PIAZZA

a RESTful SAAS messaging system

04 April 2021

CLOUD COMPUTING CONCEPTS

SALIK TARIQ
Student ID: 12516369

ACADEMIC DECLARATION

"I have read and understood the sections of 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 report to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software."

External Resources:

Following resources were consulted while developing the Piazza application:

Learning Django – Video series by Caleb Smith on Linkedin Learning

Django Rest Framework by Code Environment on YouTube.com

Cloud Computing Concepts – Lab4 - Birkbeck College

Stackoverflow resources:

https://stackoverflow.com/questions/52575523/how-to-capture-the-model-doesnotexist-exception-in-django-rest-framework

https://stackoverflow.com/questions/66825340/retrieving-and-posting-data-from-to-a-related-table-in-django-apiview/66828272#66828272 *(I am original poster of this question)*

https://stackoverflow.com/questions/66808629/setting-djando-field-default-value-based-on-another-field/66809204#66809204 *(I am original poster of this question)*

https://stackoverflow.com/questions/66792442/django-rest-framework-oauth2-apiview-error/66793234#66793234 *(I am original poster of this question)*

Application License

Environment Version Requirements

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
pycparser==2.20
pytz==2021.1
requests==2.25.1
six==1.15.0
sqlparse==0.4.1
urllib3==1.26.4

Application Structure

Piazza ("the application") is deployed in python environment using Django framework. The application costs of following sub apps:

- Api: This is the main Django project, the authorization server is defined in the main project
- Users: This is the application server that manages usernames and passwords and also sends authorization requests for oAuth 2.0 authentication.
- Messaging: This is the resource server that hosts and serves piazza data. Business logic is also defined in this sub-application.
- Testing application: This is a python file outside of Django project used to test the functionality of Piazza application.

For the purpose of this coursework, the Piazza application is deployed on virtual machine provided by college.

- The IP address of virtual machine is **10.61.64.150**.
- Piazza project called 'api' is saved under /home/student/coursework/src
- Users application inside Piazza project is saved at /home/student/coursework/src/users
- Piazza resource server application called 'messaging' is saved at ~/coursework/src/messaging
- Testing application is saved under /home/student/coursework_testing_application/ which is outside of the Piazza application.
- The account details for admin panel are **username**: admin **password**: secret1234

Following is *simplified* tree structure of Piazza application.

```
├── api
|   ├── __init__.py
|   ├── __pycache__
|   ├── asgi.py
|   ├── settings.py
|   ├── urls.py
|   └── wsgi.py
├── db.sqlite3
├── manage.py
├── messaging
|   ├── __init__.py
|   ├── admin.py
|   ├── apps.py
|   ├── migrations
|   |   ├── __init__.py
|   |   └── __pycache__
|   ├── models.py
|   ├── serializers.py
|   ├── tests.py
|   └── views.py
└── users
    ├── __init__.py
    ├── __pycache__
    ├── admin.py
    ├── apps.py
    ├── migrations
    |   ├── __init__.py
    |   └── __pycache__
    ├── models.py
    ├── serializers.py
    ├── tests.py
    ├── urls.py
    └── views.py
```

OAuth 2.0 Implementation

OAuth2 and rest framework applications are included in main project's settings.py file after installing along with OAuth2 middleware token entry into MIDDLEWARE section of settings.py. Rest framework, default permission classes along with authentication backends are also defined in settings.py

The base URL for OAuth2 provision is 10.61.64.150:8000/o/ which is defined in **api**'s url.py file.

Finally, the application server is implemented in the users app, that holds Client ID and Client secret to register users app on the OAuth2 authentication server.

The base URLs for application server's authentication requests is 10.61.64.150:8000/authentication/

Messaging app – Database design

The data structure of the application is defined in the models.py file inside messaging folder. The classes represent database tables, and fields correspond to table columns.

Messages class (table) has many to many relationship with Topics class(table), whereas Messages class has one to many relationship with Feedback class.

The database structure is summarized in the diagram below:

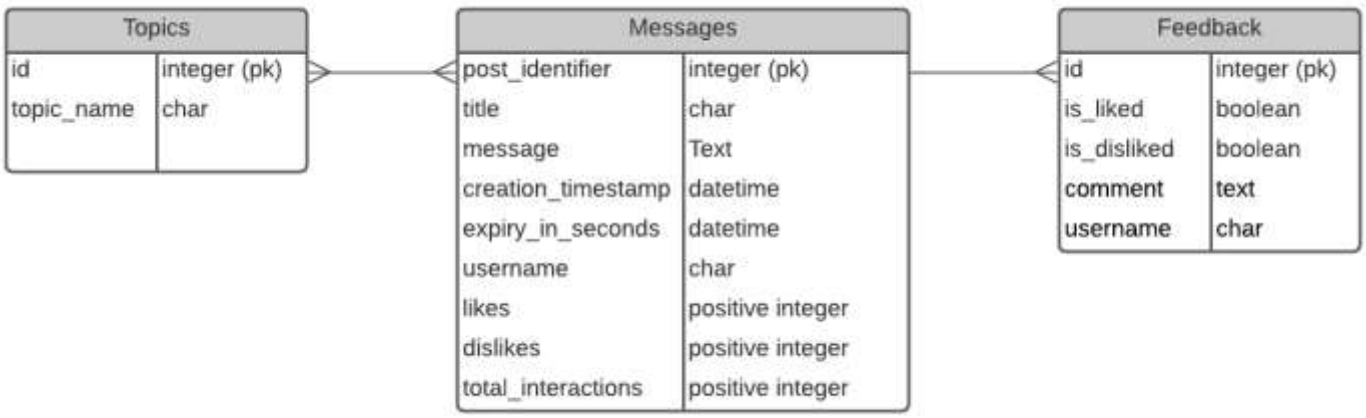

Figure 1: Messaging app models ERM diagram

Kindly note, Topics table has many: many relationship with Messages as a message can have more than one topics, and each topic is related to many messages.

REST API Endpoints

The base URL for restful API endpoints is http://10.61.64.150:8000/v1/

There are following REST API endpoints in service in Piazza application:

http://10.61.64.150:8000/v1/message/

http://10.61.64.150:8000/v1/feedback/

http://10.61.64.150:8000/v1/topic/

http://10.61.64.150:8000/v1/sortedmessages/

http://10.61.64.150:8000/v1/messagebytopic/

http://10.61.64.150:8000/v1/messagebytopicsorted/

http://10.61.64.150:8000/v1/expiredmessagebytopic/

1) http://10.61.64.150:8000/v1/message/

This API endpoint is served by messaging.views.MessagesViewset. This would fetch ALL records stored in messages table. Kindly note this endpoint would fetch all messages from ALL topics. If post_identifier is added to the end of endpoint, it would fetch that particular message irrespective of the topic it belongs to. This endpoint is also used to POST messages to Messages table.

In the following example a call was made to fetch message with post_identifier = 116 that resulted in that particular message record.

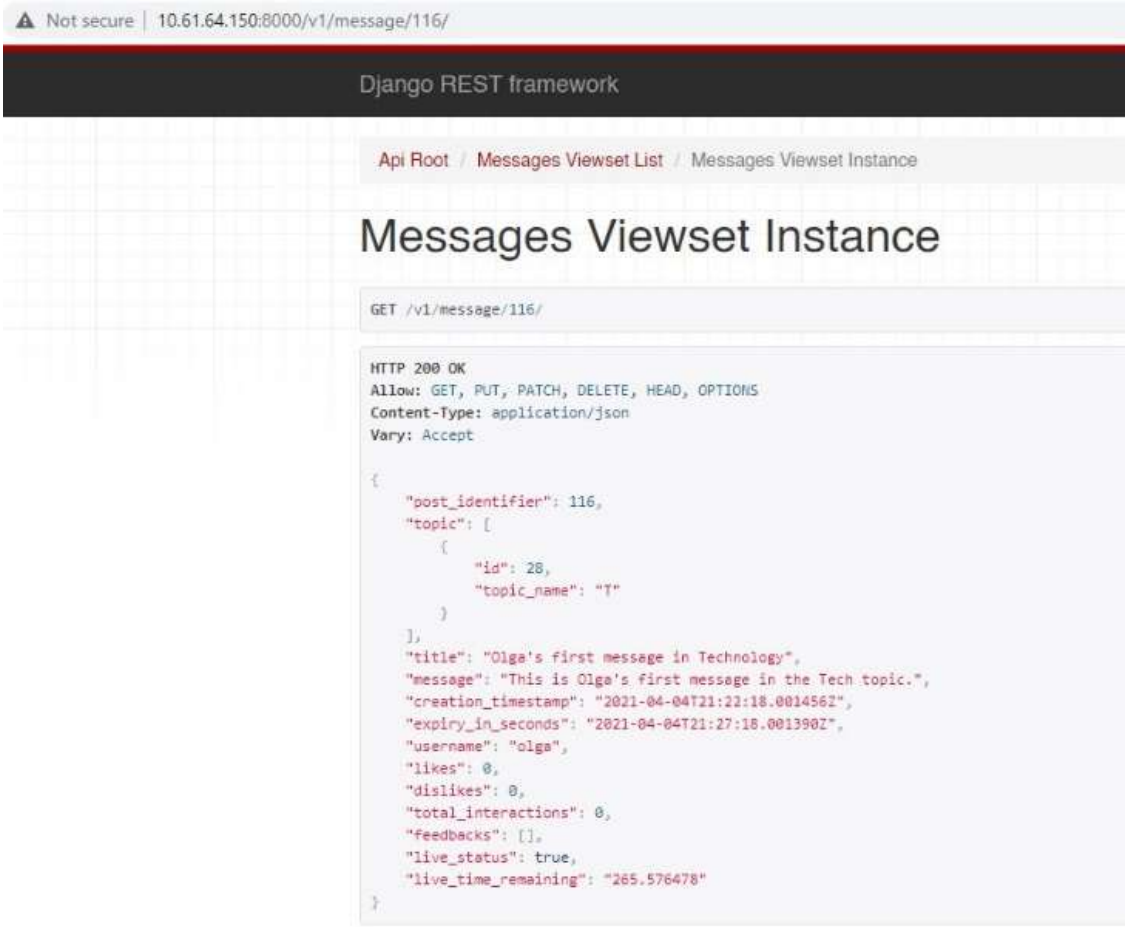

Figure 2: Call to fetch message with post_identifier 116

2) http://10.61.64.150:8000/v1/messagebytopic/

This API endpoint is served by messaging.views.SearchMessageByTopic. It would fetch all the messages that belong to a particular topic by adding that topic initial at the end of the api endpoint.

For example, if we want to fetch all messages belonging to Technology topic, we will query the following endpoint url http://10.61.64.150:8000/v1/messagebytopic/T/ as shown in the example below.

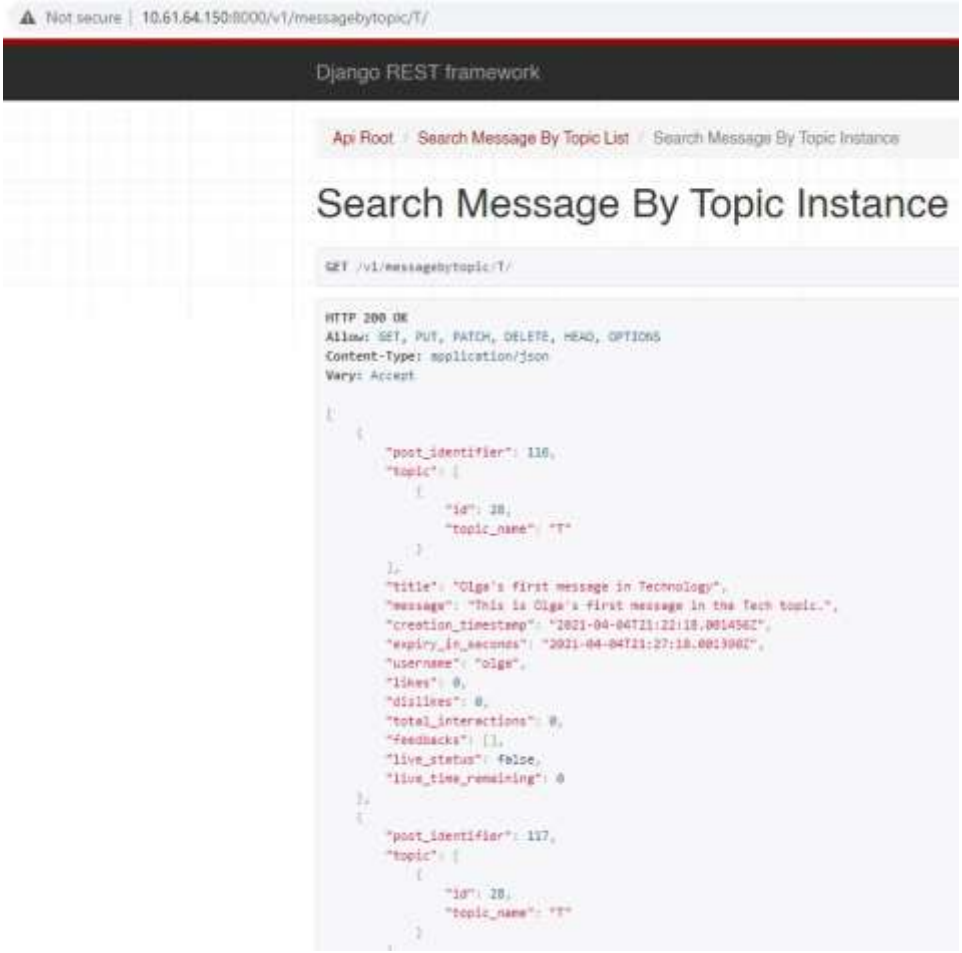

Figure 3 Call resulting in all messages belonging to Technology topic

3) http://10.61.64.150:8000/v1/messagebytopicsorted/

This endpoint works in a similar way as /messagebytopic/ endpoint except this endpoint would produce sorted result in descending order of level of activeness. Hence, the post with most likes, dislikes and comments will come on top. In order to retrieve the top post from response, the record with index [0] is retrieved. To retrieve top five posts, records with indices 0-4 are to be extracted from response.

This behavior can be observed in test case 20 as follows:

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



----------TOP POST-------------------
{'post_identifier': 118, 'topic': [{'id': 28, 'topic_name': 'T'}], 'title': "Mary's first
message in Technology", 'message': "This is Mary's first message in the Tech topic.",
'creation_timestamp': '2021-04-04T21:22:18.232757Z', 'expiry_in_seconds':
'2021-04-04T21:27:18.232701Z', 'username': 'mary', 'likes': 2, 'dislikes': 1,
'total_interactions': 7, 'feedbacks': [{'id': 200, 'is_liked': True, 'is_disliked': False,
'comment': '', 'username': 'nick', 'message': 118}, {'id': 201, 'is_liked': True,
'is_disliked': False, 'comment': '', 'username': 'olga', 'message': 118}, {'id': 203,
'is_liked': False, 'is_disliked': True, 'comment': '', 'username': 'nester', 'message':
118}, {'id': 204, 'is_liked': False, 'is_disliked': False, 'comment': "Nick's first
comment on Mary's post", 'username': 'nick', 'message': 118}, {'id': 205, 'is_liked':
False, 'is_disliked': False, 'comment': "Olga's first comment on Mary's post", 'username':
'olga', 'message': 118}, {'id': 206, 'is_liked': False, 'is_disliked': False, 'comment':
"Nick's second comment on Mary's post", 'username': 'nick', 'message': 118}, {'id': 207,
'is_liked': False, 'is_disliked': False, 'comment': "Olga's second comment on Mary's
post", 'username': 'olga', 'message': 118}], 'live_status': True, 'live_time_remaining':
'294.869896'}
------------------------------------------------
```

Figure 4 Querying the most active post

4) http://10.61.64.150:8000/v1/feedback/

This API endpoint is served by messaging.views.FeedbackViewset. It is used to POST feedback on a message by providing its post_identifier.

In the following example this endpoint was used to post feedbacks on a message in Piazza application.

```
TC 8. Nick and Olga "likes" Mary's post in the Tech topic


{'id': 200, 'is_liked': True, 'is_disliked': False, 'comment': '', 'username': 'nick',
'message': 118}
{'id': 201, 'is_liked': True, 'is_disliked': False, 'comment': '', 'username': 'olga',
'message': 118}
```

Figure 5: Feedbacks successfully posted

5) http://10.61.64.150:8000/v1/expiredmessagebytopic/

This endpoint is served by messaging.views.SearchExpiredMessageByTopic viewset. As name suggests, it provides messages per topic that are expired. Hence a query http://10.61.64.150:8000/v1/expiredmessagebytopic/T/ will retrieve all expired messages in Technology topic.

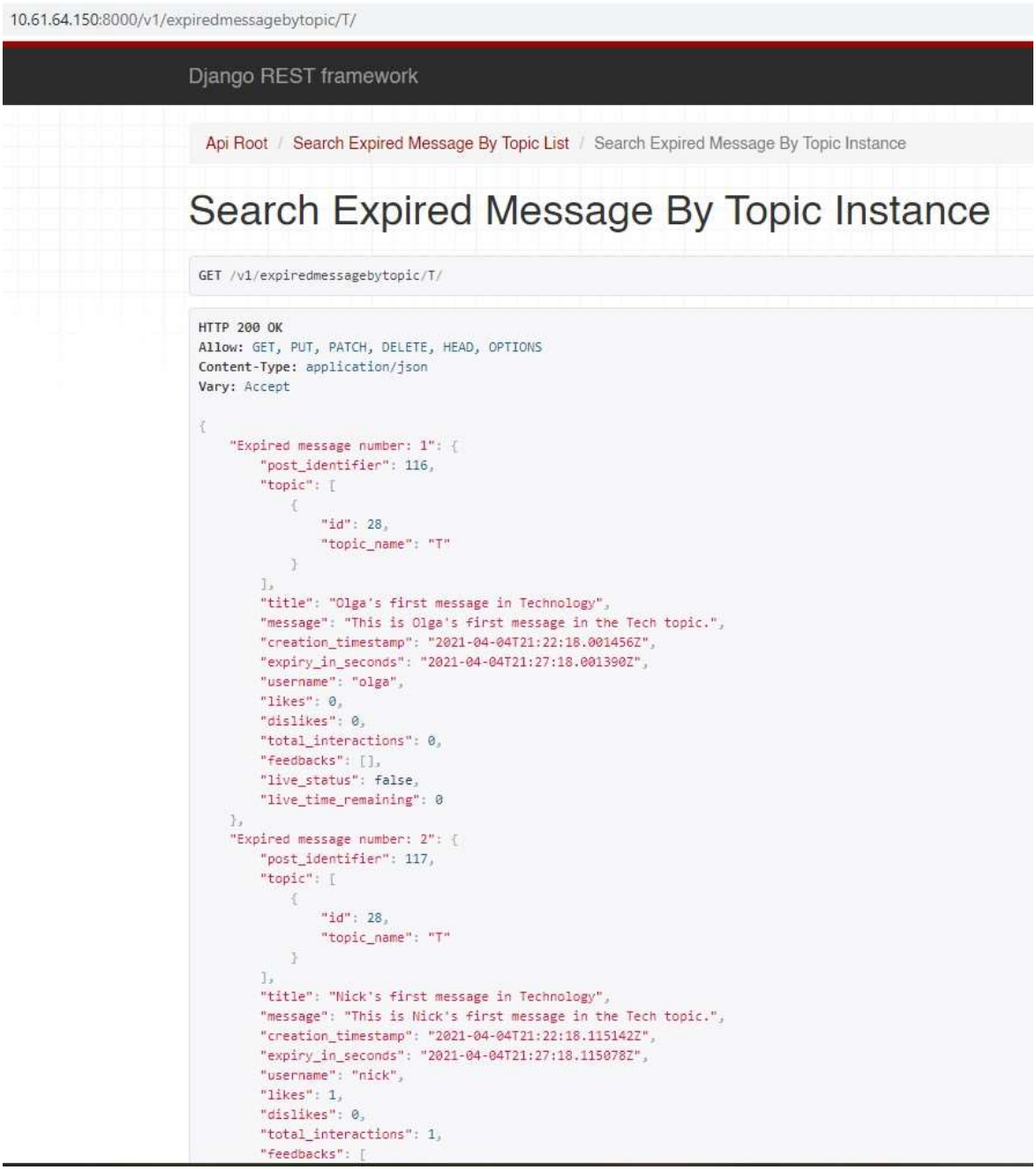

Figure 6: Retrieved list of expired messages in Technology topic

In the following example, a call was made to retrieve all expired messages in Sports topic, but as there was no expired topic, so call returned without a result.

```
TC 19. Nick browses all the expired messages in Sport topic. These should be empty

******No records found******
```

Figure 7: Empty result as no expired message is present in Sports topic

6) http://10.61.64.150:8000/v1/topic/

This endpoint is served by messaging.views.TopicsViewset. It is used to POST topics. Kindly note that all topics must be unique. This endpoint may also be used to retrieve all messages belonging to a particular topic by adding topic name initial at the end of endpoint. Kindly see the following example:

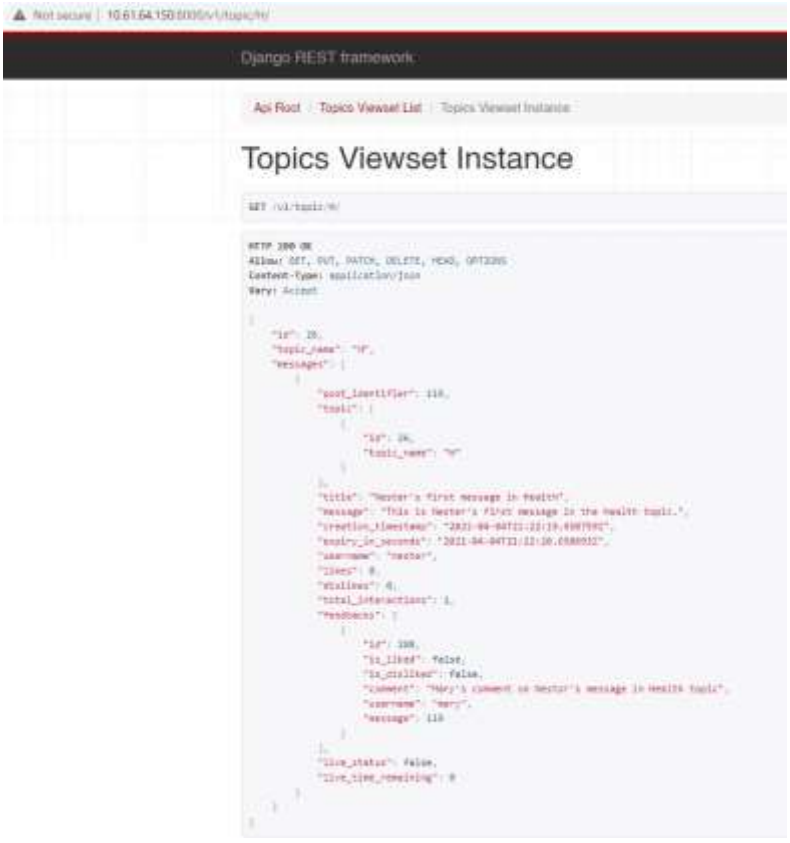

Figure 8: Retrieving all messages in topic health

7) http://10.61.64.150:8000/v1/sortedmessages/

This endpoint sorts messages based on their 'activeness' from all topics. It is served by

MessagesSortedByInteractionViewSet in messaging.views.

NO ACCESS WITHOUT VALID TOKEN

Kindly note, none of these endpoints will fetch or post any data unless a valid token is sent in API call. Any call made without providing valid token will result in following error(s):

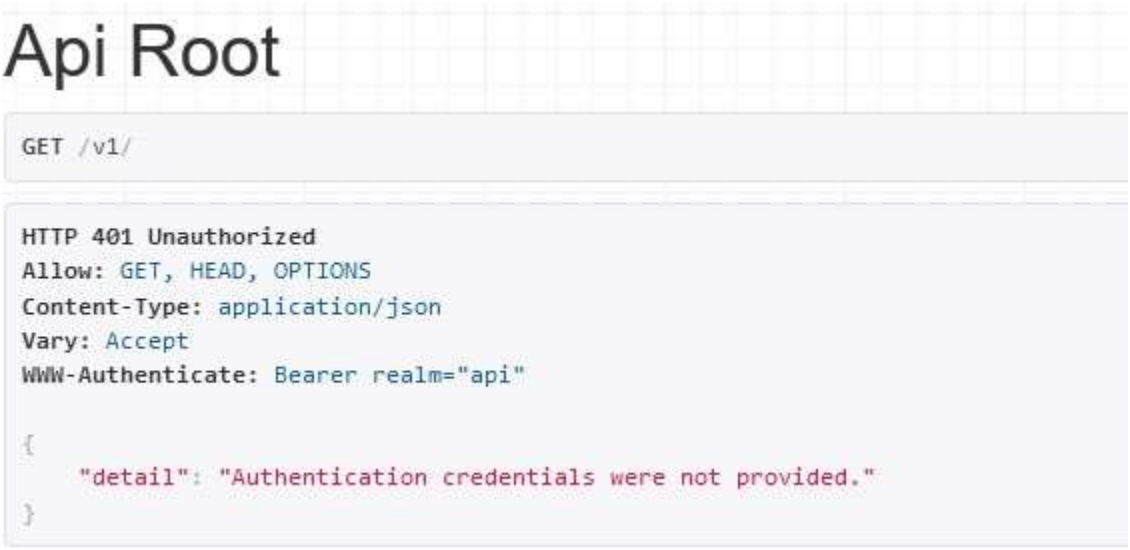

Figure 9 Admin site API call without valid token

This behavior was also observed in test case 3 as follows:

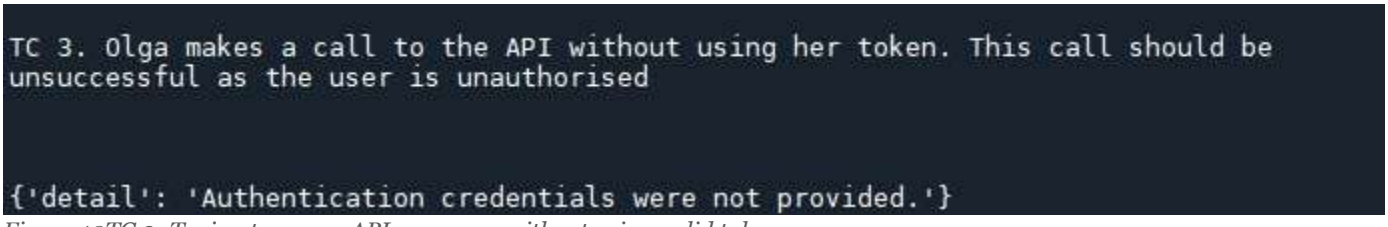

Figure 10TC 3: Trying to access API resources without using valid token

NO FEEDBACK ON OWN POST

Piazza system does not allow a user to post feedback on their own posts. Please see the example below:

```
TC 11. Mary likes her post in the Tech topic. This call should be unsuccessful, as in
Piazza a post owner cannot like their own message.


{'Error: ': 'Can not give feedback on own post'}
```

Figure 11: Piazza does not allow to post feedback on own post

NO FEEDBACK AFTER POST EXPIRY

Piazza system does not allow a user to post feedback on a post after it is expired.

```
TC 17. Mary dislikes Nester's message in the Health topic after the end of post expiration
time. This should fail.


{'Error: ': 'Post is expired now, can not add comments.'}
```

Figure 12: Can not post feedback on an expired post