CIS641
Fall 2023

Homework 2 (Individual)
Due by: September 27th by 11:59pm
175 points

This homework is also to be done individually. It will serve as (1) getting your team repositories setup, (2) ensuring everybody has the correct access to a repository, (3) getting practice with git, and (4) getting started on your term project.

Prereqs – Get git

As mentioned, everybody is most likely going to have a different computer configuration, so I am going to assume that each of you can install git to your liking on your machines. I will be using terminal-based git (Git for Windows, Linux Subsystem for Windows, etc.), however feel free to use a graphical client of your choosing.

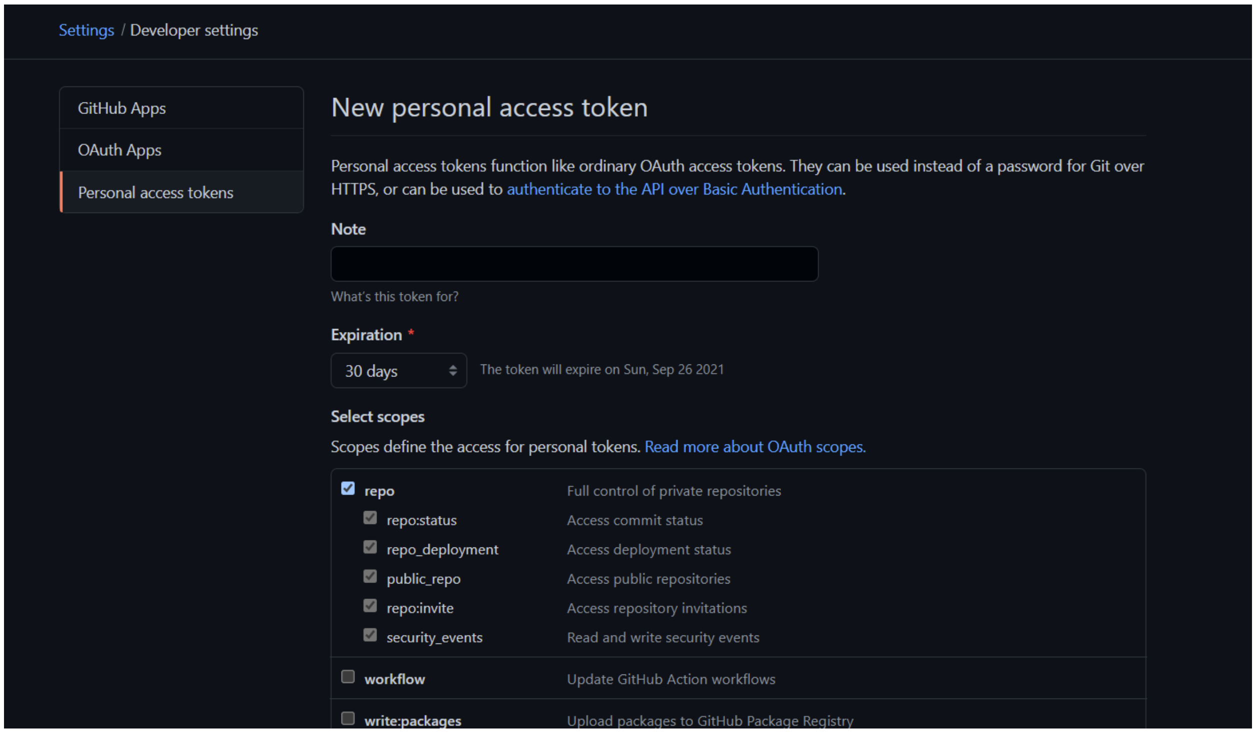
Prereqs PART 2:

Recently, GitHub has moved to a key-only method for logging in via the terminal. You must go to your Settings -> Developer -> Personal Access Tokens OR setup SSH authentication.

Token access guide: https://docs.github.com/en/authentication/authenticating-with-a-passkey/
SSH access guide: https://docs.github.com/en/authentication/connecting-to-github-with-ssh

If going the key route, create one, save it to a password manager (I use PasswordSafe … nice local-only storage), and use it in place of your usual password.

The page looks like this. All you really need are the `repo` permissions.


Settings / Developer settings

GitHub Apps
OAuth Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

What's this token for?

Expiration *

30 days — The token will expire on Sun, Sep 26 2021

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓ repo	Full control of private repositories
✓ repo:status	Access commit status
✓ repo_deployment	Access deployment status
✓ public_repo	Access public repositories
✓ repo:invite	Access repository invitations
✓ security_events	Read and write security events
☐ workflow	Update GitHub Action workflows
☐ write:packages	Upload packages to GitHub Package Registry

You can also set it up to use SSH, though that's a slightly lengthier process
Git guides:

General:	https://github.com/git-guides
The "simple" guide:	http://rogerdudler.github.io/git-guide/

The "Simple" Git Workflow

Refer to this as your high-level guide for a typical process. **I assume you have git installed at this point.**

IF YOU HAVE NO REPOSITORY CLONED TO YOUR MACHINE:

You need to clone a repository to your local machine. Git is distributed in that you'll have a local (your computer) and a remote (GitHub, in this case). You only need to do this the first time (unless if you royally screw things up --- which will happen, don't feel bad if it does).

```
git clone <remote>
```

This creates a local folder, with a hidden folder (.git) that makes it a git repository. Congrats, you now have a local repository. For reference, <remote> is the URL where your repository is hosted. It might be GitHub or a private git server. The process is the same.

ONCE YOU HAVE A LOCAL REPOSITORY CLONED TO YOUR MACHINE:

Now you can work on things locally. Treat it as your local sandbox. You first check to see if any changes exist on your remote (i.e., a teammate made a change), make your personal changes, make sure they work, then send them back up to the remote for your team to use.

First, check to see if there are any updates to the remote repository:

```
git pull
```

This pulls down any local changes. It's important to do this, otherwise you get into things like merge conflicts.

Next, make your changes. This means editing source code, creating documentation, etc. Whatever "work" you are presently doing.

Then, you need to **stage** your changes for committing. This means that you're telling git that you want to track whatever changes you've made. Otherwise, git doesn't know.

The following command has git tracking "all" changes you've made.

```
git add .
```

This command only tracks the files you want it to track.

```
git add <file-1> <file-2> <file-n>
```

Either are fine.

Now it is time to commit your changes. Here, you are effectively making a revision. This is a snapshot of the project at any given point in time. (Usually it's good to tag them with an easy to remember name, but we'll just use the ID for now). The ID is a long hex string that uniquely identifies each revision. You can checkout this revision later if you ever need to go back to a 'working' state.

```
git commit -m 'YOUR COMMIT MESSAGE'
```

Make your commit message short but descriptive. Otherwise, nobody will have any idea what your commit is about.

Last, push your changes up to the remote repository. This means that the rest of your team (and the world, if it is a public repository) can see what you've done.

```
git push
```

And you've done the process. Good job. If you get into merge conflicts or branches obviously you'll need to do a bit more, however this is your process you'll do 99.9% of the time.

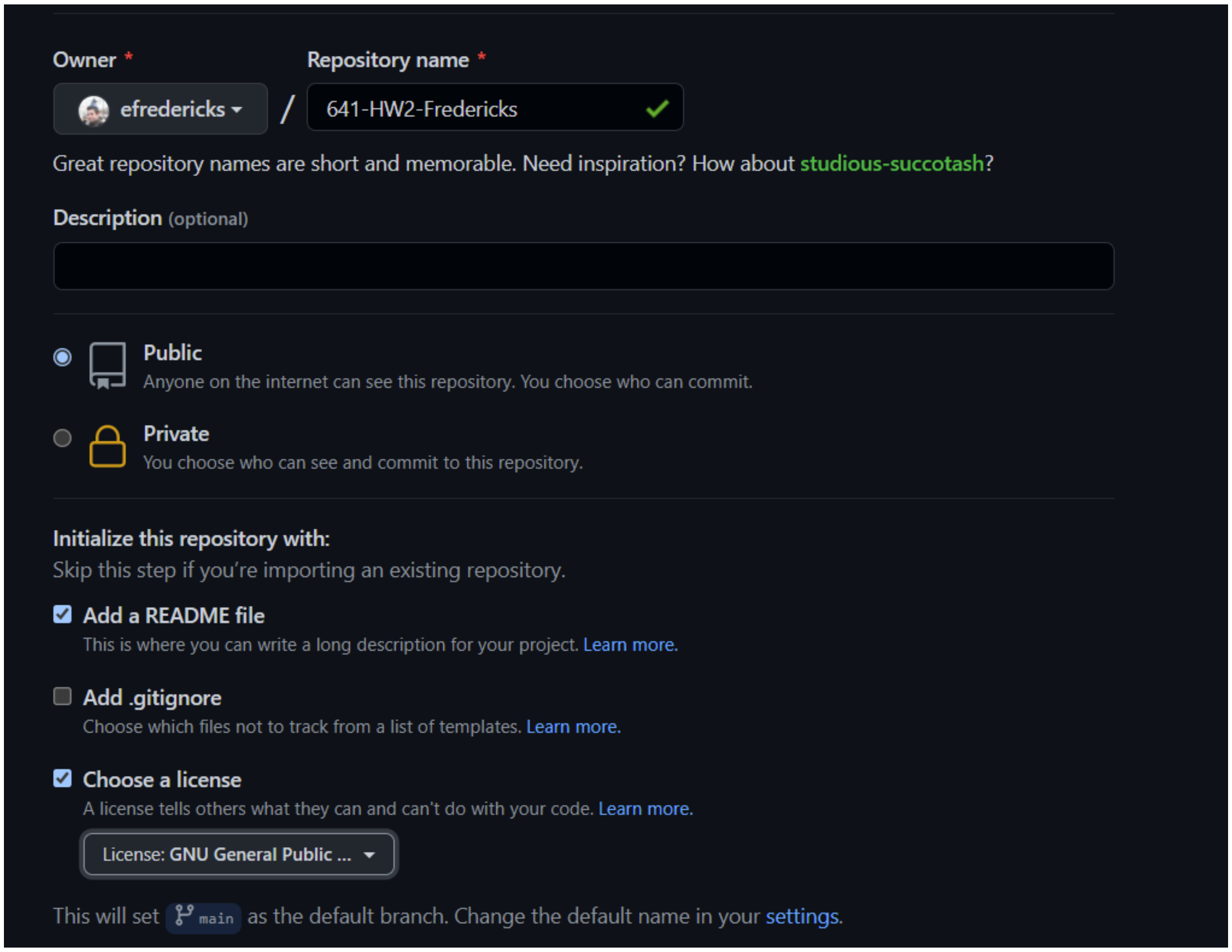
tl;dr:

```
1. git pull
2. Make changes
3. git add .
4. git commit -m 'commit message'
5. git push
```

Lab instructions:

(Individual homework)

1) Using your GitHub account, create a new repository. Name it `CIS641-HW2-<YourLastName>`. Replace <YourLastName> with … your last name obviously. Set it to be Public, add a README by default, and select whichever license you prefer. You don't need a .gitignore for this lab (basically … this tells git what *not* to track). Should look similar to this:



Owner * Repository name *

efredericks / 641-HW2-Fredericks ✓

Great repository names are short and memorable. Need inspiration? How about studious-succotash?

Description (optional)

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

[✓ Add a README file] This is where you can write a long description for your project. Learn more.

Add .gitignore — Choose which files not to track from a list of templates. Learn more.

[✓ Choose a license] A license tells others what they can and can't do with your code. Learn more.

License: GNU General Public …

This will set `main` as the default branch. Change the default name in your settings.

Copy and paste the URL of your GitHub repository in the homework questions at the end.

2) Add me as a contributor, so that I can update it, etc. Go into **Settings → Manage Access → Invite a Collaborator**. My username is efredericks.

3) Clone the repository to your local machine and update the README file. Remember, README files are in Markdown format. Here is a reference for Markdown formatting (https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax). In your README file, add the following (**I will check that styles match exactly**):

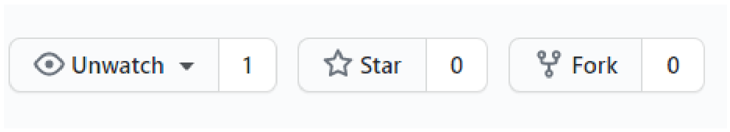
   a. A **section heading** comprising your full name
   b. At least **3 bullet points** describing your interests (either personal, technical, or research-related)
   c. A **sub-section called Technical Background,** where the **content** of the sub-section describes your programming experience (or technical experience)
   d. A **link** to a picture of your favorite meme

4) **Push all your changes back to GitHub** and ensure you're happy with how it looks.

(Term project homework)

5) Nominate one member of your term project team to fork the project repository (https://github.com/gvsu-cis641/base) and assign access to all team members. **Write that person's name in the Lab Writeup.**

6) Whoever was selected should **fork** the following repository. All you have to do is click the 'Fork' button on the repository's webpage here:



   a. Invite all your teammates to be collaborators. You can find this in the 'Settings' page. Unfortunately for personal account repositories there can only be one owner, so ensure that whoever is in charge is up for managing the repository for your team (note, you can transfer ownership if necessary). The invite will show up in whichever email account you used to sign up for GitHub.
   b. Invite me as well (efredericks). I will need to manage some things as well behind the scenes.

7) Each team member should now clone the team repository. **Individually**, add your name to the list of team members in the README.md file, where your name is a link to your HW2 repository. (I should see a list of people that, if I click your name, will take me to your HW2 repo). **Each person is responsible for adding their own name. I will be checking GitHub commits!**

   (Hint: you have to do the git pull/change/add/commit/push steps listed above to do this).

8) Ensure that your project repository is named GVSU-CIS641-<YOURTEAMNAME> (replace <YOURTEAMNAME> with your project team's name). Also ensure that the first heading in your README.md file reflects your team name as well.

9) You now must start tracking your weekly meetings after this assignment is due. You'll see a folder called **meetings** with a template for what I expect to see each week. Each week I will check to see if there is a `minutes-YYY-MM-DD` file that has been correctly filled out in this folder for each team.

10) Last but not least, work together to fill out your Project Proposal. **You will find the template in your docs folder.** This will be a living document in that I will ask for revisions later, however you have some key points for this assignment. Essentially, fill in the blanks in the proposal template with the requisite information (at a grad student level – you are now planning your project).

LAB WRITEUP:

The rubric here is simple. Ensure you've followed all the steps above, making sure that the correct permissions, content, etc., has been added. (This will be split amongst two grade items to reflect individual vs. team efforts).

(Individual points)

1) (40 points) Paste the URL of your personal GitHub repository here:
RE) https://github.com/srigeethavegi99/CIS641-HW2-Vegi

2) *Read Chapter 3 in the textbook and answer the following questions **individually***

(5 points) How can you differentiate between facts and opinions? Why can both be useful?
RE) Facts are such statements that have evidence and can be proven and doesn't vary by the person using it in a conversation. On the other hand, Opinions have no significant evidence and might vary from a person to a person based on their beliefs, these are more of personal interpretations. Facts are used to convey the actual information and are not subject to change over time or situation. Opinions are used to express one's point of view and might change over time or situation.

(10 points) What is the purpose of the requirements definition? What is the difference between a functional and a non-functional requirement?
RE) The purpose of requirements definition is to provide the information regarding what the system is expected to deliver and to understand the scope of the system which means understanding whether it is possible to deliver that requirement output or not. The requirements are grouped into functional and non-functional requirements.
Functional requirements are related to what a system needs to perform while using and what kind of functionality the system should possess whereas Non-functional requirements means how the system should behave under several circumstances, it's more related to performance, stability, reliability etc…

(3.K) (20 points) Review the gvsu.edu/computing website. Develop the requirements definition for the site. Create a list of functional business requirements that the system meets. What different kinds of non-functional business requirements does the system meet? Provide examples for each kind. ***Provide 5 functional and 3 non-functional at minimum.***
RE) Requirements Definition:
a) The site should have a homepage which displays the school of computing's overview.
b) Each module in the overview should have a dropdown which contains the things included in that module and this should be displayed when cursor is clicked on the module.

c) There should be a section which displays the news and events related to the department and have URLs to each article in the news.
d) There should be a search functionality to help the visitors or students to quickly search for a particular thing.
e) There should be a block to help the students to register courses, get information regarding courses etc…

Functional Requirements:
a) search functionality
b) displaying of homepage with URLs
c) course Registration
d) Having modules of what the department offers and able to explore the category with a click.
e) directing to social media pages when we click on the icons.

Non-functional requirements:
a) Images on the website and user experience.
b) Testimonials.
c) accessing the website using URL.

3) (Extra credit (up to 25 points, depending on effort)) Add a basic GitHub pages website to your **personal** repository (the HW2 portion). This part is going to serve as a forced personal portfolio for you. Copy/paste the link to your GitHub.io page here:

(Team project points)

4) (10 points) Which team member is responsible for forking the project repository:
RE) Mani Chandana Gopireddy.

5) (90 points) Paste the URL of your **team** project repository (I will be checking your proposals for this question):
RE) https://github.com/GManichandana/GVSU-CIS641-WE-4