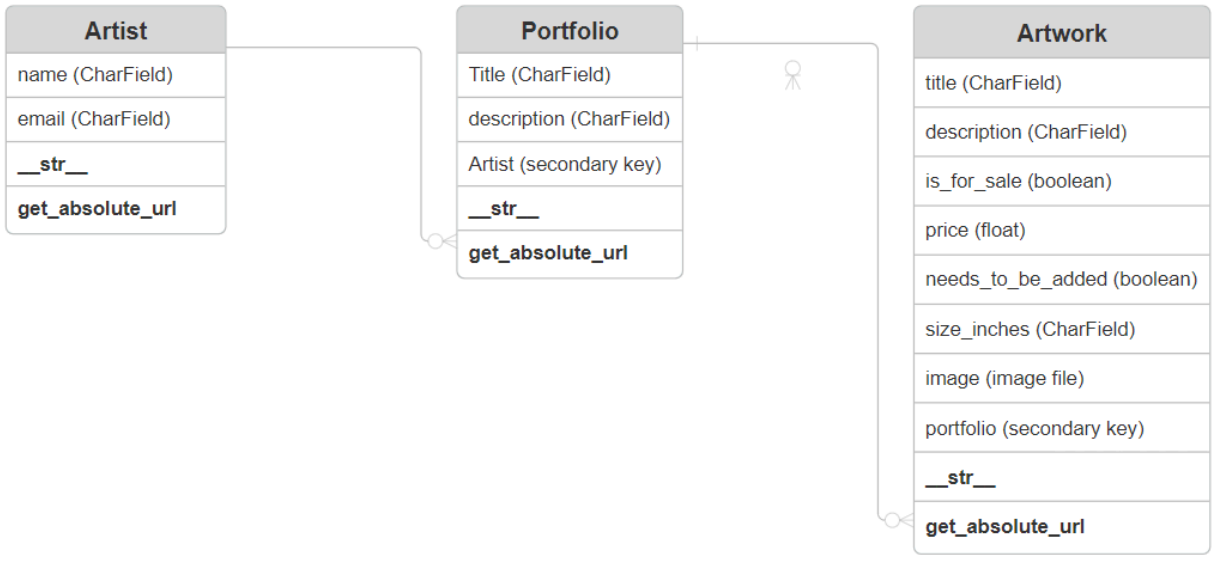
User Stories
https://github.com/users/BrettAF/projects/1/views/1
Repository
https://github.com/BrettAF/ArtProfileProject



Breaking user stories into tasks can be challenging, especially when determining the level of granularity. I find it time-consuming, but I understand its importance in defining the work needed for each user story. It's a balancing act between detailed tasks for better tracking and broader tasks for flexibility, depending on the developer's preferences and the complexity of the user story.

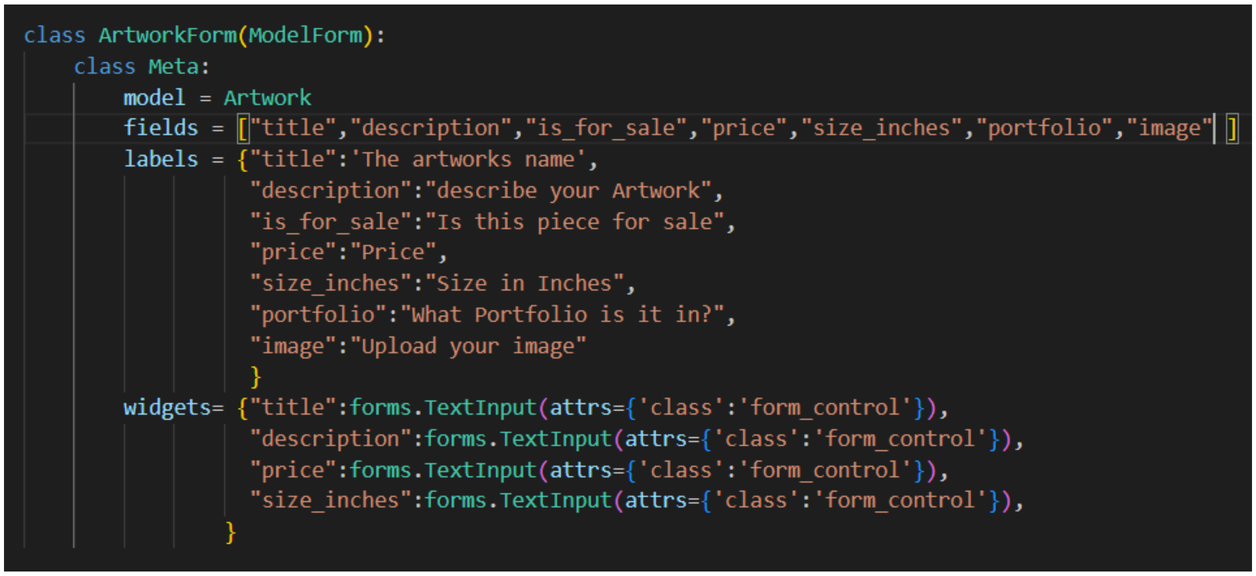
My estimate for task completion time doesn't always match the actual time spent. For example, I estimated 450 minutes for this task but ended up spending around 900 minutes, not including time spent on user stories and documentation. While missing the estimate by double was more than I hoped for, it is honestly better than I expected. It's a reminder to continuously refine my estimation skills.

Learning to use GitHub has been a gradual process, as it's a powerful but complex tool. Despite its complexity, I'm slowly gaining proficiency. GitHub's version control capabilities are invaluable for tracking changes and collaborating on projects.

1. Explain how you utilized version control for your spint01 development.

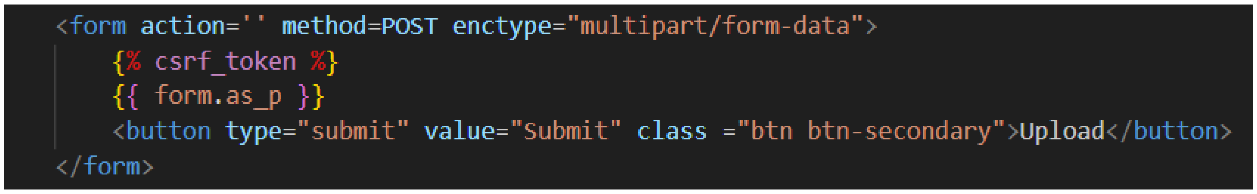
   Every time I made another component work successfully, I immediately committed it to ensure the work wouldn't be lost. I committed and pushed the branch at the end of each work session so that I could continue where I left off. I made good use of commit comments so that I could back up if I needed to. But because I saved only when I had things working, I never needed to back up to older versions.

2. Explain how you implemented the functionality for the user to create an item.
   The creation of an Artwork Item was accomplished with a form added to the forms.py file.



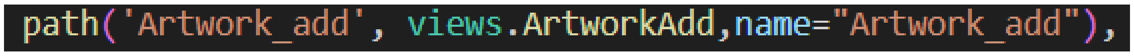
```
class ArtworkForm(ModelForm):
    class Meta:
        model = Artwork
        fields = ["title","description","is_for_sale","price","size_inches","portfolio","image"]
        labels = {"title":'The artworks name',
                  "description":"describe your Artwork",
                  "is_for_sale":"Is this piece for sale",
                  "price":"Price",
                  "size_inches":"Size in Inches",
                  "portfolio":"What Portfolio is it in?",
                  "image":"Upload your image"
                  }
        widgets= {"title":forms.TextInput(attrs={'class':'form_control'}),
                  "description":forms.TextInput(attrs={'class':'form_control'}),
                  "price":forms.TextInput(attrs={'class':'form_control'}),
                  "size_inches":forms.TextInput(attrs={'class':'form_control'}),
                }
```

   This had information to tell the program what to display and style information on how to display it.
   The Form also had a template that was placed in the templates folder. That template had the line form.as_p, which causes the form to display from the forms page



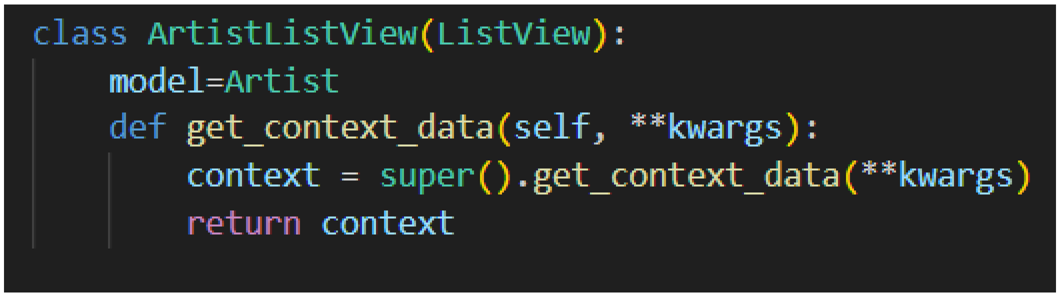
```
    <form action='' method=POST enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit" value="Submit" class ="btn btn-secondary">Upload</button>
    </form>
```

   This template had to be registered under Urls and under Views to make it work correctly.



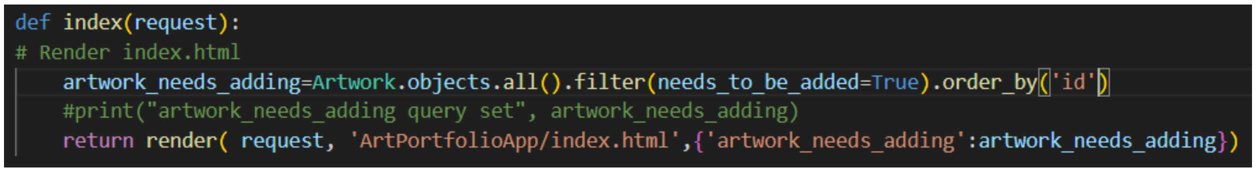
```
path('Artwork_add', views.ArtworkAdd,name="Artwork_add"),
```

3. Explain how you implemented the functionality to display a list of items on a web page.
   I implemented two different types of lists in my programming. First was a generic view which I used to make a view for each of the three models. They looked like this:



```
class ArtistListView(ListView):
    model=Artist
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        return context
```

The second was the method I used to display the list on the home page.



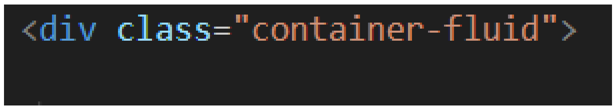
```
def index(request):
# Render index.html
    artwork_needs_adding=Artwork.objects.all().filter(needs_to_be_added=True).order_by('id')
    #print("artwork_needs_adding query set", artwork_needs_adding)
    return render( request, 'ArtPortfolioApp/index.html',{'artwork_needs_adding':artwork_needs_adding})
```

This method needed to only include artwork that needs to be added(the variable needs_to_be_added == True) and be sorted by id number.

4. Explain how you used Bootstrap's framework of html and css (elements and classes) to implement the navigation menu in the user interface.

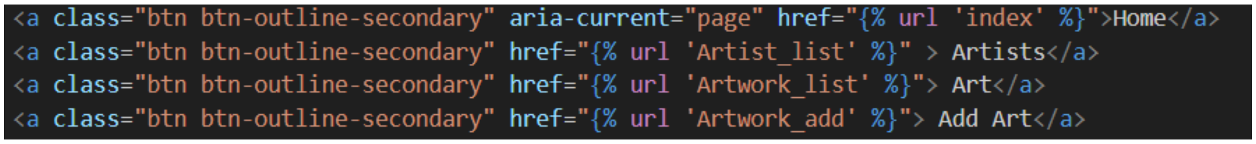
   Bootstrap helped me to easily create my styles for my website. It has many options, but I was only able to implement a couple on my navigation bar.

   I used Bootstrap and div to try to make the page responsive



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

I also used buttons that are from Bootstrap to make the links on the navigation bar look clearer.



```
<a class="btn btn-outline-secondary" aria-current="page" href="{% url 'index' %}">Home</a>
<a class="btn btn-outline-secondary" href="{% url 'Artist_list' %}" > Artists</a>
<a class="btn btn-outline-secondary" href="{% url 'Artwork_list' %}"> Art</a>
<a class="btn btn-outline-secondary" href="{% url 'Artwork_add' %}"> Add Art</a>
```