Lab 4 - report

Assignment 4

Group 19 - se24

Table of contents

Project

Name: openverse
URL: https://www.github.com/WordPress/openverse

Openverse is a search engine for GPL-compatible images, audio, and more. Openverse is live at openverse.org. It is built in Python and JavaScript along with some utilities. The issue we worked on concerned the API's admin panel and thus involved mostly Python/Django.

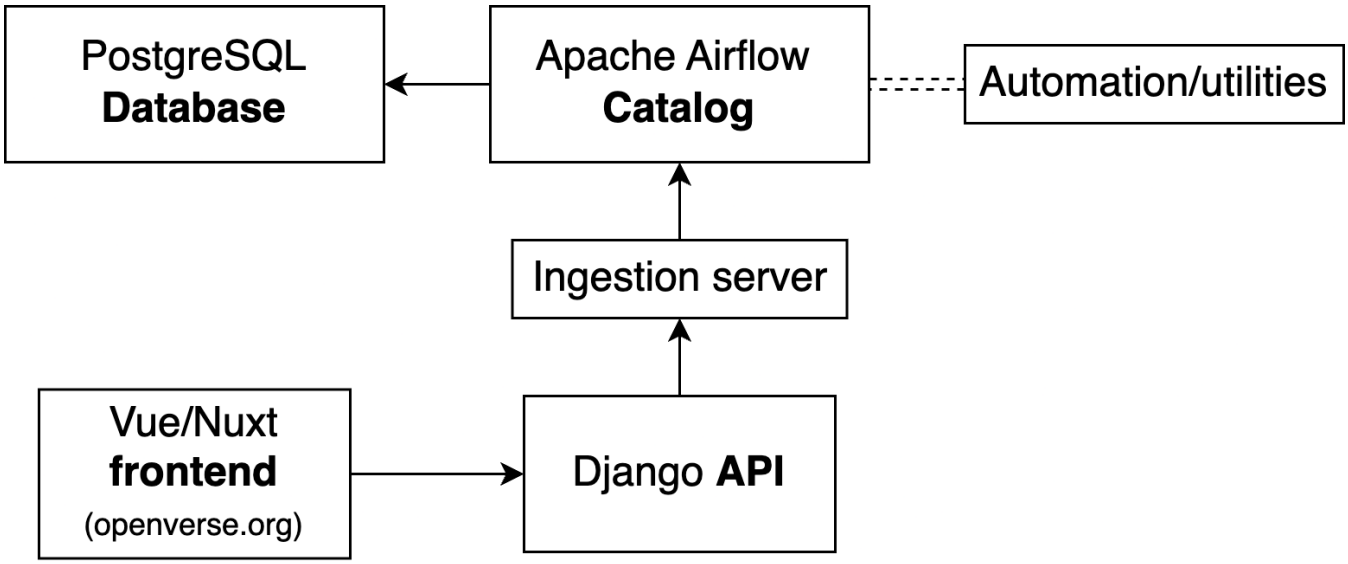
How the projects fit in an ecosystem of open-source and closed-source software

Depending on how broad of a look one takes at the project and its goals it fits into different categories. Openverse is inherently a search engine, which specializes in free-to-use images. As such if we were to compare it to search engines as a whole, like Google,

Bing, Duckduckgo, and Yahoo then it would not be considered an emulation of those, since it has its niche. There are, however, 2 other projects called Unsplash and Pexels, which were released 4 years before Openverse that fill the same problem and are proprietary.

The main “selling point” of Openverse is not that it is supposed just to be better than those previously mentioned projects, but that WordPress creates it. As such it should feel better and interact more easily for people whose websites are built using WordPress, which is around 40% of all websites online. As such it is mostly targeting its niche and is not just a poor imitation of previous projects.

System architecture



The architecture of the openverse search engine consists of several, relatively discrete, parts, all well-documented and logically separated. This makes the system architecture easy to understand. These different parts were previously developed separately, but are all hosted in the main monorepo since a few years. Here is the short overview provided in the repository's Readme file, with hyperlinks to the relevant repository folders:

- Catalog | The Apache Airflow-powered system for downloading and storing Openverse's metadata
- Ingestion server | The mechanism for refreshing the data from the catalogue to the API
- API | The Django REST API for querying the database, used by the frontend
- Frontend | The public search engine at openverse.org, built with Vue and Nuxt
- Automations | Scripts used for various workflows around Openverse repositories and processes
- Utilities | Scripts or utilities which are useful across multiple projects or don't necessarily fit into a specific project.

Catalog

Openverse uses Apache Airflow along with custom API puller scripts to download, identify, and organize the over 1.4 billion Creative Commons works that the search engine

can access. It uses a combination of web crawling and pulling APIs on a schedule to pull information from large providers, such as Wikimedia Commons and museums. Apache Airflow is also used to provide data normalization, refresh, maintenance, and other type of utilities.

PostgreSQL

The Catalog is also backed by a PostgreSLQ database, hosted on AWS. The postgres instance is the primary data warehouse for all ingested media data.

Django API

The API is built in Python using Django. It is used both by the frontend and is publically accessible at https://api.openverse.engineering/v1/, for use in 3rd party applications. It uses a relatively standard Django setup, with API endpoints for getting audio, images, and other media as well as for authentication towards the API for rate-limiting purposes. If self-hosting the API (as we have during development), one can access the Django admin panel to manually browse, add, and delete media, configure content providers, and throttled applications.

Frontend

The frontend for the live version of the web application at https://openverse.org is also hosted in the same repository. It uses Vue/Nuxt as the JavaScript framework, and node.js for development.

Onboarding experience

After some consideration, we ultimately chose to start fresh with a new project for this lab as compared to assignment 3. Mostly due to the complexity of the code base we chose for the previous lab as well as the lack of issues and active maintainers of it.

The experience between the two projects was night and day, differing in almost every way imaginable. The first major difference was the language, with the former project being written in Java whilst the new is primarily Python. The second was the amount of resources available. The project we chose for assignment 3 was primarily written by a single person over 7 years ago, as such the onboarding experience and the contribution guidelines were hazy at best. With the new project, we could easily get in contact with maintainers who actively wanted our contributions and had detailed contribution guidelines and quickstart instructions. While openverse is a large and complex project, the excellent documentation, guides, and general attitude toward new contributors made the experience much more pleasant.

We first tried to choose another project called Mixxx but did not find a way to build it on Mac which a large part of the group was using. There were also some issues when building openverse from the start, especially related to Docker. However using their onboarding documentation in conjunction with googling was enough to get us up to speed in the end.

Effort spent

The biggest effort in terms of hours was spent on group meetings. However, that time was spent on different tasks, such as finding and discussing the choice of project, collaborating on planning, setting up a local development environment, etc.

Time in hours	Sam K	Eric J	Tore N	Sam S	Björn T
Group meetings	8	8	6	8	8
Discussions	0	0.5	0,5	1	0
Finding project	3	3	3,5	3	3
Reading documentation	2	2	1	1,5	3
Config and setup	2	1	2	1,5	1
Analyzing code/output	0.5	0.25	0,5	0,5	1
Writing documentation	0	0	0	0	0
Writing code	2	2	0,5	2	2
Running code	1	1	0	0	1
Writing report	2	2.25	3	2	2
Sum	20.5	20	17	19,5	21

It took some time to configure the correct Python version locally, to install dependencies and to get the containerized dev environment set up. Installing and downloading also took time, but the just files made configuration easy.

Overview of issue(s) and work done.

Title: Add access token and throttled application models to Django admin
URL: https://github.com/WordPress/openverse/issues/3711

The issue concerns adding two Django models to the admin panel with the correct fields editable, one custom model and one from a 3rd party package.

Requirements

The two requirements are defined in the original issue, and below. Note that the “allow editing of certain fields” was not something we explicitly included in our PR, as we did not have enough information insight into the system to conclude which fields made sense to have as editable. This is something that we mention in our draft PR (which as of the time of writing has not been responded to by maintainers).

We trace both our tests to these two requirements, by asserting through requests to the local API that the new tables have been added to the admin panel.

Requirement 1: add ThrottledApplication table to Django Admin UI. api.models.oauth.ThrottledApplication needs view in admin.init. enable editing of certain fields so that access can be revoked by maintainers using the Admin UI as well. Otherwise read-only.

Test:
The test for this function consisted of mocking the website and then asserting that the ThrottledApplication was present in the admins site registry.

The hard part here was not writing the test, but to understand how to access the AccessToken and the ThrottledApplication instance as well as how to access the admin panel registry.

Output before adding ThrottledApplication table to Django Admin UI:

```
[+] Running 9/9
 ✔ Container assignment-4-upstream_db-1         Running
0.0s
 ✔ Container assignment-4-cache-1               Running
0.0s
 ✔ Container assignment-4-db-1                  Running
0.0s
 ✔ Container assignment-4-es-1                  Running
0.0s
 ✔ Container assignment-4-web-1                 Running
0.0s
 ✔ Container assignment-4-indexer_worker-1      Running
0.0s
 ✔ Container assignment-4-proxy-1               Started
0.0s
 ✔ Container assignment-4-nginx-1               Running
0.0s
 ✔ Container assignment-4-ingestion_server-1  Runni...
0.0s

================================================================================
==
                                Service Ports
================================================================================
==
================================================================================
==
just ../ingestion_server/wait # API profile includes ingestion server
just wait # API waits for ES in entrypoint
env DC_USER="opener" just ../exec web pytest
just dc exec -u opener  web pytest
env COMPOSE_PROFILES="api,ingestion_server,frontend,catalog" docker-compose -f
docker-compose.yml exec -
u opener web pytest
Test session starts (platform: linux, Python 3.11.8, pytest 7.4.4, pytest-
```

```
sugar 0.9.7)
django: version: 4.2.9, settings: conf.settings (from env)
rootdir: /api
configfile: pytest.ini
plugins: django-4.8.0, time-machine-2.13.0, Faker-22.6.0, anyio-4.2.0, sugar-
0.9.7, raises-0.11
collected 581 items

 test/integration/test_audio_integration.py ✓✓
0%
 test/integration/test_auth.py ✓
1% |
 test/integration/test_auth.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
4% ▌
 test/integration/test_dead_link_filter.py ✓✓✓✓✓✓
5% ▌
 test/integration/test_image_integration.py ✓✓✓✓✓
6% ▌
 test/integration/test_media_integration.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓ 14% █
                                                ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
17% █
 test/unit/controllers/test_search_controller.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓ 23% ██

✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓ 37% ███

✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
46% ████▋
 test/unit/controllers/test_search_controller_search_query.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
48% ████▊
 test/unit/controllers/elasticsearch/test_related.py ✓
49% ████▊
 test/unit/management/commands/test_generatewaveforms.py ✓✓✓✓✓✓
50% ████▊
 test/unit/models/test_accesstoken.py ✓
50% █████
 test/unit/models/test_audio.py ✓✓✓✓
50% █████
 test/unit/models/test_media_report.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
56% █████▌

━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
```

```
ThrottledApplicationAdminTest.test_throttled_application_in_admin
―――――――――――――――――――――――――――――――――――――――

self = <test.unit.models.test_throttle.ThrottledApplicationAdminTest
testMethod=test_throttled_application_in_admin>

        def test_throttled_application_in_admin(self):
        #test that ThrottledApplication is in Django admin.
>       self.assertIn(ThrottledApplication, admin.site._registry)
E       AssertionError: <class 'api.models.oauth.ThrottledApplication'> not
found in {<class 'api.models.image.Image'>: <ImageAdmin: model=Image
 site=OpenverseAdmin(name='admin')>, <class 'api.models.audio.Audio'>:
<AudioAdmin: model=Audio site=OpenverseAdmin(name='admin')>, <class 'api.
models.image.ImageReport'>: <ImageReportAdmin: model=ImageReport
site=OpenverseAdmin(name='admin')>, <class 'api.models.audio.AudioReport'>:
<Au
dioReportAdmin: model=AudioReport site=OpenverseAdmin(name='admin')>, <class
'api.models.audio.MatureAudio'>: <MediaSubreportAdmin: model=Mature
Audio site=OpenverseAdmin(name='admin')>, <class
'api.models.image.MatureImage'>: <MediaSubreportAdmin: model=MatureImage
site=OpenverseAdmin(na
me='admin')>, <class 'api.models.audio.DeletedAudio'>: <MediaSubreportAdmin:
model=DeletedAudio site=OpenverseAdmin(name='admin')>, <class 'api.
models.image.DeletedImage'>: <MediaSubreportAdmin: model=DeletedImage
site=OpenverseAdmin(name='admin')>, <class 'api.models.models.ContentProvi
der'>: <ProviderAdmin: model=ContentProvider
site=OpenverseAdmin(name='admin')>, <class
'oauth2_provider.models.AccessToken'>: <AccessTokenAdmin
: model=AccessToken site=OpenverseAdmin(name='admin')>}

test/unit/models/test_throttle.py:11: AssertionError

 test/unit/models/test_throttle.py ×
56% 
 test/unit/serializers/test_audio_serializers.py ✓✓✓
56% 
 test/unit/serializers/test_media_serializers.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
80% 
 test/unit/templates/test_admin_base_html.py ✓✓✓
62% 
 test/unit/utils/test_image_proxy.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓              98% 
```

```
 test/unit/utils/test_search_context.py ✓✓✓✓✓✓✓✓✓✓
65% ██████
 test/unit/utils/test_throttle.py ✓✓✓✓✓✓✓✓✓✓✓✓✓
68% ██████
 test/unit/views/test_health_views.py ✓✓✓✓✓✓
100% ██████████
 test/unit/views/test_image_views.py ✓✓✓✓✓✓
69% ██████
 test/unit/views/test_media_views.py ✓✓✓✓✓✓✓✓✓✓✓✓
71% ███████
 test/test_examples.py ✓✓✓✓✓✓✓✓✓
73% ███████
 test/integration/test_auth.py ✓
73% ███████
 test/integration/test_deprecations.py ✓✓✓✓✓✓
74% ███████
 test/unit/configuration/test_link_validation_cache.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
77% ███████
 test/unit/models/test_media.py ✓✓✓✓✓✓✓✓✓✓
78% ███████
 test/unit/utils/test_aiohttp.py ✓✓✓
81% ███████
 test/unit/utils/test_attribution.py ✓✓✓✓✓✓✓
82% ████████
 test/unit/utils/test_check_dead_links.py ✓✓✓✓✓✓
83% ████████
 test/unit/utils/test_drf_renderer.py ✓✓✓✓✓✓✓✓✓✓✓
85% ████████
 test/unit/utils/test_help_text.py ✓✓✓✓
85% ████████
 test/unit/utils/test_tallies.py ✓✓✓✓✓
99% ██████████
 test/unit/utils/test_watermark.py ✓✓
99% ██████████
 test/unit/utils/test_waveform.py ✓
99% ██████████
=============================================================== short test summary
info ================================================================FAILED
test/unit/models/test_throttle.py::ThrottledApplicationAdminTest::test_throttl
ed_application_in_admin - AssertionError: <class
'api.models.oauth.ThrottledApplication'> not found in {<class
'api.models.image.Image'>: <ImageAdmin: model=Image si...

Results (87.43s (0:01:27)):
```

```
        580 passed
        1 failed
        - test/unit/models/test_throttle.py:9
ThrottledApplicationAdminTest.test_throttled_application_in_admin
error: Recipe `dc` failed on line 160 with exit code 1
error: Recipe `exec` failed on line 225 with exit code 1
error: Recipe `test` failed on line 115 with exit code 1
```

Requirement 2: add AccessToken table to Django Admin UI
`oauth_provider.models.AccessToken` needs view in admin.init. enable editing of certain fields so that access can be revoked by maintainers using the Admin UI as well. Otherwise read-only.

Test:
The test for this function consisted of mocking the website and then asserting that the ThrottledApplication was present in the admins site registry.

The hard part here was not writing the test, but to understand how to access the AccessToken and the ThrottledApplication instance as well as how to access the admin panel registry.

Output before adding AccessToken:

```
[+] Running 9/9
 ✔ Container assignment-4-cache-1             Running
0.0s
 ✔ Container assignment-4-db-1                Running
0.0s
 ✔ Container assignment-4-es-1                Running
0.0s
 ✔ Container assignment-4-upstream_db-1       Running
0.0s
 ✔ Container assignment-4-web-1               Running
0.0s
 ✔ Container assignment-4-nginx-1             Running
0.0s
 ✔ Container assignment-4-indexer_worker-1    Running
0.0s
 ✔ Container assignment-4-proxy-1             Started
0.0s
 ✔ Container assignment-4-ingestion_server-1  Running
0.0s


================================================================================
==
                                Service Ports
```

```
==============================================================================
==
==============================================================================
==
just ../ingestion_server/wait # API profile includes ingestion server
just wait # API waits for ES in entrypoint
env DC_USER="opener" just ../exec web pytest
just dc exec -u opener  web pytest
env COMPOSE_PROFILES="api,ingestion_server,frontend,catalog" docker-compose -f
docker-compose.yml exec -u opener web pytest
Test session starts (platform: linux, Python 3.11.8, pytest 7.4.4, pytest-
sugar 0.9.7)
django: version: 4.2.9, settings: conf.settings (from env)
rootdir: /api
configfile: pytest.ini
plugins: django-4.8.0, time-machine-2.13.0, Faker-22.6.0, anyio-4.2.0, sugar-
0.9.7, raises-0.11
collected 581 items

 test/integration/test_audio_integration.py ✓✓
0%
 test/integration/test_auth.py ✓✓✓
1% |
 test/integration/test_auth.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
4% ▌           test/integration/test_dead_link_filter.py ✓✓✓✓✓✓
5% ▌           test/integration/test_image_integration.py ✓✓✓✓✓
6% ▌           test/integration/test_media_integration.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
17% █          test/unit/controllers/test_search_controller.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓ 42% ████
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
46% █████       test/unit/controllers/test_search_controller_search_query.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
48% █████       test/unit/controllers/elasticsearch/test_related.py ✓
49% █████       test/unit/management/commands/test_generatewaveforms.py ✓✓✓✓✓✓
50% █████
 test/integration/test_dead_link_filter.py ✓✓✓✓✓✓
5% ▌
 test/integration/test_image_integration.py ✓✓✓✓✓
6% ▌
 test/integration/test_media_integration.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
17% █
```

```
 test/unit/controllers/test_search_controller.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓ 42% ████
                                                 ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
46% █████
 test/unit/controllers/test_search_controller_search_query.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
48% █████
 test/unit/controllers/elasticsearch/test_related.py ✓
49% █████
 test/unit/management/commands/test_generatewaveforms.py ✓✓✓✓✓✓
50% █████
──────────────────────────────────────────────────────────────────────────────
─── AccesstokenAdminTest.test_accesstoken_in_admin
──────────────────────────────────────────────────────────────────────────────
───
self = <test.unit.models.test_accesstoken.AccesstokenAdminTest
testMethod=test_accesstoken_in_admin>

    def test_accesstoken_in_admin(self):
        #test that accestoken is in Django admin.
>       self.assertIn(AccessToken, admin.site._registry)
E       AssertionError: <class 'oauth2_provider.models.AccessToken'> not found
in {<class 'api.models.image.Image'>: <ImageAdmin: model=Image
site=OpenverseAdmin(name='admin')>, <class 'api.models.audio.Audio'>:
<AudioAdmin: model=Audio site=OpenverseAdmin(name='admin')>, <class
'api.models.image.ImageReport'>: <ImageReportAdmin: model=ImageReport
site=OpenverseAdmin(name='admin')>, <class 'api.models.audio.AudioReport'>:
<AudioReportAdmin: model=AudioReport site=OpenverseAdmin(name='admin')>,
<class 'api.models.audio.MatureAudio'>: <MediaSubreportAdmin:
model=MatureAudio site=OpenverseAdmin(name='admin')>, <class
'api.models.image.MatureImage'>: <MediaSubreportAdmin: model=MatureImage
site=OpenverseAdmin(name='admin')>, <class 'api.models.audio.DeletedAudio'>:
<MediaSubreportAdmin: model=DeletedAudio site=OpenverseAdmin(name='admin')>,
<class 'api.models.image.DeletedImage'>: <MediaSubreportAdmin:
model=DeletedImage site=OpenverseAdmin(name='admin')>, <class
'api.models.models.ContentProvider'>: <ProviderAdmin: model=ContentProvider
site=OpenverseAdmin(name='admin')>, <class
'api.models.oauth.ThrottledApplication'>: <ThrottledApplicationAdmin:
model=ThrottledApplication site=OpenverseAdmin(name='admin')>}

test/unit/models/test_accesstoken.py:11: AssertionError

 test/unit/models/test_accesstoken.py ×
50% █████      test/unit/models/test_audio.py ✓✓✓✓
```

```
50%        test/unit/models/test_media_report.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
56%        test/unit/models/test_throttle.py ✓
56%        test/unit/serializers/test_audio_serializers.py ✓✓✓
56%        test/unit/serializers/test_media_serializers.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
80%        test/unit/templates/test_admin_base_html.py ✓✓✓
62%        test/unit/utils/test_image_proxy.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓
98%        test/unit/utils/test_search_context.py ✓✓✓✓✓✓✓✓✓✓
65%        test/unit/utils/test_throttle.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓
68%        test/unit/views/test_health_views.py ✓✓✓✓✓✓
test/unit/views/test_image_views.py ✓✓✓✓✓✓
test/unit/views/test_media_views.py ✓✓✓✓✓✓✓✓✓✓✓✓✓
test/test_examples.py ✓✓✓✓✓✓✓✓✓
test/integration/test_auth.py ✓
test/integration/test_deprecations.py ✓✓✓✓✓✓
test/unit/configuration/test_link_validation_cache.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
test/unit/models/test_media.py ✓✓✓✓✓✓✓✓✓✓
test/unit/utils/test_aiohttp.py ✓✓✓
test/unit/utils/test_attribution.py ✓✓✓✓✓✓✓
test/unit/utils/test_check_dead_links.py ✓✓✓✓✓✓
test/unit/utils/test_drf_renderer.py ✓✓✓✓✓✓✓✓✓✓✓
test/unit/utils/test_help_text.py ✓✓✓✓
test/unit/utils/test_tallies.py ✓✓✓✓✓
test/unit/utils/test_watermark.py ✓✓
test/unit/utils/test_waveform.py ✓
 test/unit/models/test_audio.py ✓✓✓✓
50%
 test/unit/models/test_media_report.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
56%
 test/unit/models/test_throttle.py ✓
56%
 test/unit/serializers/test_audio_serializers.py ✓✓✓
56%
 test/unit/serializers/test_media_serializers.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
80%
 test/unit/templates/test_admin_base_html.py ✓✓✓
62%
 test/unit/utils/test_image_proxy.py
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓
✓✓
```

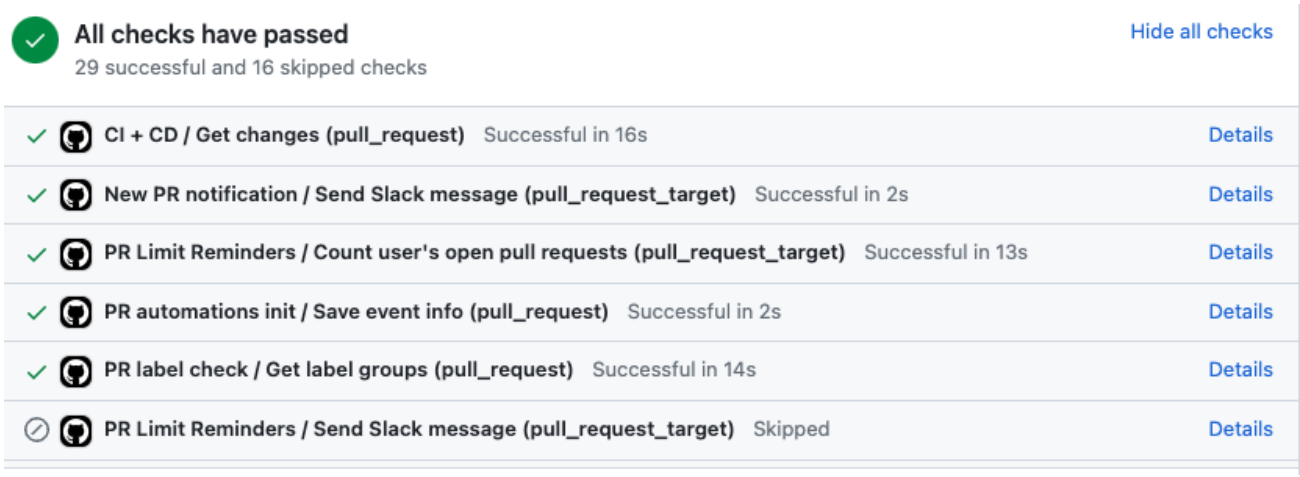
```
98% ████████
 test/unit/utils/test_search_context.py ✓✓✓✓✓✓✓✓✓✓
65% ██████
 test/unit/utils/test_throttle.py ✓✓✓✓✓✓✓✓✓✓✓✓✓
68% ███████
 test/unit/views/test_health_views.py ✓✓✓✓✓✓
 test/unit/views/test_image_views.py ✓✓✓✓✓✓
 test/unit/views/test_media_views.py ✓✓✓✓✓✓✓✓✓✓✓✓
 test/test_examples.py ✓✓✓✓✓✓✓✓✓
 test/integration/test_auth.py ✓
 test/integration/test_deprecations.py ✓✓✓✓✓✓
 test/unit/configuration/test_link_validation_cache.py ✓✓✓✓✓✓✓✓✓✓✓✓✓✓
 test/unit/models/test_media.py ✓✓✓✓✓✓✓✓✓✓
 test/unit/utils/test_aiohttp.py ✓✓✓
 test/unit/utils/test_attribution.py ✓✓✓✓✓✓✓
 test/unit/utils/test_check_dead_links.py ✓✓✓✓✓✓
 test/unit/utils/test_drf_renderer.py ✓✓✓✓✓✓✓✓✓✓✓
 test/unit/utils/test_help_text.py ✓✓✓✓
 test/unit/utils/test_tallies.py ✓✓✓✓✓
 test/unit/utils/test_watermark.py ✓✓
 test/unit/utils/test_waveform.py ✓

================================================================================
============== short test
FAILED
test/unit/models/test_accesstoken.py::AccesstokenAdminTest::test_accesstoken_i
n_admin - AssertionAdmin: model=Image site=OpenverseAdmin(name='admin')>,
<class 'api.models.audio.Audio'>: <...

Results (94.93s (0:01:34)):
     580 passed
       1 failed
         - test/unit/models/test_accesstoken.py:9
AccesstokenAdminTest.test_accesstoken_in_admin
error: Recipe `dc` failed on line 160 with exit code 1
error: Recipe `exec` failed on line 225 with exit code 1
error: Recipe `test` failed on line 115 with exit code 1
```

Test output after fixes:

```
Results (87.40s (0:01:27)):
         581 passed
```

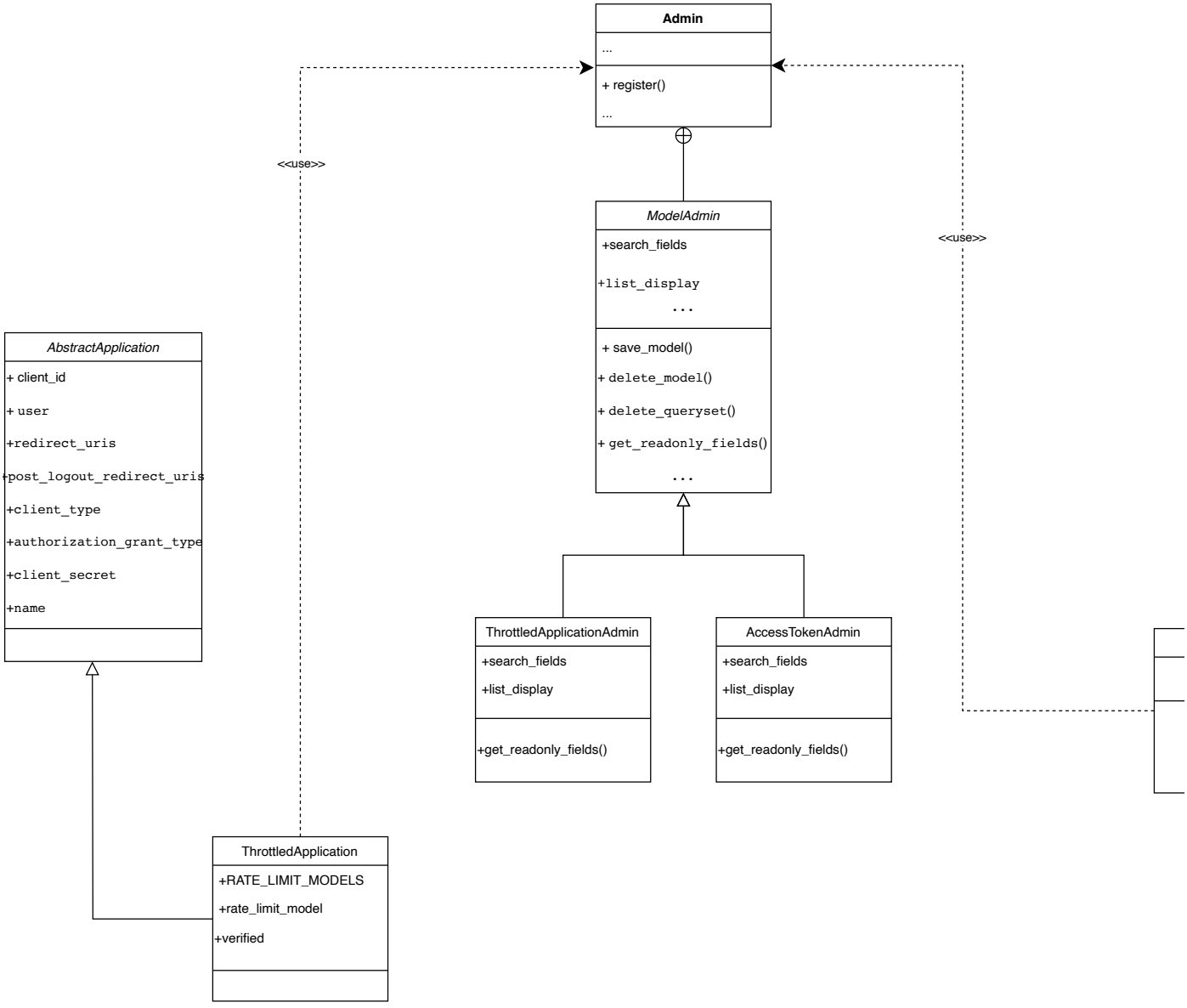
Pull Request and discussion

All tests pass:



Discussion of PR at https://github.com/WordPress/openverse/pull/3836

UML class diagram



You can argue critically about the benefits, drawbacks, and limitations of your work carried out, in the context of current software engineering practice, such as the SEMAT kernel (covering alphas other than Team/Way of Working).

The team was previously in the collaborating state for lab2 and lab3. However, the team is now performing, as we have since before:
consistently met our commitments
Adapted to changing contexts and different assignments
identify problems without outside help

But we needed to achieve these two as well:
Effective progress is being achieved with minimal avoidable backtracking and reworking.
Wasted work and the potential for wasted work are continuously identified and eliminated.

This was possible as we have for this task collaborated a lot more than previous, as it was also harder to just split it up. Furthermore, the group worked together to understand the problem, which minimized wasted work as 5 brains together understand the problem better and faster than 1 alone.

Furthermore, the task of contributing to open source, the team has to understand the requirements and also implement these. Right now, the requirements are in the coherent state, as they are
captured and shared with the team and the stakeholders.
This was done through github, and the requirement is the issue given to us by the stakeholder.
The origin of the requirements is clear.
This is also explained in the issue
The rationale behind the requirements is clear.
This is also explained in the issue
The requirements communicate the essential characteristics of the system to be delivered.
Yes, communicated in the issue and explains what to do but still leaves out a bit for own understanding
The most important usage scenarios for the system can be explained.
The team had to understand how parts of the system work before implementation
The priority of the requirements is clear.
As given by the assignment, we start with one issue and take on more if possible and time is there.
The impact of implementing the requirements is understood.
Yes, the impact will be another page/choice on the admin panel.
The team understands what has to be delivered and agrees to deliver it
Yes

However, the last part that needs to be identified is:
Conflicting requirements are identified and attended to.
This seems intuitive as there might not be any real conflicting requirements as its a pretty niche issue.

Furthermore, by looking at the state of the software system, we can see that it is clearly operational.

This as all the following criteria is achieved:
Usable:
The system can be operated by stakeholders who use it.
The functionality provided by the system has been tested.
The performance of the system is acceptable to the stakeholders.
Defect levels are acceptable to the stakeholders.
The system is fully documented.
Release content is known.
The added value provided by the system is clear.
Ready:
Installation and other user documentation are available.
The stakeholder representatives accept the system as fit-for-purpose.
The stakeholder representatives want to make the system operational.
Operational support is in place.
Operational:
The system has been made available to the stakeholders intended to use it.
At least one example of the system is fully operational.
The system is fully supported to the agreed service levels.

Lastly, when looking at way of working:
The team was previously in the state of “In Use” according to the Essence standard, and was on a good path to the next stage, which is the “In Place” state. The conclusion was drawn because the team was working well together as one unit, with 2-3 hour coworking sessions 2-3 times a week, as well as using the tools and standards set in place regarding commits and pull requests. The group members were also open about their issues and what they need help with, as well as directing each other and assessing work with pull requests. The team has also grown well and through many discussions on Discord, they have become closer.

The team was on a good track to the next stage, and the following points were reached:

- The team consistently meets its commitments.
- The team continuously adapts to the changing context.
- The team identifies and addresses problems without outside help.
- Wasted work and the potential for wasted work are continuously identified and eliminated, through pull request feedback and constructive feedback.
  However, to fully reach the next stage, the team would need to do the following:
- Effective progress is being achieved with minimal avoidable backtracking and reworking, by pair programming more and collaborating.

This is now achieved, as for this assignment, there was a lot of understanding and less coding, which led to the team having to put together their brains to understand the system and how Django worked as well, as this was a new framework for everyone.

Therefore, the way of working is now in place, as “all team members are using the way of working to accomplish their work.”

Overall experience

What are your main take-aways from this project? What did you learn?
A well-structured and documented project that has a clear and positive attitude toward new contributors is much more pleasant to work with.

We have also learned about many different open-source projects when searching for suitable ones, which will help if/when we decide to actively contribute to such projects in the future. The experience of comparing many different repositories and different documentation will help when determining which projects are worth contributing to, and which would be a waste of time.

How did you grow as a team, using the Essence standard to evaluate yourself?

The team could be seen as being in the “Performing” part of the Essence standard. The group collaborates as a team and supports each other when necessary. Potential time sinks are effectively identified and avoided, for example by changing what project was worked on from assignment 3 to this assignment, as well as through using a structured but flexible project plan. Active communication helps us achieve this.

Updates in the source are put into context with the overall software architecture and discussed, relating them to design patterns and/or refactoring patterns.

When implementing the feature the structure of the existing repository was taken into account to align with the previous programmers’ established design patterns. Not doing this would make it much more difficult for the maintainers of the project to keep track of the changes and structure over time. In turn this would likely force major refactorings in order to keep the project in check, something that might often be left for the maintainers to do (rather than feature contributors). Adopting a preemptive approach with atomic functions where the code complexity is kept reasonably low does not add a lot of work upfront while at the same time decreases the risk of those later problems appearing.

The architecture and purpose of the system are presented in an overview of about 1–1.5 pages; consider using a diagram.

The main purpose of the entire project is to allow developers and creative artists to be able to more easily find GPL-compatible media items. It acts as a search engine for media libraries for this purpose (as a meta-search engine). The hosted webpage for this project, openverse.org, has api endpoints to search in other databases such as freesound, wikimedia and flickr. In turn openverse itself has an api for users and developers to search in all these databases simultaneously. To combat malicious actors from overloading these

apis it can be helpful to be able to access the statuses of these api tokens and how they are used. This issue attempts to help mitigate those problems.