**B.Eng in Software and Electronic Engineering**

**Year 1**

**Web Technologies**

**Lab 6 – Using GIT and GitHub**

#### **1. Create a GitHub account**

In order to use GitHub, you’ll need a GitHub account. You can create a free GitHub account [here](https://github.com/join) and start using GitHub right away.

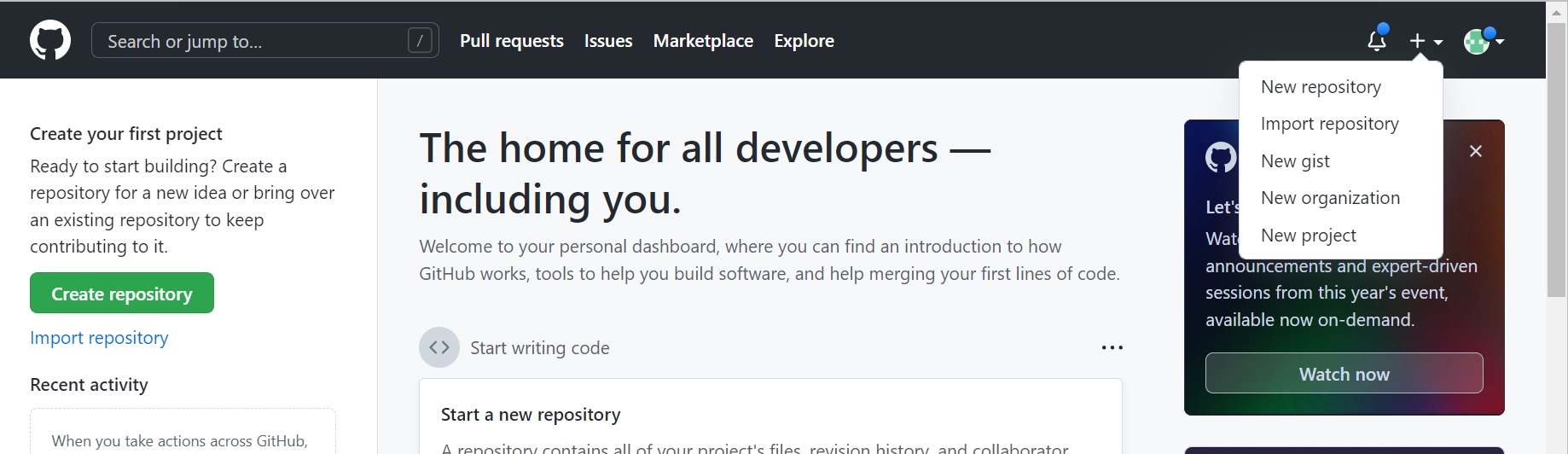
#### Git and GitHub

A quick aside: git and GitHub are **not**the same thing. Git is an open-source, version control tool created in 2005 by developers working on the Linux operating system; GitHub is a company founded in 2008 that makes tools which integrate with git. You do not need GitHub to use git, but you cannot use GitHub without using git. There are many other alternatives to GitHub, such as GitLab, BitBucket, and “host-your-own” solutions such as gogs and gittea. All of these are referred to in git-speak as “remotes”, and all are completely optional. You do not need to use a remote to use git, but it will make sharing your code with others easier.

## **2: Create a new repository on GitHub**

If you only want to keep track of your code locally, you don't need to use GitHub. But if you want to work with a team, you can use GitHub to collaboratively modify the project's code.

To create a new repo on GitHub, log in and go to the GitHub home page. You can find the “New repository” option under the “+” sign next to your profile picture, in the top right corner of the navbar:



After clicking the button, GitHub will ask you to name your repo and provide a brief description:

Graphical user interface, text, application, email

Description automatically generated

***Note****: Public repositories are accessible to everyone on the internet. Private repositories are only accessible to you, people you explicitly share access with, and, for organization repositories, certain organization members.*

When you're done filling out the information, press the 'Create repository' button to make your new repo.

GitHub will ask if you want to create a new repo from scratch or if you want to add a repo you have created locally. In this case, since we've already created a new repo locally (in Section A), we want to push that onto GitHub so follow the '....or push an existing repository from the command line' section:

**Graphical user interface, application

Description automatically generated**

i.e. do the following

**Text

Description automatically generated**

**git remote add origin https://github.com/Mairtin1974/NewRepo.git**

I want to set up a remote location and I want to give it an alias of ***origin***, so in future I won’t need to type in the full URL, ***origin*** will do!!!!

**Note**: *Make sure to change the URL in the first command line to what GitHub lists in this section since your GitHub username and repo name are different.*

**git branch -M main**

Master branch is changed to main. Why? Up to about 2 years ago, GitHub used **master** but now it uses **main**!! Git is **master** and GitHub is **main**!!! Make both the same, makes things easier.

**git push -u origin main**

Pushes all the code to the remote/GitHub repository (URL = origin)

-u means your branch is up to date with origin main

On pushing, it will prompt you to login!!

Sign in with your GitHub account and click on **‘Authorize GitCredentialManager’**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

The goal of [**Git Credential Manager (GCM)**](https://aka.ms/gcm) is to make the task of authenticating to your remote Git repositories easy and secure, no matter where your code is stored or how you choose to work.

It needs to know who you are /where to push the code before it pushes the code to your repository!!

## **3: Push a branch to GitHub**

Now we'll push the commit in your branch to your new GitHub repo. This allows other people to see the changes you've made. If they're approved by the repository's owner, the changes can then be merged into the primary branch.

To push changes onto a new branch on GitHub, you'll want to run

[**git push**](http://git-scm.com/docs/git-push)**-u origin yourbranchname**.

For me this command will look like the following

git push origin feature1

GitHub will automatically create the branch for you on the remote repository:

Text

Description automatically generated

(If this is your first time using GitHub locally, it might prompt you to log in with your GitHub username and password.)

If you refresh the GitHub page, you'll see note saying a branch with your name has just been pushed into the repository. You can also click the 'branches' link to see your branch listed there.

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

Now click the green button in the screenshot above. We're going to make a **pull request**!

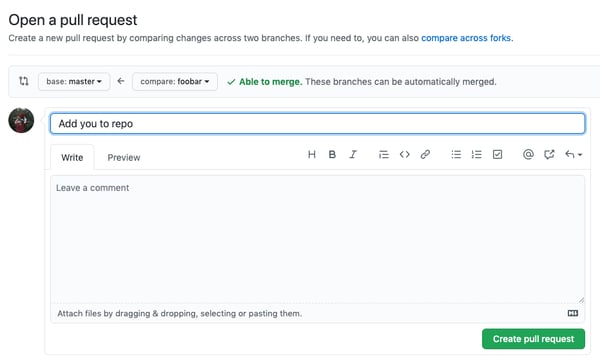
## **4. Create a pull request (PR)**

A pull request (or PR) is a way to alert a repo's owners that you want to make some changes to their code. It allows them to review the code and make sure it looks good before putting your changes on the primary branch.

This is what the PR page looks like before you've submitted it:

Graphical user interface, text, application, email

Description automatically generated

[[](https://cloud.githubusercontent.com/assets/5241432/9189500/4688c07e-3fb7-11e5-99ed-d75b50ed9e48.png)](https://cloud.githubusercontent.com/assets/5241432/9189500/4688c07e-3fb7-11e5-99ed-d75b50ed9e48.png)

And this is what it looks like once you've submitted the PR request:

Graphical user interface, text, application, email

Description automatically generated

You might see a big green button at the bottom that says **'Merge pull request'**. Clicking this means you'll merge your changes into the primary branch.

Sometimes you'll be a co-owner or the sole owner of a repo, in which case you may not need to create a PR to merge your changes. However, it's still a good idea to make one so you can keep a more complete history of your updates and to make sure you always create a new branch when making changes.

## **5:Merge a PR**

Go ahead and click the green 'Merge pull request' button. This will merge your changes into the primary branch after you hit **Confirm merge**.

Graphical user interface, text, application, email

Description automatically generated

**When you're done, I recommend deleting your branch (too many branches can become messy), so hit that grey 'Delete branch' button as well.**

**Graphical user interface, text, application, email

Description automatically generated**

You can double check that your commits were merged by clicking on the 'Commits' link on the first page of your new repo.

This will show you a list of all the commits in that branch. You can see the one I just merged right up top (Merge pull request #1).

You can also see the [hash code](https://git-scm.com/docs/git-hash-object) of the commit on the right hand side. A hash code is a unique identifier for that specific commit. It's useful for referring to specific commits and when undoing changes (use the [git revert](http://git-scm.com/docs/git-revert)<hash code number> command to backtrack).

A picture containing application

Description automatically generated

## **6: Get changes on GitHub back to your computer**

Right now, the repo on GitHub looks a little different than what you have on your local machine. For example, the commit you made in your branch and merged into the primary branch doesn't exist in the primary branch on your local machine.

In order to get the most recent changes that you or others have merged on GitHub, use the git pull origin master command (when working on the primary branch). In most cases, this can be shortened to “git pull”.

Text

Description automatically generatedThis shows you all the files that have changed and how they've changed.

Now we can use the [git log](http://git-scm.com/docs/git-log) command again to see all new commits.

# **7. Inviting collaborators to a personal repository**

You can invite users to become collaborators to your personal repository.

If you're using GitHub Free, you can add unlimited collaborators on public and private repositories.

Repositories owned by an organization can grant more granular access.

Pending invitations will expire after 7 days, restoring any unclaimed licenses.

**Note:** GitHub limits the number of people who can be invited to a repository within a 24-hour period. If you exceed this limit, either wait 24 hours or create an organization to collaborate with more people.

1. Ask for the username of the person you're inviting as a collaborator. If they don't have a username yet, they need to sign up for GitHub
2. On GitHub.com, navigate to the main page of the repository.
3. Under your repository name, click  **Settings**.
4. In the "Access" section of the sidebar, click **Collaborators**.

Graphical user interface, text, application, email

Description automatically generated

1. Click **Add people**.
2. In the search field, start typing the name of person you want to invite, then click a name in the list of matches.

Graphical user interface, text, application

Description automatically generated

1. Click **Add NAME to REPOSITORY**.

Graphical user interface, application

Description automatically generated

1. The user will receive an email inviting them to the repository. Once they accept your invitation, they will have collaborator access to your repository.

**8. When the user accepts….**

They now have access to the project. Try this with someone beside you in the lab.

Once you have access to their repository on GitHub you can click on Code and copy the URL for the project.

Then create a new folder for this project, open up Git Bash in the folder and type the following

$ git clone <Paste the URL here >

and type ls, git status and git log to see what we have!

Git clone is a command for downloading existing source code from a remote repository like Github. In other words, Git clone basically makes an identical copy of the latest version of a project in a repository and saves it to your computer.

Now we need to make changes to this project. Create a new branch, **ALWAYS** create a new branch to make changes.

You will have to run git config to add your user name and email address.

Push all the changes to your friends repository using a command similar to the following:

git push -u origin moconghaile

And then as before we must now **Create a Pull Request** to merge our changes onto the main branch This can be done by you or the invited user, depending on how we set administrative rights.

**Exercise – Part 1**

1. Create a local repository and add/commit three files.
2. Push the repository to your GitHub account. (You need a Github account)
3. Create a branch using your id as the name for the branch. Add/commit a file to the branch.
4. Push all the changes on this branch to the same branch on your Github repository.
5. Create a pull request to push all the changes on your branch to the main branch on your github repository.

**Exercise – Part 2**

1. Make friends with someone and ask if they can add you as a collaborator to their project.
2. Make a clone of their project on your pc.
3. Add a branch to the cloned repository using your id as the branch name.
4. Add/commit some files to the branch.
5. Push your branch to your new friends repository
6. Create a pull request to add your changes in your branch to their main branch
7. Ask your new friend to approve this request

**Submission :** Submit the two URL links

1. GitHub repository used for Part 1
2. GitHub repository used for Part 2
3. ghp\_ioONz9itcI0IUMROsX0Tmv3WcnVwBR1H21CY