  2. **Modify new Branch**

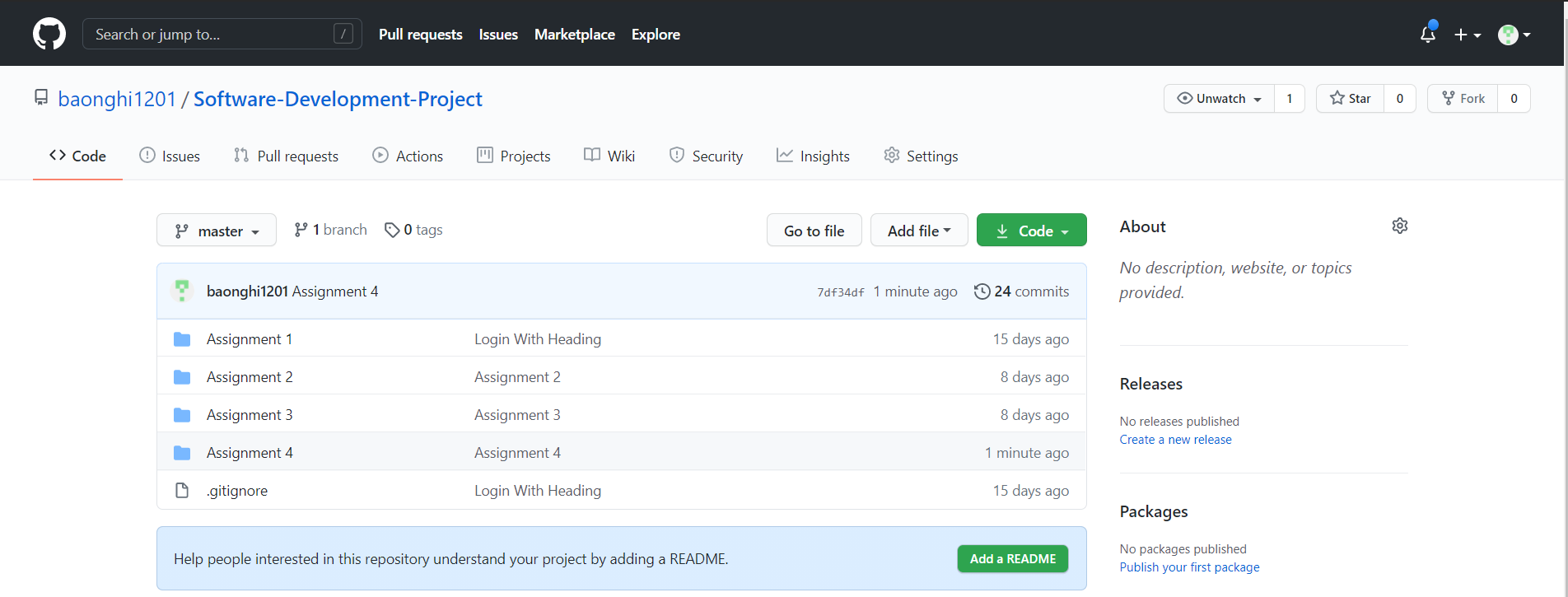
By modifying this new branch, is actually changing some of the context in the project. But the only difference here, is that we will first need to select this new branch we just created called, in this case **“style”** from there, we can now modify whatever we want. Then after that commit it to the **“style”** branch, in this case the changes will be adding another header to the project.

* 1. **Merge to Master**

After we done playing around with our project, let us merge these branches together into one **master** branch.

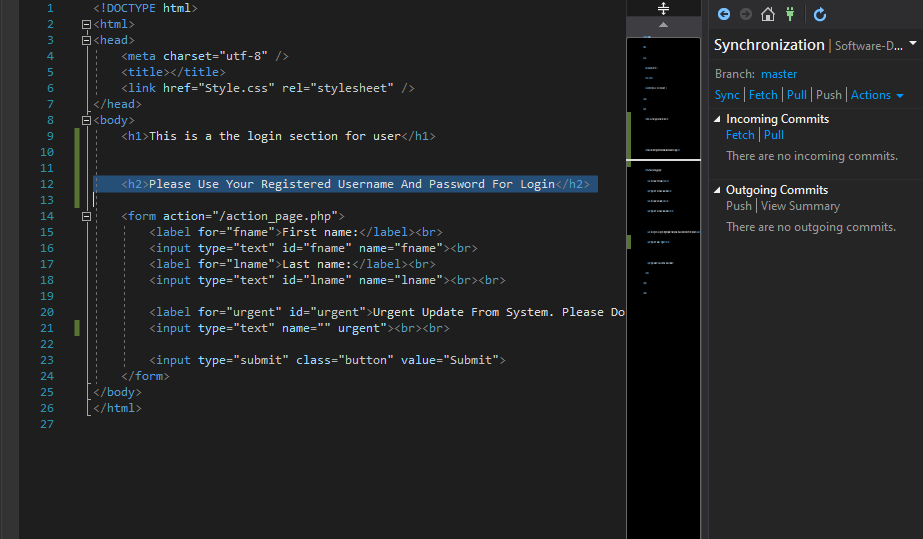
* 1. **Publish Project to GitHub**

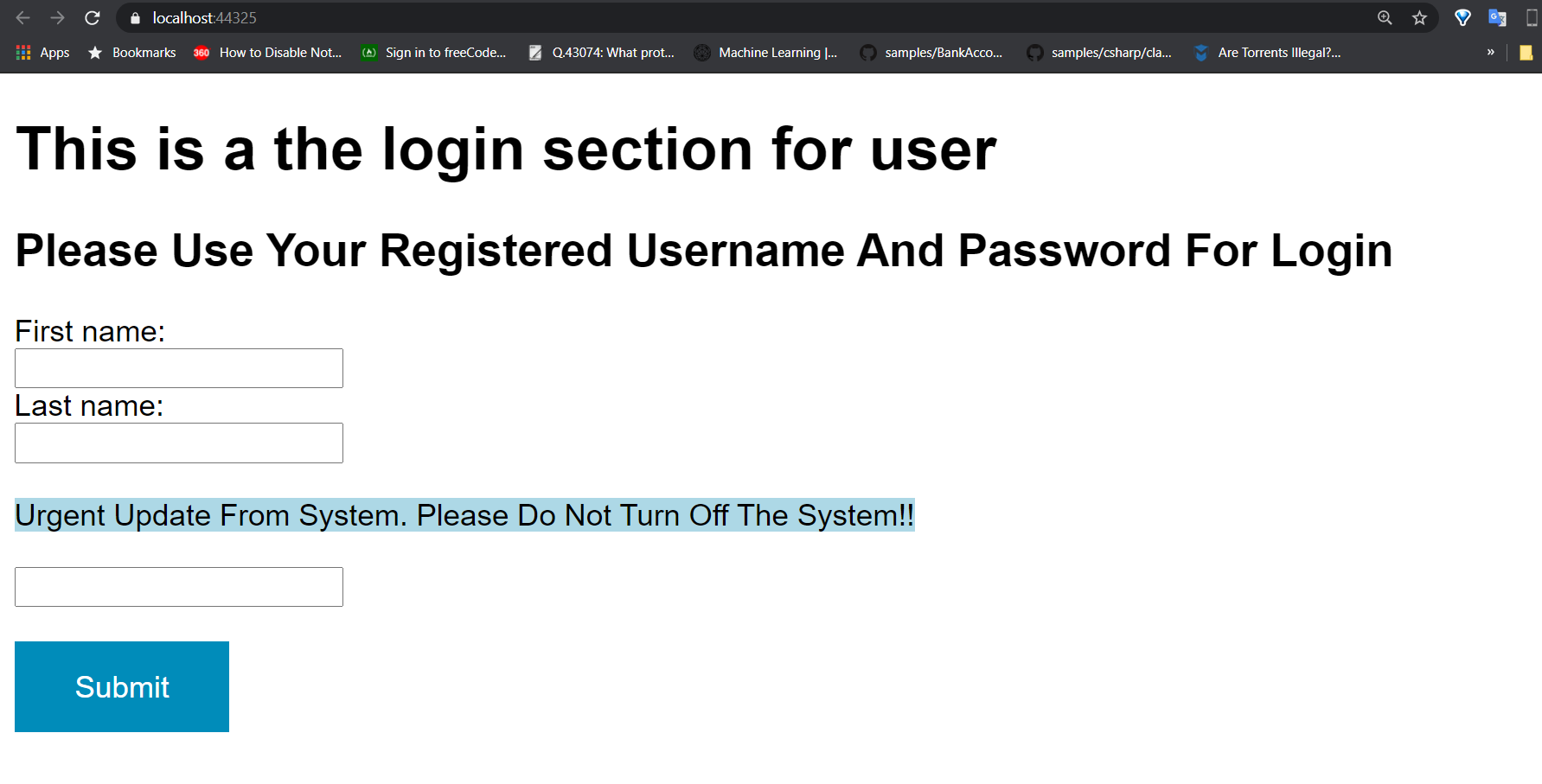
Having everything well-prepared, we then pushed the project to the **GitHub**, where it should look like this:



* 1. **Modify Project Locally**

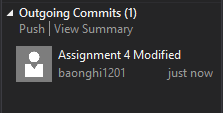
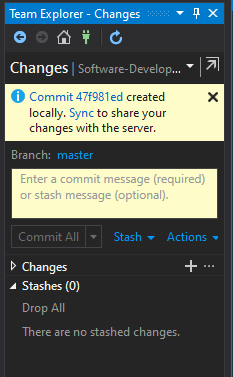
Try adding in something to the local project.

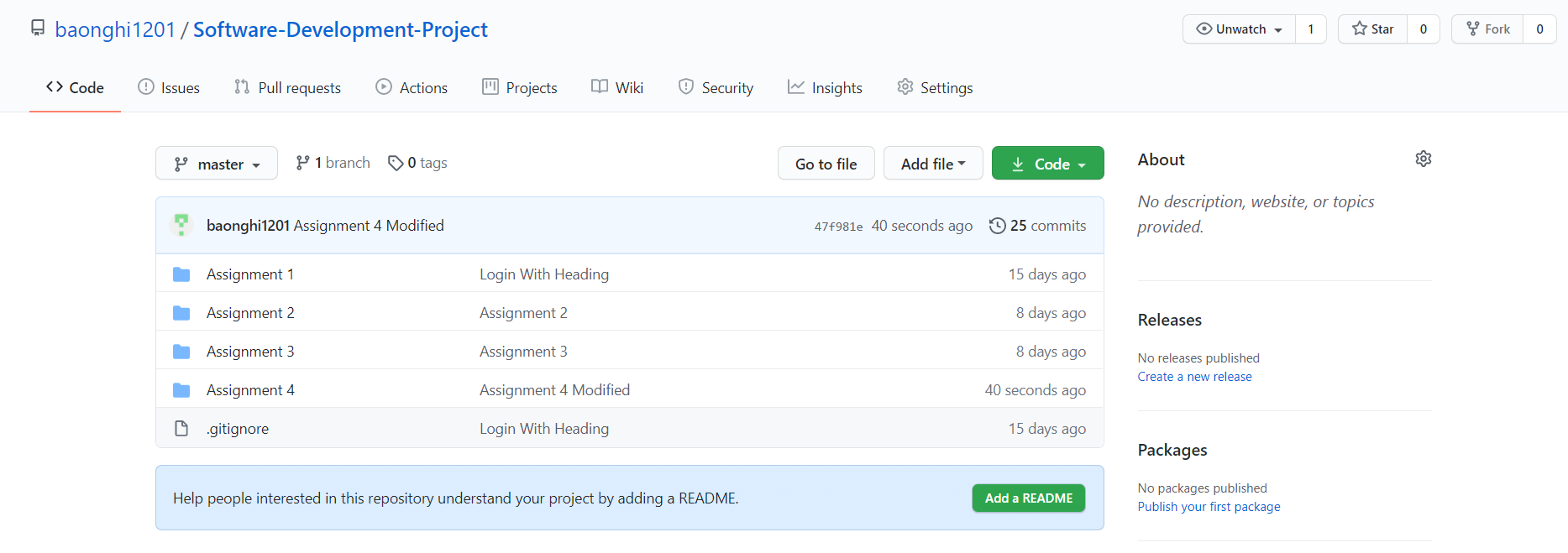


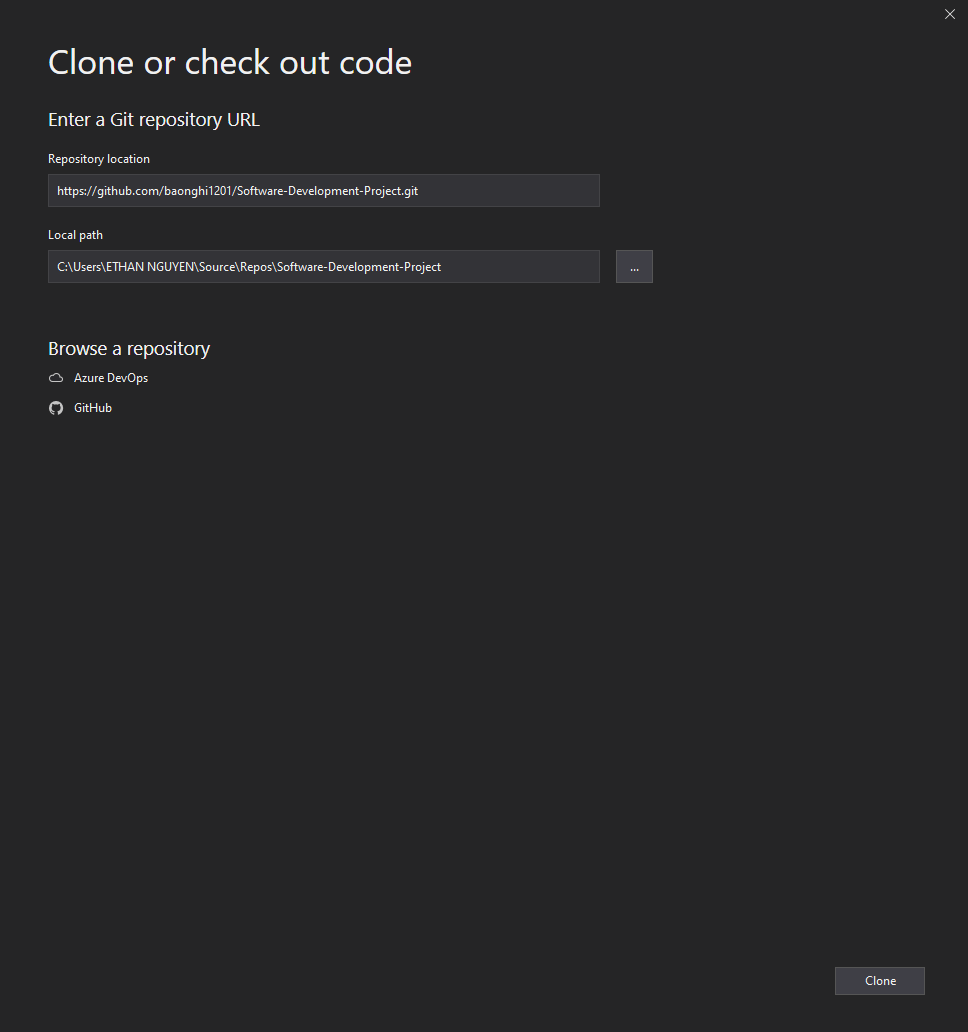


* 1. **Synchronize to GitHub**

Then we synchronize to the **Git Repository.**

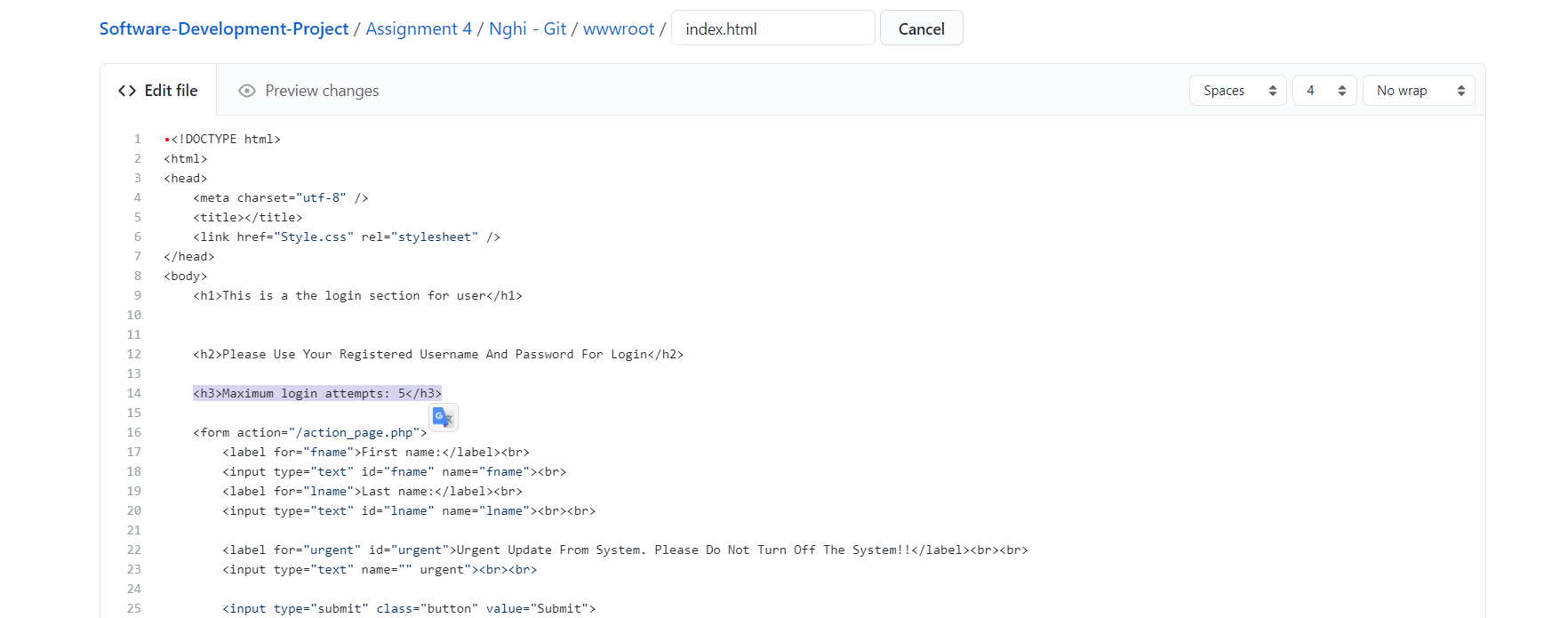
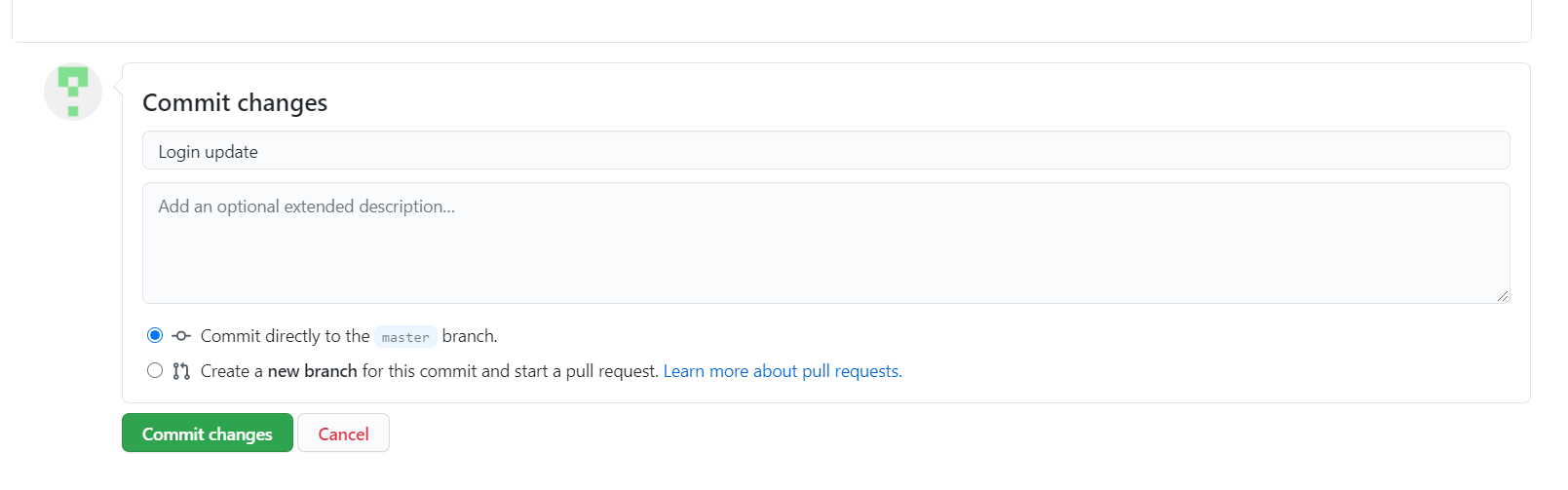


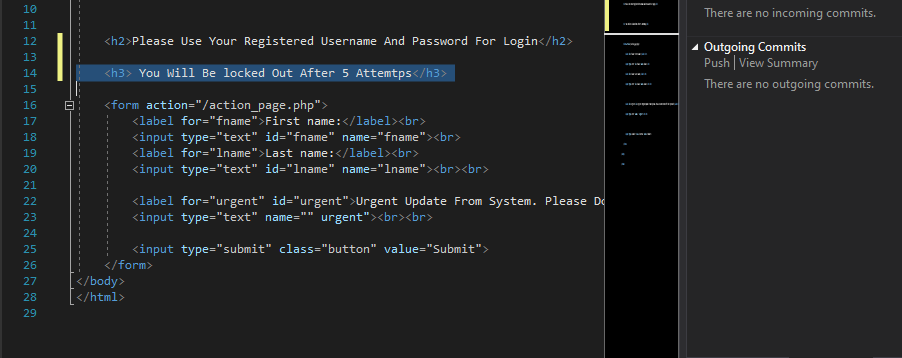
****

* 1. **Clone the Project from GitHub**
  2. **Git Conflict**

The definition of **Git Conflict** happens when the two persons, at the same modify the same file, but from different source. For instance, one person is current modifying the project on the local machine, while the other is modifying on **Git Repository**. So, when the they finished modifying their file and submitted, the conflict will instantly occur, due to the asynchronization between them.

For demonstration of this conflict, we first go to the **Git Repository** and modify something there and commit the changes:



At the local project we also modify something:

Now the conflict should happen as we expected:

* 1. **Solve the Conflict**

To solve out the conflict, the solution for solving the conflict was quite easy, the two person that specifically modify the file must decide which modified source should be applied and subtract the other. Something like this: