Technical Documentation CS 3300

[Creating a GitHub Repo from an existing local folder](#_Toc158403260)

[Django Portfolio App setup](#_Toc158403261)

[Branching](#_Toc158403262)

[Quick Commands](#_Toc158403263)

[Virtual Env Setup](#_Toc158403264)

[Manage project in VS Code](#_Toc158403265)

[Git & Github setup](#_Toc158403266)

[Creating a GitHub repo on the website](#_Toc158403267)

[Add a .gitignore file to the new repo](#_Toc158403268)

[Django book: patterns](https://learning.oreilly.com/library/view/django-design-patterns/9781783986644/ch01s03.html)

[Django tutorial](https://www.w3schools.com/django/django_views.php)

## Creating a GitHub Repo from an existing local folder

Resources:

<https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github>

<https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository>

git init

echo “# A GitHub repo” >> README.md

git add .

git commit -m “Initial commit”

git branch -M main

git remote add main <URL of GitHub repo>

git push

**To get the URL of the GitHub repo, you need to create a repo on the website:**

Go to GitHub.com and sign in. Click on Repositories and click on the New button.

A screenshot of a computer

Description automatically generated

Name the repo



Make it private

A black background with white text

Description automatically generated

Do not select add a README file and do not add a .gitignore or license. Just leave those options as they are.

Click “Create repository”

Find the SSH link under Quick setup and use that in the

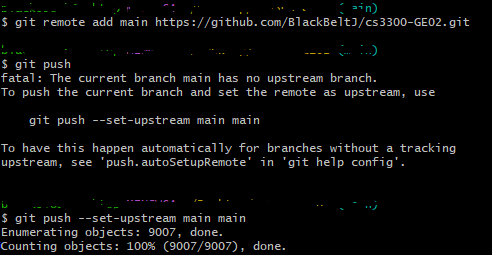
git remote add main <URL of GitHub repo>

command above

A screenshot of a computer error

Description automatically generated

**NOTE: You may have to set the upstream branch before you can push to the new repo**



## Django Portfolio App setup

## Branching

1. From the command line in your repository on your computer check the log and what branch you are on.
2. Create a branch called sprint01 and check the log and branch

Copy and paste the commands you used

|  |
| --- |
| A black background with white text  Description automatically generated <- to check the current branches  git branch sprint01 <- to create a branch  A screen shot of a computer  Description automatically generated <- to check the branches again  git switch sprint01 <- to switch branches  A black screen with white text  Description automatically generated <- to verify switched branch |

1. Switch to sprint01 branch to check out code:

git checkout 'sprint01'

git branch

git status

1. Modify python file and Add the file to the staging area and update the version in your local directory.

Copy and paste the command(s) you used

|  |
| --- |
| git add .  git commit -m “Modified GE01.py on sprint01 branch” |

1. Share the changes with the remote repository on the new sprint01 branch. Go to your github and you will see you now have two branches. Click to view the branches. Now others working on the branch could pull your updates from the sprinto1 branch.

git push --set-upstream origin sprint01

git status

git log

A screenshot of a computer

Description automatically generated

1. Switch to the main branch and update the remote main branch repository with the change from sprint01 branch. Then go to github to see the versioning.

Copy and paste the commands you used

|  |
| --- |
| git push --set-upstream main sprint01  git branch  git status  git log |

1. Tag the main branch ‘v1.0’ then view the tag and push to the remote repository. When you go to the remote repository you should see the tag listed.

Copy and paste the commands you used

|  |
| --- |
| git tag v1.0  git tag -l <- to verify the tag was created  git push main –tags  A screenshot of a computer  Description automatically generated  A screenshot of a computer  Description automatically generated |

For example

A screenshot of a computer

Description automatically generated

## Quick Commands

To activate a virtual environment (must be run from portfolio folder):

source <venv name>/Scripts/activate

To start web server (must be run from within portfolio folder):

py -3 manage.py runserver

Server url: <http://localhost:8000/>

## Virtual Env Setup

**NOTE: for all Python commands on Windows, use py -3 instead of python3 as in the GE.**

Open a command/terminal window (I used GitBash) and check your Python version.

Python version command for Windows:

# for Windows

py -3 --version

Somewhere on your computer create a folder called cs3300 (I created mine in my cs3300-version-control Git repo from the first GE). Then make a portfolio folder for this GE.

# run all these commands in order

mkdir cs3300

cd cs3300

mkdir portfolio

cd portfolio

Then create a virtual environment. If you want more information on Python virtual envs, check this article: [Python Virtual Environments](https://python.land/virtual-environments/virtualenv). After creating the venv (virtual environment), activate it. **You need to be in the cs3300 directory to have this command work correctly. If you are in the djvenv folder, it will not work.**

Note: On windows assuming you are not using bash or mingw you will not be able to use “source activate” and will need to instead run “./venv/activate.bat’

# create venv

py -3 -m venv djvenv

# activate it

source djvenv/Scripts/activate

# NOTE: in GE, the command is "source djvenv/bin/activate" which does not work, # you need to change the "bin" to "Scripts" like the command above

You should see something like this in the command line if you have activated the venv correctly:



Then install django in the venv with this command: **Make sure to specify the version 4.2**

# specify version 4.2 with django==4.2

pip install django**==4.2**

# upgrade pip

py -3 -m pip install --upgrade pip

# also run this to be safe

python.exe -m pip install --upgrade pip

# create django proj

django-admin startproject django\_project

Reorder directory structure for ease of use

Change from A screenshot of a computer

Description automatically generated

to

A screenshot of a computer

Description automatically generated

# you can do the same thing with these three commands

mv django\_project/manage.py ./

mv django\_project/django\_project/\* django\_projectrm -r django\_project/django\_project/

(Basically moving the manage.py file and django\_project folder back one directory into the portfolio folder)  
Portfolio folder should now look like this:

A screenshot of a computer

Description automatically generated

# run server

python manage.py runserver

# if that command doesn't work use this

py -3 manage.py runserver

A screen shot of a computer

Description automatically generated

To verify the installation was successful, go to <http://localhost:8000/> or <http://127.0.0.1:8000/> and you should see this:

A screenshot of a computer

Description automatically generated

Note: It may still say Django 5.0 on the top right even if you have version 4.2 installed. To verify django version:

django-admin --version



**To stop the server, go back to your command/terminal and type** CTRL+C**.**

Open another terminal and activate the venv there with:

source djvenv/Scripts/activate

Once in the venv, create a requirements file of [what is installed](https://note.nkmk.me/en/python-pip-list-freeze/):

pip freeze > requirements.txt

### Manage project in VS Code

Open VSCode (Or your IDE) and open the folder that contains the django project. In VS Code go to ‘File -> Open Folder’ and select the portfolio folder.

A screenshot of a computer

Description automatically generated

## Git & Github setup

NOTE: The GE document says to create a new private github repo to store the django project. However, I believe we all already have a private github repo from the first GE called cs3300-version-control so I just created the portfolio and subsequent django project in that existing repo. If this is not ok, I will update the documentation, but for now I did not have to re-create a new private repo for the django project.

**If you still want to create a new repo, here’s how:**

Open the folder where your portfolio folder is, right click and select ‘Open Git Bash here’ or open your preferred terminal/command line interface.

Type these commands

git init # initializes a repository in the current folder

echo "#CS3300 Private Repo" >> README.md # created a README file

git add . # adds all files to the staging area

git commit -m "Initial commit"

git branch -M main # creates a main branch

git remote add main <URL of GitHub repo>

git push

You will need to go to GitHub.com and create a repository there to get the repository URL.

## Creating a GitHub repo on the website

Resources:

<https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github>

<https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository>

**To get the URL of the GitHub repo, you need to create a repo on the website:**

1. Go to GitHub.com and sign in. Click on Repositories and click on the New button.
2. Name the repo and make it private
3. Do not select add a README file and do not add a .gitignore or license. Just leave those options as they are.
4. Click “Create repository”
5. Find the SSH link under Quick Setup and use this command from earlier A screenshot of a computer

   Description automatically generated

git remote add main <URL of GitHub repo>

1. Then you can push your local repo

**NOTE: You may have to set the upstream branch before you can push to the new repo**

**A computer screen with white text

Description automatically generated**

### [Add a .gitignore file to the new repo](https://docs.github.com/en/get-started/getting-started-with-git/ignoring-files)

Use this article to add and format a .gitignore file

touch .gitignore

vi .gitignore

Once in the vi editor

* 1. press the “i” key to enter the Insert mode

1. and paste (right click and select paste) the contents of [this file](https://github.com/github/gitignore/blob/main/Python.gitignore) in.
2. Then find the # Environments section and add “/djvenv” or /”whatever you called your virtual environment”

A screen shot of a computer

Description automatically generated

1. Then press the “Esc” key, you should see the “Insert” at the bottom go away
2. and then the colon “:” key
3. followed by “wq” and hit enter.

A screenshot of a computer

Description automatically generated

1. That combination of keys will escape you from Insert mode, open the command window, and wq will “write” and then “quit” or save the file. After that you should be back to the normal terminal screen. Check [this article](https://stackoverflow.com/questions/39933600/how-to-ignore-folder-in-github-correctly) if a file is not being ignored.