

Git

1. What is Global information tracker (Git)

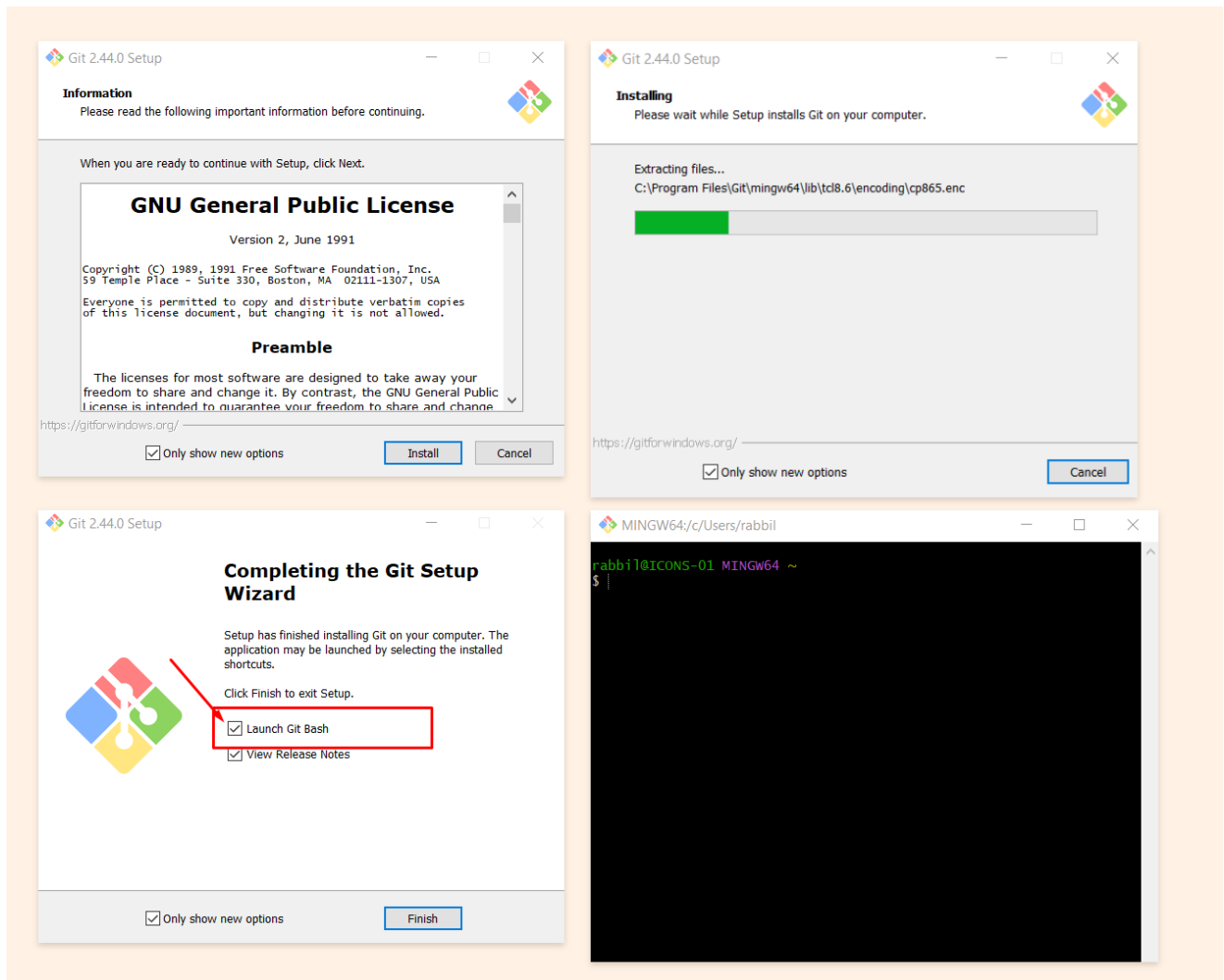
Git is a tool that helps developers work together on projects by keeping track of changes to files. It makes teamwork smoother and allows for better organization of code.

2. Use Case

1. **Version Control:** Git helps track changes made to files over time, allowing developers to revert to previous versions if needed, ensuring code stability and history tracking.
2. **Collaboration:** Git enables multiple developers to work on the same project simultaneously, managing conflicts and merging changes seamlessly, facilitating teamwork.
3. **Backup and Recovery:** Git serves as a backup system, storing project code on remote repositories like GitHub, GitLab, or Bitbucket, ensuring data safety and providing a platform for disaster recovery.

3. Git Download & Install

Download Link <https://www.git-scm.com/>



4. Your First Git Command

```
git --version
```

```
MINGW64:/c/Users/rabbil

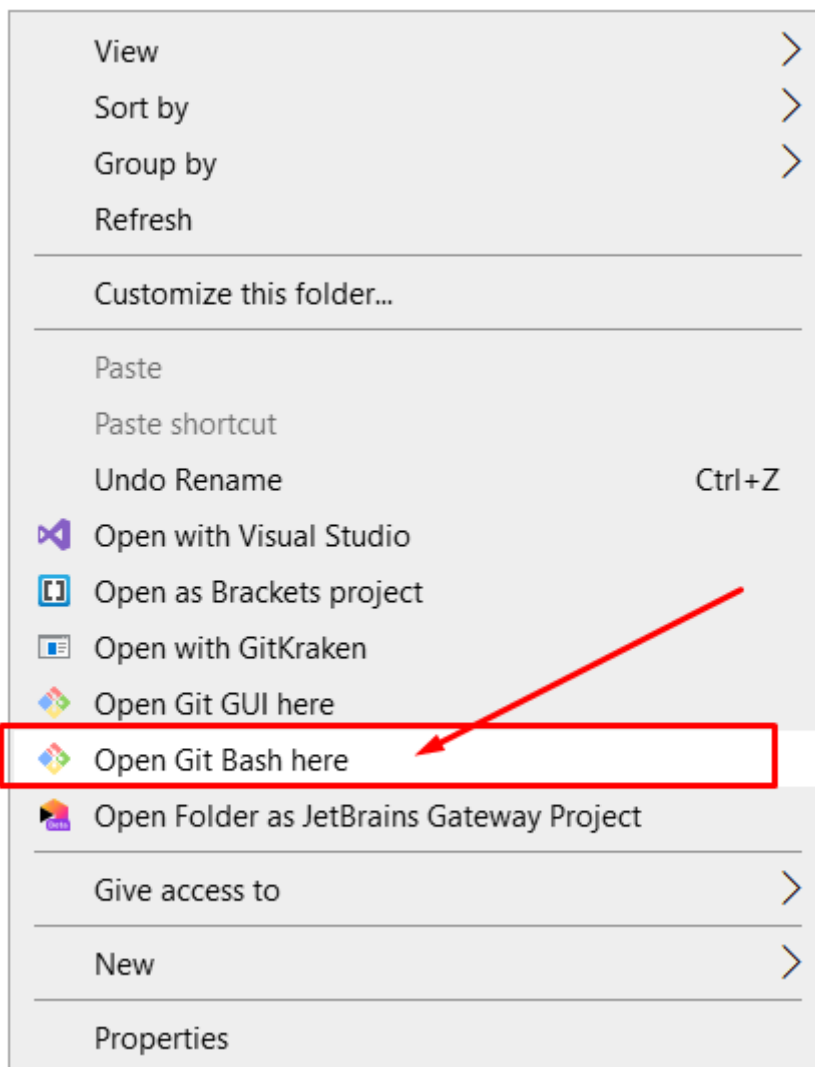
rabbil@ICONS-01 MINGW64 ~
$ git --version
git version 2.44.0.windows.1

rabbil@ICONS-01 MINGW64 ~
$
```

5. Configure Git that who your are ?

```
git config --global user.name "rabbil"
git config --global user.email "mrrabbilhasan@gmail.com"
```

6. Create your first local git repository

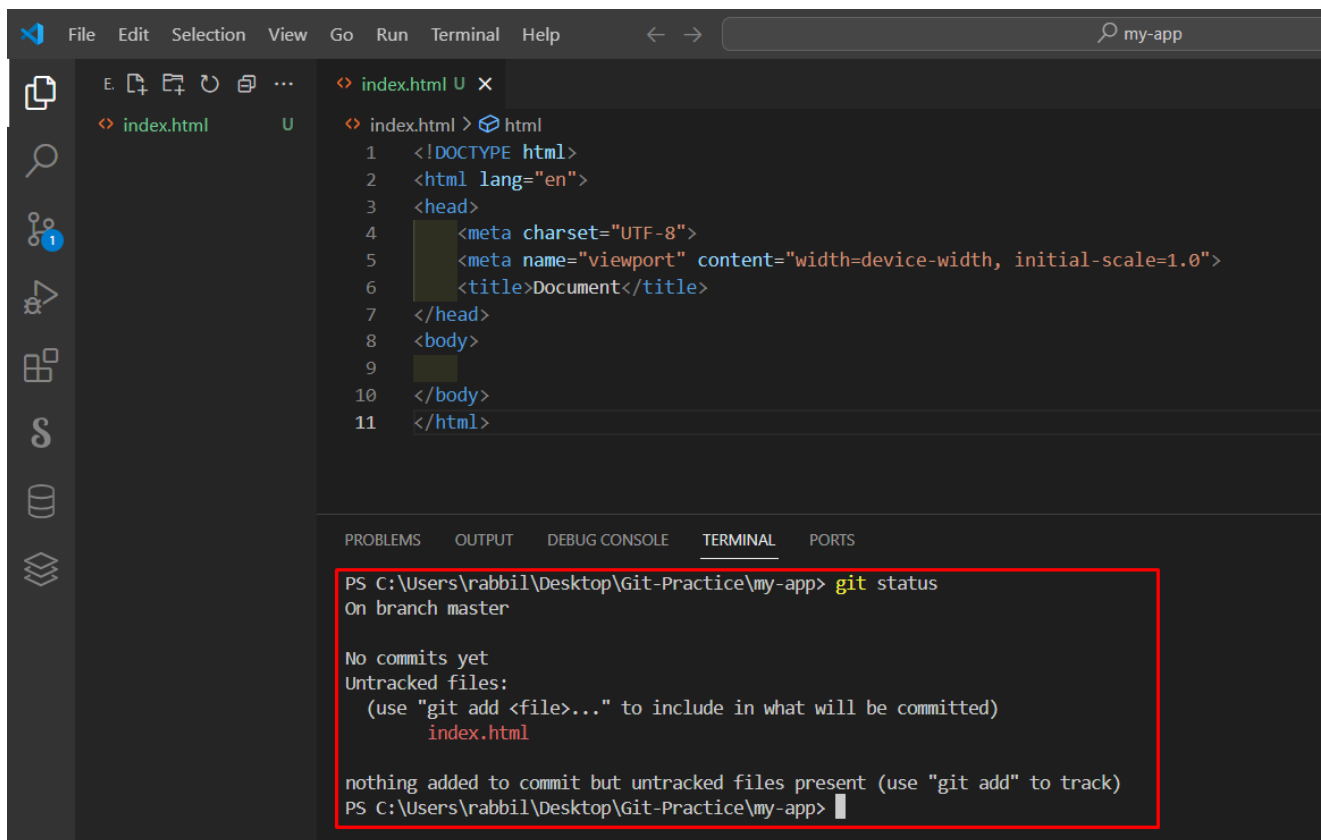


```
mkdir my-app
cd my-app
git init
```

7. Lets create your first coding project & start version controlling

■ STEP-01: Understanding Git Status Tracked & Untracked Files

- Open my-app folder in vs code
- Create simple html file index.html
- Lets check your project git status `git status`



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a file named `index.html` with a green 'U' icon, indicating it is untracked. The Editor pane displays the content of `index.html`, which is a basic HTML document with a doctype, meta tags for charset and viewport, and a title 'Document'. The Terminal pane at the bottom shows the output of the `git status` command. The output indicates that there are no commits yet and that `index.html` is an untracked file. It also provides instructions on how to use `git add` to track the file.

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git status
On branch master

No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

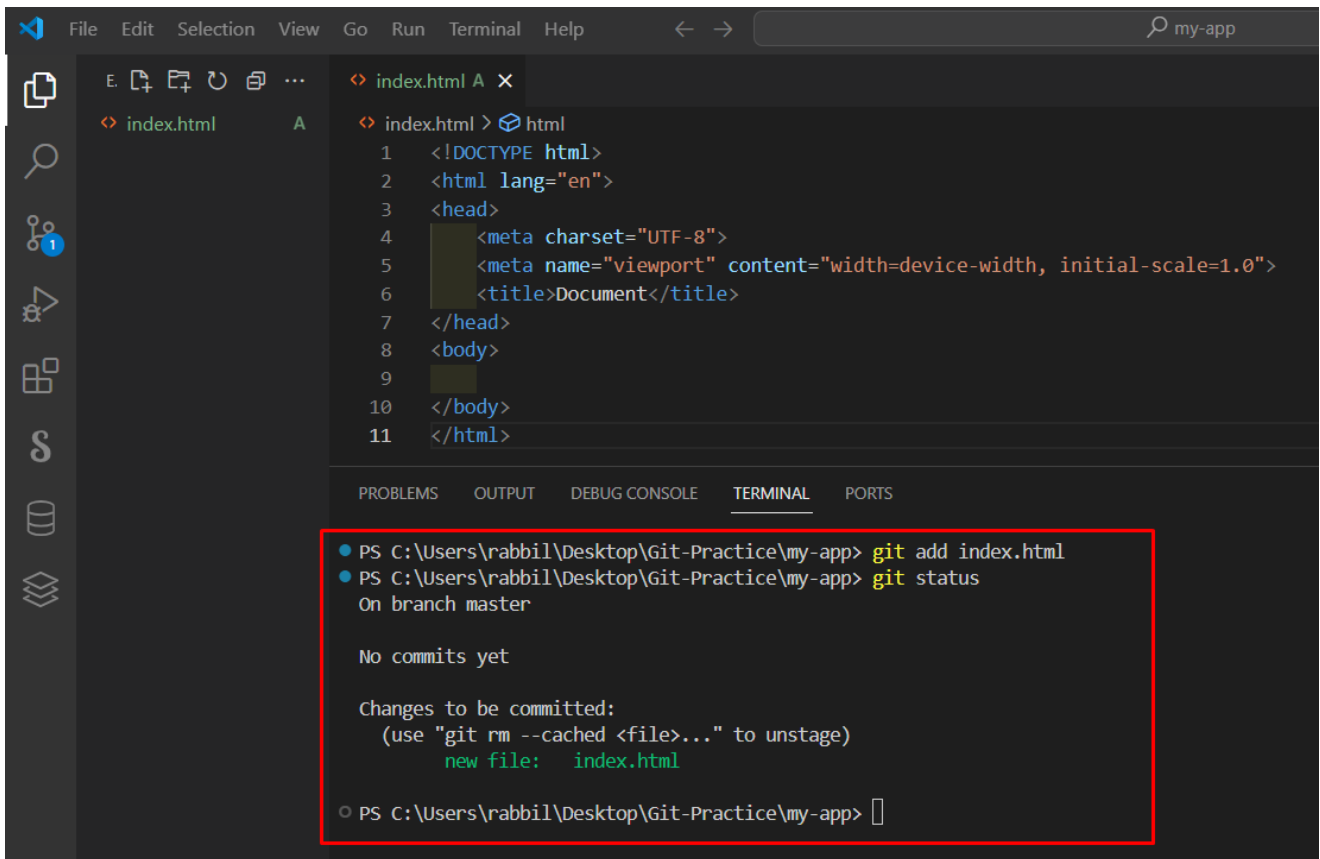
nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

- Files in your Git repository folder can be in one of **2 states**
- **Tracked** - files that Git knows about and are added to the repository
- **Untracked** - files that are in your working directory, but not added to the repository
- When you first add files to an empty repository, they are all untracked
- To get Git to track them, you need to add them

■ STEP-02: Understanding Git Staging

- To add specific files or directories, you would typically follow the `git add` command with the names of the files or directories you want to add.

```
git add index.html
```



The screenshot shows the Visual Studio Code editor with a file named `index.html` open. The file content is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10 </body>
11 </html>
```

The terminal window at the bottom shows the following commands and output:

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git add index.html
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git status
On branch master

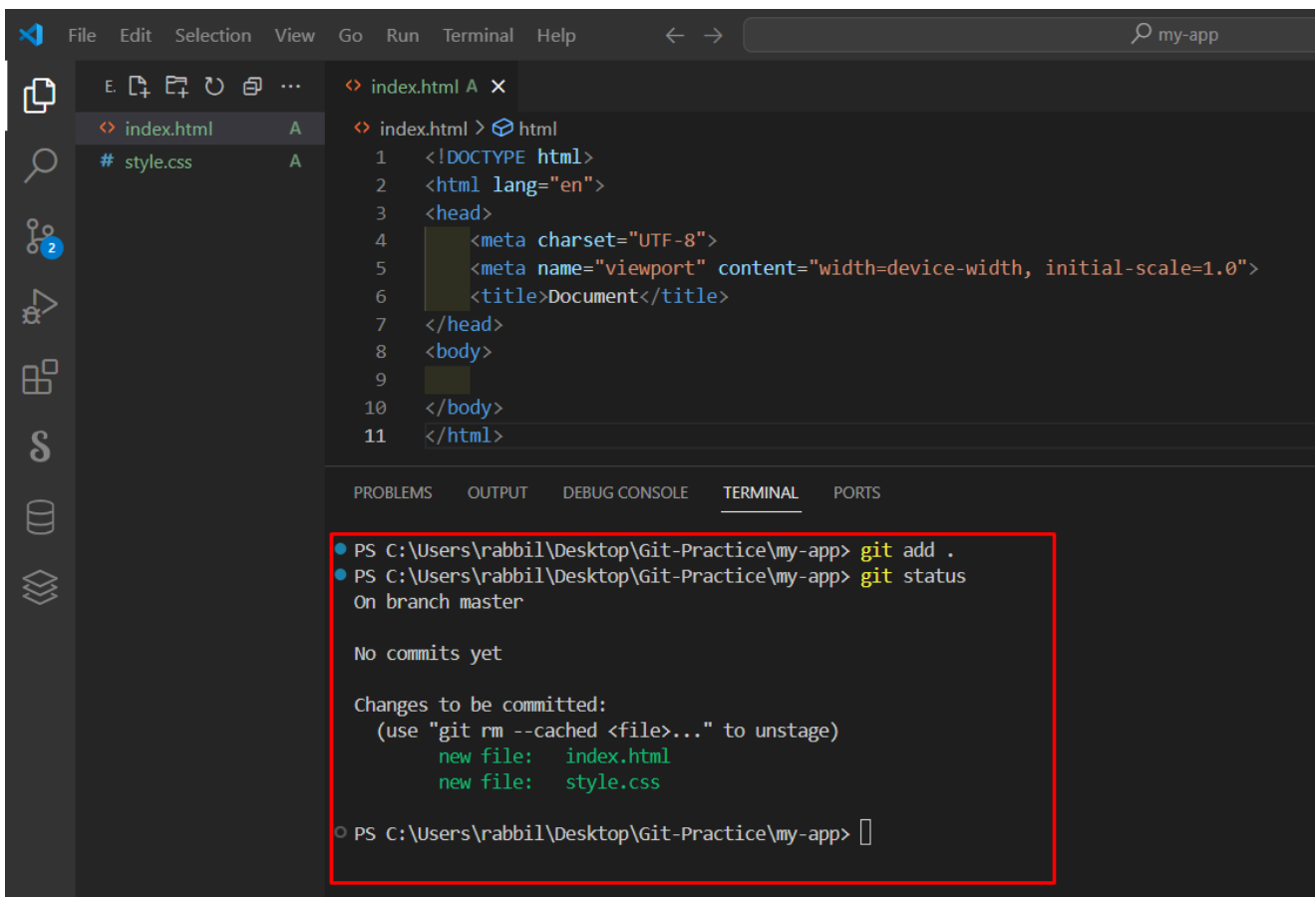
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html

PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

- If you want to add all changes in the current directory and its subdirectories, you can use

```
git add .
// or
git add --all
```



The screenshot shows the Visual Studio Code editor with two files open: `index.html` and `style.css`. The `index.html` content is the same as in the previous screenshot. The `style.css` file is currently empty.

The terminal window at the bottom shows the following commands and output:

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git add .
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git status
On branch master

No commits yet

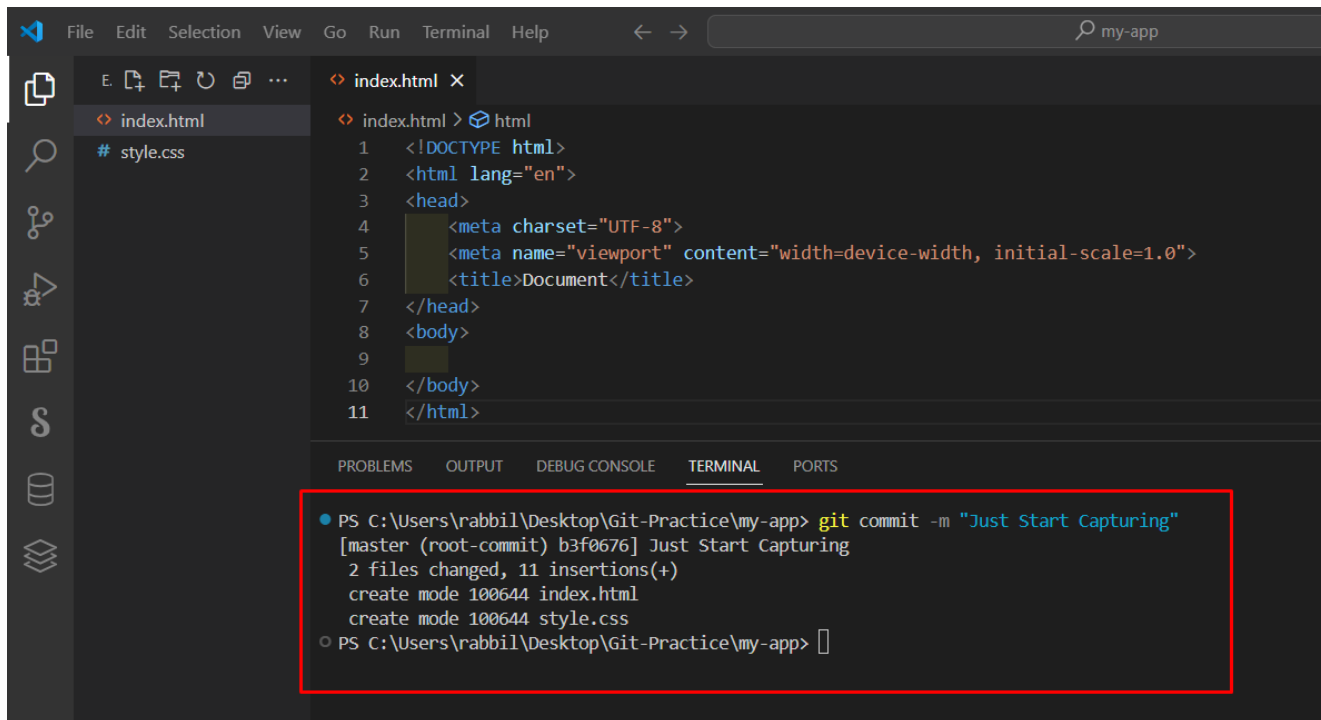
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css

PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

STEP-03: Understanding Git Commit

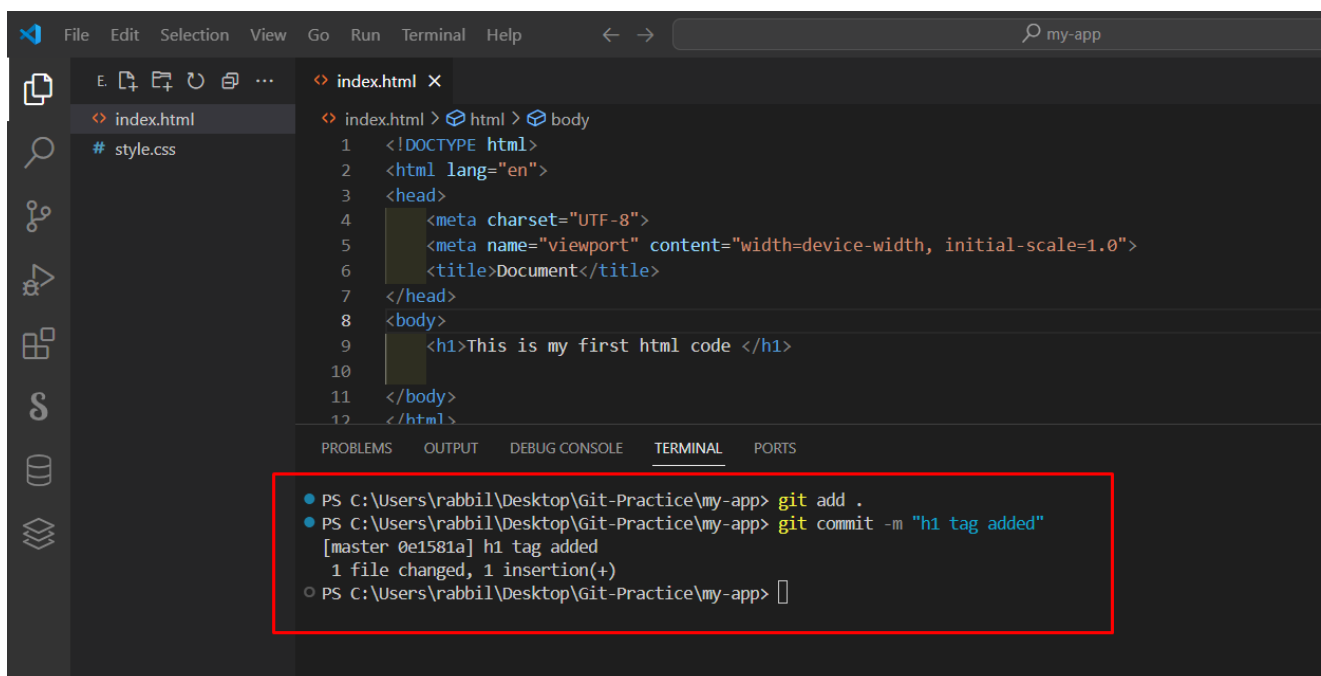
- A Git commit is like taking a photo of your project at a specific moment. It captures all the changes you've made, who made them, and a short message explaining what was done. It's like saving a checkpoint in your project's history.

```
git commit -m "Just Start Capturing"
```



- Make some changes capture again

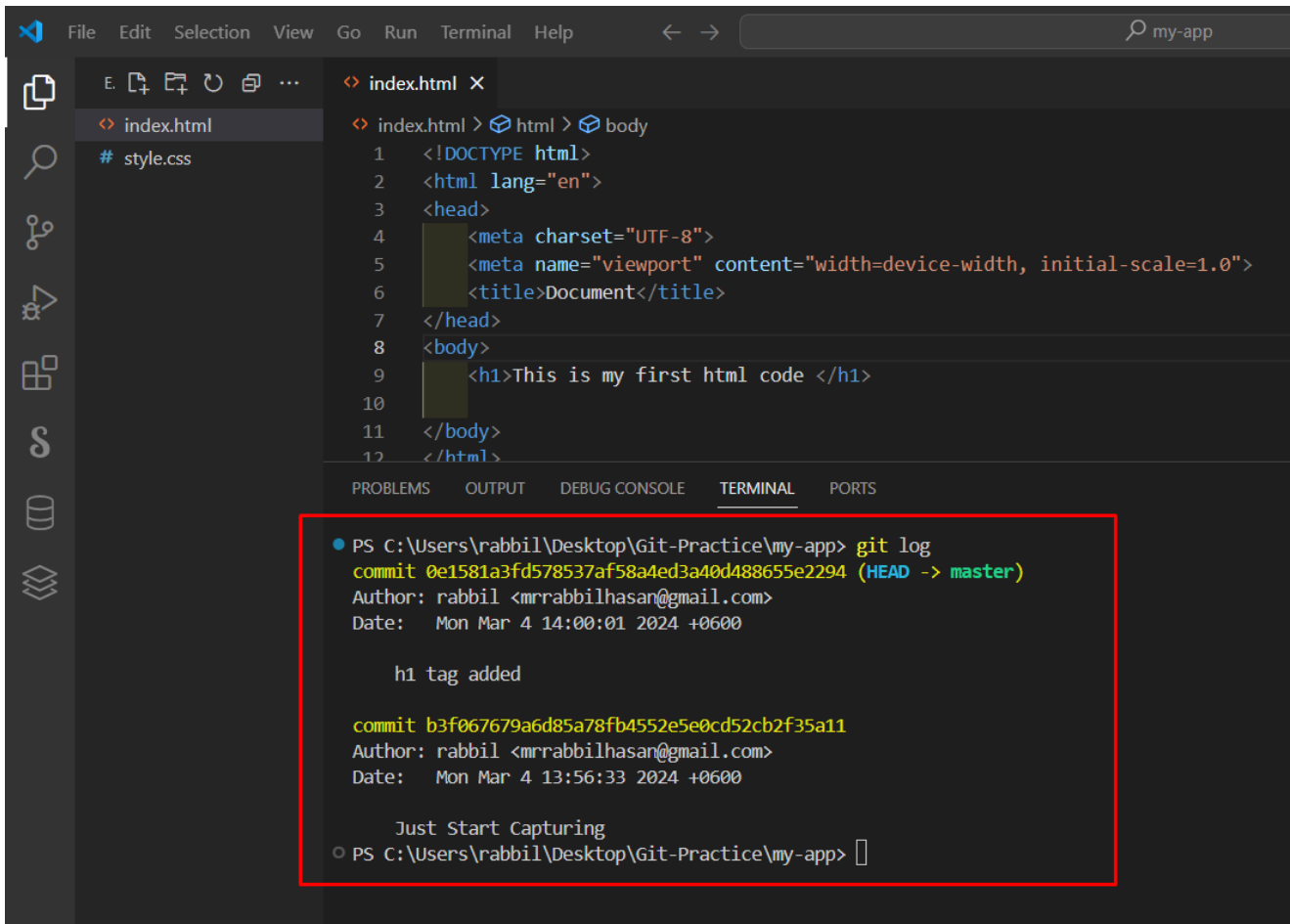
```
git add .
git commit -m "h1 tag added"
```



STEP-04: Understanding Git Log

Command used to view the history of captures in a Git repository.

git log



The screenshot shows the Visual Studio Code interface. The editor is open to `index.html`, which contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>This is my first html code </h1>
</body>
</html>
```

The terminal window at the bottom shows the output of `git log`:

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git log
commit 0e1581a3fd578537af58a4ed3a40d488655e2294 (HEAD -> master)
Author: rabbil <mrrabbilhasan@gmail.com>
Date: Mon Mar 4 14:00:01 2024 +0600

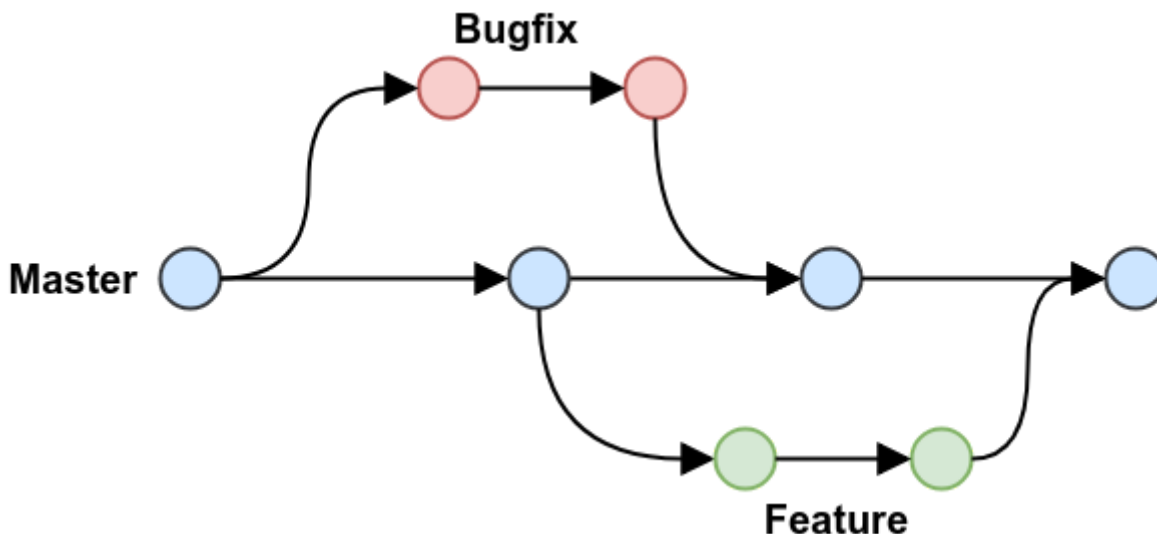
    h1 tag added

commit b3f067679a6d85a78fb4552e5e0cd52cb2f35a11
Author: rabbil <mrrabbilhasan@gmail.com>
Date: Mon Mar 4 13:56:33 2024 +0600

    Just Start Capturing
PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

8. Lets Understand Branch

Branch is a new/separate version of the main repository.

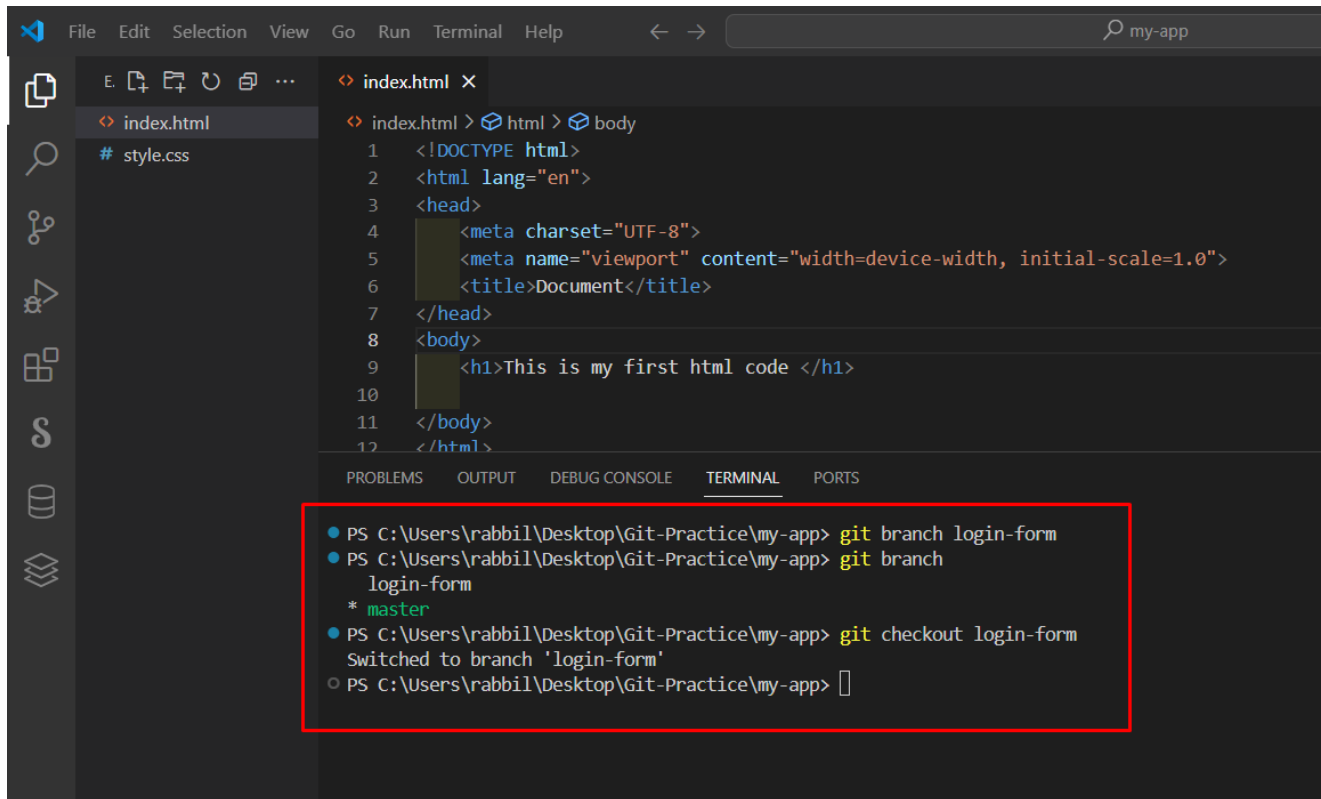


1. **Feature Development:** Creating a branch to work on a new feature or enhancement without disrupting the main codebase.
2. **Bug Fixes:** Isolating bug fixes in separate branches to ensure they can be tested and deployed independently.
3. **Experimentation:** Trying out new ideas or approaches without altering the main project until they're proven successful.
4. **Versioning:** Maintaining different versions of the project for different purposes (e.g., stable releases, development versions)

9. Playing with branch

■ STEP-01: Create new branch & confirm that is created & switching to new branch

```
git branch login-form
git branch
git checkout login-form
```



■ **STEP-02: Say I added a login form feature inside index.html

```
git status
git add --all
git status
git commit -m "login html form created"
```

The image shows a Visual Studio Code editor window with a file explorer on the left showing 'index.html' and 'style.css'. The main editor area displays the content of 'index.html', which is an HTML form with a title 'Document', a heading 'This is my first html code', and a login form with fields for 'username', 'password', and a 'Login' button. Below the editor, the 'TERMINAL' tab is active, showing a series of git commands and their outputs. The terminal output is as follows:

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git status
On branch login-form
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

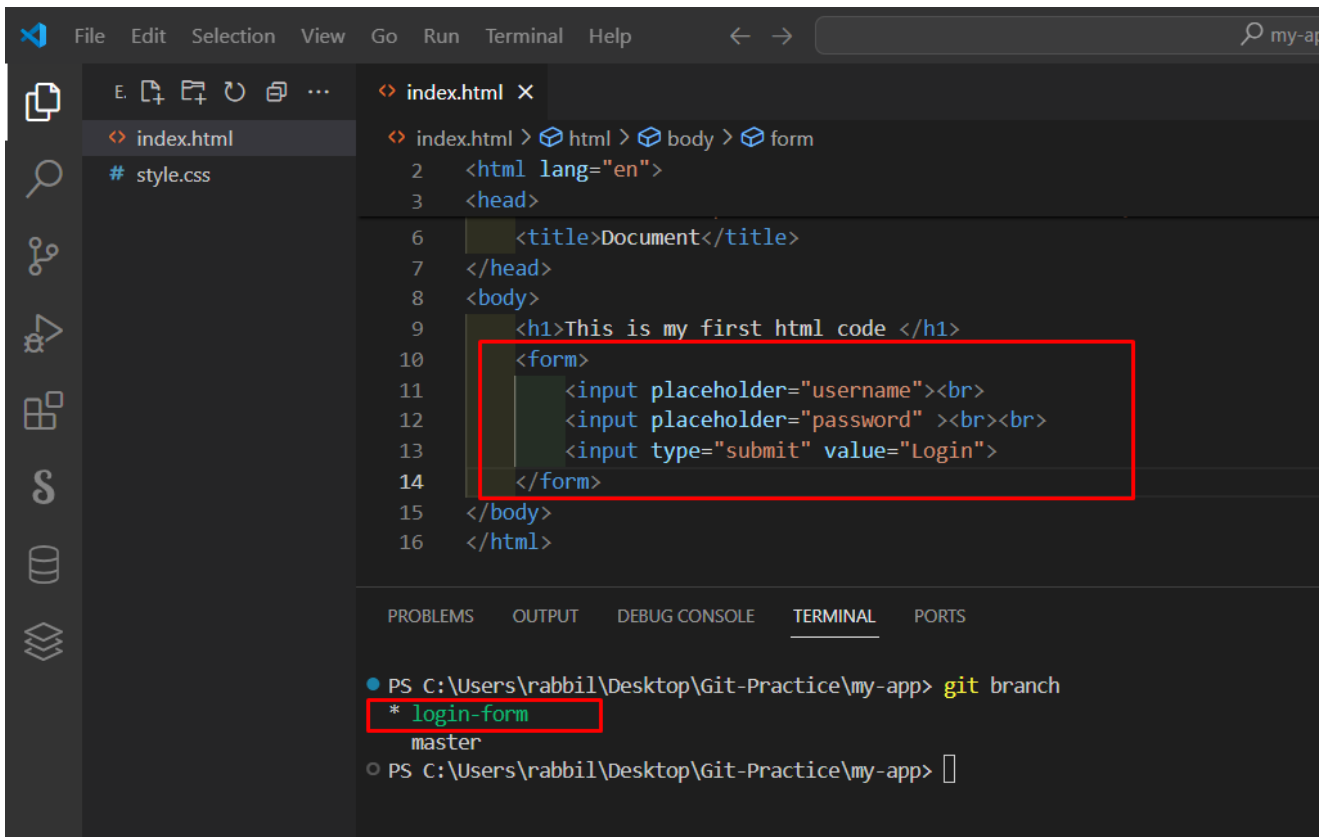
no changes added to commit (use "git add" and/or "git commit -a")
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git add --all
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git status
On branch login-form
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git commit -m "login html form created"
[login-form 3847463] login html form created
1 file changed, 5 insertions(+), 1 deletion(-)
PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

■ **STEP-03: Now discover index.html at main branch & login-form branch

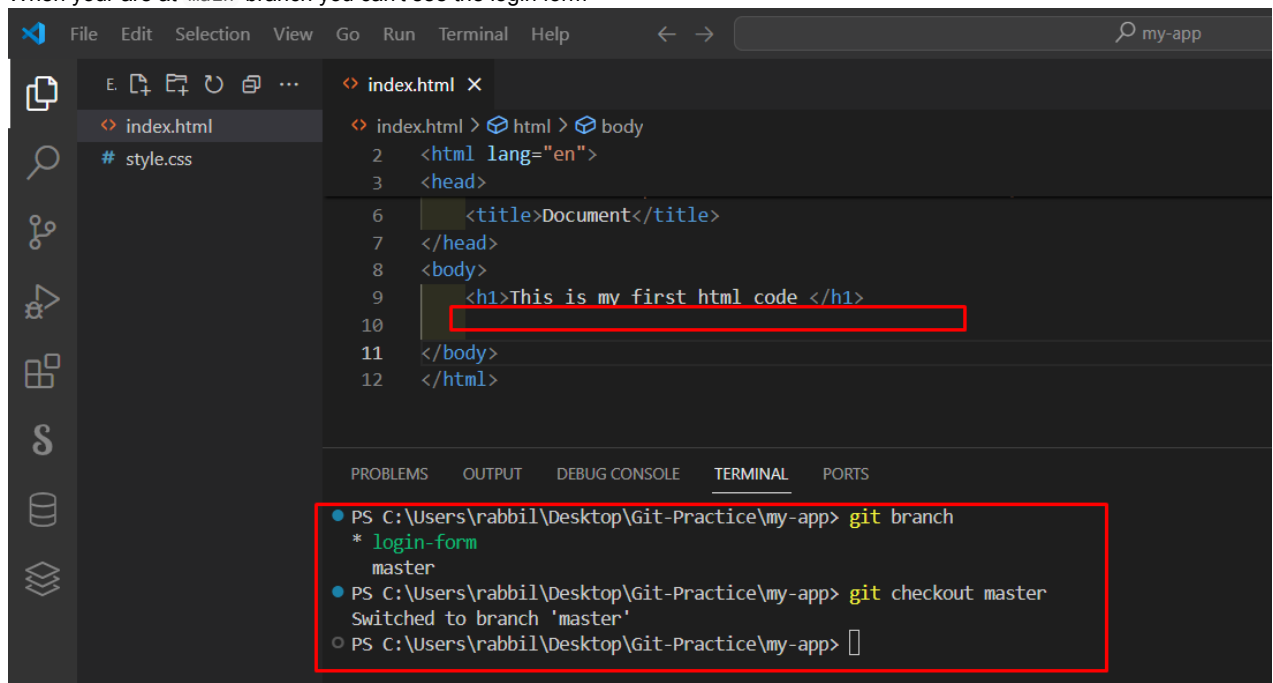
```
git branch
```

- When you are at login-form branch you can see the login form



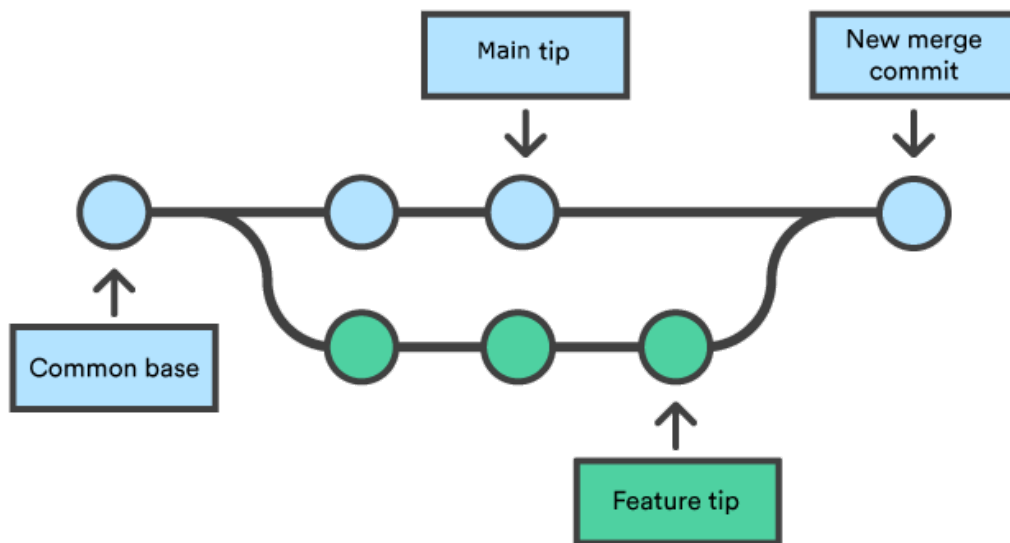
`git checkout master`

- When you are at `main` branch you can't see the login form



10. Git Branch Merge

Git branch merge refers to the process of combining the changes from one branch into another. This is typically done to incorporate the changes made in a feature branch (or any other branch) back into the main branch of the repository, such as `master` or `main`.



STEP: 01 Let merge the login from html codes from `login-form` branch to `main` branch

```
git branch
git checkout master
git merge login-form
```

The screenshot shows the Visual Studio Code interface. The left sidebar displays the file explorer with `index.html` and `style.css`. The main editor shows the `index.html` file with the following content:

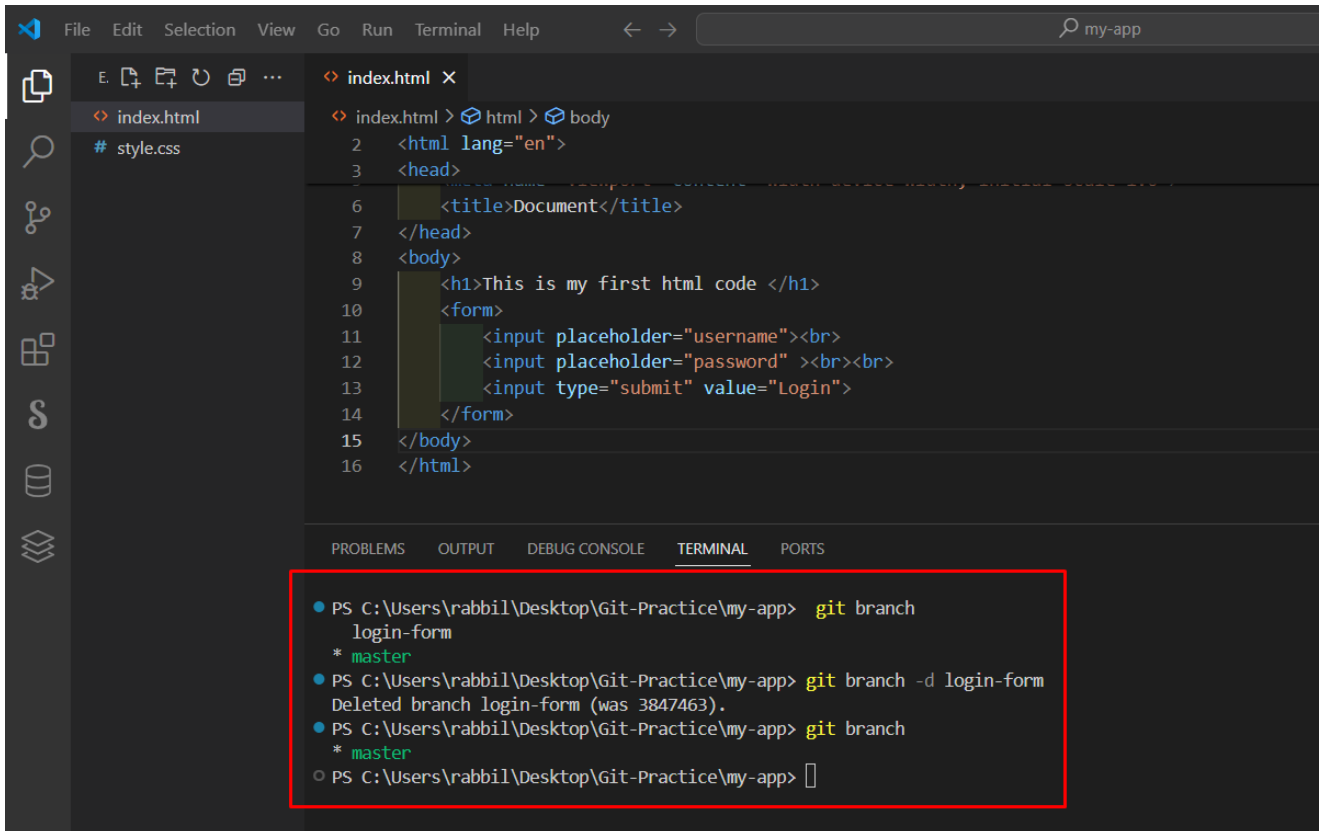
```
<html lang="en">
<head>
  <title>Document</title>
</head>
<body>
  <h1>This is my first html code </h1>
  <form>
    <input placeholder="username"><br>
    <input placeholder="password" ><br><br>
    <input type="submit" value="Login">
  </form>
</body>
</html>
```

The `<form>` block is highlighted with a red box. The bottom panel shows the terminal output of the merge command:

```
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git branch
* login-form
  master
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git checkout master
Switched to branch 'master'
PS C:\Users\rabbil\Desktop\Git-Practice\my-app> git merge login-form
Updating 0e1581a..3847463
Fast-forward
 index.html | 6 +++++
 1 file changed, 5 insertions(+), 1 deletion(-)
PS C:\Users\rabbil\Desktop\Git-Practice\my-app>
```

STEP: 02 Delete the `login-form` branch after merge

```
git branch
git branch -d login-form
git branch
```



Git Ignore and .gitignore

`.gitignore` is a file used in Git repositories to specify intentionally untracked files and directories that Git should ignore.

- Log files
- Vendor files
- Node modules
- Temporary files
- Hidden files

```
touch .gitignore
```

Rules for .gitignore

- **File and directory matching:** You can specify individual files, directories, or patterns using wildcard characters.
 - Example: `logs/` ignores the `logs` directory.
 - Example: `*.log` ignores all files with the `.log` extension.
 - Example: `build/` ignores all directories named `build`.
- **Negation:** You can use negation to exclude specific files or directories that would otherwise match a pattern.
 - Example: `!important.log` ignores all `.log` files except `important.log`
- **Comments:** Lines starting with `#` are considered comments and are ignored.
 - Example: `# Ignore compiled files` is a comment.
- **Ignore a specific file:** If you want to ignore a file named `example.txt`, just add this line to your `.gitignore` file:
 - Example: write `example.txt` inside

