Summary Code Review

List any system services that the site appears to use (based on settings.py, requirements.pip, etc.)? If applicable, make note of any potential setup complications that these services pose.

Overview

- Django 3.2
- Python 3.8 (per Dockerfile)
- Wagtail 5.0
- PostgreSQL (per base.py)
- Hosted: AWS

First Party Applications

- home
- search
- open_hours
- header_footer
- page
- webform
- library_programs
- online_resource
- person

Select Third Party Modules

- django-crispy-forms==1.14.0
- djangorestframework==3.13.1 (doesn't appear to be used directly)
- django-taggit==2.1.0
- xlwt, xlsxwriter, xlrd (Excel reading/export libraries)
- defusedxml, et-xmlfile (XML file handling)

JS/CSS Notes

No django-compressor present. No evidence of LESS/SCSS or similarly either. Most of the static JS/CSS is in the root static folder. No larger JS frameworks like React or Angular seem to be in use.

What version control is in use?

- Git on GitHub
- Number of Contributors: 5

Are they using best-practices for their site setup?

Examples:

Requirements file included	Yes
Unit or integration tests	No
Evidence of virtualenv	Unknown (1)
Configuration separate from code	Yes
Makes use of generic CBVs	Yes
Makes use of provisioning system	Unknown (2)
Makes use of CMS	Yes

(1) No evidence of virtualenv but Dockerfile is present
(2) Dockerfile but no indication of Ansible or Saltstack configurations

How well is the code and setup documented? < 0 1 [✓ 2] 3 4 5 6 7 8 9 >

The codebase does not includes a README file, nor was there any Sphinx or external documentation provided. There is some commenting within the code but not enough to be useful. Commenting within the repository is a mixed bag

- some comments provide nice descriptions with issue number reference, others are less helpful such as, “update user interface.”

Overall opinion, can we work with this? < 0 1 2 3 4 5 6 7 [✓ 8] 9 >

The codebase is relatively small. First-party modules mostly consist of very concrete definitions of Wagtail pages. Most features are basic implementations of Wagtail built-in (or installed modules add-ons) features. Not a lot of layers of indirection in first-party code.

A naive CLOC count.

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
CSS                            125           5773           1244          32934
JavaScript                     237           4379           4870          32581
SQL                              1           2740           2585          11398
Python                          70            512            187           2671
HTML                            82            267             14           2148
SVG                             43              0              1            261
Dockerfile                       1             12             26             22
Markdown                         1              4              0             17
JSON                             1              0              0              1
-------------------------------------------------------------------------------
SUM:                           561          13687           8927          82033
```

Note there are only 2,671 lines of Python. JavaScript and CSS count here is almost certainly at least double its actual size (since copied files were included) and much of the count there comes from imported/copied code from third-party features brought into the static files. The handwritten JS I noticed was minimal. The actual amount of first-party code is relatively small overall.

General Observations

This is a relatively small Django/Wagtail codebase. Not many dependencies and no 3rd party services that I noticed. No notable odd hacking of Django/Wagtail internals to make unusual dependencies work or other similar code smells that old projects tend to develop (e.g. no forked dependencies).

Documentation is lacking but the simplicity of the Python codebase means that should not be a major roadblock. There is a Dockerfile which suggests we may want to use that for local setup/deployment. But if that isn't in a working state, importing a DB dump and running "runserver" should suffice.

JavaScript assets actually written in the codebase are either jQuery or vanilla. Everything else is copied in as a library/widget from elsewhere. Handwritten JS is pretty minimal. Short functions handling basic events on single pages mostly.

Most of the 3rd party modules are pretty typical for a Django site. Things like image libraries, common Django add-ons, and basic modules needed to run a Django app (gunicorn, psycopg2 etc.).

The code seems generally well formatted. This is a relatively standard implementation of a Wagtail based site. No real long functions or significant code smells. This should be a pretty straightforward project to work on.