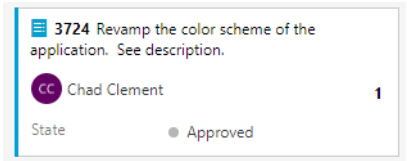
The Tech Academy Live Project 1 Code Retrospective

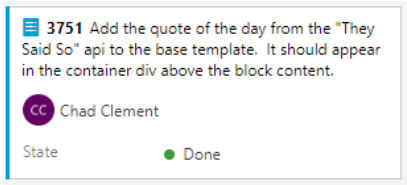
In the first half of the live project, I worked on a site called DataScrape. It is a Django site that uses APIs to collect information based on the user's location



In this task, I used CSS to modify the color scheme of the website to use a colors that were provided.

```
#eventList {
    border-style: solid;
    border-color: black;
    border-width: 1px;
    background-color: rgb(230, 230, 230);
}
.divider {
    background-color: black !important;
}
.navbar-nav > li > .dropdown-menu {
    background-color: darkseagreen !important;
    border-color: black;
  }
.navbar-nav > li > .dropdown-toggle {
            background-color: darkseagreen !important;
            border-color: black;
             }
body {
    text-align: center;
    background-color: dimgray !important;
}

```



In this task, I modified the homepage to show a quote of the day. It uses an API from theysaidso.com

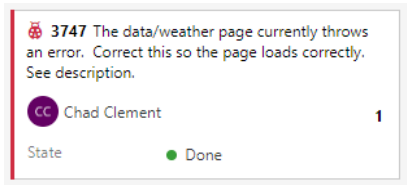
```
class QodScraper:

    def __init__(self):

        try:
            #Convert null to none for JSON file
            null = None
            # URL Quote of the Day
            # Open the URL and save the content to a variable.
            url = 'http://quotes.rest/qod.json'
            qoddata = urllib.request.urlopen(url).read()
            # load the json data
            qoddata = json.loads(qoddata)
            author = qoddata["contents"]["quotes"][0]['author']
            quote = qoddata["contents"]["quotes"][0]['quote']
            self.quote = ('"{}", {}'.format(quote,author))

        except urllib.error.URLError as e:
            self.quote = '"Too many requests on Quote of the Day"'
            print(e)

```



This bug was caused by a line in the data/weather viewmodel that imported the datetime module twice.

```
import time
import datetime
import urllib.request
import json
import re
import requests
from datetime import datetime
from django.shortcuts import get_object_or_404
```

3773 Style the traffic page. Bring the image size down. You can use col-6 from bootstrap. Insure that the height of the image responds appropriately. Add a margin-bottom to the header.

Chad Clement

State Approved

In this task, I ran into difficulties resizing the map on the traffic page because the API is not responsive to screen size. To achieve the desired effect, I created div tags for three different screen sizes.

```
<div class="hidden-md hidden-sm hidden-xs">
<!-- giving the embed tag a src of the API call will render the map on the page -->
<embed src="{{traffic.flow}}" type="" style="width:1140px; height:600px">
</div>
<div class="hidden-lg hidden-xl hidden-xs col-sm-offset-2 col-md-offset-3">
        <embed src="{{traffic.flow}}" type="" style="width:500px; height:600px">
</div>
<div class="hidden-lg hidden-xl hidden-md hidden-sm col-xs-offset-1">
        <embed src="{{traffic.flow}}" type="" style="width:300px; height:600px">
</div>
```

In the second half of the live project I worked on a Django site for The Tech Academy site. It is a forum where staff and students can post in threads.

3859 Adjust data migrations file "0004_TestDataLoad" to make every test user's avatar the default-avatar.png file.

Chad Clement

State Committed

Here the 0004_TestDataLoad file was modified to include avatar pictures in the test data for user profiles.

```
    # Instantiate and create 20 UserProfiles for the users created above.
    UserProfile = apps.get_model('Forum', 'UserProfile')
    for i in range(1, 21):
        user_profile = UserProfile(
            User_id=userData[i]['id'],
            Signature="TestSignature_{}".format(i),
            Avatar='avatar/default-avatar.png',
        )
        user_profile.save()
```

3783 Add a register page that allows a user to register a new account. The view should create a new Django User and save to the database.

Chad Clement

State Done

Here I created a view to register user data to the database, based off inputs from a registration page.

```
def register(request):
    if request.method == 'POST':

        username = request.POST.get('username')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        email = request.POST.get('email')
        password = request.POST.get('password')
        date_joined = datetime.datetime.now()
        u1 = User.objects.create_user(username=username, first_name=first_name, \
        last_name=last_name, email=email, password=password, date_joined=date_joined)
        u1.save()

        return redirect('home_view')
    else:
        return render(request, 'Accounts/register.html')
```

3835 Create a homepage for the application. It should show a list of the ten most recently active threads in the database.

Chad Clement

State Done

Here a view was created for the site to give it a basic homepage. To display the ten most recently active threads, I used the Thread model to query the database.

```
def home_view(request):
    #collect data for latest 10 threads on homepage
    data = Thread.objects.values().order_by('-DateUpdate')[:10]
    #convert to dictionary to pass variable
    threads = {"threads" : data}
    return render(request, 'index.html', threads)
class UserProfileListView(generic.ListView):
    template_name = 'user_profile/index.html'
    context_object_name = 'users'
```