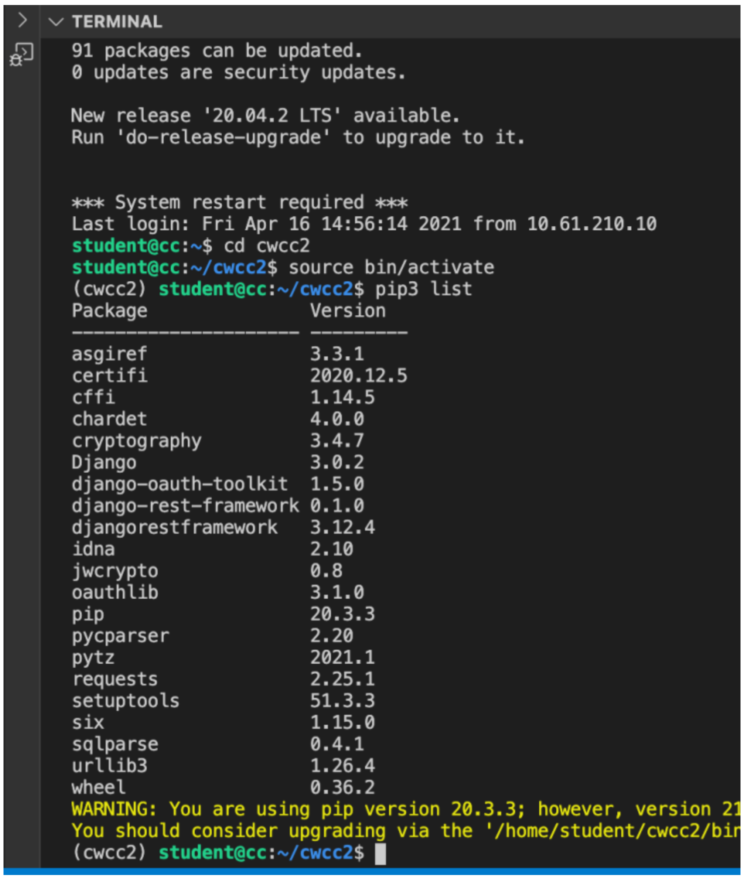
<u>Cloud Computing Coursework</u>

Student: Rohan Khanolkar   Student ID: 13199041

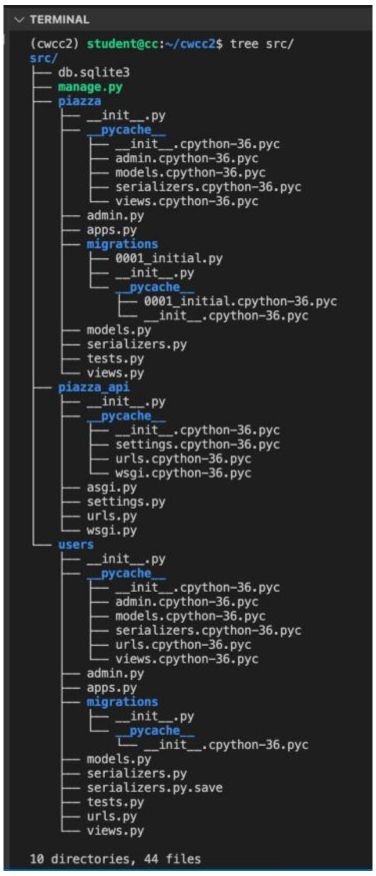
Phase A: Install and deploy software in virtual environments:

For commencing this project, we have to first ‘ssh’ (shell secure) login to our virtual machine by the IP address (10.61.64.56) login given to us by our module tutor. Then we created a virtual environment (named ‘cwcc2’) to isolate a python environment which will not affect the whole virtual machine. After creating the virtual environment, we need to activate it to enter inside the ‘cwcc2’ virtual environment and install the ‘Django==3.0.2’ (Django framework), ‘django-rest-framework’ (Django rest framework) and ‘django-oauth-toolkit’ (Django OAuth 2.0 protocol for authorization). The below screen shot (fig. 1) shows the list of packages installed in the virtual environment.



```
91 packages can be updated.
0 updates are security updates.

New release '20.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


*** System restart required ***
Last login: Fri Apr 16 14:56:14 2021 from 10.61.210.10
student@cc:~$ cd cwcc2
student@cc:~/cwcc2$ source bin/activate
(cwcc2) student@cc:~/cwcc2$ pip3 list
Package                Version
---------------------- ---------
asgiref                3.3.1
certifi                2020.12.5
cffi                   1.14.5
chardet                4.0.0
cryptography           3.4.7
Django                 3.0.2
django-oauth-toolkit   1.5.0
django-rest-framework  0.1.0
djangorestframework    3.12.4
idna                   2.10
jwcrypto               0.8
oauthlib               3.1.0
pip                    20.3.3
pycparser              2.20
pytz                   2021.1
requests               2.25.1
setuptools             51.3.3
six                    1.15.0
sqlparse               0.4.1
urllib3                1.26.4
wheel                  0.36.2
WARNING: You are using pip version 20.3.3; however, version 21
You should consider upgrading via the '/home/student/cwcc2/bir
(cwcc2) student@cc:~/cwcc2$ 
```

fig 1: List of applications installed



```
(cwcc2) student@cc:~/cwcc2$ tree src/
src/
├── db.sqlite3
├── manage.py
├── piazza
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   ├── admin.cpython-36.pyc
│   │   ├── models.cpython-36.pyc
│   │   ├── serializers.cpython-36.pyc
│   │   └── views.cpython-36.pyc
│   ├── admin.py
│   ├── apps.py
│   ├── migrations
│   │   ├── 0001_initial.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       ├── 0001_initial.cpython-36.pyc
│   │       └── __init__.cpython-36.pyc
│   ├── models.py
│   ├── serializers.py
│   ├── tests.py
│   └── views.py
├── piazza_api
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-36.pyc
│   │   ├── settings.cpython-36.pyc
│   │   ├── urls.cpython-36.pyc
│   │   └── wsgi.cpython-36.pyc
│   ├── asgi.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── users
    ├── __init__.py
    ├── __pycache__
    │   ├── __init__.cpython-36.pyc
    │   ├── admin.cpython-36.pyc
    │   ├── models.cpython-36.pyc
    │   ├── serializers.cpython-36.pyc
    │   ├── urls.cpython-36.pyc
    │   └── views.cpython-36.pyc
    ├── admin.py
    ├── apps.py
    ├── migrations
    │   ├── __init__.py
    │   └── __pycache__
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── serializers.py
    ├── serializers.py.save
    ├── tests.py
    ├── urls.py
    └── views.py

10 directories, 44 files
```

fig 2: Folder structure

Please note that the python package is pre-installed in out virtual machine, thus we did not install it for doing this project.

We now created a Django project called ‘piazza_api’ and changed the name of the folder (project name ‘piazza_api’) from ‘piazza_api’ to ‘src’ (source). So now we have the project name as ‘src’ and there is a folder inside called as ‘piazza_api’(as seen in fig. 2). After this, we open the ‘settings.py’ file which consist of apps and database settings which is inside ‘piazza_api’, we then need to add the IP number in the allowed host section i.e. ‘ALLOWED_HOST = [’10.61.64.56’]’ . By adding the IP number in will allow Django to accept connection.

Phase B: Development of authorisation and authentication services

Now we configure OAuth in this Django REST framework, by continuing in the same ‘settings.py’ file we will add ‘oauth2_provider’ and ‘rest_framework’ in ‘INSTALLED_APPS’ section and under the ‘MIDDLEWARE’ section we will add ‘oauth2_provider.middleware.OAuth2TokenMiddleware ’ which will all a user to access resource with token (for HTTP calls). Now to support the REST authentication process we have added ‘REST_FRAMEWORK’ and ‘AUTHENTICATION_BACKEND’ for specifying the authentication is required for every user request. The ‘REST_FRAMEWORK’ consist of two classes (an authentication class and a permission) for each request to check to user token and allow it to access. The ‘AUTHENTICATION_BACKEND’ specifies which authentication system to be used.

Now to add the OAuth2 provider, we will open ‘urls.py’ file of ‘piazza_api’ and first add import ‘include’ from the ‘django.urls’ library and then under ‘urlpatterns’ we will add a path for ‘o/’ (resource for performing OAuth authentication process) and link it with the installed app ‘oauth2_provider’ (provided in the setting file). Thus, we created a new url for authentication link i.e. ‘http:// 10.61.68.56:8000/o/ ‘.

After completing the OAuth2 protocol from ‘urls.py’, we will now create a super user i.e ‘admin’ user which will help us for further configuration of ‘Authorisation Server’. So now we run the server and open the admin panel on the web browser (link: http://10.61.64.56:8000/admin/)and login with ‘admin’ user name and password. After we login to the admin panel we can see the ‘Django OAuth Toolkit’ below the ‘Authentication and Authorisation’ header. We can now say that Django Auth Toolkit is ready to start building the application server with a default OAuth configuration.

Up till now, we have deployed an authorisation server which will need a registration for every application and request access token to use the resources. This token will be used every time an application performs a request action to the ‘application server’. Every time a user is registered with the system a new access token is generated by the ‘Authorization Server’ which later authenticates this token with its authentication database, this means that a user’s request is validated using the security token.

Now we start an application inside the ‘src’ folder and call it ‘users’ and register it under ‘INSTALLED_APPS’ in the ‘piazza_api/settings.py’ file. This application will help us to save the tokens generated by the server for every user. To do this we will have to create two files ‘serializers.py’ and ‘views.py’ where this serializers help to render python data types into JASON, XML and other type of languages. In this serializer we use ‘extra_kwargs’ which is a shortcut to specify additional arbitrary arguments which help us to explicitly assert fields on the serializer. On the other hand ‘views.py’ help us to create API views for our application to register, login refresh and revoke tokens. This ‘views.py’ registers the client id and client secret tokens along with ‘IP_token’ and ‘IP_revoke_token’ (i.e. our server number with OAuth token). For this course work I have used the ‘views.py’ codes given as an attachment in the week 4 lab. Now in the admin panel we will first click an application and add a new application where we can see the client id token and client secret token along with keeping the client type as confidential and resource as password based. We will now create ‘urls.py’ file in ‘users’ folders and give path to ‘register/’, ‘token/’, ‘token/refresh/’ and ‘token/revoke/’ to allow to user to register, refresh and revoke their token. We now accord this authentication with ‘piazza_api/urls.py’ for user token interaction. In this way we have created our authentication server that authorises piazza RESTful API for storing data on behalf of the authorised user in the database which can register and authenticate the user each time and action is performed. This authorisation server will not allow any user to query database without a token, this is implemented by the OAuth V2 protocol which we had learnt in lab 4 of this module.

Phase C: Development of Piazza RESTful APIs

After completing the authorization server which could ‘GET’, ‘POST’, ‘DELETE’ and ‘PATCH’ data from an API end point we will now proceed in development of the resource server (piazza Restful API). To do this we will start a new application called ‘piazza’ and create models in ‘piazza/models.py’ which will help us to build SQLite database that can help us to retrieve, create, update, delete a record. Further, we will validate the models in ‘piazza/serializer.py’ which would help us to convert to JSON format which can be accessed by the API end point.

In our ‘models.py’ we have created two classes ‘piazza’ and ‘action’, where ‘piazza’ class allows user to create a post along with a topic and post it with an expiration time so that other users can view the post, like/dislike and/or comment on the post within the given expiration time. Figure 3 (Table a & b) shows the SQLite database schema which I have created in the application.

fig. 3 (Table a: ‘piazza’ class SQLite database scheme)

‘Title’	‘Politics’	‘Health’	‘Sports’	‘Sports’	‘Message_body’	‘TimesStamp_Post’	‘ExpDateTime_Post’	‘UserID_Post’

fig. 3 (Table b: ‘action’ class SQLite database scheme)

'postInteractionID'	'UserID_Post'	'Post_ID'	'Actions'	'comments'	'interactionTimeStamp'

The above SQLite database (fig 3 Table a) takes input from the models classes created in ‘models.py. The class ‘piazza’ has the following functions:

- ‘Title’- is a character field which takes 100 characters as input.
- ‘Politics’, ‘Health’, ‘Sports’, ‘Sports’– are individual boolean fields which takes input as true or false.
- ‘Message_body’ - is a character field which takes 1000 characters as input.
- ‘Post_id’ – this is an auto field which takes the primary key and connects it with the ‘action’ class so that the post id is maintained with the same number in both the SQLite tables.
- ‘TimesStamp_Post’ – This gives the post a current time stamp which shows the data and time when the post was created.
- ‘ExpDateTime_Post’ – This is a date time field which calculates the difference between the expiration time and the current time and tells the users if the post is live or expired. This is also connected with a defined property ‘status1’ which is an if/else condition that compares current time with the expiration time and informs the user about the post being live or expired.
- ‘UserID_Post’ – This is a unique foreign key which the user gets when he/she gets registered with API. This is usually registering a user with a unique number (user identification number) which is helpful for us to connect with ‘action’ class to maintain the same user id number in both SQLite database.
- ‘total_likes’ – This is a define statement which checks the status of ‘Actions’ from the ‘action’ class and increment a count each time a user ‘Like’ a post and display the total likes on the application end point (browser screen)
- ‘total_dislikes’ - This is a define statement which checks the status of ‘Actions’ from the ‘action’ class and increment a count each time a user ‘Dislike’ a post and display the total dislikes on the application end point (browser screen)
- ‘total_comments’ – This is a define statement which checks the status of ‘Actions’ from the ‘action’ class and increment a count each time a user ‘comments’ a post and display the total comments on the application end point (browser screen)
- ‘status1’ – As mentioned above in this is an if/else condition that compares current time with the expiration time and informs the user about the post being live or expired.

The class ‘action’ (fig 3 Table b) has the following functions:

- 'postInteractionID' – This is an auto filled which takes primary key form the ‘piazza’ class and saves a unique id for every interaction a user makes on the post.
- 'UserID_Post' – This is a foreign key which takes the unique user id from ‘piazza’ class to maintain the same user id number in both the tables.
- 'Post_ID' - This is a foreign key which takes the unique post id from ‘piazza’ class to maintain the same post id number in both the tables. This gives a dropdown list to select which post ID the user want to interact with.

- 'Action_list' – This creates a drop-down list from where a user can select ‘Like’ or ‘Dislike’ a post.
- 'Actions' – This is a character field which chooses the input from 'Action_list' and displays the result on the API end point (browser)
- 'comments' – This is a character field which takes 600 characters as input.
- 'interactionTimeStamp' – This is a date time field which notes the interaction date and time and compares it with validation conditions in the ‘serializers.py’ that checks if the post is live or expired and raise a validation error if the post is expired.

The ‘serializers.py’ is created in the in the piazza app which has two classes ‘piazzaSeralizer’ and ‘actionSerializer’. The ‘piazzaSerializer’ consist class ‘meta’ which takes input from the ‘piazza’ class of the ‘models.py’ to convert it into JSON format so that the API endpoint can interact with it. The ‘piazzaSerializer’ also consist of two validate statement ‘validate’ and ‘status_vaidate’

- ‘validate’ – This is a define function that first creates a post expiry date and time (i.e. post created date & time + the expiration time input by the user who posted it)
- ‘status_validate' - This is a define function that first checks if the current time is larger that the expiration time and also checks if status is expired then is compares the current status with status expired and returns post expired to the ‘piazza’ class.

The ‘actionSerializer’ consist class ‘meta’ which takes input from the ‘action’ class of the ‘models.py’ to convert it into JSON format so that the API endpoint can interact with it. The ‘actionSerializer’ also consist of three validate statement ‘validate1’, ‘validate2’ and ‘validate_Post_ID’

- ‘validate1’ – This checks the ‘Post_ID’ assigned to the post assigns it to the ‘Title’ from the ‘piazza’ class and display the post ID to the users to select in the ‘action’ API endpoint browser.
- ‘validate2’ – This compares the user id with user names from the ‘piazza’ class and display the user names in the ‘action’ API endpoint browser so we can select which user wants to interact with the post.
- ‘validate_Post_ID’ – This is a if condition which checks if the ‘status1’ from the ‘piazza’ models is live or expired. If its expired then it will raise a validation error to the user that it cannot interact with the post.

Now we open the ‘piazza/views.py’ we will add both the ‘piazza’ class and ‘piazzaSerializer’ from the ‘model.py’ and ‘serializer.py’ under ‘piazzaViewset’ Class. Also, we will add both the ‘action’ class and ‘actionSerializer’ from the ‘model.py’ and ‘serializer.py’ under ‘actionViewset’ Class. This help to render the class objects in JSON format which can query the database for the objects and then pass it to the sterilizer to create a new JSON object.

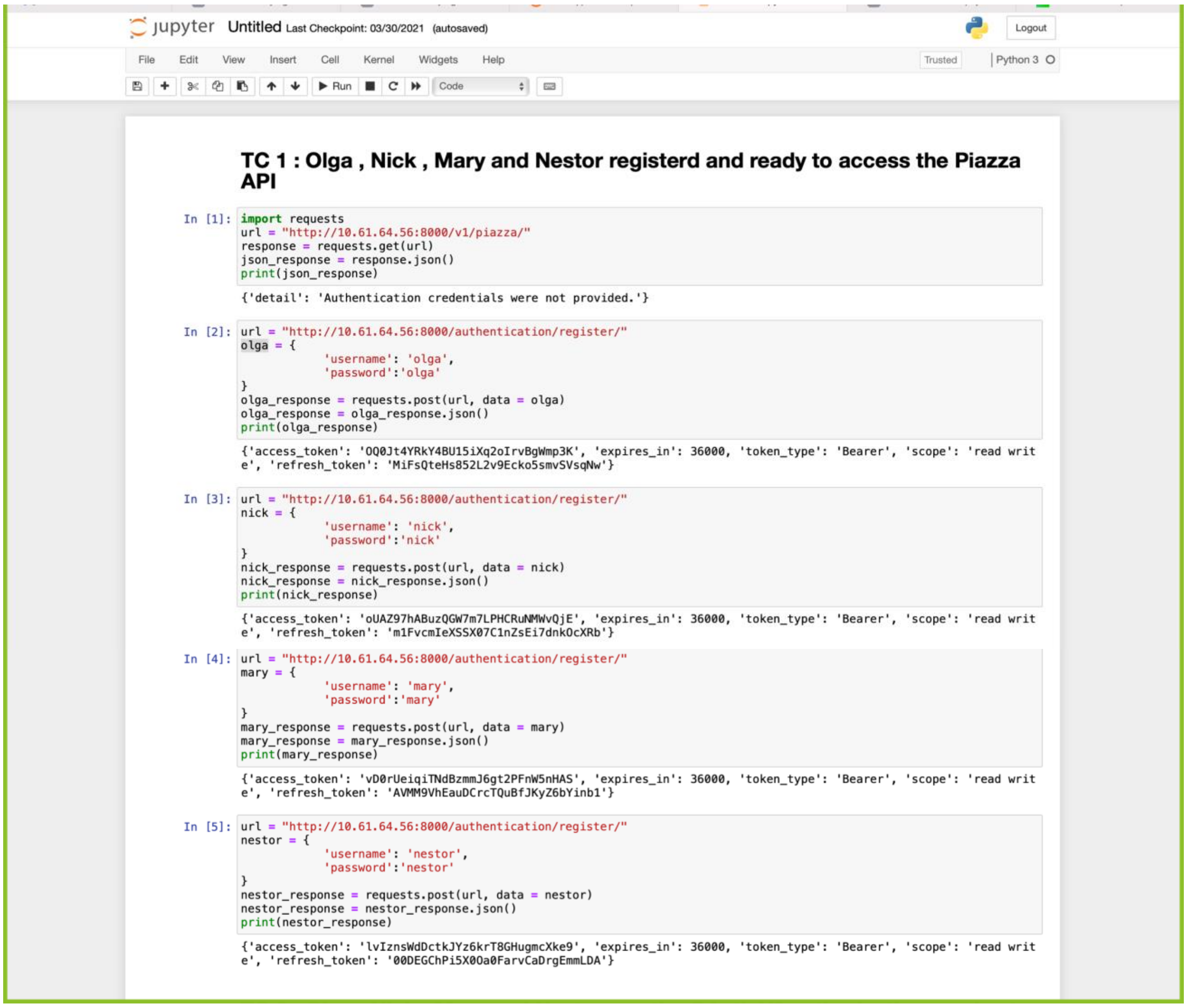
Now we open the ‘piazza_api/urls.py’ to import the ViewSet of from the ‘views.py’ and assign a path ‘v1’to create the url endpoints. In our case it would be http://10.61.64.56:8000/v1/piazza/ and http://10.61.64.56:8000/v1/action/ for the API endpoint browser. And after this step we go to the ‘piazza_api/settings.py’ and add the application ‘piazza’ inside ‘INSTALLED_APPS’.

We are now ready to test our application.

Phase D: Development of a testing application:

For this phase I have user ‘juypter notebook’ to interact with the applications. Below are the test cases performed.

TC 1 : Olga , Nick , Mary and Nestor registered and ready to access the Piazza API

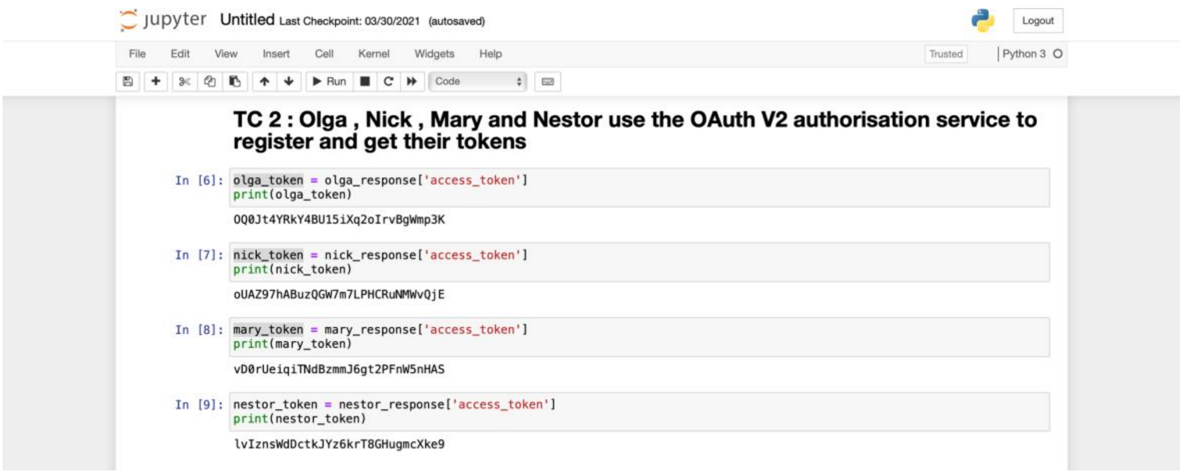

jupyter Untitled Last Checkpoint: 03/30/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 1 : Olga , Nick , Mary and Nestor registerd and ready to access the Piazza API

```
In [1]: import requests
        url = "http://10.61.64.56:8000/v1/piazza/"
        response = requests.get(url)
        json_response = response.json()
        print(json_response)

        {'detail': 'Authentication credentials were not provided.'}

In [2]: url = "http://10.61.64.56:8000/authentication/register/"
        olga = {
                    'username': 'olga',
                    'password':'olga'
        }
        olga_response = requests.post(url, data = olga)
        olga_response = olga_response.json()
        print(olga_response)

        {'access_token': 'OQ0Jt4YRkY4BU15iXq2oIrvBgWmp3K', 'expires_in': 36000, 'token_type': 'Bearer', 'scope': 'read writ
        e', 'refresh_token': 'MiFsQteHs852L2v9Ecko5smvSVsqNw'}

In [3]: url = "http://10.61.64.56:8000/authentication/register/"
        nick = {
                    'username': 'nick',
                    'password':'nick'
        }
        nick_response = requests.post(url, data = nick)
        nick_response = nick_response.json()
        print(nick_response)

        {'access_token': 'oUAZ97hABuzQGW7m7LPHCRuNMWvQjE', 'expires_in': 36000, 'token_type': 'Bearer', 'scope': 'read writ
        e', 'refresh_token': 'm1FvcmIeXSSX07C1nZsEi7dnkOcXRb'}

In [4]: url = "http://10.61.64.56:8000/authentication/register/"
        mary = {
                    'username': 'mary',
                    'password':'mary'
        }
        mary_response = requests.post(url, data = mary)
        mary_response = mary_response.json()
        print(mary_response)

        {'access_token': 'vD0rUeiqiTNdBzmmJ6gt2PFnW5nHAS', 'expires_in': 36000, 'token_type': 'Bearer', 'scope': 'read writ
        e', 'refresh_token': 'AVMM9VhEauDCrcTQuBfJKyZ6bYinb1'}

In [5]: url = "http://10.61.64.56:8000/authentication/register/"
        nestor = {
                    'username': 'nestor',
                    'password':'nestor'
        }
        nestor_response = requests.post(url, data = nestor)
        nestor_response = nestor_response.json()
        print(nestor_response)

        {'access_token': 'lvIznsWdDctkJYz6krT8GHugmcXke9', 'expires_in': 36000, 'token_type': 'Bearer', 'scope': 'read writ
        e', 'refresh_token': '00DEGChPi5X0Oa0FarvCaDrgEmmLDA'}
```

In the above screenshot of TC 1, you can see that we posted a request to the ‘application server’ and the ‘application server’ forwards our request to the ‘authentication server’. The ‘authentication server’ receives it and save the user details and give an ‘access_token’ to the ‘application server’ and the ‘application server’ returns the ‘access_token’ to the ‘application browser’ and hence we received the ‘access_token’ and ‘refresh_token’ and this token is valid for 3600 min (i.e. 60 hours).

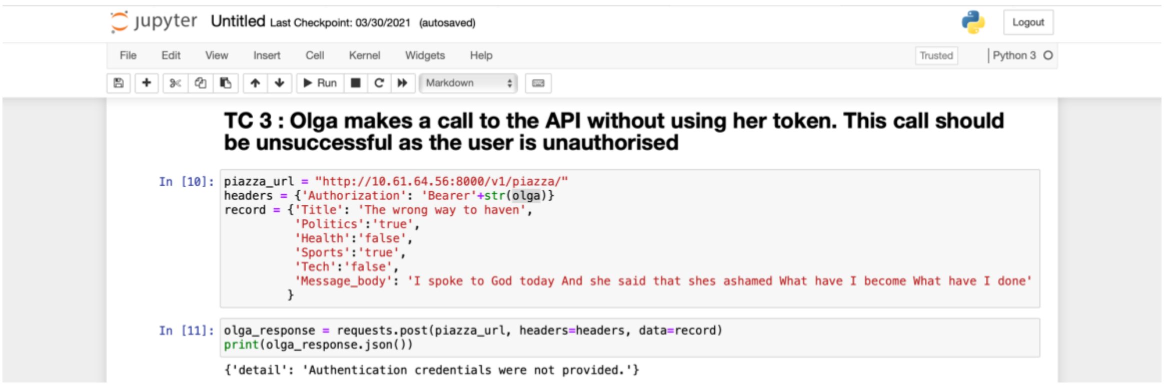
TC 2 : Olga , Nick , Mary and Nestor use the OAuth V2 authorisation service to register and get their tokens


jupyter Untitled Last Checkpoint: 03/30/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 2 : Olga , Nick , Mary and Nestor use the OAuth V2 authorisation service to register and get their tokens

```
In [6]: olga_token = olga_response['access_token']
        print(olga_token)

        OQ0Jt4YRkY4BU15iXq2oIrvBgWmp3K

In [7]: nick_token = nick_response['access_token']
        print(nick_token)

        oUAZ97hABuzQGW7m7LPHCRuNMWvQjE

In [8]: mary_token = mary_response['access_token']
        print(mary_token)

        vD0rUeiqiTNdBzmmJ6gt2PFnW5nHAS

In [9]: nestor_token = nestor_response['access_token']
        print(nestor_token)

        lvIznsWdDctkJYz6krT8GHugmcXke9
```

TC 3 : Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised

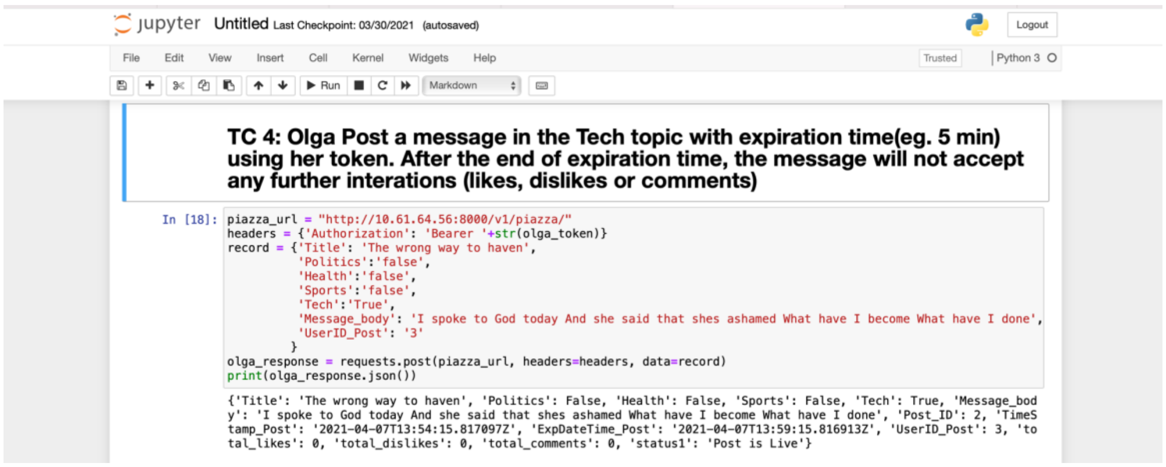

jupyter Untitled Last Checkpoint: 03/30/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

TC 3 : Olga makes a call to the API without using her token. This call should be unsuccessful as the user is unauthorised

```
In [10]: piazza_url = "http://10.61.64.56:8000/v1/piazza/"
         headers = {'Authorization': 'Bearer'+str(olga)}
         record = {'Title': 'The wrong way to haven',
                   'Politics':'true',
                   'Health':'false',
                   'Sports':'true',
                   'Tech':'false',
                   'Message_body': 'I spoke to God today And she said that shes ashamed What have I become What have I done'
                  }

In [11]: olga_response = requests.post(piazza_url, headers=headers, data=record)
         print(olga_response.json())

         {'detail': 'Authentication credentials were not provided.'}
```

Here you can see that this call went unsuccessful and token was not provided.

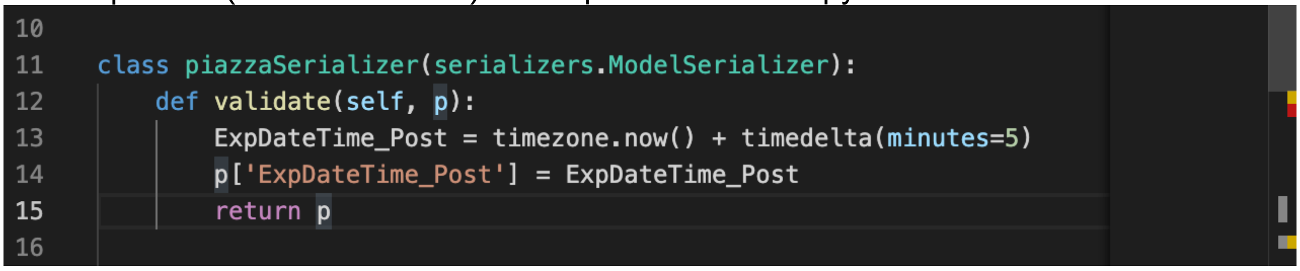
TC 4: Olga Post a message in the Tech topic with expiration time(eg. 5 min) using her token. After the end of expiration time, the message will not accept any further interations (likes, dislikes or comments)


jupyter Untitled Last Checkpoint: 03/30/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

TC 4: Olga Post a message in the Tech topic with expiration time(eg. 5 min) using her token. After the end of expiration time, the message will not accept any further interations (likes, dislikes or comments)

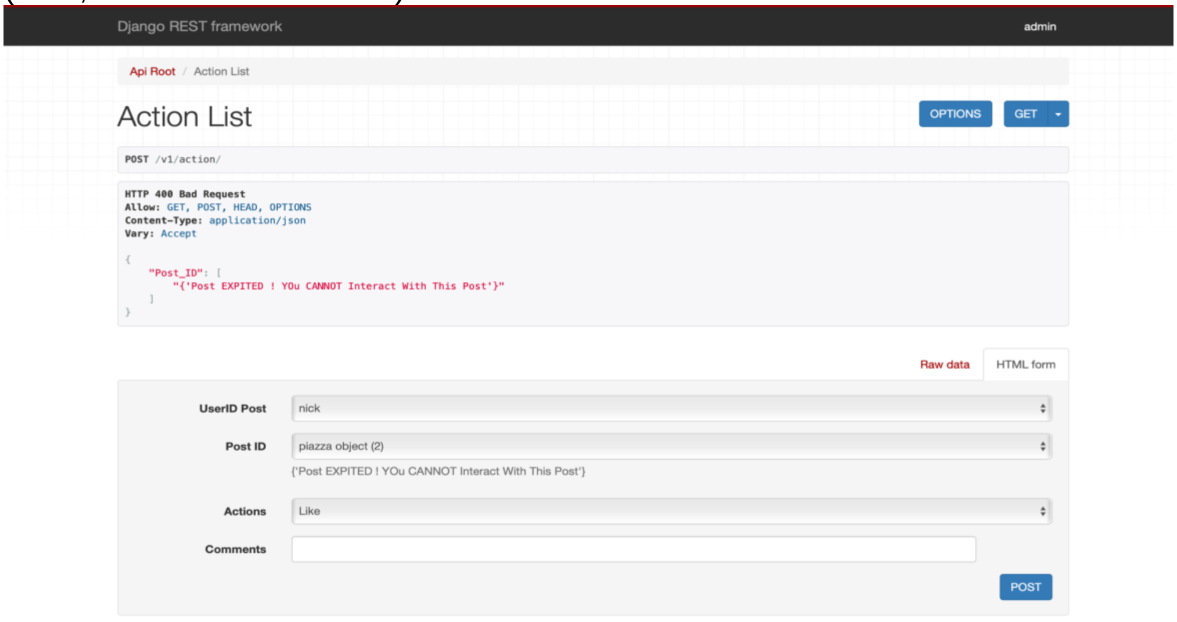
```
In [18]: piazza_url = "http://10.61.64.56:8000/v1/piazza/"
         headers = {'Authorization': 'Bearer '+str(olga_token)}
         record = {'Title': 'The wrong way to haven',
                   'Politics':'false',
                   'Health':'false',
                   'Sports':'false',
                   'Tech':'True',
                   'Message_body': 'I spoke to God today And she said that shes ashamed What have I become What have I done',
                   'UserID_Post': '3'
                  }
         olga_response = requests.post(piazza_url, headers=headers, data=record)
         print(olga_response.json())

         {'Title': 'The wrong way to haven', 'Politics': False, 'Health': False, 'Sports': False, 'Tech': True, 'Message_bod
         y': 'I spoke to God today And she said that shes ashamed What have I become What have I done', 'Post_ID': 2, 'TimeS
         tamp_Post': '2021-04-07T13:54:15.817097Z', 'ExpDateTime_Post': '2021-04-07T13:59:15.816913Z', 'UserID_Post': 3, 'to
         tal_likes': 0, 'total_dislikes': 0, 'total_comments': 0, 'status1': 'Post is Live'}
```

Post expiration (time set to 5 min) in the ‘piazza/serializer.py’ in ‘validate’



```

class piazzaSerializer(serializers.ModelSerializer):
    def validate(self, p):
        ExpDateTime_Post = timezone.now() + timedelta(minutes=5)
        p['ExpDateTime_Post'] = ExpDateTime_Post
        return p

```

After the end of expiration time, the message will not accept any further integrations (likes, dislikes or comments)



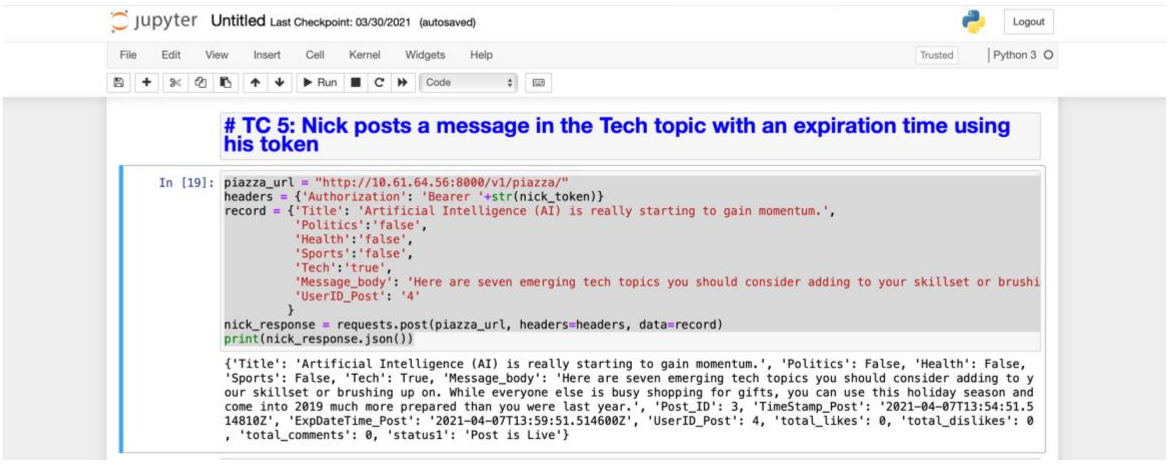
Action List

```
POST /v1/action/

HTTP 400 Bad Request
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "Post_ID": [
        "{'Post EXPITED ! YOu CANNOT Interact With This Post'}"
    ]
}
```

UserID Post	nick
Post ID	piazza object (2) {'Post EXPITED ! YOu CANNOT Interact With This Post'}
Actions	Like
Comments	

As you can note from the above screenshots of TC 4, we have put post expiration of 5 min for the Olga’s post. As the time expired no user was able to interact with the post.

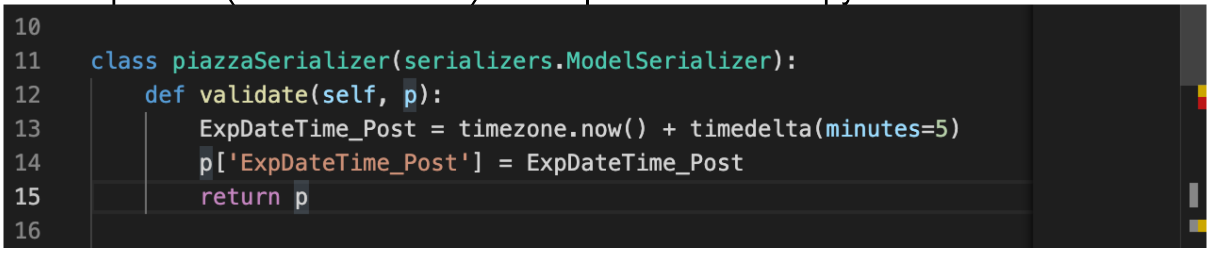
TC 5: Nick posts a message in the Tech topic with an expiration time using his token


jupyter Untitled Last Checkpoint: 03/30/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 5: Nick posts a message in the Tech topic with an expiration time using his token

```
In [19]: piazza_url = "http://10.61.64.56:8000/v1/piazza/"
         headers = {'Authorization': 'Bearer '+str(nick_token)}
         record = {'Title': 'Artificial Intelligence (AI) is really starting to gain momentum.',
                   'Politics':'false',
                   'Health':'false',
                   'Sports':'false',
                   'Tech':'true',
                   'Message_body': 'Here are seven emerging tech topics you should consider adding to your skillset or brushi
                   'UserID_Post': '4'
                  }
         nick_response = requests.post(piazza_url, headers=headers, data=record)
         print(nick_response.json())

         {'Title': 'Artificial Intelligence (AI) is really starting to gain momentum.', 'Politics': False, 'Health': False,
         'Sports': False, 'Tech': True, 'Message_body': 'Here are seven emerging tech topics you should consider adding to y
         our skillset or brushing up on. While everyone else is busy shopping for gifts, you can use this holiday season and
         come into 2019 much more prepared than you were last year.', 'Post_ID': 3, 'TimeStamp_Post': '2021-04-07T13:54:51.5
         14810Z', 'ExpDateTime_Post': '2021-04-07T13:59:51.514600Z', 'UserID_Post': 4, 'total_likes': 0, 'total_dislikes': 0
         , 'total_comments': 0, 'status1': 'Post is Live'}
```

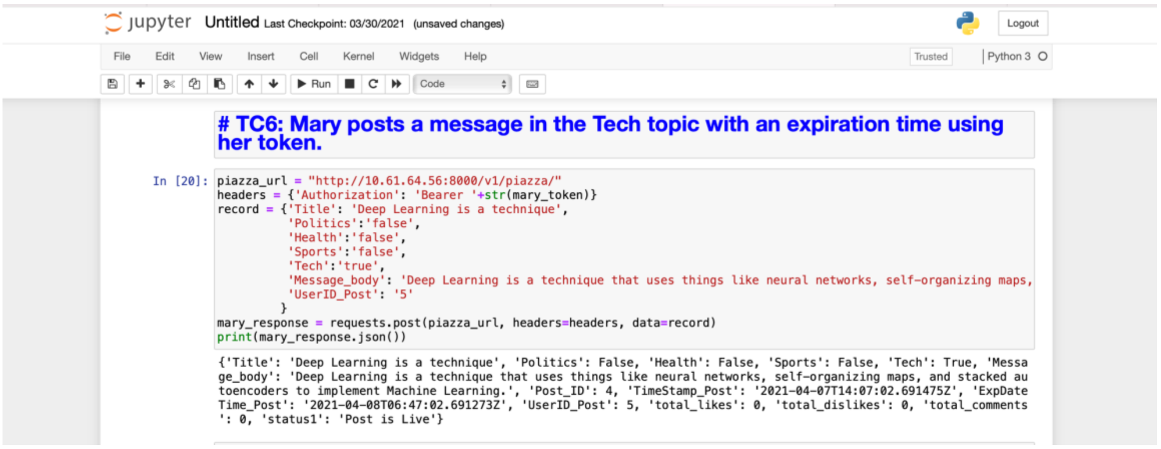
Post expiration (time set to 5 min) in the ‘piazza/serializer.py’ in ‘validate’



```
class piazzaSerializer(serializers.ModelSerializer):
    def validate(self, p):
        ExpDateTime_Post = timezone.now() + timedelta(minutes=5)
        p['ExpDateTime_Post'] = ExpDateTime_Post
        return p
```

As you can note from the above screenshots of TC 5, we have put post expiration of 5 min for the Nick’s post.

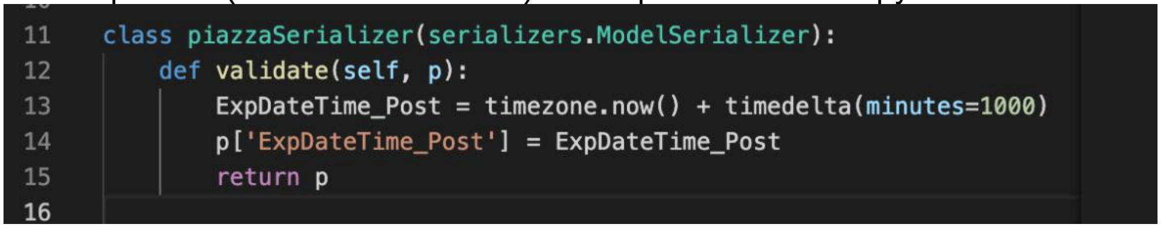
TC6: Mary posts a message in the Tech topic with an expiration time using her token.


jupyter Untitled Last Checkpoint: 03/30/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC6: Mary posts a message in the Tech topic with an expiration time using her token.

```
In [20]: piazza_url = "http://10.61.64.56:8000/v1/piazza/"
         headers = {'Authorization': 'Bearer '+str(mary_token)}
         record = {'Title': 'Deep Learning is a technique',
                   'Politics':'false',
                   'Health':'false',
                   'Sports':'false',
                   'Tech':'true',
                   'Message_body': 'Deep Learning is a technique that uses things like neural networks, self-organizing maps,
                   'UserID_Post': '5'
                  }
         mary_response = requests.post(piazza_url, headers=headers, data=record)
         print(mary_response.json())

         {'Title': 'Deep Learning is a technique', 'Politics': False, 'Health': False, 'Sports': False, 'Tech': True, 'Messa
         ge_body': 'Deep Learning is a technique that uses things like neural networks, self-organizing maps, and stacked au
         toencoders to implement Machine Learning.', 'Post_ID': 4, 'TimeStamp_Post': '2021-04-07T14:07:02.691475Z', 'ExpDate
         Time_Post': '2021-04-08T06:47:02.691273Z', 'UserID_Post': 5, 'total_likes': 0, 'total_dislikes': 0, 'total_comments
         ': 0, 'status1': 'Post is Live'}
```

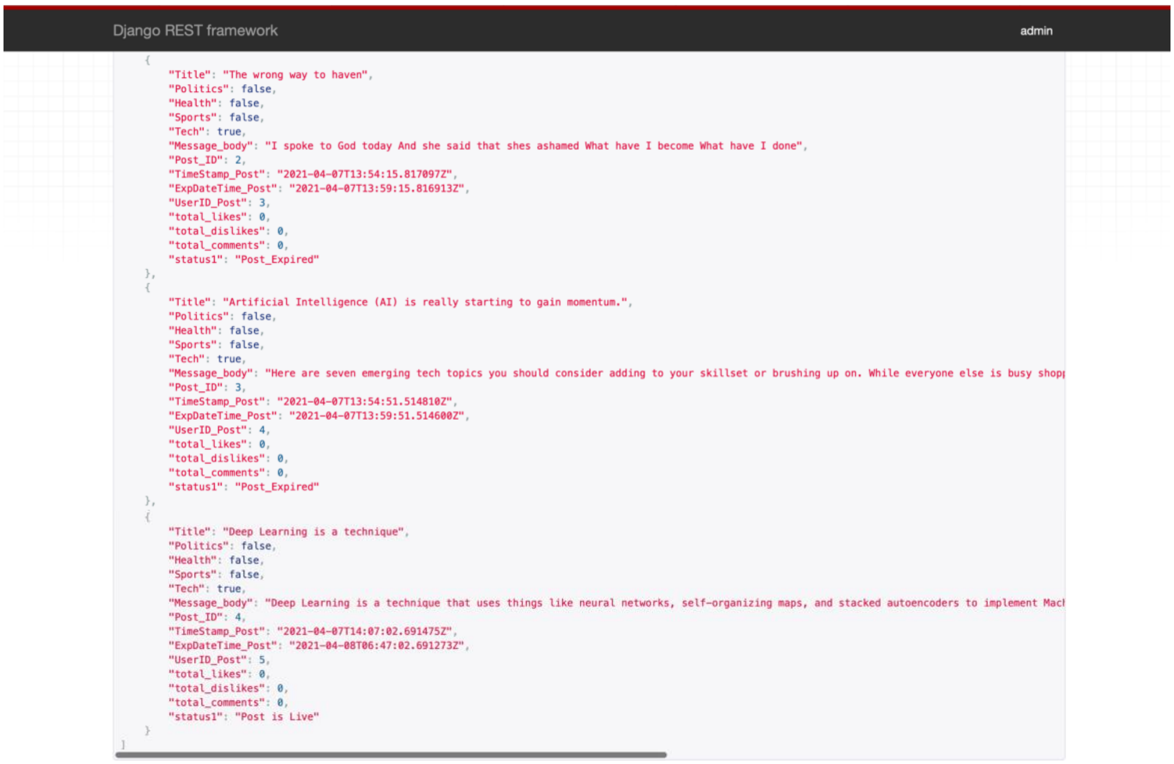
Post expiration (time set to 1000 min) in the ‘piazza/serializer.py’ in ‘validate’



```
class piazzaSerializer(serializers.ModelSerializer):
    def validate(self, p):
        ExpDateTime_Post = timezone.now() + timedelta(minutes=1000)
        p['ExpDateTime_Post'] = ExpDateTime_Post
        return p
```

As you can note from the above screenshots of TC 6, we have put post expiration of 1000min for the Mary’s post.

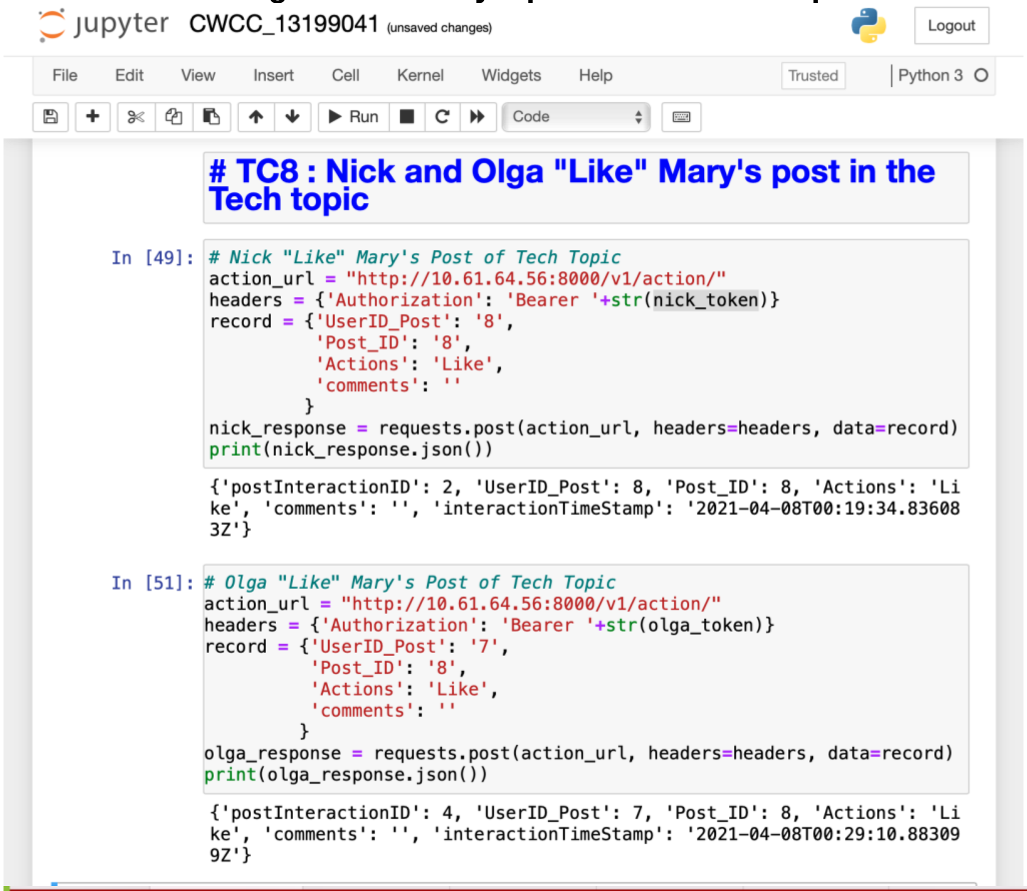
TC7 : Nick and Olga Browse all the available post in the Tech topics, there should be three post available with zero likes, zero dislikes and without and comments.


Django REST framework admin

```
    {
        "Title": "The wrong way to haven",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "I spoke to God today And she said that shes ashamed What have I become What have I done",
        "Post_ID": 2,
        "TimeStamp_Post": "2021-04-07T13:54:15.817097Z",
        "ExpDateTime_Post": "2021-04-07T13:59:15.816913Z",
        "UserID_Post": 3,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Artificial Intelligence (AI) is really starting to gain momentum.",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Here are seven emerging tech topics you should consider adding to your skillset or brushing up on. While everyone else is busy shop
        "Post_ID": 3,
        "TimeStamp_Post": "2021-04-07T13:54:51.514810Z",
        "ExpDateTime_Post": "2021-04-07T13:59:51.514600Z",
        "UserID_Post": 4,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Deep Learning is a technique",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Deep Learning is a technique that uses things like neural networks, self-organizing maps, and stacked autoencoders to implement Mac
        "Post_ID": 4,
        "TimeStamp_Post": "2021-04-07T14:07:02.691475Z",
        "ExpDateTime_Post": "2021-04-08T06:47:02.691273Z",
        "UserID_Post": 5,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post is Live"
    }
]
```

As you can note from the above screenshots of TC 7, we can see the three post with zero likes, zero dislikes and without and comments.

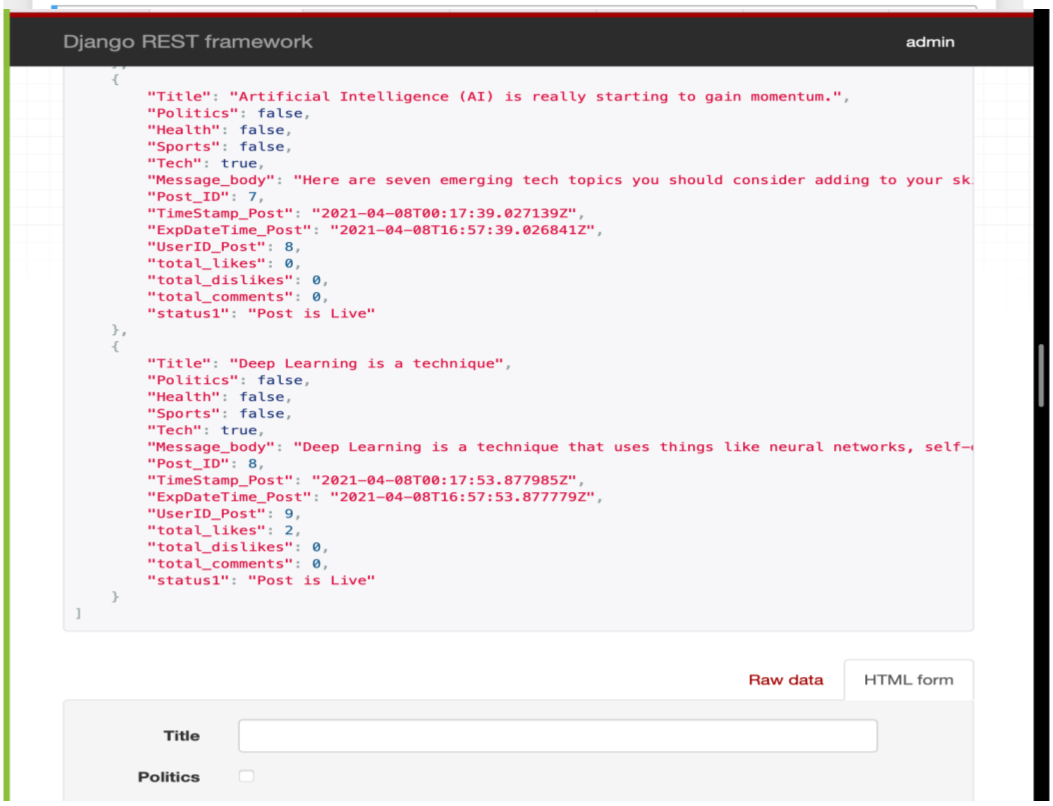
TC8 : Nick and Olga "Like" Mary's post in the Tech topic


jupyter CWCC_13199041 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC8 : Nick and Olga "Like" Mary's post in the Tech topic

```
In [49]: # Nick "Like" Mary's Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(nick_token)}
record = {'UserID_Post': '8',
          'Post_ID': '8',
          'Actions': 'Like',
          'comments': ''
         }
nick_response = requests.post(action_url, headers=headers, data=record)
print(nick_response.json())

{'postInteractionID': 2, 'UserID_Post': 8, 'Post_ID': 8, 'Actions': 'Li
ke', 'comments': '', 'interactionTimeStamp': '2021-04-08T00:19:34.83608
3Z'}
```

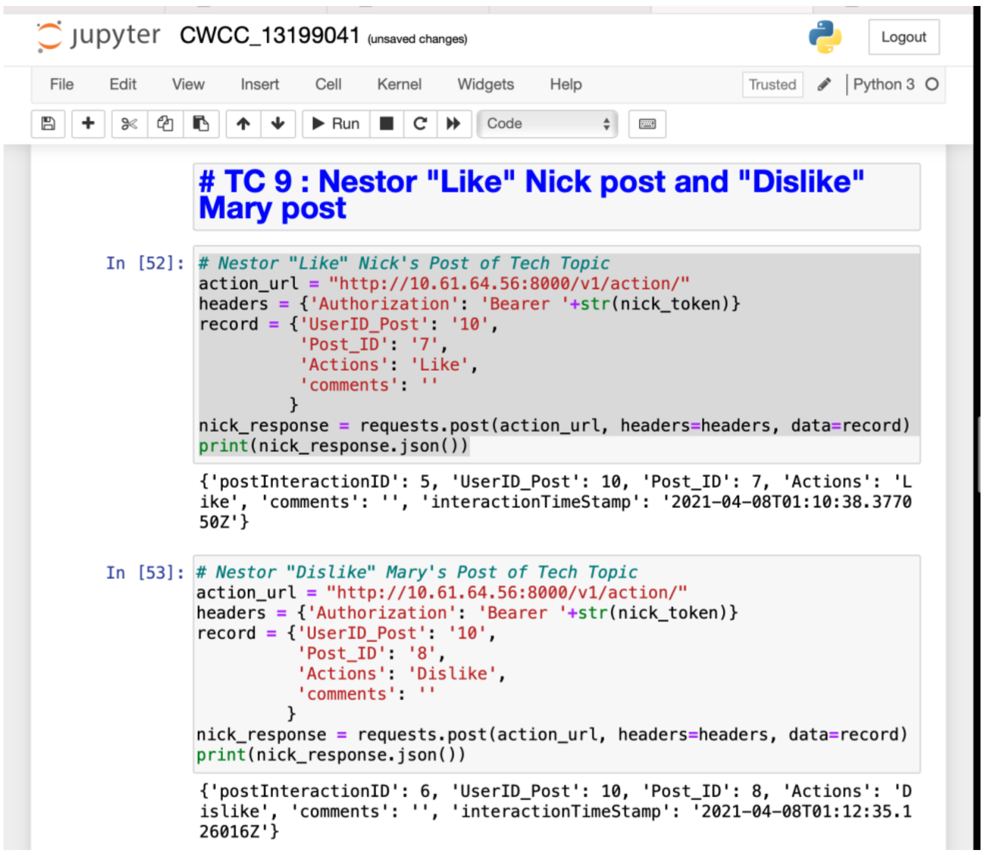
```
In [51]: # Olga "Like" Mary's Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(olga_token)}
record = {'UserID_Post': '7',
          'Post_ID': '8',
          'Actions': 'Like',
          'comments': ''
         }
olga_response = requests.post(action_url, headers=headers, data=record)
print(olga_response.json())

{'postInteractionID': 4, 'UserID_Post': 7, 'Post_ID': 8, 'Actions': 'Li
ke', 'comments': '', 'interactionTimeStamp': '2021-04-08T00:29:10.88309
9Z'}
```



```
    {
        "Title": "Artificial Intelligence (AI) is really starting to gain momentum.",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Here are seven emerging tech topics you should consider adding to your sk
        "Post_ID": 7,
        "TimeStamp_Post": "2021-04-08T00:17:39.027139Z",
        "ExpDateTime_Post": "2021-04-08T16:57:39.026841Z",
        "UserID_Post": 8,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post is Live"
    },
    {
        "Title": "Deep Learning is a technique",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Deep Learning is a technique that uses things like neural networks, self-
        "Post_ID": 8,
        "TimeStamp_Post": "2021-04-08T00:17:53.877985Z",
        "ExpDateTime_Post": "2021-04-08T16:57:53.877779Z",
        "UserID_Post": 9,
        "total_likes": 2,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post is Live"
    }
]
```

As you can note from the above screenshots of TC 8, Nick and Olga "Like" Mary's post and the total likes on Mary’s Post = 2.

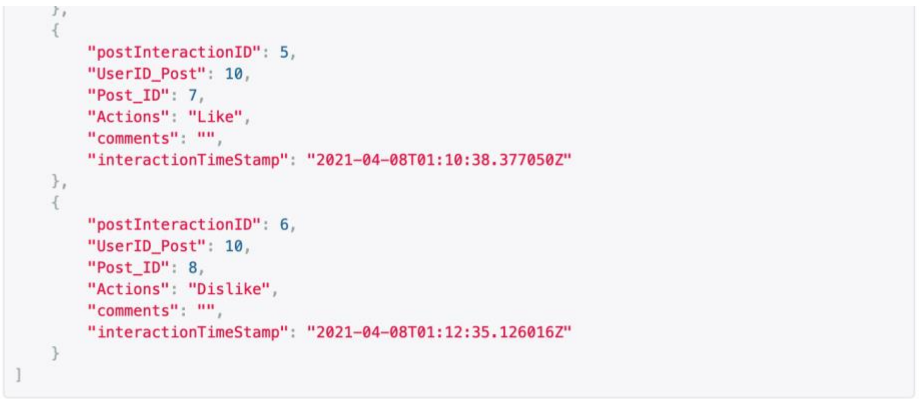
TC 9 : Nestor "Like" Nick post and "Dislike" Mary post in the Tech topic


jupyter CWCC_13199041 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 9 : Nestor "Like" Nick post and "Dislike" Mary post

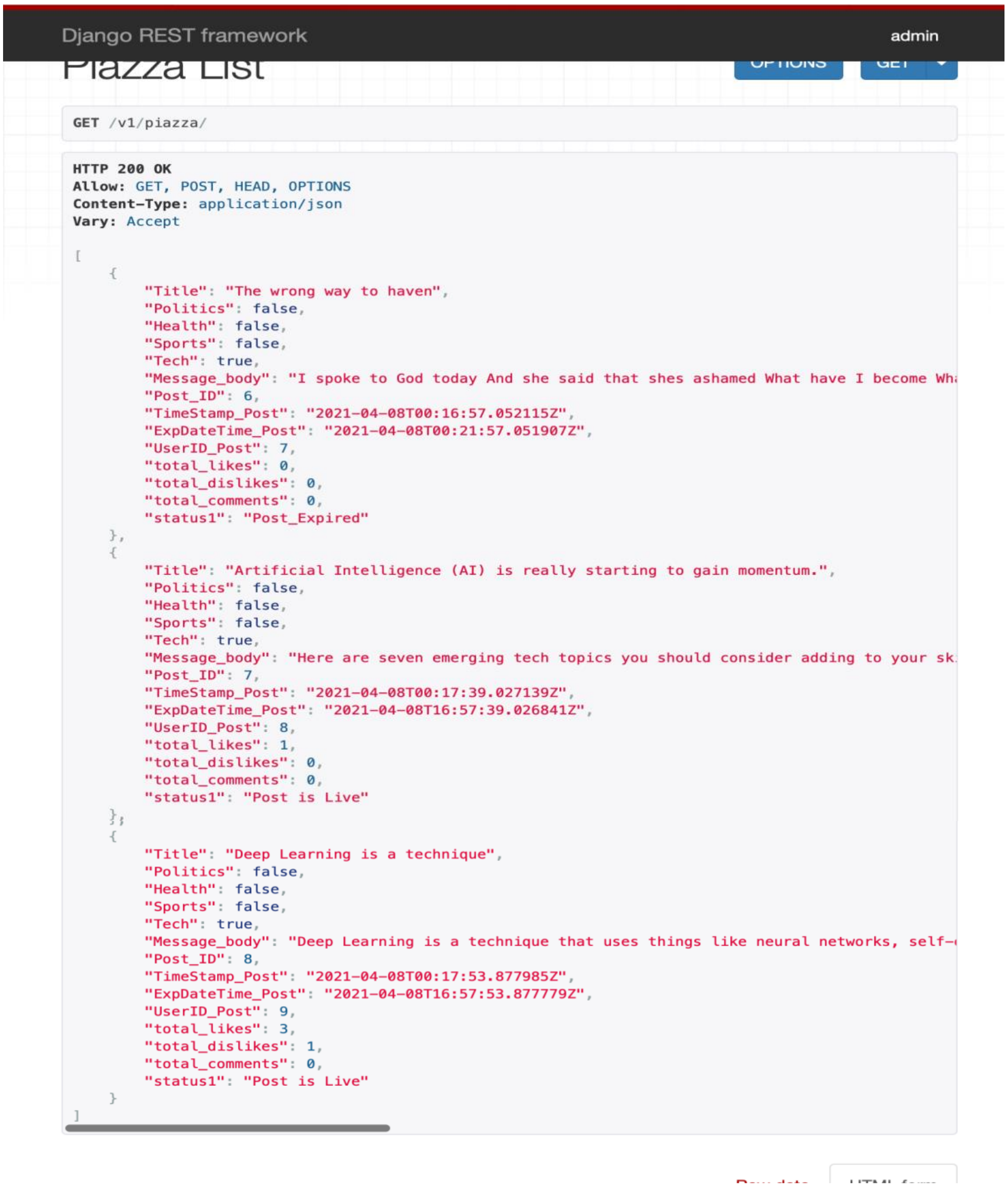
```
In [52]: # Nestor "Like" Nick's Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(nick_token)}
record = {'UserID_Post': '10',
          'Post_ID': '7',
          'Actions': 'Like',
          'comments': ''
         }
nick_response = requests.post(action_url, headers=headers, data=record)
print(nick_response.json())

{'postInteractionID': 5, 'UserID_Post': 10, 'Post_ID': 7, 'Actions': 'L
ike', 'comments': '', 'interactionTimeStamp': '2021-04-08T01:10:38.3770
50Z'}
```

```
In [53]: # Nestor "Dislike" Mary's Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(nick_token)}
record = {'UserID_Post': '10',
          'Post_ID': '8',
          'Actions': 'Dislike',
          'comments': ''
         }
nick_response = requests.post(action_url, headers=headers, data=record)
print(nick_response.json())

{'postInteractionID': 6, 'UserID_Post': 10, 'Post_ID': 8, 'Actions': 'D
islike', 'comments': '', 'interactionTimeStamp': '2021-04-08T01:12:35.1
26016Z'}
```


```
    },
    {
        "postInteractionID": 5,
        "UserID_Post": 10,
        "Post_ID": 7,
        "Actions": "Like",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T01:10:38.377050Z"
    },
    {
        "postInteractionID": 6,
        "UserID_Post": 10,
        "Post_ID": 8,
        "Actions": "Dislike",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T01:12:35.126016Z"
    }
]
```

As you can note from the above screenshots of TC 9, Nestor(with ‘UserID_post’ :10) "Like" Nick post (‘Post_ID’ :7)and "Dislike" Mary post (‘Post_ID’ :8).

TC 10: Nick browses all the available posts in the Tech topic; at this stage he can see the number of likes and dislikes for each post (Mary has 2 likes and 1 dislike and Nick has 1 like). There are no comments made yet.

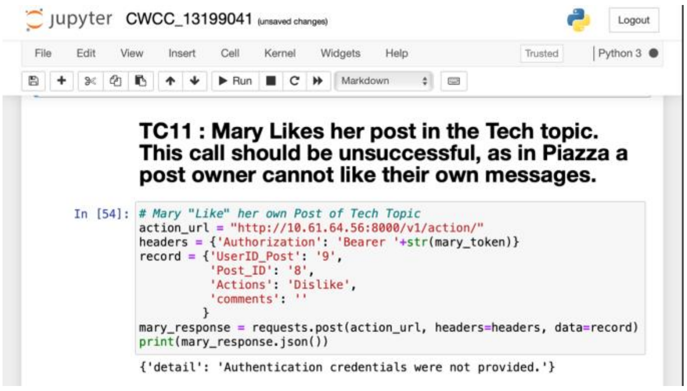


```
GET /v1/piazza/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "Title": "The wrong way to haven",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "I spoke to God today And she said that shes ashamed What have I become Wh
        "Post_ID": 6,
        "TimeStamp_Post": "2021-04-08T00:16:57.052115Z",
        "ExpDateTime_Post": "2021-04-08T00:21:57.051907Z",
        "UserID_Post": 7,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Artificial Intelligence (AI) is really starting to gain momentum.",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Here are seven emerging tech topics you should consider adding to your sk
        "Post_ID": 7,
        "TimeStamp_Post": "2021-04-08T00:17:39.027139Z",
        "ExpDateTime_Post": "2021-04-08T16:57:39.026841Z",
        "UserID_Post": 8,
        "total_likes": 1,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post is Live"
    },
    {
        "Title": "Deep Learning is a technique",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Deep Learning is a technique that uses things like neural networks, self-
        "Post_ID": 8,
        "TimeStamp_Post": "2021-04-08T00:17:53.877985Z",
        "ExpDateTime_Post": "2021-04-08T16:57:53.877779Z",
        "UserID_Post": 9,
        "total_likes": 3,
        "total_dislikes": 1,
        "total_comments": 0,
        "status1": "Post is Live"
    }
]
```

Note for TC 10: Mary's Post has 3 likes, because in the previous Test case 8, I had accidently liked the post of Mary again after taking the screenshots. Thus you can see Mary's post will have 3 likes and 1 dislike.

TC11 : Mary Likes her post in the Tech topic. This call should be unsuccessful, as in Piazza a post owner cannot like their own messages.

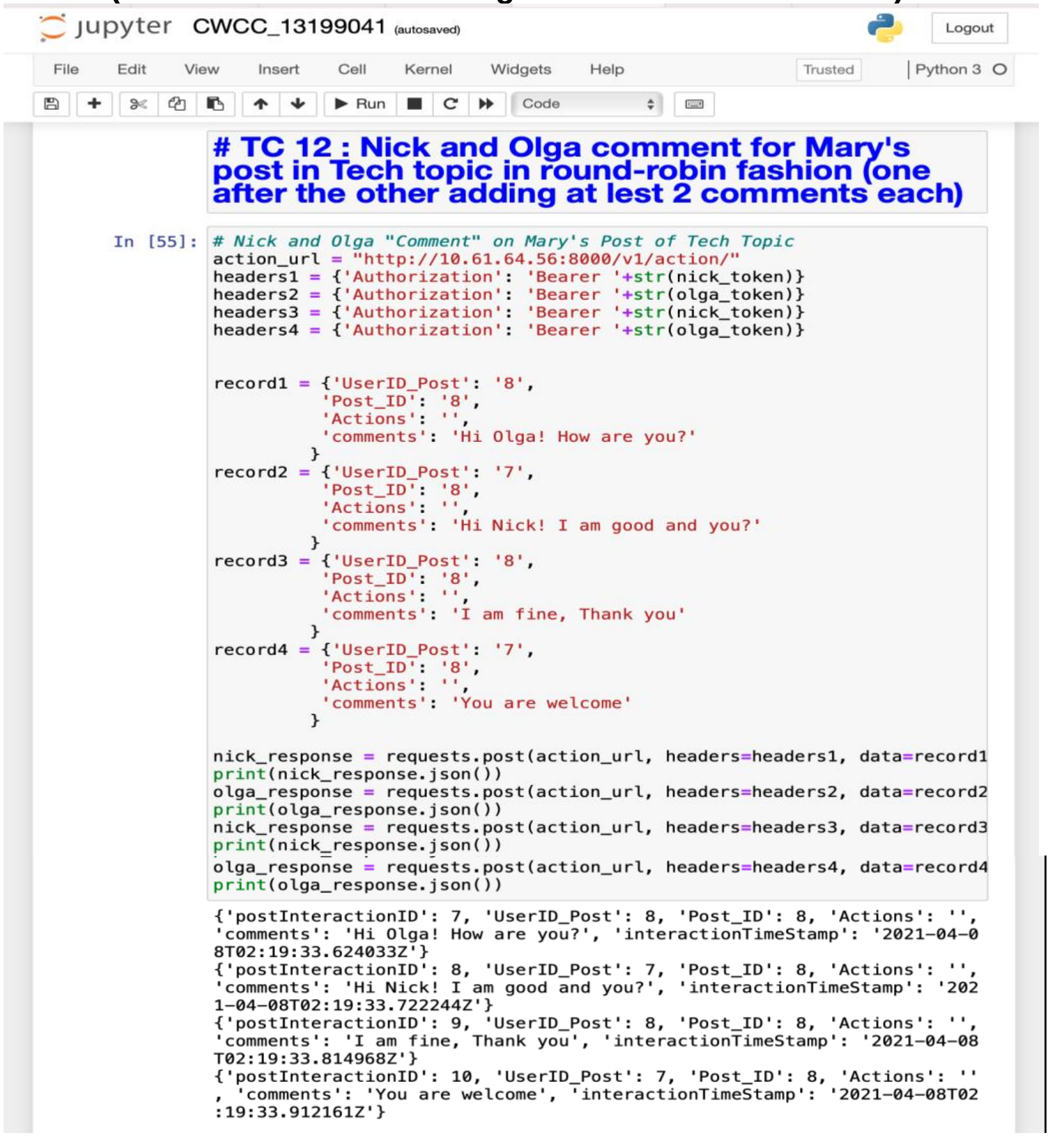

jupyter CWCC_13199041 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

TC11 : Mary Likes her post in the Tech topic. This call should be unsuccessful, as in Piazza a post owner cannot like their own messages.

```
In [54]: # Mary "Like" her own Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(mary_token)}
record = {'UserID_Post': '9',
          'Post_ID': '8',
          'Actions': 'Dislike',
          'comments': ''
         }
mary_response = requests.post(action_url, headers=headers, data=record)
print(mary_response.json())

{'detail': 'Authentication credentials were not provided.'}
```

As you can note from the above screenshots of TC 9, Mary is not able to Like her own post

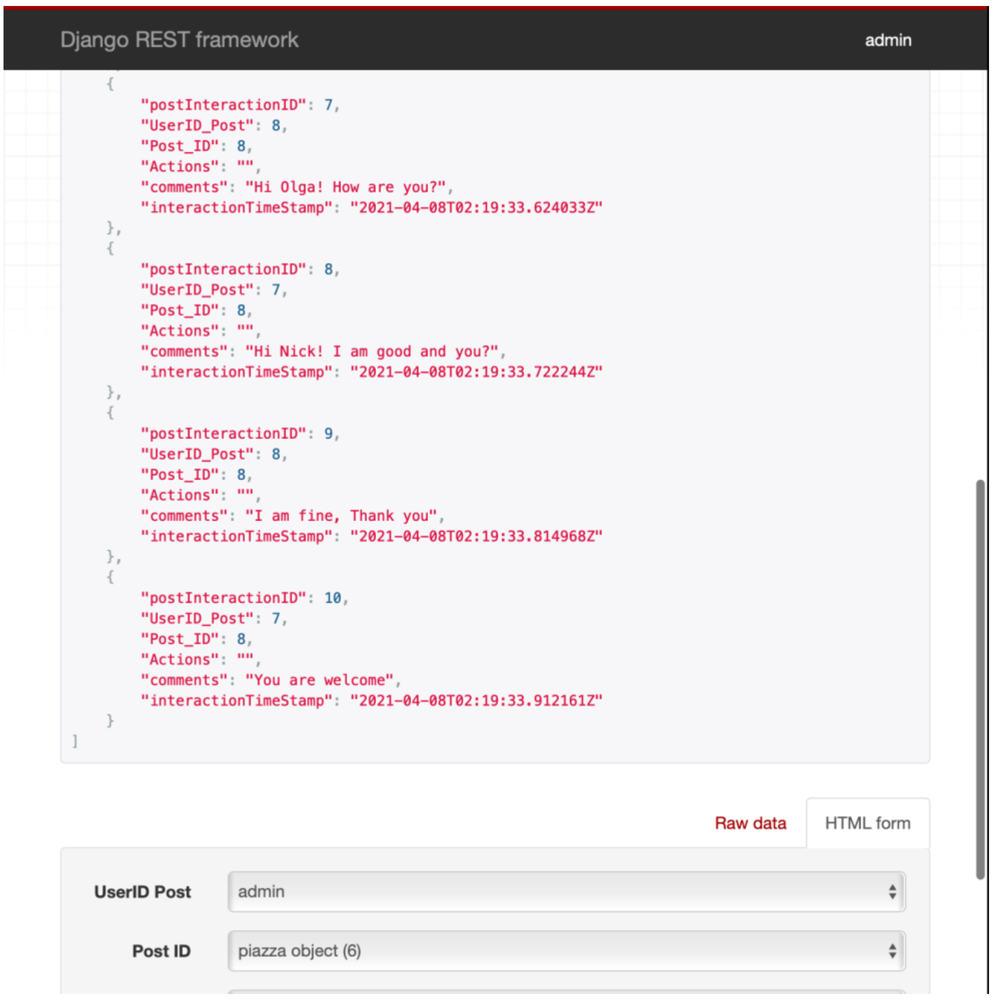
TC 12 : Nick and Olga comment for Mary's post in Tech topic in round-robin fashion (one after the other adding at lest 2 comments each)


jupyter CWCC_13199041 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

# TC 12 : Nick and Olga comment for Mary's post in Tech topic in round-robin fashion (one after the other adding at lest 2 comments each)

```
In [55]: # Nick and Olga "Comment" on Mary's Post of Tech Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers1 = {'Authorization': 'Bearer '+str(nick_token)}
headers2 = {'Authorization': 'Bearer '+str(olga_token)}
headers3 = {'Authorization': 'Bearer '+str(nick_token)}
headers4 = {'Authorization': 'Bearer '+str(olga_token)}


record1 = {'UserID_Post': '8',
           'Post_ID': '8',
           'Actions': '',
           'comments': 'Hi Olga! How are you?'
          }
record2 = {'UserID_Post': '7',
           'Post_ID': '8',
           'Actions': '',
           'comments': 'Hi Nick! I am good and you?'
          }
record3 = {'UserID_Post': '8',
           'Post_ID': '8',
           'Actions': '',
           'comments': 'I am fine, Thank you'
          }
record4 = {'UserID_Post': '7',
           'Post_ID': '8',
           'Actions': '',
           'comments': 'You are welcome'
          }

nick_response = requests.post(action_url, headers=headers1, data=record1
print(nick_response.json())
olga_response = requests.post(action_url, headers=headers2, data=record2
print(olga_response.json())
nick_response = requests.post(action_url, headers=headers3, data=record3
print(nick_response.json())
olga_response = requests.post(action_url, headers=headers4, data=record4
print(olga_response.json())

{'postInteractionID': 7, 'UserID_Post': 8, 'Post_ID': 8, 'Actions': '',
'comments': 'Hi Olga! How are you?', 'interactionTimeStamp': '2021-04-0
8T02:19:33.624033Z'}
{'postInteractionID': 8, 'UserID_Post': 7, 'Post_ID': 8, 'Actions': '',
'comments': 'Hi Nick! I am good and you?', 'interactionTimeStamp': '202
1-04-08T02:19:33.722244Z'}
{'postInteractionID': 9, 'UserID_Post': 8, 'Post_ID': 8, 'Actions': '',
'comments': 'I am fine, Thank you', 'interactionTimeStamp': '2021-04-08
T02:19:33.814968Z'}
{'postInteractionID': 10, 'UserID_Post': 7, 'Post_ID': 8, 'Actions': ''
, 'comments': 'You are welcome', 'interactionTimeStamp': '2021-04-08T02
:19:33.912161Z'}
```

Back to back comments added by Nick and Olga

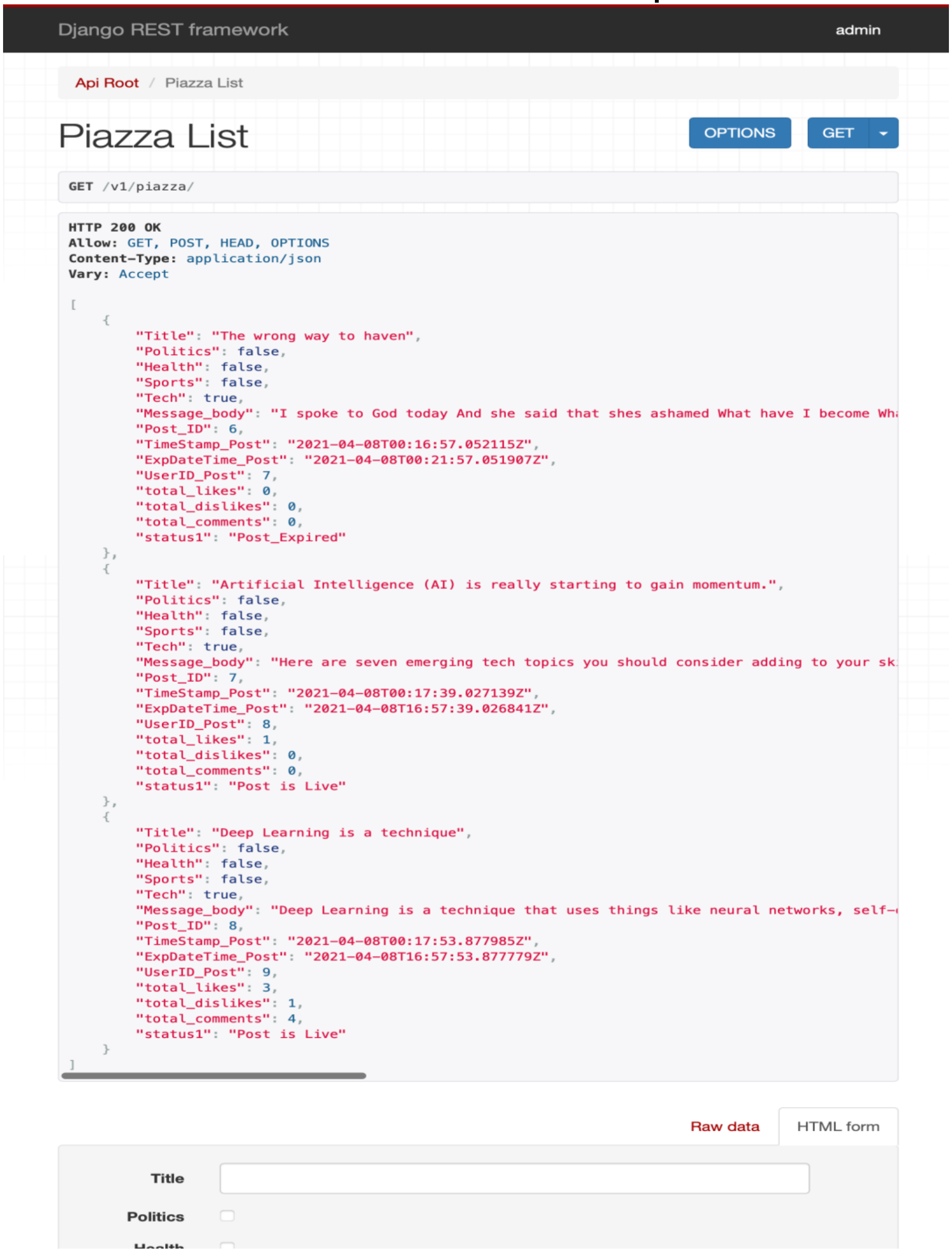


```
    {
        "postInteractionID": 7,
        "UserID_Post": 8,
        "Post_ID": 8,
        "Actions": "",
        "comments": "Hi Olga! How are you?",
        "interactionTimeStamp": "2021-04-08T02:19:33.624033Z"
    },
    {
        "postInteractionID": 8,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "",
        "comments": "Hi Nick! I am good and you?",
        "interactionTimeStamp": "2021-04-08T02:19:33.722244Z"
    },
    {
        "postInteractionID": 9,
        "UserID_Post": 8,
        "Post_ID": 8,
        "Actions": "",
        "comments": "I am fine, Thank you",
        "interactionTimeStamp": "2021-04-08T02:19:33.814968Z"
    },
    {
        "postInteractionID": 10,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "",
        "comments": "You are welcome",
        "interactionTimeStamp": "2021-04-08T02:19:33.912161Z"
    }
]
```

UserID Post	admin
Post ID	piazza object (6)

As you can note from the above screenshots of TC 12, the comment are posted in round-robin fashion on Mary’s post.

TC 13: Nick browses all the available post in the Tech topic; at this stage he can see the number of likes and dislikes of each post and the comments made.

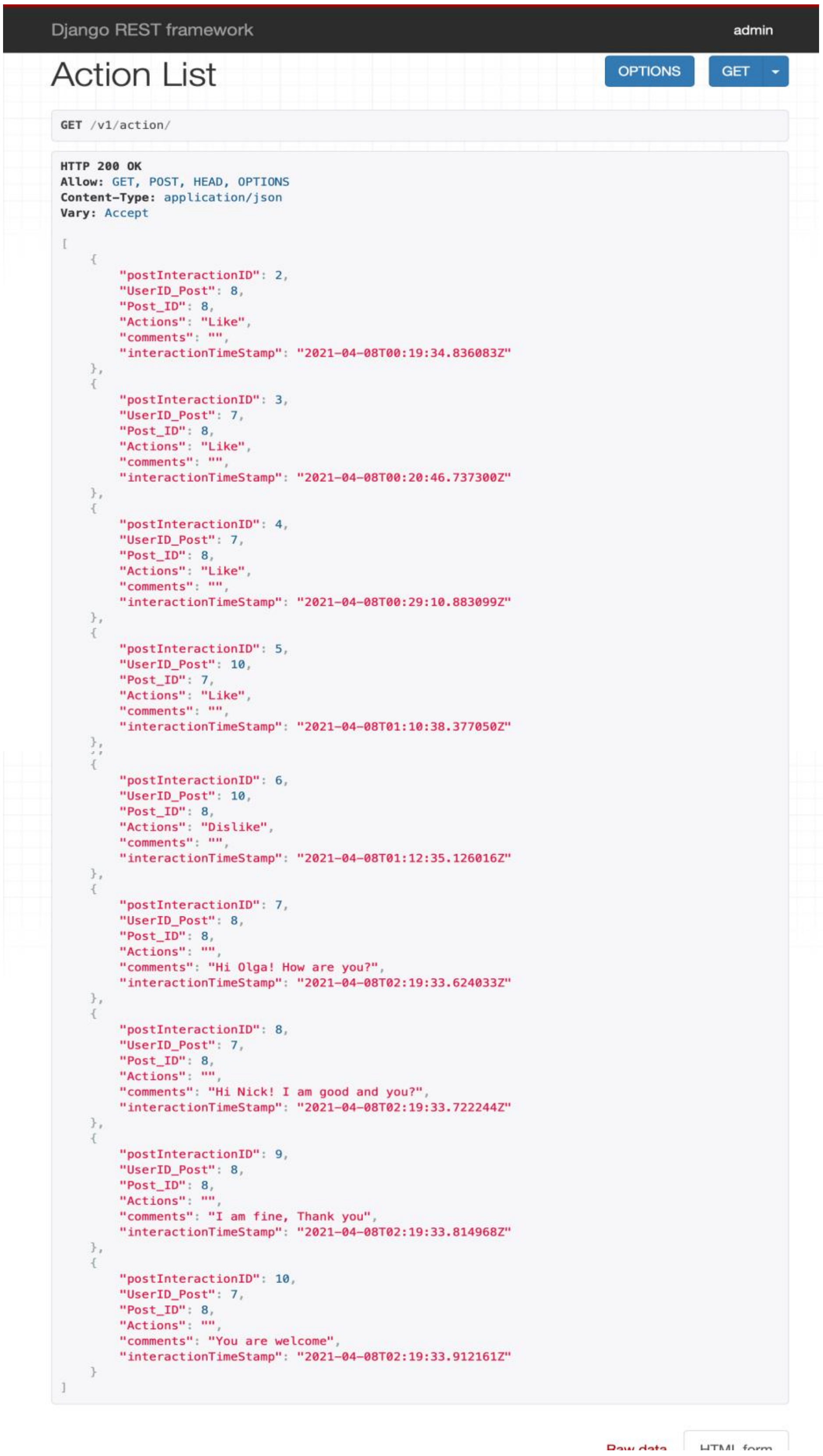


Piazza List

```
GET /v1/piazza/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "Title": "The wrong way to haven",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "I spoke to God today And she said that shes ashamed What have I become Whi
        "Post_ID": 6,
        "TimeStamp_Post": "2021-04-08T00:16:57.052115Z",
        "ExpDateTime_Post": "2021-04-08T00:21:57.051907Z",
        "UserID_Post": 7,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Artificial Intelligence (AI) is really starting to gain momentum.",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Here are seven emerging tech topics you should consider adding to your sk
        "Post_ID": 7,
        "TimeStamp_Post": "2021-04-08T00:17:39.027139Z",
        "ExpDateTime_Post": "2021-04-08T16:57:39.026841Z",
        "UserID_Post": 8,
        "total_likes": 1,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post is Live"
    },
    {
        "Title": "Deep Learning is a technique",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Deep Learning is a technique that uses things like neural networks, self-
        "Post_ID": 8,
        "TimeStamp_Post": "2021-04-08T00:17:53.877985Z",
        "ExpDateTime_Post": "2021-04-08T16:57:53.877779Z",
        "UserID_Post": 9,
        "total_likes": 3,
        "total_dislikes": 1,
        "total_comments": 4,
        "status1": "Post is Live"
    }
]
```

Title
Politics

Detailed Like and comment of each post can be seen in Action List below


Django REST framework admin

Action List

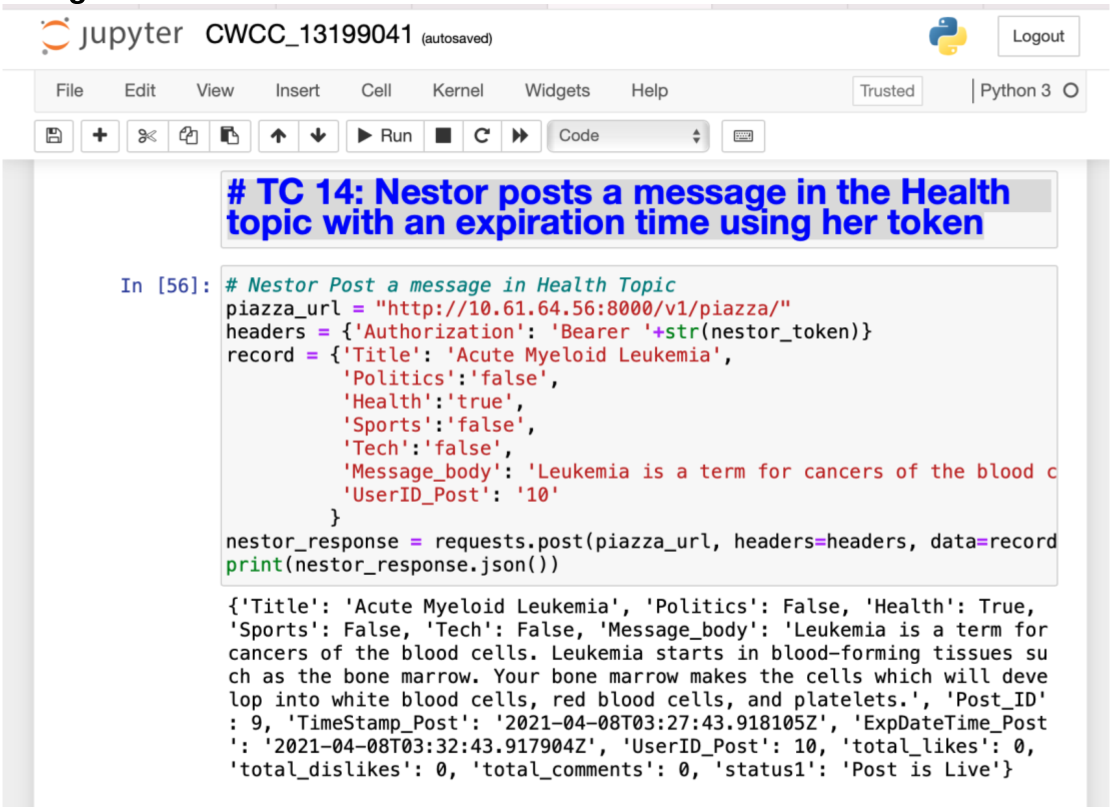
OPTIONS GET

```
GET /v1/action/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "postInteractionID": 2,
        "UserID_Post": 8,
        "Post_ID": 8,
        "Actions": "Like",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T00:19:34.836083Z"
    },
    {
        "postInteractionID": 3,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "Like",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T00:20:46.737300Z"
    },
    {
        "postInteractionID": 4,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "Like",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T00:29:10.883099Z"
    },
    {
        "postInteractionID": 5,
        "UserID_Post": 10,
        "Post_ID": 7,
        "Actions": "Like",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T01:10:38.377050Z"
    },
    {
        "postInteractionID": 6,
        "UserID_Post": 10,
        "Post_ID": 8,
        "Actions": "Dislike",
        "comments": "",
        "interactionTimeStamp": "2021-04-08T01:12:35.126016Z"
    },
    {
        "postInteractionID": 7,
        "UserID_Post": 8,
        "Post_ID": 8,
        "Actions": "",
        "comments": "Hi Olga! How are you?",
        "interactionTimeStamp": "2021-04-08T02:19:33.624033Z"
    },
    {
        "postInteractionID": 8,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "",
        "comments": "Hi Nick! I am good and you?",
        "interactionTimeStamp": "2021-04-08T02:19:33.722244Z"
    },
    {
        "postInteractionID": 9,
        "UserID_Post": 8,
        "Post_ID": 8,
        "Actions": "",
        "comments": "I am fine, Thank you",
        "interactionTimeStamp": "2021-04-08T02:19:33.814968Z"
    },
    {
        "postInteractionID": 10,
        "UserID_Post": 7,
        "Post_ID": 8,
        "Actions": "",
        "comments": "You are welcome",
        "interactionTimeStamp": "2021-04-08T02:19:33.912161Z"
    }
]
```

As you can note from the above screenshots of TC 12, Nick can see the number of likes and dislikes of each post and the comments made.

TC 14: Nestor posts a message in the Health topic with an expiration time using her token

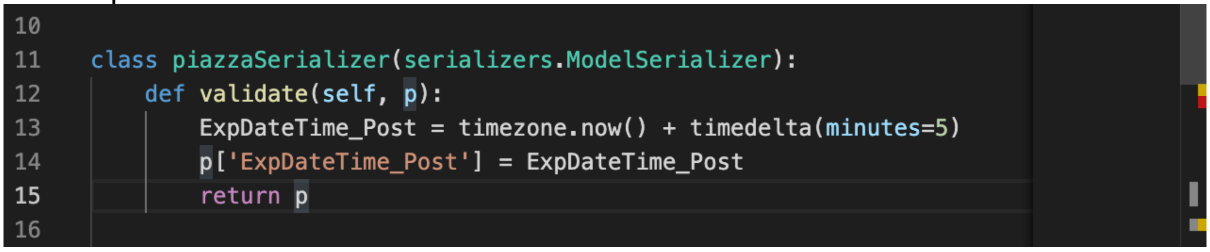

jupyter CWCC_13199041 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 14: Nestor posts a message in the Health topic with an expiration time using her token

```
In [56]: # Nestor Post a message in Health Topic
piazza_url = "http://10.61.64.56:8000/v1/piazza/"
headers = {'Authorization': 'Bearer '+str(nestor_token)}
record = {'Title': 'Acute Myeloid Leukemia',
          'Politics':'false',
          'Health':'true',
          'Sports':'false',
          'Tech':'false',
          'Message_body': 'Leukemia is a term for cancers of the blood c
          'UserID_Post': '10'
         }
nestor_response = requests.post(piazza_url, headers=headers, data=record
print(nestor_response.json())
```

```
{'Title': 'Acute Myeloid Leukemia', 'Politics': False, 'Health': True,
'Sports': False, 'Tech': False, 'Message_body': 'Leukemia is a term for
cancers of the blood cells. Leukemia starts in blood-forming tissues su
ch as the bone marrow. Your bone marrow makes the cells which will deve
lop into white blood cells, red blood cells, and platelets.', 'Post_ID'
: 9, 'TimeStamp_Post': '2021-04-08T03:27:43.918105Z', 'ExpDateTime_Post
': '2021-04-08T03:32:43.917904Z', 'UserID_Post': 10, 'total_likes': 0,
'total_dislikes': 0, 'total_comments': 0, 'status1': 'Post is Live'}
```

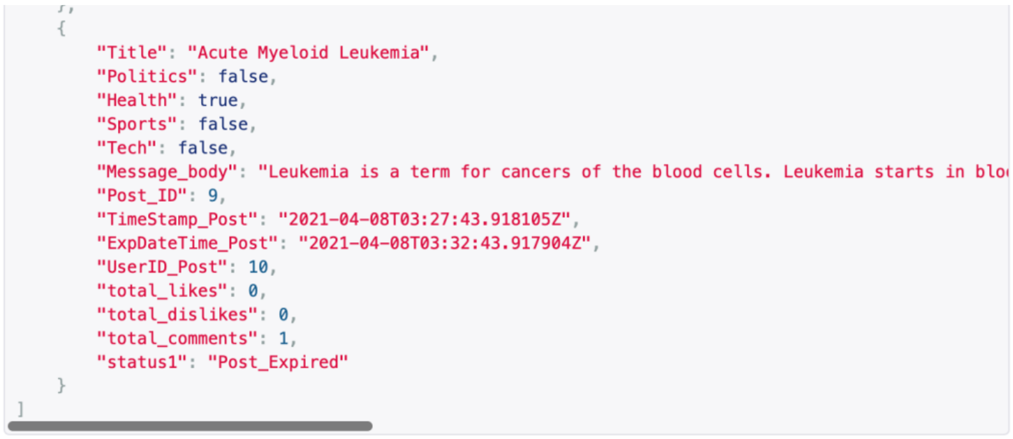
Post expire time 5min.



```

class piazzaSerializer(serializers.ModelSerializer):
    def validate(self, p):
        ExpDateTime_Post = timezone.now() + timedelta(minutes=5)
        p['ExpDateTime_Post'] = ExpDateTime_Post
        return p

```

As you can note from the above screenshots of TC 14, we have put post expiration of 5min for the Nestor’s post.

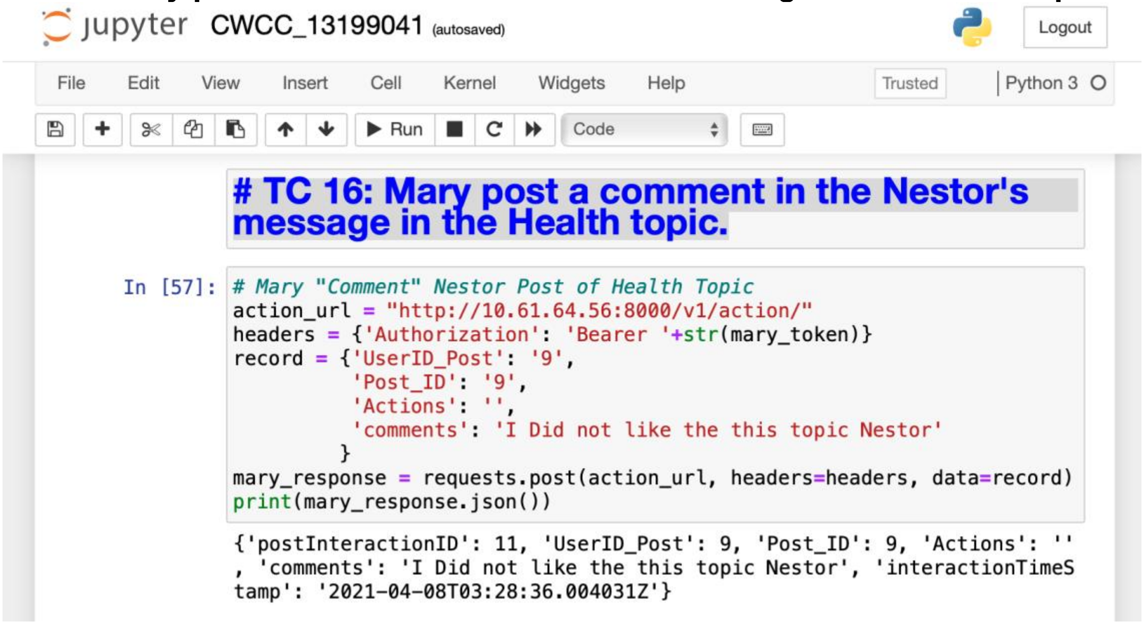
TC 15: Mary browses all the available post in the Health topic; at this stage she can see only Nestor's post



```
    },
    {
        "Title": "Acute Myeloid Leukemia",
        "Politics": false,
        "Health": true,
        "Sports": false,
        "Tech": false,
        "Message_body": "Leukemia is a term for cancers of the blood cells. Leukemia starts in blo
        "Post_ID": 9,
        "TimeStamp_Post": "2021-04-08T03:27:43.918105Z",
        "ExpDateTime_Post": "2021-04-08T03:32:43.917904Z",
        "UserID_Post": 10,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 1,
        "status1": "Post_Expired"
    }
]
```

NOTE for TC 15: This screenshot was taken after performing TC 16 & TC 17. Thus, you can see that the post is expired and there is one comment of Mary

TC 16: Mary post a comment in the Nestor's message in the Health topic.

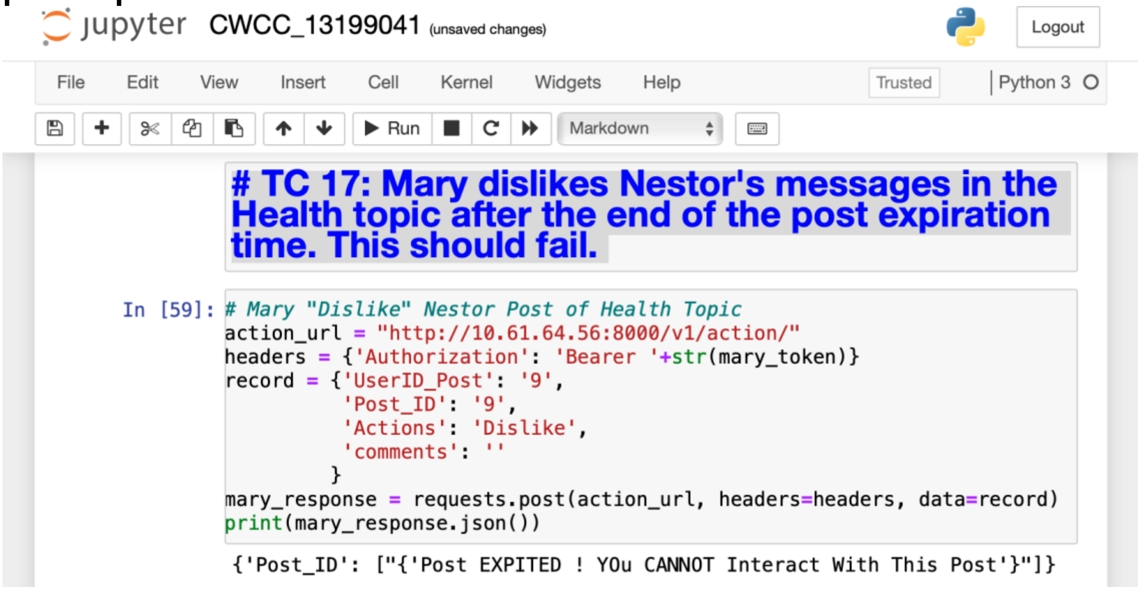

jupyter CWCC_13199041 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

TC 16: Mary post a comment in the Nestor's message in the Health topic.

```
In [57]: # Mary "Comment" Nestor Post of Health Topic
         action_url = "http://10.61.64.56:8000/v1/action/"
         headers = {'Authorization': 'Bearer '+str(mary_token)}
         record = {'UserID_Post': '9',
                   'Post_ID': '9',
                   'Actions': '',
                   'comments': 'I Did not like the this topic Nestor'
                  }
         mary_response = requests.post(action_url, headers=headers, data=record)
         print(mary_response.json())

         {'postInteractionID': 11, 'UserID_Post': 9, 'Post_ID': 9, 'Actions': ''
         , 'comments': 'I Did not like the this topic Nestor', 'interactionTimeS
         tamp': '2021-04-08T03:28:36.004031Z'}
```

As you can note from the above screenshots of TC 16, Mary post a comment in the Nestor's message.

TC 17: Mary dislikes Nestor's messages in the Health topic after the end of the post expiration time. This should fail.

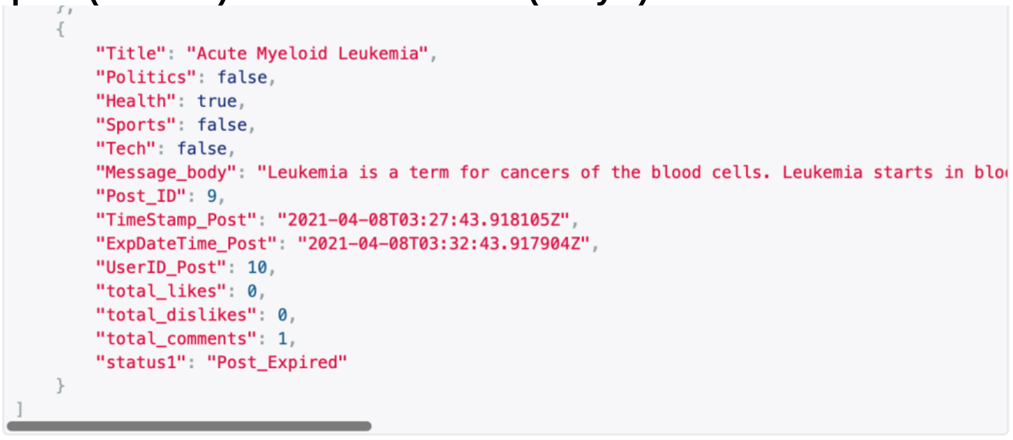

jupyter CWCC_13199041 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown

TC 17: Mary dislikes Nestor's messages in the Health topic after the end of the post expiration time. This should fail.

```
In [59]: # Mary "Dislike" Nestor Post of Health Topic
action_url = "http://10.61.64.56:8000/v1/action/"
headers = {'Authorization': 'Bearer '+str(mary_token)}
record = {'UserID_Post': '9',
          'Post_ID': '9',
          'Actions': 'Dislike',
          'comments': ''
         }
mary_response = requests.post(action_url, headers=headers, data=record)
print(mary_response.json())

 {'Post_ID': ["{'Post EXPITED ! YOu CANNOT Interact With This Post'}"]}
```

As you can note from the above screenshots of TC 17, Mary dislikes Nestor's messages and fail to post as the post expired.

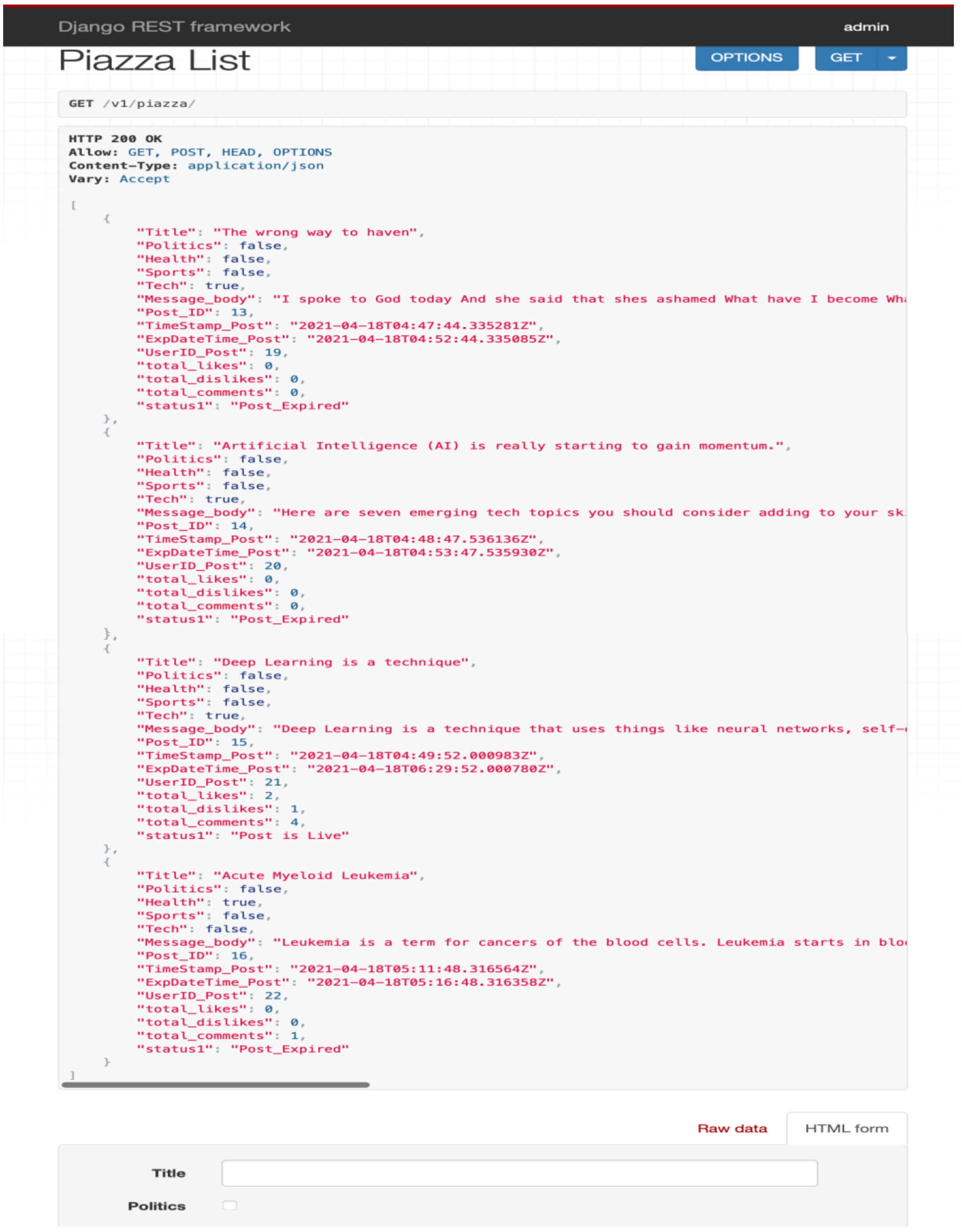
TC 18: Nestor browses all the messages in the Health topic. There should be only post (his own) with one comment (Mary's).



```
    {
        "Title": "Acute Myeloid Leukemia",
        "Politics": false,
        "Health": true,
        "Sports": false,
        "Tech": false,
        "Message_body": "Leukemia is a term for cancers of the blood cells. Leukemia starts in blo
        "Post_ID": 9,
        "TimeStamp_Post": "2021-04-08T03:27:43.918105Z",
        "ExpDateTime_Post": "2021-04-08T03:32:43.917904Z",
        "UserID_Post": 10,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 1,
        "status1": "Post_Expired"
    }
]
```

As you can note from the above screenshots of TC 18, there is only one post (his own) with one comment on it.

TC 19: Nick browses all the expired messages in the Sports topic. These should be empty.



Piazza List

```
GET /v1/piazza/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "Title": "The wrong way to haven",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "I spoke to God today And she said that shes ashamed What have I become Wh
        "Post_ID": 13,
        "TimeStamp_Post": "2021-04-18T04:47:44.335281Z",
        "ExpDateTime_Post": "2021-04-18T04:52:44.335085Z",
        "UserID_Post": 19,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Artificial Intelligence (AI) is really starting to gain momentum.",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Here are seven emerging tech topics you should consider adding to your sk
        "Post_ID": 14,
        "TimeStamp_Post": "2021-04-18T04:48:47.536136Z",
        "ExpDateTime_Post": "2021-04-18T04:53:47.535930Z",
        "UserID_Post": 20,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 0,
        "status1": "Post_Expired"
    },
    {
        "Title": "Deep Learning is a technique",
        "Politics": false,
        "Health": false,
        "Sports": false,
        "Tech": true,
        "Message_body": "Deep Learning is a technique that uses things like neural networks, self-
        "Post_ID": 15,
        "TimeStamp_Post": "2021-04-18T04:49:52.000983Z",
        "ExpDateTime_Post": "2021-04-18T06:29:52.000780Z",
        "UserID_Post": 21,
        "total_likes": 2,
        "total_dislikes": 1,
        "total_comments": 4,
        "status1": "Post is Live"
    },
    {
        "Title": "Acute Myeloid Leukemia",
        "Politics": false,
        "Health": true,
        "Sports": false,
        "Tech": false,
        "Message_body": "Leukemia is a term for cancers of the blood cells. Leukemia starts in blo
        "Post_ID": 16,
        "TimeStamp_Post": "2021-04-18T05:11:48.316564Z",
        "ExpDateTime_Post": "2021-04-18T05:16:48.316358Z",
        "UserID_Post": 22,
        "total_likes": 0,
        "total_dislikes": 0,
        "total_comments": 1,
        "status1": "Post_Expired"
    }
]
```

Title

Politics

As you can note from the above screenshots of TC 19, there is no post in sports topic.

TC 20: Nestor queries for an active post having the highest interest (maximum sum of likes and dislikes) in the Tech topic. This should be Mary's post.

I WAS NOT ABLE TO DO THIS TEST CASE

Phase E: Report your solution in a technical annex

In this report a detailed implementations of how the packages were installed, structure of the folders and creating authorization server an oAuth v2 protocol is mentioned in Phase A & B of this coursework.

A detailed design of database design with description of services are discussed in detail inside the Phase C of the coursework. This creating the models the codes were referred through many online sources , they are:

1) models.CharField [1] [2]
2) models.BooleanField [1]
3) models.AutoField [3]
4) models.DateTimeField(default=timezone.now) [1] [4]
5) models.DateTimeField(null=True) [1] [5]
6) models.ForeignKey [6] [7]
7) @property [8]
8) Action_list = [('',''),('Like', 'Like'), ('Dislike', 'Dislike')] [9]
9) validator statements [10]

Phase F: Submit quality scripts

I have attached the folder of my virtual environment which consist of all the folders and subfolders of the code used to complete this coursework. The ‘piazza/models.py’ and ‘piazza/serializers.py’ consist of codes with comments wherever required.

Conclusion:

In conclusion to the coursework, I would like to say that it was a great learning experiences. I was not able to build a perfect model to run all the test cases, but I have given my best to complete it.

Academic Declaration:

“I have read and understood the sections on plagiarism in the College Policy on assessment offences and confirm that the work is my own, with the work of others clearly acknowledged. I give my permission to submit my work to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software.”

References:

[1] “Model field reference,” *Djangoproject.com*. [Online]. Available: https://docs.djangoproject.com/en/3.2/ref/models/fields/. [Accessed: 18-Apr-2021].

[2] “django.db.models CharField Example Code,” *Fullstackpython.com*. [Online]. Available: https://www.fullstackpython.com/django-db-models-charfield-examples.html. [Accessed: 18-Apr-2021].

[3] “AutoField - Django Models - GeeksforGeeks,” *Geeksforgeeks.org*, 04-Oct-2019. [Online]. Available: https://www.geeksforgeeks.org/autofield-django-models/. [Accessed: 18-Apr-2021].

[4] “DateTimeField - Django Models - GeeksforGeeks,” *Geeksforgeeks.org*, 09-Oct-2019. [Online]. Available: https://www.geeksforgeeks.org/datetimefield-django-models/. [Accessed: 18-Apr-2021].

[5] “How to make Django’s DateTimeField optional?,” *Stackoverflow.com*. [Online]. Available: https://stackoverflow.com/questions/11351619/how-to-make-djangos-datetimefield-optional/11351690. [Accessed: 18-Apr-2021].

[6] “Many-to-one relationships,” *Djangoproject.com*. [Online]. Available: https://docs.djangoproject.com/en/3.2/topics/db/examples/many_to_one/. [Accessed: 18-Apr-2021].

[7] “Foreign key Django model,” *Stackoverflow.com*. [Online]. Available: https://stackoverflow.com/questions/14663523/foreign-key-django-model. [Accessed: 18-Apr-2021].

[8] “What does Django’s @property do?,” *Stackoverflow.com*. [Online]. Available: https://stackoverflow.com/questions/58558989/what-does-djangos-property-do. [Accessed: 18-Apr-2021].

[9] “Dropdown in Django Model,” *Stackoverflow.com*. [Online]. Available: https://stackoverflow.com/questions/31130706/dropdown-in-django-model. [Accessed: 18-Apr-2021].

[10] “Simple Power of Django Validators // Python Django Tutorial // Form Validation // Model Validation” *Youtube.com*. [Online]. Available: https://www.youtube.com/watch?v=gEbbso4XG00. [Accessed: 18-Apr-2021].