CS 696A Full-Stack Entrepreneurs App Development - Lab 2

Prof. Arya Boudaie - Fall 2025
Pace University

OBJECTIVE

In this lab, you will create a GitHub Pages Website, and learn how to use GitHub command line.

DUE DATE

Tuesday September 16th at 6:00 PM EST

SUBMISSION INSTRUCTIONS

- Create a word document (word, google docs, pages, etc).
- Complete four (4) exercises below by answering ten (10) questions inside the word document created.
- Upload the document to the assignment section of the course.

INSTRUCTIONS

EXERCISE 1: CREATE YOUR GITHUB ACCOUNT & WEBSITE

If you already have a Github account, log in and skip to Step 4. If you already have a Github pages, make another repository for this page.

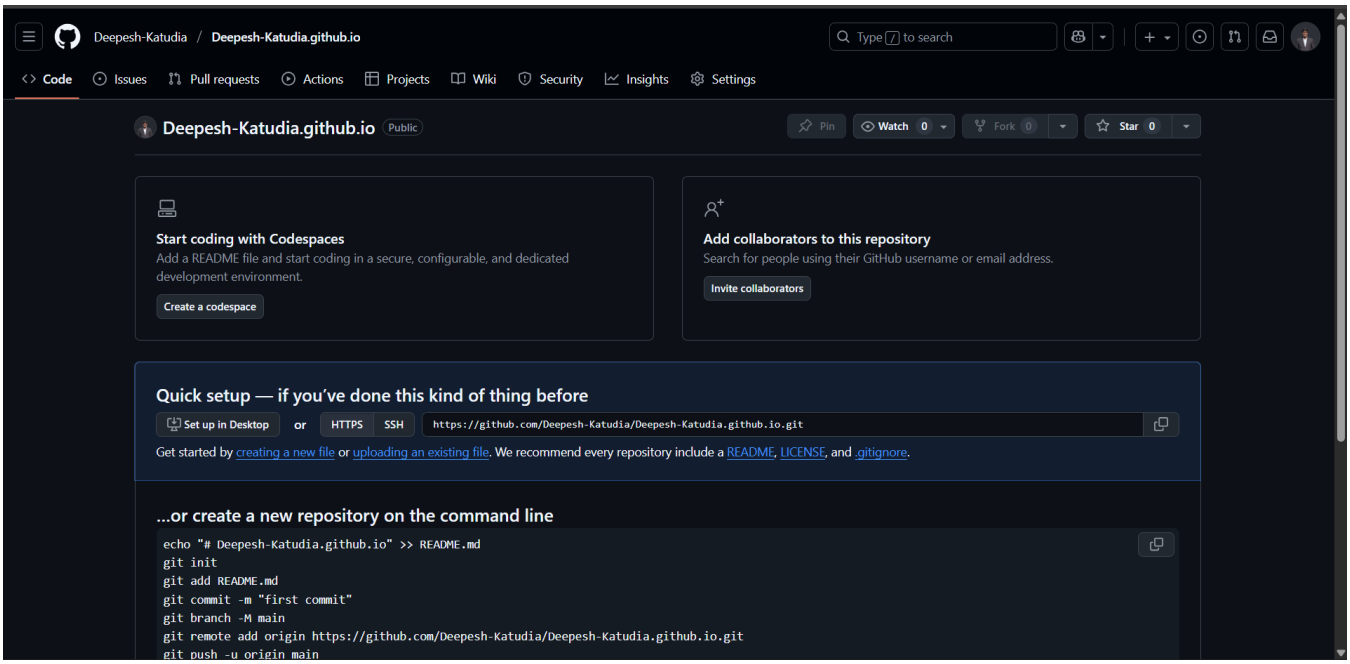
1. Go to `https://www.github.com`
2. Click 'Sign Up' button at the top right corner.
3. Create your Github account following the steps; remember the username of the account. Make the username professional.
4. Sign up for the GitHub student pack.
5. Create a new repository by clicking 'New' button at the top left corner (next to "Top Repositories").

(a) **IMPORTANT:** When creating a new repository, the repository name **MUST** be {username}.github.io.
(i.e., if your username is janedoe123, your repository name **MUST** be janedoe123.github.io).

(b) Do not include any starter files (README, license, etc).

6. Go to the repository you just created.

Question 1: Take a screenshot of the repository created from the web browser. Take a screenshot and place it within the document.



6. Click a green 'Code' button and find a URL under HTTPS; it is a URL representing your website's git repository.

Question 2: Paste the URL within the document.

- https://github.com/Deepesh-Katudia/Deepesh-Katudia.github.io.git

7. Read this guide to set up your repository's settings correctly with GitHub pages.

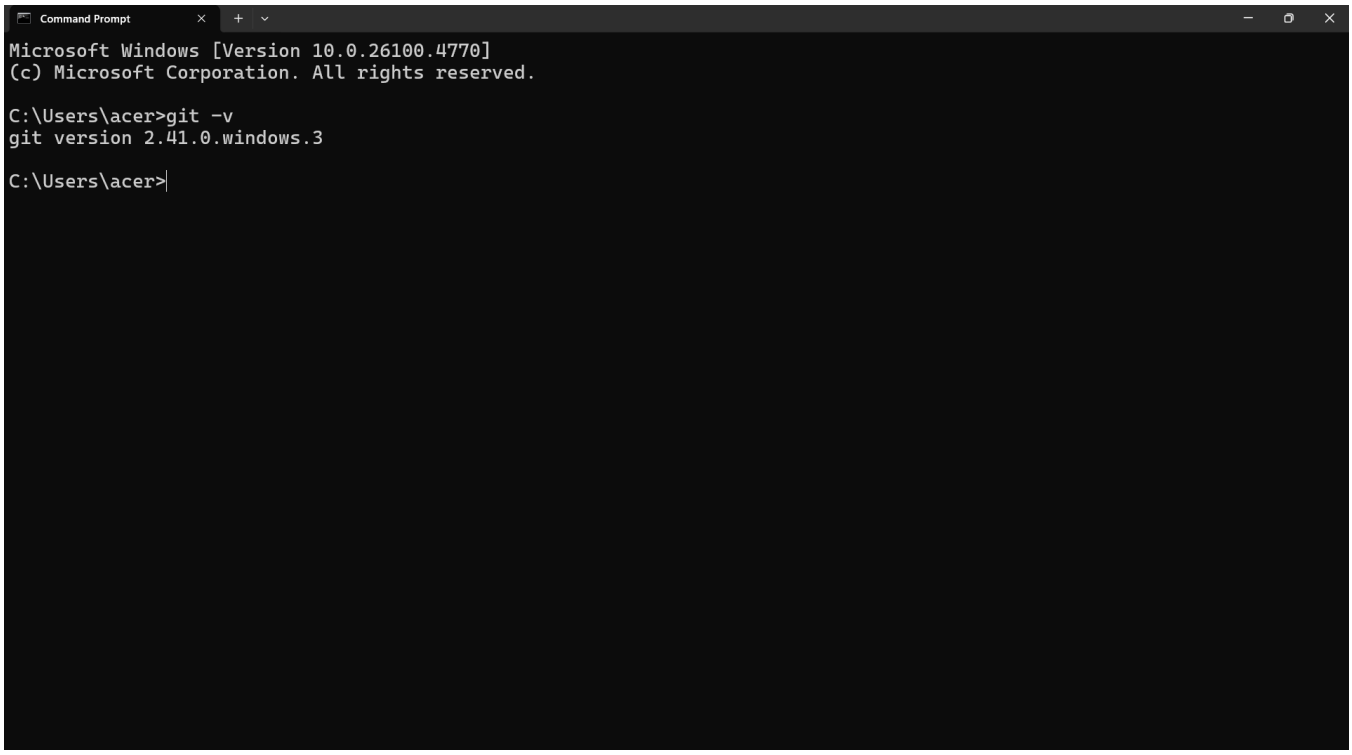
EXERCISE 2: INSTALL GIT

If you are using a Macbook or run Linux or Windows Subsystem for Linux (WSL), you already have Git, skip to the confirmation step. For Windows users, it is **HIGHLY** recommended you install Windows Subsystem for Linux which comes with git. Follow the steps here to install WSL.

To confirm Git is installed, Within the Terminal window (On Windows, Git Bash; on Macbook,

Terminal), run a command ‘git -v’ to make sure git is installed.

Question 3: Take a screenshot of the output of the command ‘git -v’ within the Terminal window. Take a screenshot and place it within the document.



5. Run the following two (2) command to set up your information within Git; this personal information will be used to keep track of the people who made the latest changes inside the repository.

Note: Replace `<YOUR NAME>` and `<YOUR EMAIL>` with your own details.

```
git config --global user.name "<YOUR NAME>"
git config --global user.email "<YOUR EMAIL>"
```

EXERCISE 3: PRACTICE GIT WORKFLOW

1. We will clone a git repository corresponding to your Github website; run the following command within the Terminal.
   Note: Replace `<REPOSITORY URL>` with the URL identified for **Question 2** above.

```
git clone <REPOSITORY URL>
```

2. Now, you will see all the contents inside the repository are copied down to the new folder created; under the folder, use any text editor and create a file, index.html, with the following contents.

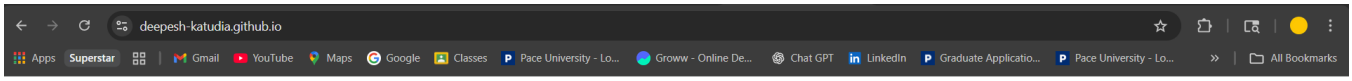
```
<html>
    <body>
        Hello, World!
    </body>
</html>
```

3. Within the Terminal, change directory to the folder where index.html above is created.
4. We will commit and push the new file now; run the following commands:

```
git status (Note: the command may show fatal error; read the
instructions and take actions accordingly).

git add .

git commit -m "Create index.html"

git push (Note: the command may ask you to log in; read the
instructions and take actions).
```

5. Refresh Github repository on the web browser; confirm 'index.html' exists now; wait 1 minute if necessary.

Question 4: Go to {username}.github.io URL on the web browser (replace {username} with your actual Github username). Take a screenshot of your web browser showing "Hello, World!" and place it inside the document.



Hello, World!

---

6. Now, we will practice the usage of feature branch. Within the Terminal, run the following command to create a new feature branch, add-title.

```
git checkout -b add-title
```

7. Update your index.html file to the contents below.

```
<html>
    <head>
        <title>My Page</title>
    </head>
    <body>
```

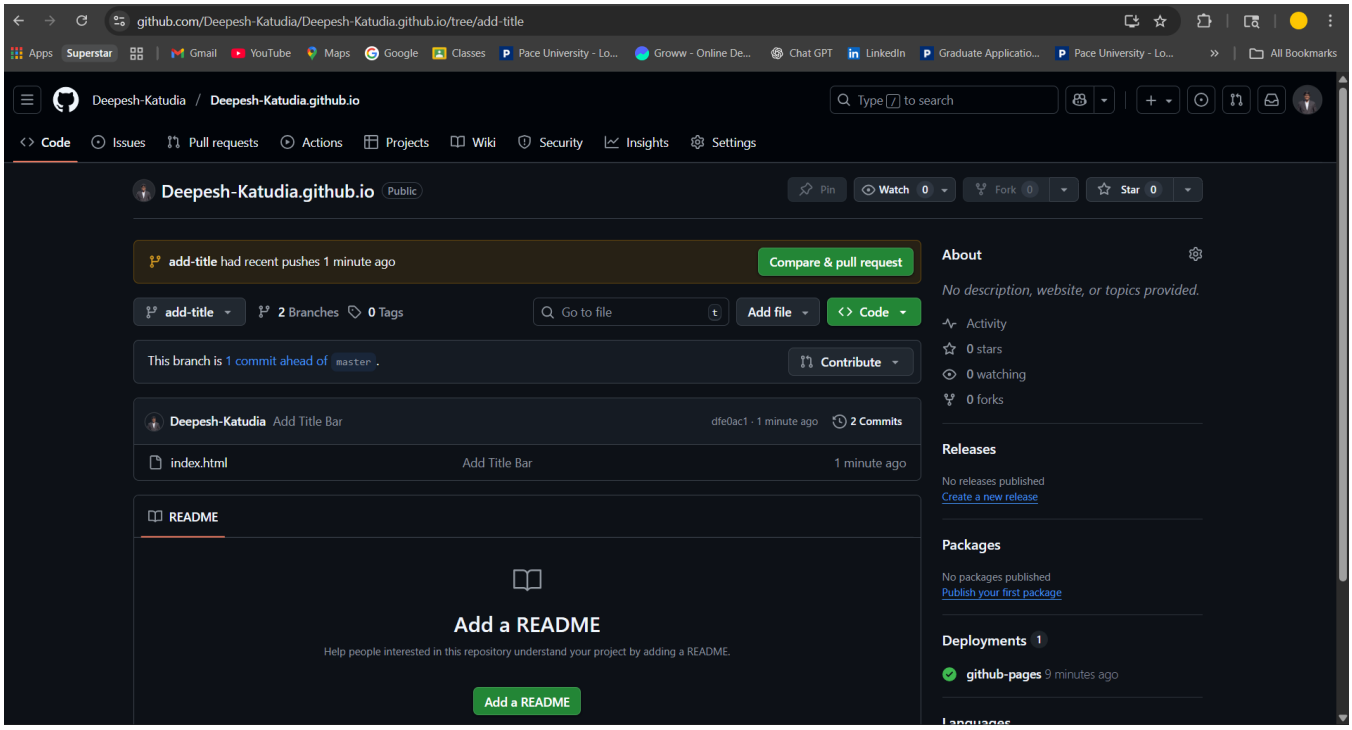
```
        Hello, World!
    </body>
</html>
```

8. Now, we will push the change in the feature branch so that a pull request can be created; a pull request is used to review a change made for a particular feature and merge the change into the main branch. Run the following commands:

```
git status (Note: the command may show fatal error; read the
instructions and take actions accordingly).

git add .

git commit -m 'Add Title Bar'

git push (Note: the command may show an error; read the
instructions and take actions accordingly).
```

9. Refresh Github repository on the web browser. Click 'main' at the top left corner; now you will see 'add-title' (a new branch we created) under the menu. Click 'add-title' branch name.

Question 5: Take a screenshot of your Github repository showing 'add-title' branch name, and place it in the document.

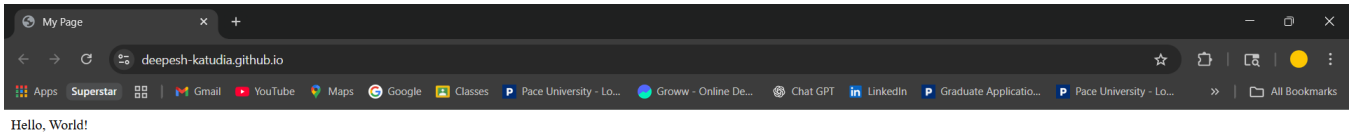


10. Click ‘Contribute’ at top right corner, and click a green ‘Open Pull Request’ button.

Question 6: Scroll down and see the change of the file; take a screenshot of the change of the file and place it in the document.

11. Click green ‘Create Pull Request’ button.
12. Click green ‘Merge Pull Request’ button; now, your changes from the feature branch are merged to ‘main’ branch that is reflected on your Github website. Wait 1 minute if necessary.

Question 7: Go to {username}.github.io URL on the web browser (replace {username} with your actual Github username). Take a screenshot of your web browser showing the title of ‘My Page’ and place it inside the document.



---

13. Within the Terminal, run the following command to go back to ‘main’ branch.

```
git checkout main
```

14. Open index.html file again – notice that the new contents added don’t exist.
15. Run the following command to pull the changes that were merged in step 12.

```
git pull
```

EXERCISE 4: RESOLVING GIT CONFLICTS

1. Create two branches from main: `add-nav` and `add-header`
2. In `add-nav` branch (line 6):

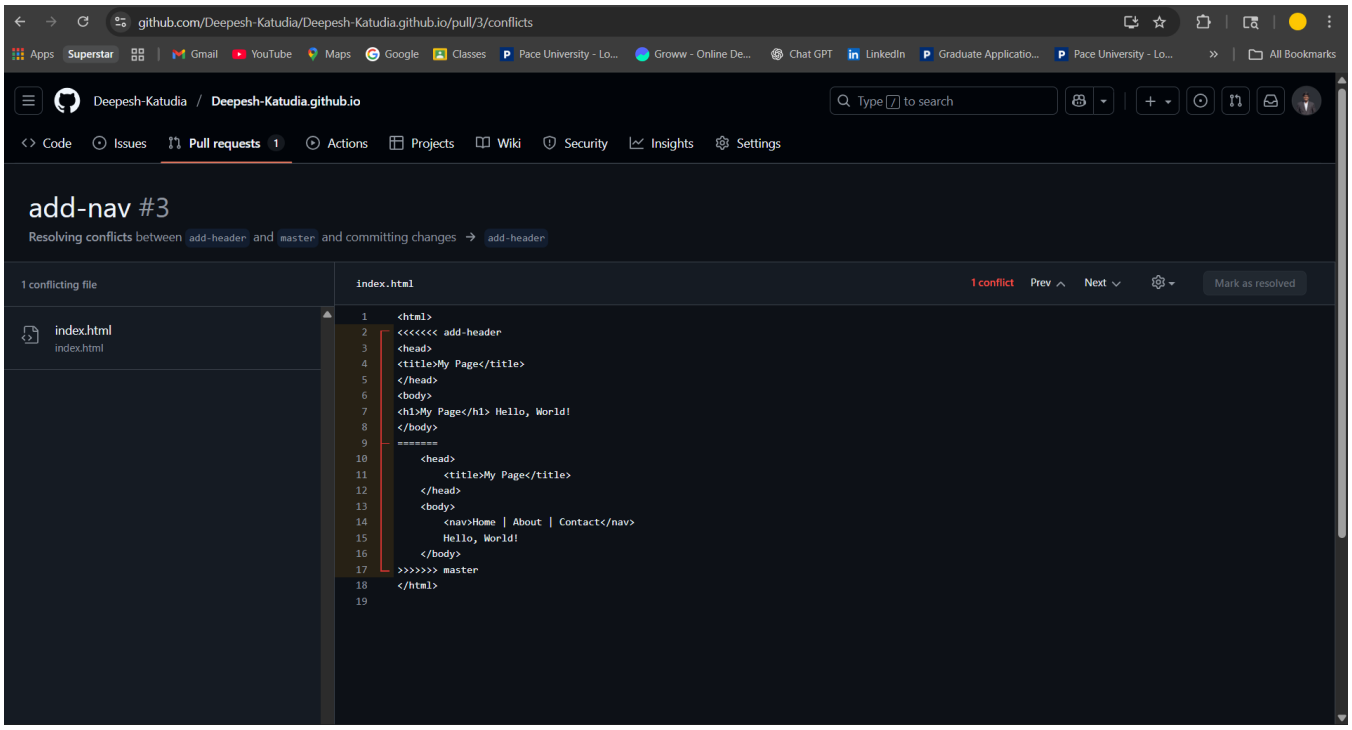
```
<html>
    <head>
        <title>My Page</title>
    </head>
    <body>
        <nav>Home | About | Contact</nav>
        Hello, World!
    </body>
</html>
```

3. In `add-header` branch, modify the same line but with this title:

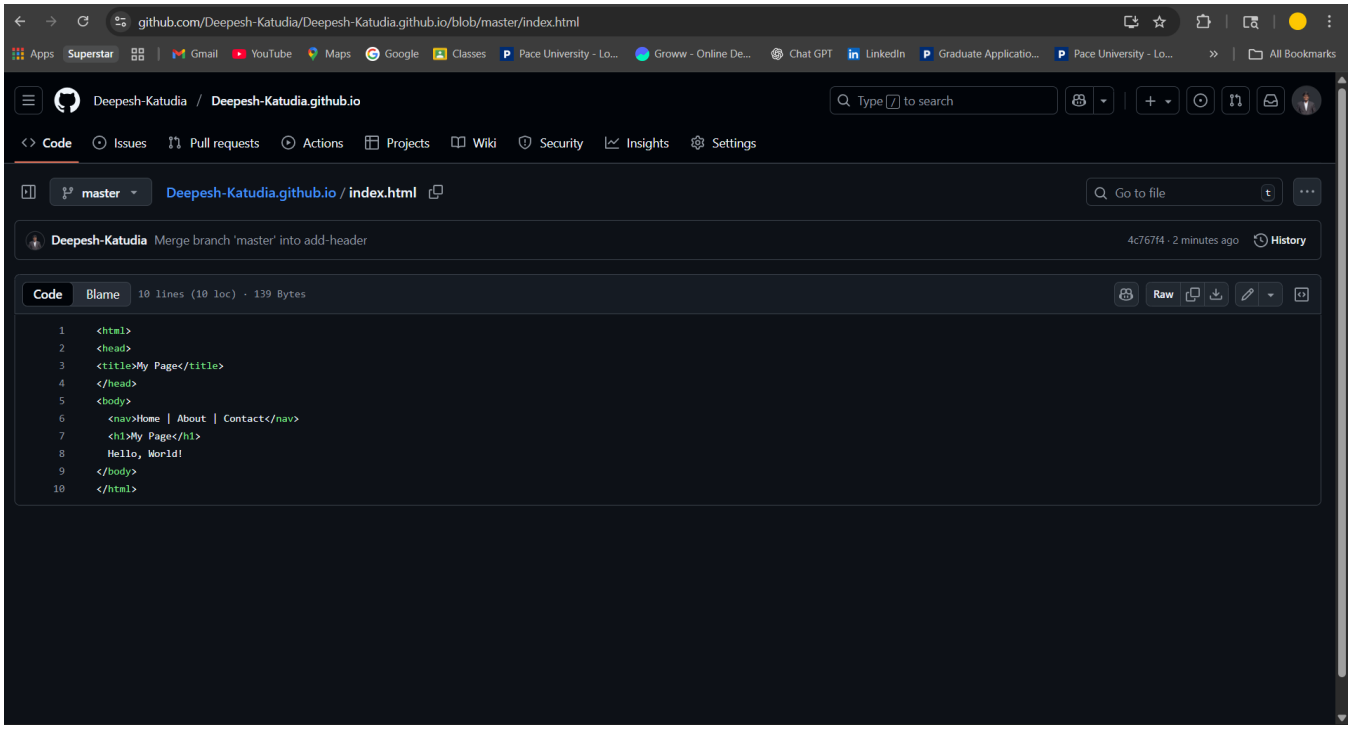
```
<html>
    <head>
        <title>My Page</title>
    </head>
    <body>
        <h1>My Page</h1>
        Hello, World!
    </body>
</html>
```

4. Switch to main branch (git checkout main), merge first branch to main (git merge add-nav), then attempt to merge second branch (git merge add-header); you'll notice an error.
5. Resolve the conflict by keeping both navigation and header elements. See here to understand what all the conflict markers mean.

Question 8: Take a screenshot of the conflict markers in your text editor before resolving.



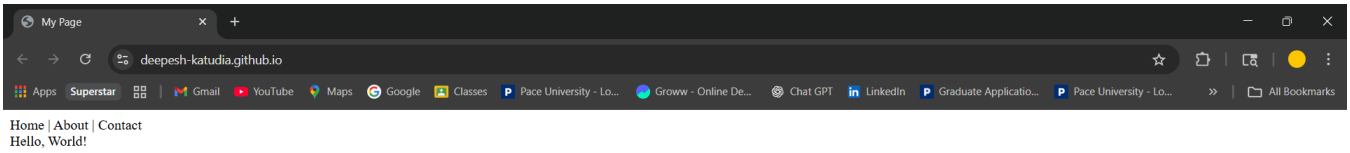
Question 9: Take a screenshot of your final resolved HTML file.



6. Once the conflict is resolved, git add index.html and commit and push the change.

Question 10: Take a screenshot of your website showing both elements working together. Include the website URL here.

- https://deepesh-katudia.github.io/



FURTHER READING

If you are interested in creating a personal portfolio, GitHub pages is the way to go! I highly recommend you spend some time either making a clean HTML page, or using a static site

generator like Jekyll to generate pages dynamically.

ACKNOWLEDGMENT

This assignment is largely copied from Doosan Baik, the former instructor of this course.