DOCUMENTATION ON THE APP.

Please kindly set up the database by picking an option then run the web before celery. Just follow the instructions as I've detailed below, relax!! you'll be cool. Please note that the db.rar file was zipped so that I won't exceed the required < 10mb criteria of the submission form. Apologies for the inconvenience in advance.

SET-UP

SETTING UP THE WEB

This is yet built to run locally so kindly follow these steps to run.

On your local machine inside the root directory ensure pip and python are installed and run:

- *pip install -r requirements.txt*

I recommend you don't run the server yet, please kindly read the database setup first. Thanks.

POPULATING THE DATABASE:

I recommend using sqlite3 as it's what I've configured by default. I've seeded enough items from the api to the db so you can play with them on the new interface i built. Kindly do one of the three methods below:

(Doing one these is recommended)

1. unzip the **db.rar** file

    a. *unzip the db.rar file and run*
    b. *python manage.py migrate*

2. Or: (if madedata.rar file exists) (if it doesn't skip this)
    a. *Unzip the madedata.rar to have madedata.json the run*
    b. *python manage.py migrate*
    c. *python manage.py loaddata madedata.json*

    (this might take a while to download)

3. Or: And if you're not comfortable working with these ready made data grab your coffee I've got you covered. I've created a custom django command for you to download the latest news and stories for the app to start with kidly run:

    a. *Deleted any db.sqlite file in root directory (if any) then run:*
    b. *python manage.py migrate*
    c. *python manage.py save_latest_items*

    (this might take a while to download)

And boom!! your database is loaded with items needed enough to start.

Running the web:

After following the earlier steps kidly run this command

python manage.py runserver

And that should do it for this stage as I've migrated the database from postgres to sqlite3 for ease of third-party integration and testing.

PERIODIC-TASK set-up (celery)

Ensure the earlier steps was observed and redis is installed up and running on your machine then run this this command on separate CLI's one after the other (make sure one finishes execution before other)

1. *redis-server*
2. *celery -A seedtest worker -l info --without-gossip --without-mingle --without-heartbeat -Ofair --pool=solo*
3. *celery -A seedtest beat -l info*

And boom!! your app is running with latest items getting downloaded automatically from the api on on port: http://127.0.0.1:8000/items/

Note: ensure redis is up and running; you can try redis-cli PING to get a PONG.

LOGIC:

VIEWS.PY - LOGIC

I used django filters to filter the items by text and manually get an object_type query-param to filter by type and set a pagination of 100 to ease workload on the database and as recommended by the test.

In the detail view I get necessary objects by id and catch where necessary. Rendering all related kid-comments.

I also created my custom template filters in the frontend to answer some complexities in the requirements.

Filtration and pagination is also implemented in my api endpoint

TASKS.PY-LOGIC

As there are different ways by which items from the API could be downloaded and for some reasons own as opinion to functions of building another gui for the api like we're doing, hence i created a function to seed the database fetching items from the api by providing two arguments: the range of objects keys or id's to iterate over and save. I've named it ***create_items_from_api(startt, endd).*** This module or function is what I call in several cases to populate the database as I like. This function is the module I used to populate the earliest items to the database. This unit module or function is the function I called in the custom command line to populate the latest news to begin with, calling the ***dowload_trending()*** module, and again I called it when I created another function called ***create_items_from_api_beat_latest()*** . This later function is what I passed to celery beat to be called periodically in the background.

NAVIGATION

The api does not provide an end point where all articles could be downloaded at once, so this makes it to be really interesting as I have to download this items one by one, I stated saving to database right from the onset of beginning of Post creation, and I save 40000+ plus earliest Posts and 20000+ latest post to easily test and check for cross functionality and have assured behavior of the relation of these items.

I have built and hosted the app on port 8000 with all functionality as described in the test requirements.

Check port:: http://127.0.0.1:8000/items/ to view the list of all latest top stories and items. On the top right corner is a filter button to filter by type and to the left is a search icon to filter by text, Here you would see a plus icon on the bottom right corner to navigate to RESTfull API endpoint @ http://127.0.0.1:8000/api/ where you can view and create new post with necessary restful requirements.

Click on an item to redirect to the detail page to view all attributes and related kids-comments. I also put nice icons to aid ux and ease of navigation. You can easily navigate to the parent post of the current item if it exists. @ http://127.0.0.1:8000/items/< id >/.

On this detail page is a pen icon to redirect to the api details where you can perform the R-U-D function provided the item was created byUS and not from the API. Kindly check and see how I have made necessary catches. http://127.0.0.1:8000/api/< id >/

BONUS

I found the API very interesting and played a little bit around with the relations. I created an author posts' page to view all posts, comments and activities performed by the respective user. On any item detail page, if you click on the author's name it will take you to this author's dashboard page. The page may be useful in the future as the author can make changes to his activities here provided they were first created by him @ http://127.0.0.1:8000/author_review/< username >/

Limitations :

This app is recommended to run locally, if for any reason you want to make it public, kindly contact the developer @ mallamsiddiq@gmail.com for further advanced considerations.

Also, the API items are downloaded once at a time and hence might not be expected that all items be found on this app soon.

And also please note that madedata.json file might be excluded if the whole file exceed what the provided form can take and hence option 2 of seeding the database might not be feasible

Thanks