

GE02 Django Portfolio App Set Up

Purpose:

- Communicate effectively in a variety of professional contexts within a team, doing oral or written presentations, and creating technical documents.
- Function effectively as a member/leader of a team engaged in scrums while participating in different roles
- Apply various tools and agile principles utilizing concepts (scrum, behavior-driven development, pair programming, version control)
- Apply computer science theory such as utilizing design patterns for software architecture, higher-order functions, metaprogramming, to improve the maintainability, modularity and reusability to build a SaaS application

Effort: Collaborative Assignment see [CS3300 Academic Integrity](#)

Points: 40 (see rubric in canvas)

Deliverables: Each person submits their own word/pdf document of the GE with their own answers. You will also submit a separate document for your professional communication of your learnings so far. DO NOT SUBMIT A ZIP FILE.

Due Date: See Canvas

[1 Team Information](#)

[2 Set Up Portfolio App \(Individual\)](#)

[2.1 Create Virtual Environment and Django Project](#)

[2.2 Create Local Git and Github Repository](#)

[2.3 Create Portfolio App](#)

[2.4 Define URI path and view](#)

[2.5 Create HTML Template](#)

[2.6 Add static files](#)

[3 Apply Technologies \(Individual\)](#)

[4 Professional Communication of Technical Content \(Individual\)](#)

Description: Explore setting up your first SaaS app using the django framework to build a portfolio app by reading documentation, troubleshooting, collaborating in your team and asking detailed questions when you need help outside of your team. Start building your understanding of Client-Server model, HTTP, URIs when developing SaaS apps.

Try setting up your environment on your own first. If an issue arises, spend 20 minutes trying to troubleshoot on your own. Then collaborate with your team. If no one on the team can help then reach out to the appropriate discord channel. Provide the following details if you want someone to help you.

- What step did you do when something went wrong?
- What have you tried so far?
- What resources did you already use to problem solve?

1 Team Information

Read your assignment for the GE02 Team Roles in Canvas before you begin.

<p>What is your role?</p> <p>Team member</p>
<p>What is your responsibility for this GE02 Sprint's Documentation?</p> <p>My responsibilities are contribute on the team's technical document for define URLpath and view, and create HTML template</p>

2 Set Up Portfolio App (Individual)

Some Resources to help you get started

[Setting up a Django development environment](#) and [Django Getting Started](#) discuss installations depending on OS.

I tried both of these virtual environments but find the first one to be easier to manage and is the one shown in steps below

- <https://python.land/virtual-environments/virtualenv> - this is the one that I included in this documentation
- <https://virtualenvwrapper.readthedocs.io/en/latest/index.html>

2.1 Create Virtual Environment and Django Project

These instructions show examples of working on a Mac for the commands. Note how you must run the commands on your environment.

○

```
BASH

# Linux/macOS
python3 -m django --version

# Windows
py -3 -m django --version
```

1. Open the command/terminal window and check your python version. You should be using 3.11.

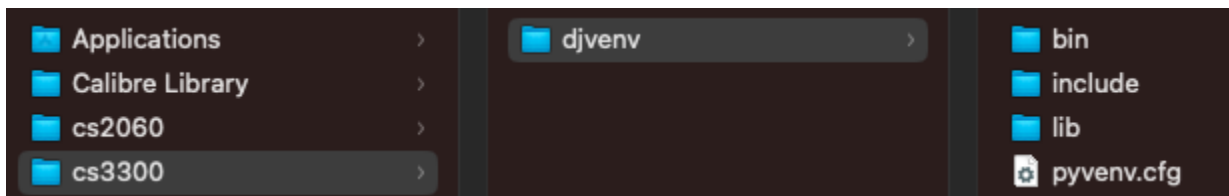
```
debteach@Debs-MBP-2 ~ % python3 --version
Python 3.11.4
debteach@Debs-MBP-2 ~ %
```

2. Create cs3300 folder then create portfolio folder for GE project

```
mkdir cs3300
cd cs3300
mkdir portfolio
cd portfolio
```

3. Create virtual env

```
python3 -m venv djvenv
```



4. Activate virtual environment

```
source djvenv/bin/activate
```

5. Install django in virtual environment

```
pip install django
```

```
(djvenv) debteach@Debs-MBP-2 portf_django % pip install django
Collecting django
  Using cached Django-4.2.4-py3-none-any.whl (8.0 MB)
Collecting asgiref<4,>=3.6.0 (from django)
  Using cached asgiref-3.7.2-py3-none-any.whl (24 kB)
Collecting sqlparse>=0.3.1 (from django)
  Using cached sqlparse-0.4.4-py3-none-any.whl (41 kB)
Installing collected packages: sqlparse, asgiref, django
Successfully installed asgiref-3.7.2 django-4.2.4 sqlparse-0.4.4

[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: pip install --upgrade pip
```

6. Upgrade pip

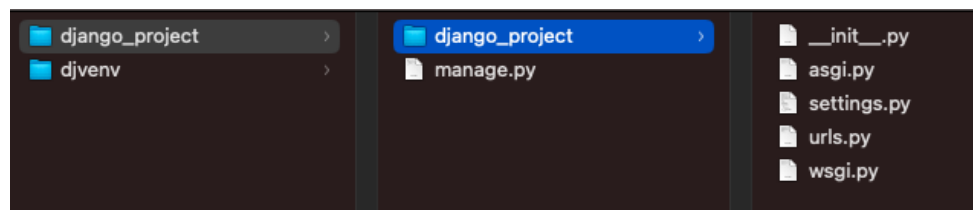
```
python3 -m pip install --upgrade pip
```

7. Create django project

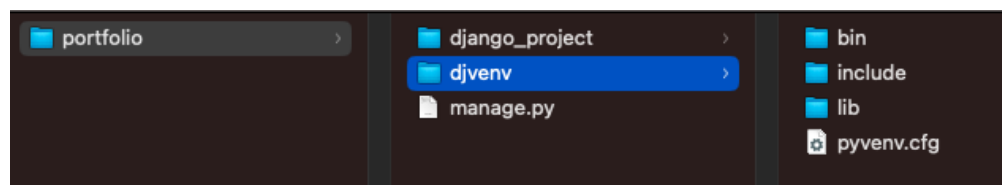
```
django-admin startproject django_project
```

8. Reorder directory structure for ease of use

Change from



to



```
mv django_project/manage.py ./
mv django_project/django_project/* django_project
rm -r django_project/django_project/
```

9. Run server and ignore migration warnings.

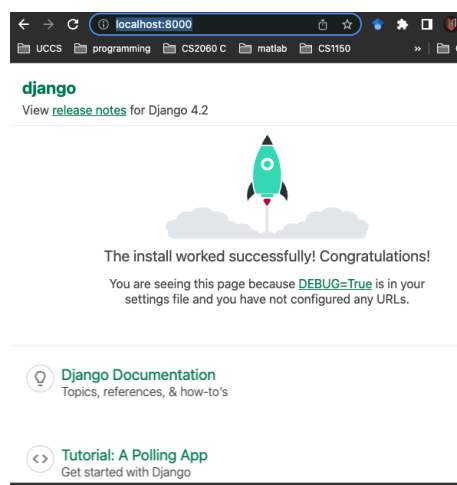
```
python manage.py runserver
```

```
(djvenv) debteach@Debs-MBP-2 django_dev % python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you
apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 05, 2023 - 19:18:26
Django version 4.2.4, using settings 'django_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

10. See if the installation worked by going to <http://localhost:8000/> and you should see. Celebrate!



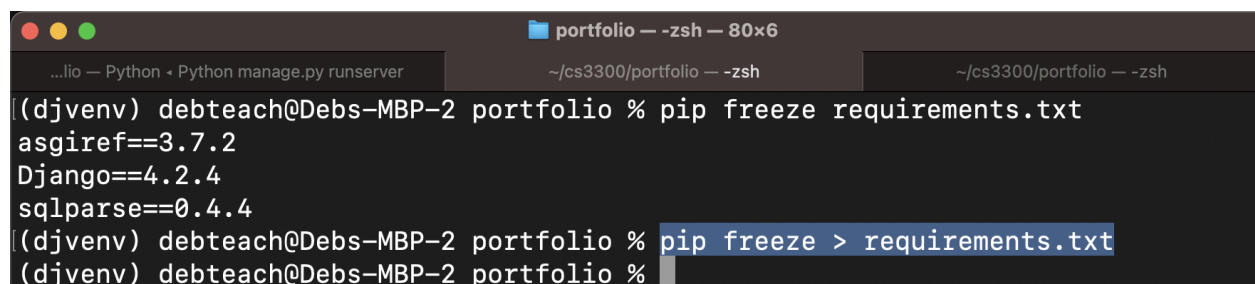
11. Open the second terminal and activate the virtual environment.

Tip: I like to have 3 terminal windows tabbed: one for

- venv to run commands
- venv starting server and stopping server
- git/github commands

12. From virtual environment create requirements file of what is installed [Check all installed Python packages with pip list/freeze](#)

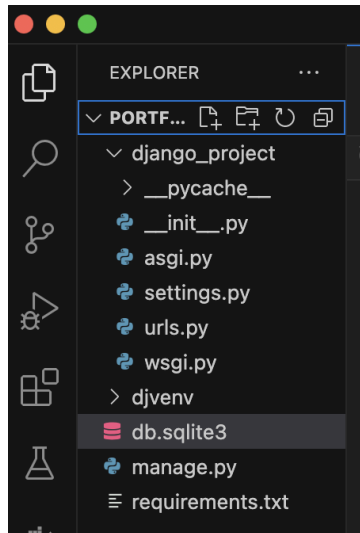
`pip freeze > requirements.txt`



Manage Project in VS Code

You can use a different IDE but I will be demonstrating with VS Code.

Open the project in Visual Studio Code: In Visual Studio Code, go to `File -> Open Folder` and select the folder that contains your Django project.



2.2 Create Local Git and Github Repository

Once your app is working you want to create a GitHub Repository from an existing Local directory/folder in your computer. This is different from the team repository. This initial versioning will be on the main branch. Search for resources to help you create your repos from an existing local directory. Read information below for using git and github for versioning your django project

- Store your code in github as a private repository.
- You will need to add a git ignore file so that when committing it doesn't check certain files in. See [Ignoring files - GitHub Docs](#) and then you can use this example for [Python.gitignore](#). Here are some updates I made to my file
 - For # Environments I added /djenv to ignore my virtual environment
 - the mac I added .DS_Store to my gitignore file.
- **Note:** Deb made mistakes when doing the gitignore and had to search for help on why it didn't ignore something.
<https://stackoverflow.com/questions/39933600/how-to-ignore-folder-in-github-correctly>

2.3 Create Portfolio App

You will add the next part for your portfolio app in a branch called Sprint01.

Here you will create a portfolio app for your project. Remember your indenting when editing python code.

1. Activate your virtual environment
2. From the django-portfolio directory run the following command to create the portfolio app. Explore what is created in the portfolio_app folder (open in VS code or your IDE).

```
django-admin startapp portfolio_app
```

3. Configure Settings: Edit your django_project's settings.py file, add your app to the INSTALLED_APPS list and add support for authenticating users:

```
INSTALLED_APPS = [  
    # ...  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'Django.contrib.staticfiles',  
    # Add your app name here  
    'portfolio_app',  
]  
  
# Add support for authenticating users  
AUTHENTICATION_BACKENDS = [  
    'django.contrib.auth.backends.ModelBackend',  
]  
]
```

2.4 Define URI path and view

Resource: Watch [URLS and Views | Django Framework \(3.0\) Crash Course Tutorials \(pt 2\)](#). Read [Django Tutorial Part 5: Creating our home page - Learn web development](#) and [Django Tutorial Part 5: Creating our home page - Learn web development](#) just to get some background information but the project directions are below.

- Open django_project urls.py to add a path below the admin path to include the specific urls that will be created in the the portfolio_app urls.py. You need to import include.

```
from django.contrib import admin  
from django.urls import path, include
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    #connect path to portfolio_app urls
    path('', include('portfolio_app.urls')),
]
```

- Update portfolio_app/views.py views by defining the following view for the home page.

```
from django.shortcuts import render
from django.http import HttpResponse
# Create your views here.
def index(request):

    # Render the HTML template index.html with the data in the
    # context variable.
    return HttpResponse('home page')
```

- Create urls.py file in portfolio_app that contains a path to the defined view

```
from django.urls import path
from . import views

urlpatterns = [
    #path function defines a url pattern
    #'' is empty to represent based path to app
    # views.index is the function defined in views.py
    # name='index' parameter is to dynamically create url
    # example in html <a href="{% url 'index' %}">Home</a>.
    path('', views.index, name='index'),
]
```

- Run Start server command

```
python manage.py runserver
```

- Open <http://127.0.0.1:8000> and you should see home page



- Update portfolio_app/views.py views by defining the following view for the home page.

```
from django.shortcuts import render
```



```

from django.http import HttpResponseRedirect
# Create your views here.
def index(request):
# Render index.html
return render( request, 'portfolio_app/index.html')

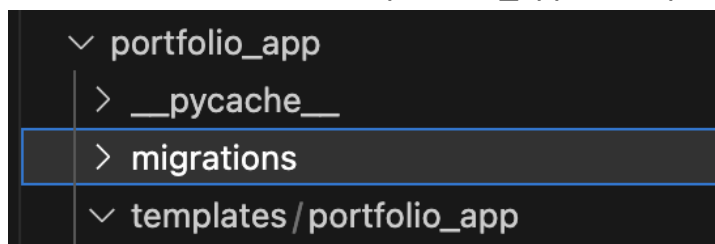
```

- Open <http://127.0.0.1:8000> What do you think is the reason for the issue? In the next part you will fix it.
- Version on your sprint01 branch and update your remote repository.

2.5 Create HTML Template

Django has a certain configuration for finding html templates. Set up the following. templates

1. Create folder called templates in portfolio_app
2. Create folder called portfolio_app in templates



3. Create a **base_template.html** file in the **templates/portfolio_app** folder that will contain the html needed on every page, such as the navigation menu. Paste the following into it. Notice the django [block tags](#) `{% block content %}{% endblock %}` that allow for inheritance. The HTML for the nav bar is from [Navbar · Bootstrap v5.3](#). Later you will learn how to implement bootstrap to create an html page. Do not worry if you do not understand the HTML code given to you below.

```

{% load static %}

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>UCCS CS Students</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <title>Bootstrap demo</title>
    <link
href="https://cdn.jsdelivrivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap
.min.css" rel="stylesheet"
integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGChEKQON+PtmoHDE
XuppvnDJzQIu9" crossorigin="anonymous">
  </head>

  <body>
    <div class="container-fluid">

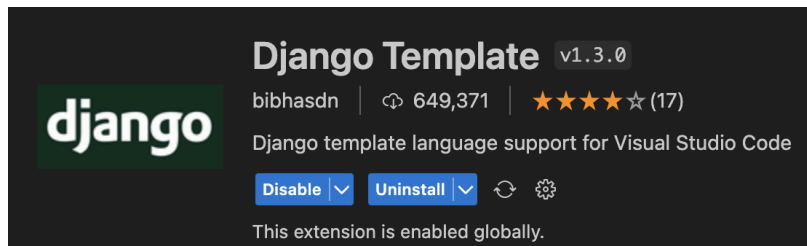
      <!-- Navbar-->
      <nav class="navbar navbar-expand-lg bg-body-tertiary">
        
        <div class="container-fluid">
          <button class="navbar-toggler" type="button"
data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup"
aria-controls="navbarNavAltMarkup" aria-expanded="false"
aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse"
id="navbarNavAltMarkup">
            <div class="navbar-nav">
              <!-- {% url 'index' %} is defined in url path to
dynamically create url -->
              <a class="nav-link active" aria-current="page" href="{%
url 'index' %}">Home</a>
              <a class="nav-link" href="#">Menu 2</a>
              <a class="nav-link" href="#">Menu 3</a>
              {% if user.is_authenticated %}
              <a class="nav-link" href="{% url 'logout' %}?next={{
request.path }}">Logout {{user}}</a>
              {% else %}
              <a class="nav-link" href="{% url 'login' %}?next={{
request.path }}">Login</a>
              {% endif %}
            </div>
          </div>
        </div>
      </nav>

      <div class="col-sm-10">
        <!-- add block content from html template -->
        {% block content %}
        {% endblock %}

```

```
</div>
</div>
</div>
</body>
</html>
```

I suggest exploring how to format your code easily. For example, I am using the extension Django Template in VS code.



When I set it up it didn't work at first so I found this solution and it worked.

<https://stackoverflow.com/questions/42170561/vscode-html-autoformat-on-django-template>

4. Create an **index.html** file in the templates/portfolio_app folder to be the home page for the app.

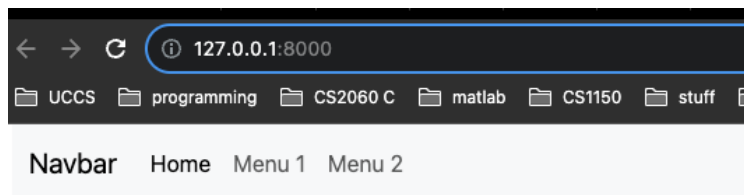
```
<!-- inherit from base.html -->
{% extends "portfolio_app/base_template.html" %}

<!-- Replace block content in base_template.html -->
{% block content %}
<h1>Computer Science Portfolios</h1>

<h2>More to come from [and your name goes here]</h2>

{% endblock %}
```

5. Restart our server and open <http://127.0.0.1:8000> and you should see your index.html using the base_template.html. For example.



Computer Science Portfolios

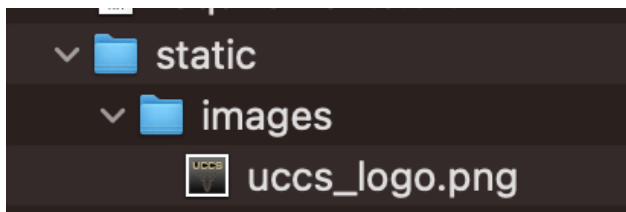
6. Version on your sprint01 branch and update your remote repository.

2.6 Add static files

Resources:

- [Static Files & Images | Django Framework \(3.0\) Crash Course Tutorials \(pt 4\)](#)
- [os.path — Common pathname manipulations — Python 3.11.5 documentation](#)

1. Create folder structure to store static files
 - a. Make a directory called “static” in portfolio folder
 - b. Create a folder called “images” in static
 - c. Add a logo. I downloaded [UCCS logo](#) and called it **uccs_logo.gif**



2. Update settings.py so Django can find the static files.
 - a. At top of settings.py file add
`import os`

```
import os

from pathlib import Path
```

- b. Then add the following near the bottom of the settings file to tell Django where to find folder based on building the path from the BASE_DIR defined in the settings.py file already

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]

MEDIA_URL = '/images/'
```

3. Add image information to base template
 - a. At top of base_template.html add {% load static %} above <!DOCTYPE html>

```
{% load static %}
```

```
<!DOCTYPE html>
```

b. Replace

```
<a class="navbar-brand" href="#">Navbar</a>
```

With

```

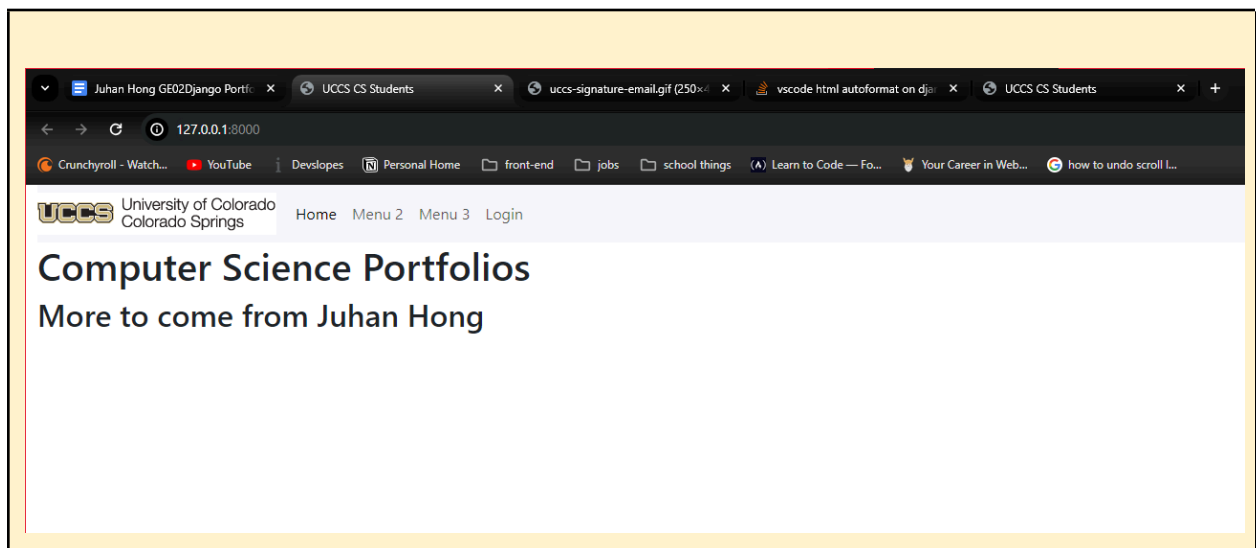
```


4. Open <http://127.0.0.1:8000> and you should see the logo.
5. Version on your sprint01 branch and update your remote repository.
6. Update your main branch by merging it with sprint01 branch code and tag the code as GE02. Do not delete sprint01 branch.

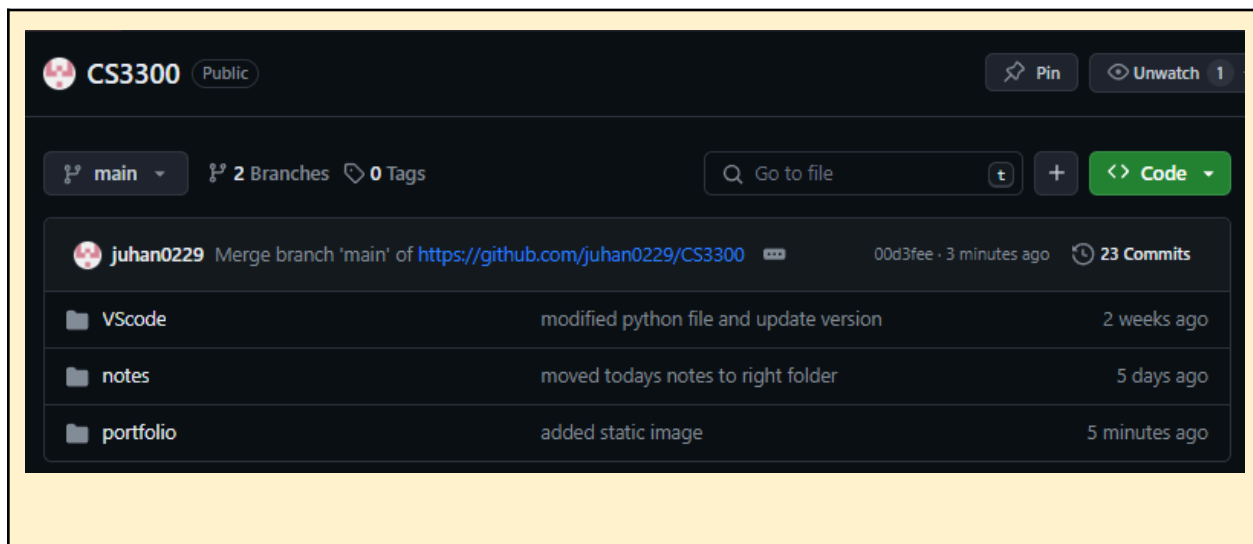
3 Apply Technologies (Individual)

These are individual responses and not team responses.

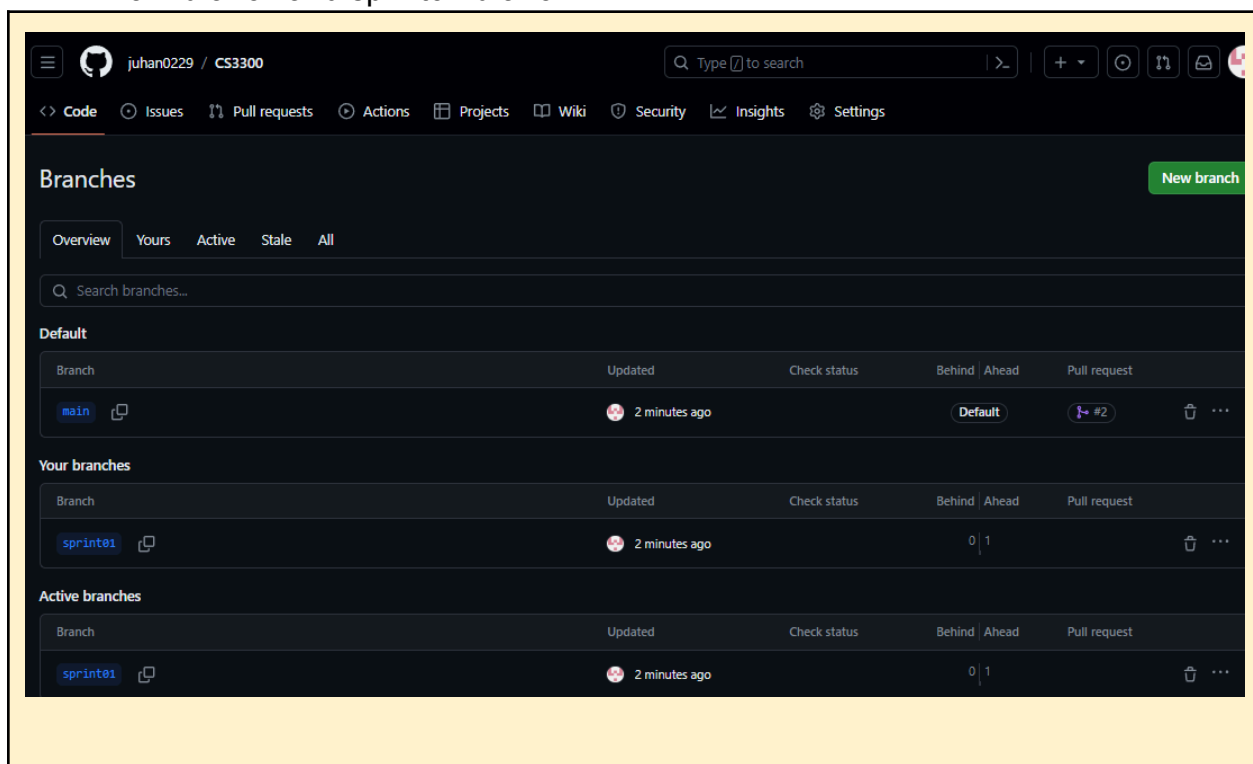
1. Put a screenshot of your home page showing the UCCS image.



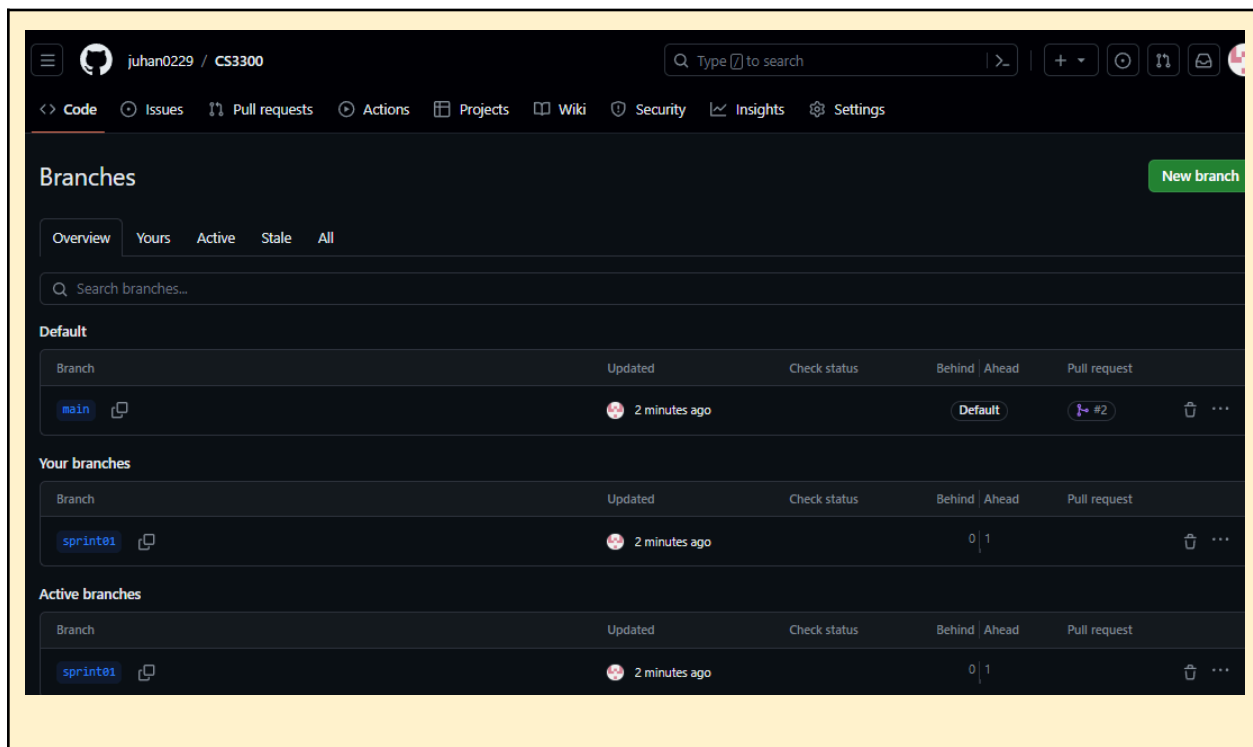
2. Go to your github remote repository, click  and take a screenshot of your code files and directories that are in the main branch



3. Go to your github remote repository, and take a screenshot showing you have a main branch and sprint01 branch



4. Take a screenshot showing your two branches in github - main and sprint01.



5. What is the requirements.txt file you created useful for? When should you update it? How do you update it?

The requirements.txt file is commonly used to specify the required software versions to run the project to run. From the GE02, the requirement.txt file i created had the following:

```
asgiref==3.7.2
Django==5.0.2
sqlparse==0.4.4
tzdata==2024.1
```

Showing the required versions of each dependencies.

Requirements.txt file should be updated in a few instances. Once it is created at the initial setup of the project, the requirements are set in stone for that version of the project. When you add or remove a dependency, for example, if i no longer need Django for the project, i can remove it from the project and update the requirements.txt file. Another instances where you should update the file is if you decide to upgrade/update, or downgrade the project with different versions of the dependencies and environment.

You can update the requirements.txt file in a couple ways. After updating the project, you can either manually change it (which probably will never do), or simply with the commands we learned in this GE:

```
Pip freeze > requirements.txt
```

6. Explain the python virtual environment you set up using venv. Include what it is and at least 2 reasons why you should use a virtual environment.

For this GE, we learned how to create a python virtual environment. To create a virtual environment, (im on Windows) you can enter `py -3 -m venv djenv(name of the virtual environment)`. As I was also keeping my eyes on the project via VScode, i noticed that once I create the virtual environment, it creates all the base necessities for the project. Once you create the virtual environment, you can activate it by typing `djenv\Scripts\activate.bat` to activate, and deactivate to deactivate.

Python virtual environment is a self contained directory that contains a specific interpreter along with the standard libraries and whatever packages you need. Here are my reasons why you should use a venv.

1. Since virtual environments are self contained directories, it is easy to delete or recreate them. I had to restart a few times from the beginning since I kept messing up at some points. It was good practice to get some repetition in, but at the same time it was very simple to delete and recreate them. Making them easy to clean up.
2. It is very easy to reproduce the same environment in your project development. Since I was working from my desktop and my laptop, I was able to look at the requirements.txt and compare them in order to make sure that I was working under the same environments on both of my computers.
3. Can create multiple virtual environments that won't have conflicts with another. From a stackoverflow post, it said that you can create as many venv as you like since they won't interfere with each other. This can and will be very useful once we get more complex into a project where we will need to create multiple virtual environments.

7. List any issues encountered by you or someone else and how you approached resolving them.

I had a few issues during this GE and it was mostly difficulties regarding actually getting the local host to work. I asked for help from my teammates via in person and discord where everyone was very helpful throughout the whole thing. It seemed like everyone was having very similar issues and we found the answers, which were mostly figuring out the right commands to input in the windows terminal

4 Professional Communication of Technical Content (Individual)

These are individual responses and not team responses.

This is just a start of your understanding of different architectures, design patterns and concepts used when learning a framework to build SaaS apps. The goal is to see how you grow over the semester in your learnings.

Create a separate professional document (word or pdf) where main ideas are clearly presented with supporting evidence.

- Present and overview of the concepts about client server, url, HTTP Requests, Django MVT(model, view, template) in relation to what you set up in Django in this GE.
- Include resources used such as lectures, books, and your own research. You do not need to include citations in the writing but list the resources at the end.
- This should contain technical vocabulary, **your own** words, code snippets, diagrams (digitally or drawn on paper). Do not use an image that someone else created.
- About 1 to 1 ½ pages. Should not be more than two pages.