## Description

The following project is a REST API built using Django and the Django Rest Framework. It offers a single endpoint that takes the URL as a query parameter at hostname/webpage/?url=https://www.defaulturl.com

- Python version 3.7
- Runs on Ubuntu 18.04
- Django 2.2.1

Upon receiving a new request, it extracts the wanted data from the page by scraping its HTML content using the library BeautifulSoup.

The results are saved in the applications database, including a column 'delete_on', which holds a date that marks 24 hours after the request. Any request with the same URL made before that date, will return the database entry instead of rescraping the site.

The library Django-Crontab is used to run regular, scheduled tasks: Every hour it checks the database and deletes any entries where the date has passed. (It is currently set to run every minute for demonstration purposes)

The request returns with different custom or generic error messages and status codes depending on what went wrong. For example, in the case of 403 or 405 error codes when trying to reach the given URL, the request will return a short description of what happened and what the reason for this could be, as well as include the error code and the error message from that URL.

## Assumptions, Problems and Explanations

- Because there is no standard on how to implement login forms, I assumed that a page contains a login form if it includes a form with input type 'password'
- I had problems trying to keep the request response quick because of the sheer amount of links that need to be pinged to check if they are accessible or not. I used the library 'httplib2' to make a request to each link without downloading its content. However, even like this it can take a while. Most pages do not take too long. Nevertheless, some sites

have over 300 links on them and if every ping takes 50 or 100 milliseconds, for example, this still adds up to 15/30 seconds.

## Conclusion

Overall, it is hard to say exactly how much time it took me. I spent about one day on and off building the majority of the application and another half a day finishing it up and polishing and testing it.