# Web Scraping

Assignment Details:

The assignment goal is to basically make a script to scrape some data from a website exposed as a Django rest framework API

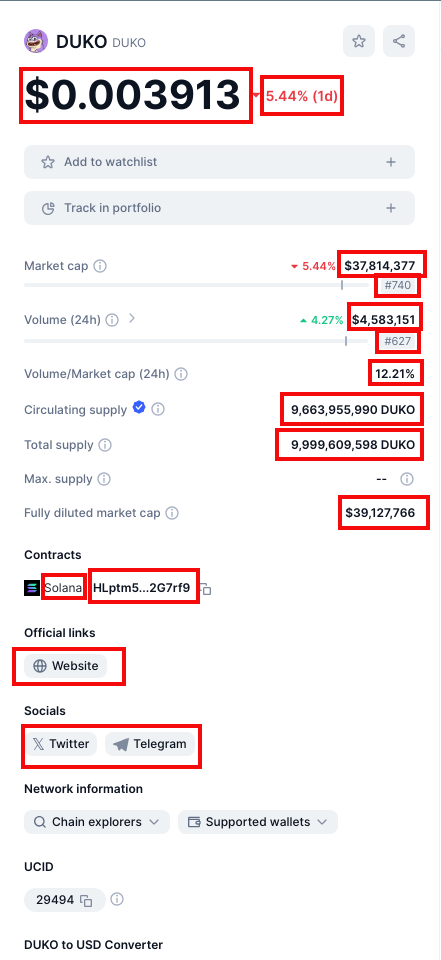
The websites assigned to you is:

<https://coinmarketcap.com/>

Write a Django rest framework API which will take in a list of crypto coin acronyms, scrape the data from the website and return back the JSON response.

Libraries to use:

* djangorestframework
* celery
* requests
* selenium

For this example, URL: <https://coinmarketcap.com/currencies/duko/> scrape the following details:

There will be 2 APIs in Django rest:

1. /api/taskmanager/start\_scraping [POST]- this will take in a list payload [“DUKO”, “NOT”, “GORILLA”] which are names of the crypto coins and submit a scraping job (celery will be used) to be run for these coins parallelly and return back a job id
2. /api/taskmanager/scraping\_status/<job\_id> [GET] - From the job\_id received in the previous API, we can query this API and it will return the currently scraped data for that job. Sample output:

{

  "job\_id": "<UUID>",

  "tasks": [

    {

      "coin": "DUKO",

      "output": {

        "price": 0.003913,

        "price\_change": -5.44,

        "market\_cap": 37814377,

        "market\_cap\_rank": 740,

        "volume": 4583151,

        "volume\_rank": 627,

        "volume\_change": 12.21,

        "circulating\_supply": 9663955990,

        "total\_supply": 9999609598,

        "diluted\_market\_cap": 39127766,

        "contracts": [

          {

            "name": "solana",

            "address": "HLptm5e6rTgh4EKgDpYFrnRHbjpkMyVdEeREEa2G7rf9"

          }

        ],

        "official\_links": [

          {

            "name": "website",

            "link": "https://dukocoin.com"

          }

        ],

        "socials": [

          {

            "name": "twitter",

            "url": "https://twitter.com/dukocoin"

          },

          {

            "name": "telegram",

            "url": "https://t.me/+jlScZmFrQ8g2MDg8"

          }

        ]

      }

    },

    {

“coin”: “NOT”,

            “output”: {

…

}

    },

    {

“coin”: “GORILLA”,

            “output”: {

…

}

    }

  ]

}

You need to create a **class** for CoinMarketCap with the functions for making requests, scraping data, processing data and sending back a JSON response (you can add any other function if needed). Submissions without proper OOPS concepts will not be preferred.

Used tools: VS Code

Content:

* Creating a Virtual Environment
* Installing the wanted Libraries
* Starting a project and an app using Django-admin
* Adding Installed apps in settings and celery settings in project folder setting
* Creating a Celery.py file in project folder and adding celery app to init file
* Update admin.py, apps.py, models.py, views.py from the GitHub code
* Create files such as scrapper.py, tasks.py, urls.py in application folder
* Add the application URLs to project URLs
* Run server & open the /api/taskmanager/start\_scraping [POST] and post the

{

"coins": ["DUKO", "NOT", "GORILLA"]

}

And wait sometime to load the values

* Then paste the Id you got in Post section in /api/taskmanager/scraping\_status/<job\_id> [GET]

Results can be seen in both API and JSON format

## Detailed Process:

**NOTE: ALL THE CODE EXECUTED IN VSCODE**

* Creating a Virtual environment:

This can help to avoid any contact with your personal computer so that it wont effect any of your personal computer data.

It can be created using the following commands:

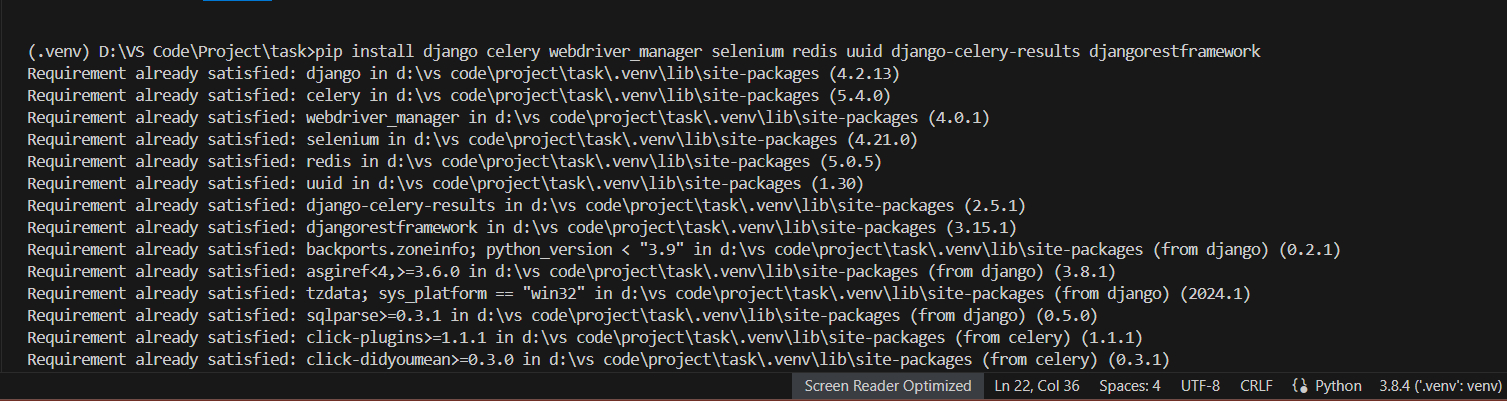
* + mkdir <Folder\_Name> [mkdir project] #to create a folder for the process
  + cd project #changing the directory to the created folder
  + python3 -m venv .venv #creating an environment called .venv
  + dir #checking if the venv is created or not
  + .\.venv\Scripts\activate #first open the folder [see if the path is set in cmd or not] then activate the environment to run further

As shown in the figure we get to create a virtual environment and activating it for further process

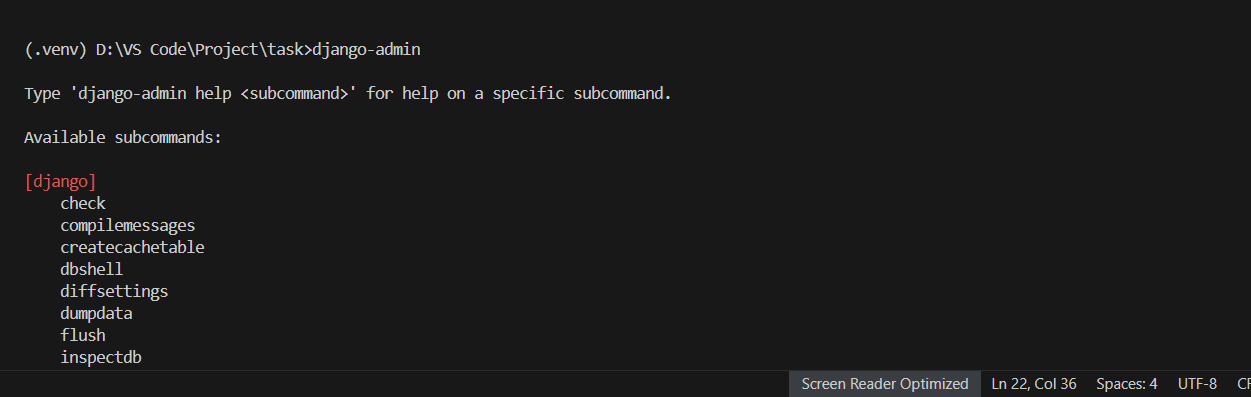
* Installing the wanted Libraries:

As already mentioned in the assignment we have to install the libraries that are needed for the project to be executed:

* pip install django celery webdriver\_manager selenium redis uuid
* pip install django-celery-results djangorestframework

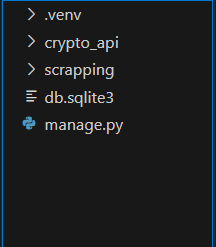
If there are any libraries missing you can just install with pip install <lib\_name> and make sure it is compatible with the version of python that is in your pc

* Starting a project and an app using Django-admin:

Before starting a project, you can check the functions of Django by running django-admin:

To start a project the following commands are used:

* django-admin startproject <project\_name> [django-admin startproject scrapping]
* python manage.py runserver #to see if the server is working or not then crtl+c
* django-admin startapp crypto\_api #for api building

Now check the folder if the project and application is created or not:

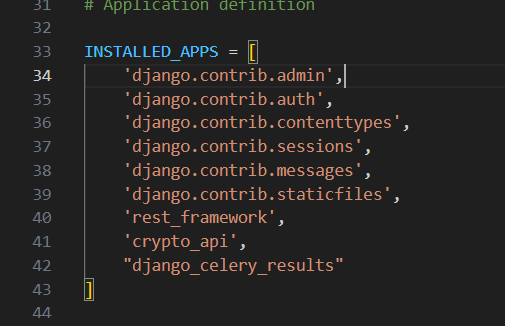
* Adding Installed apps in settings and celery settings in project folder setting:

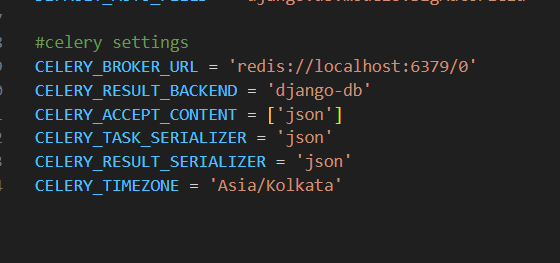
If the created project folder you will find settings.py there you have to add the installed apps which are used for further purposes and also add celery settings:

‘rest\_framework’, #for using django restframeworks

‘crypto\_api’, #created application

‘django\_celery\_results’, #for storing celery results in admin

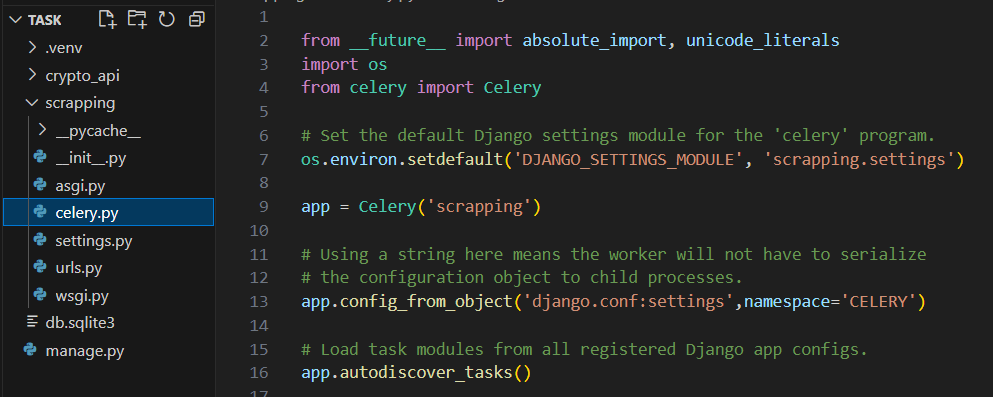


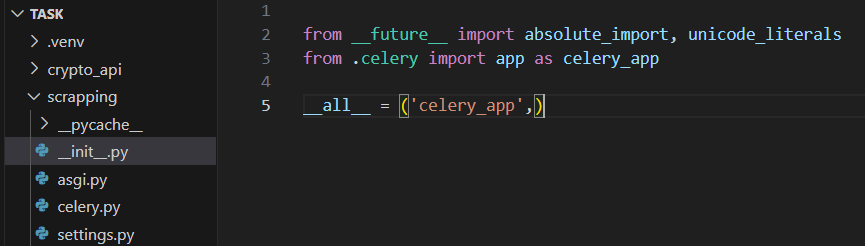


In the above picture 6379 is Redis port

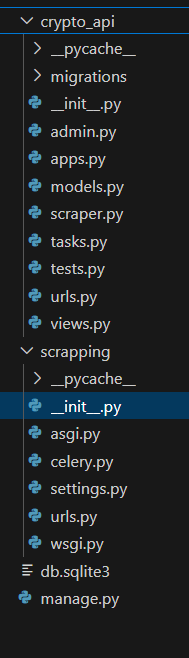
* Creating a celery.py file in project folder and adding celery app to init file:

In the project folder create a new file naming celery.py:

then add the celery details in \_init\_.py to import the application:

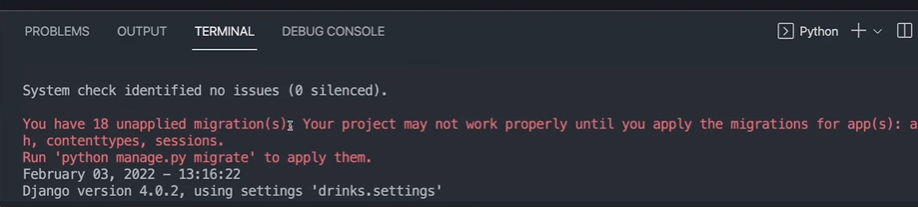
* Update admin.py, apps.py, models.py, views.py from the GitHub code in the crypto\_api which is Api application <link is given below>
* Create files such as scraper.py, tasks.py, urls.py in same application folder

NOTE: Here as the web scraping is done from the given website from the assignment so if you want to scrape from different website, please consider to change the selectors in the scraper.py file.

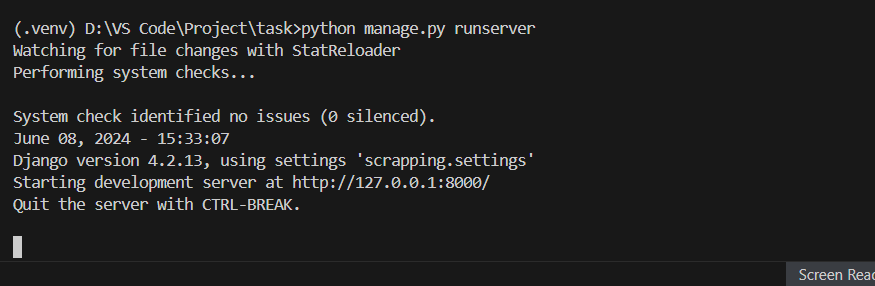
After all the files replaced and edited then we have to migrate to avoid the following error when you run the server at first:

i.e., open a new terminal and follow these commands:

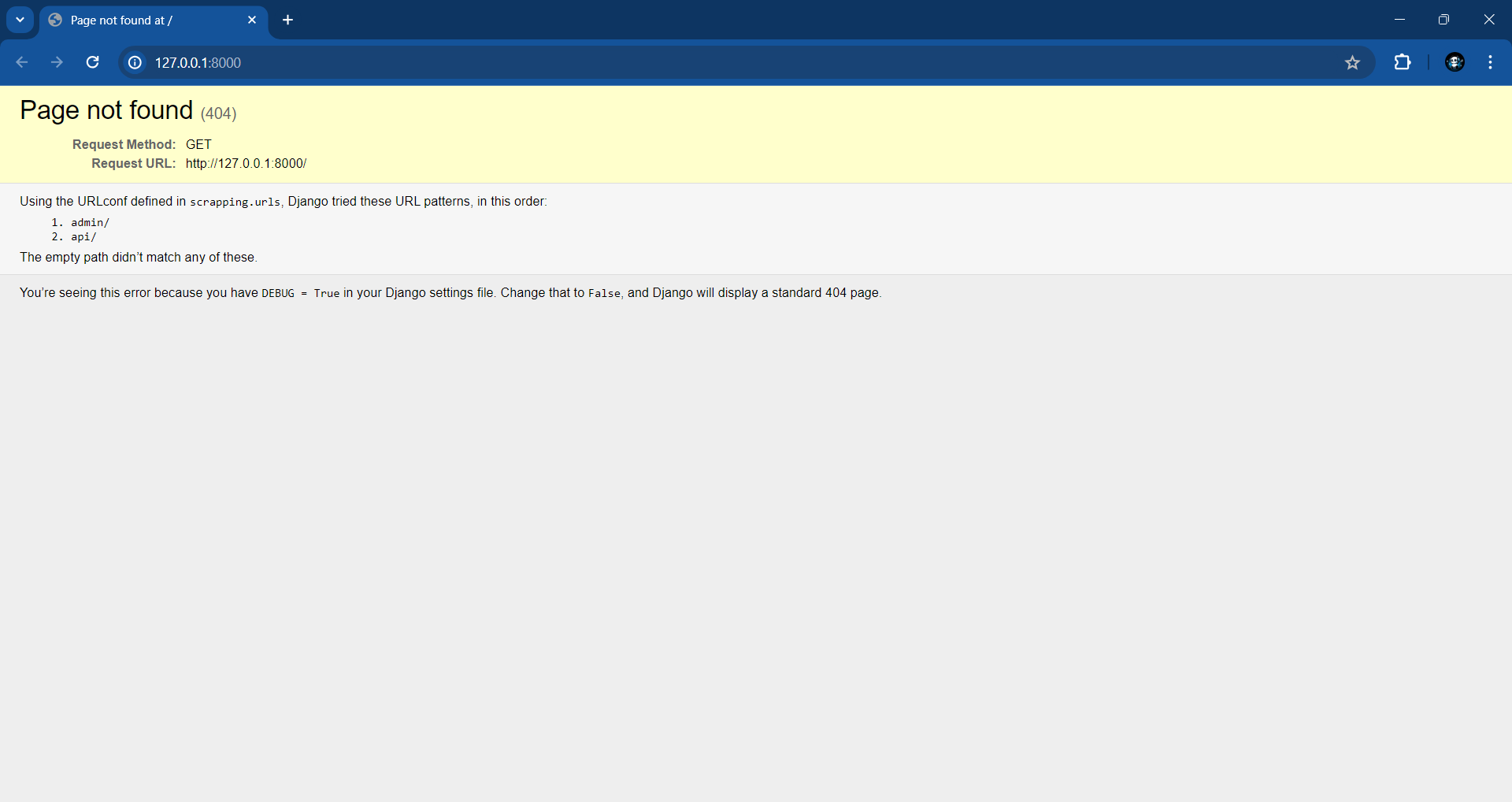
* python manage.py makemigrations
* python manage.py migrate



After this Run the server by the command:

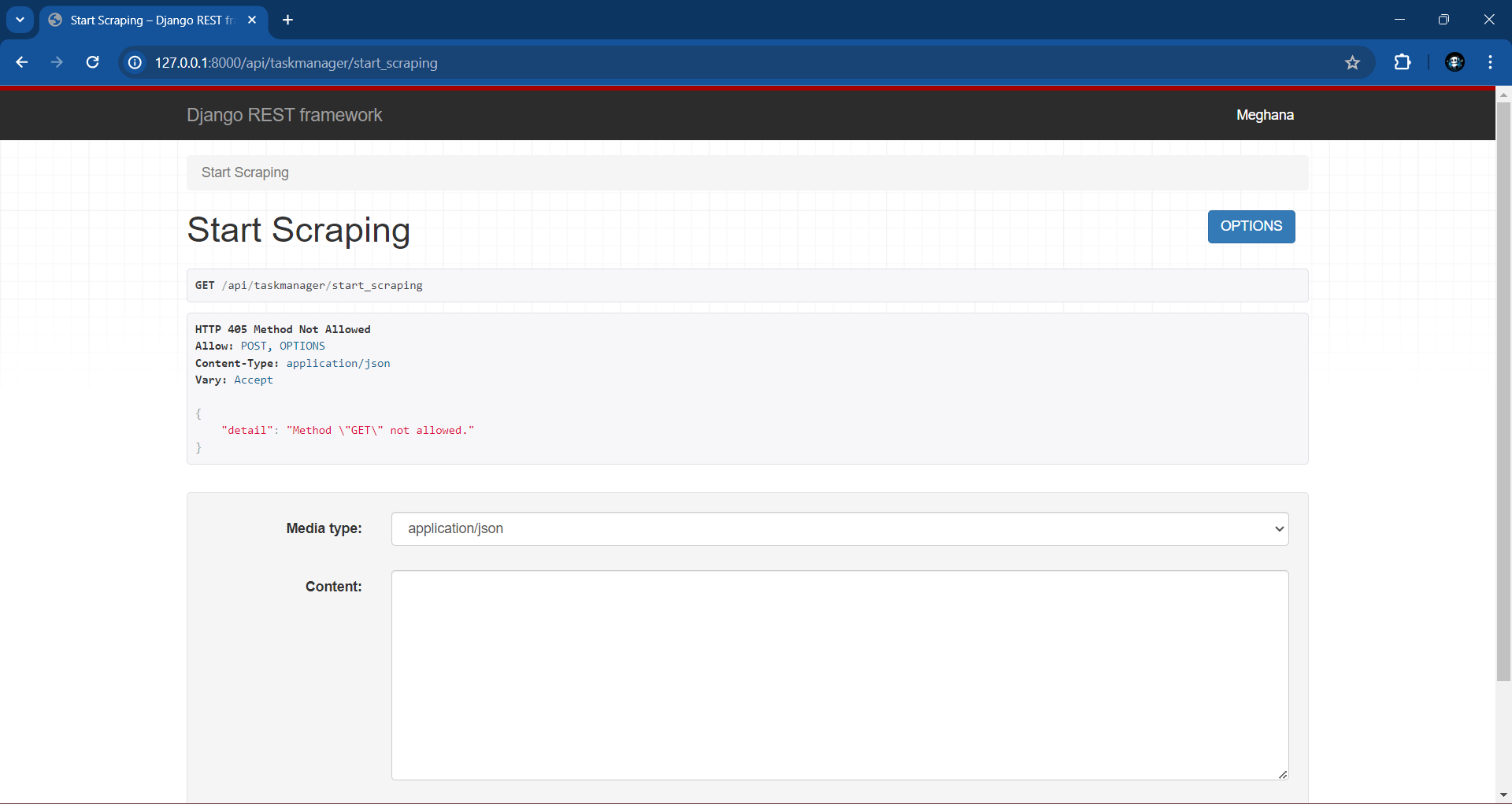
* python mange.py runserver

Then open the server given in the browser:



Then open the path that you have mentioned in the URLs in application: choosing GET path:

* path('taskmanager/start\_scraping', StartScrapingView.as\_view(), name='start\_scraping'),



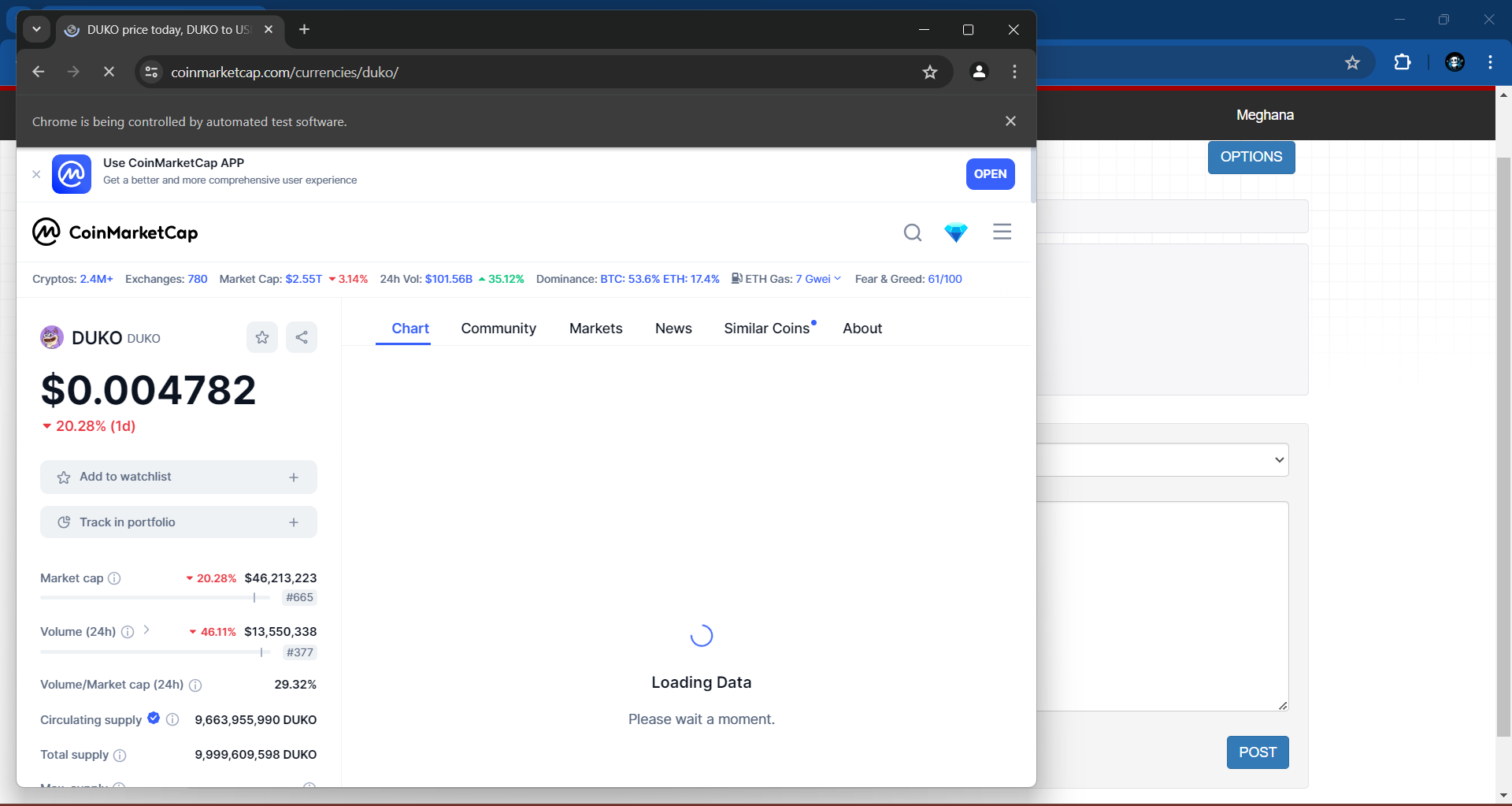
Then in content give the acronyms mentioned: as I have taken as coins I will be mentioning the key as coins:

{

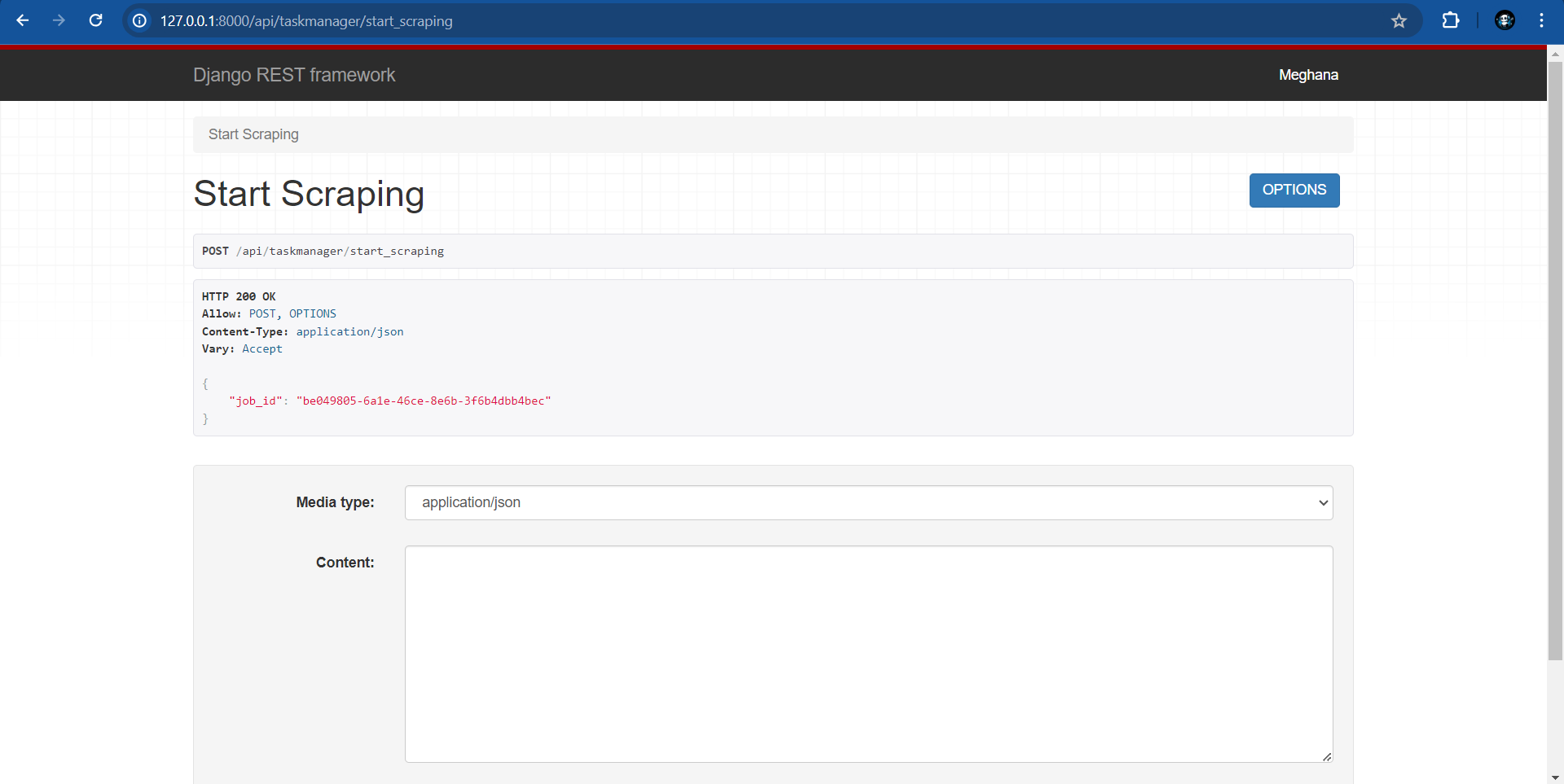
"coins": ["DUKO", "NOT", "GORILLA"]

}

Then click post and wait sometime as it loads the values given in the content regarding backend:



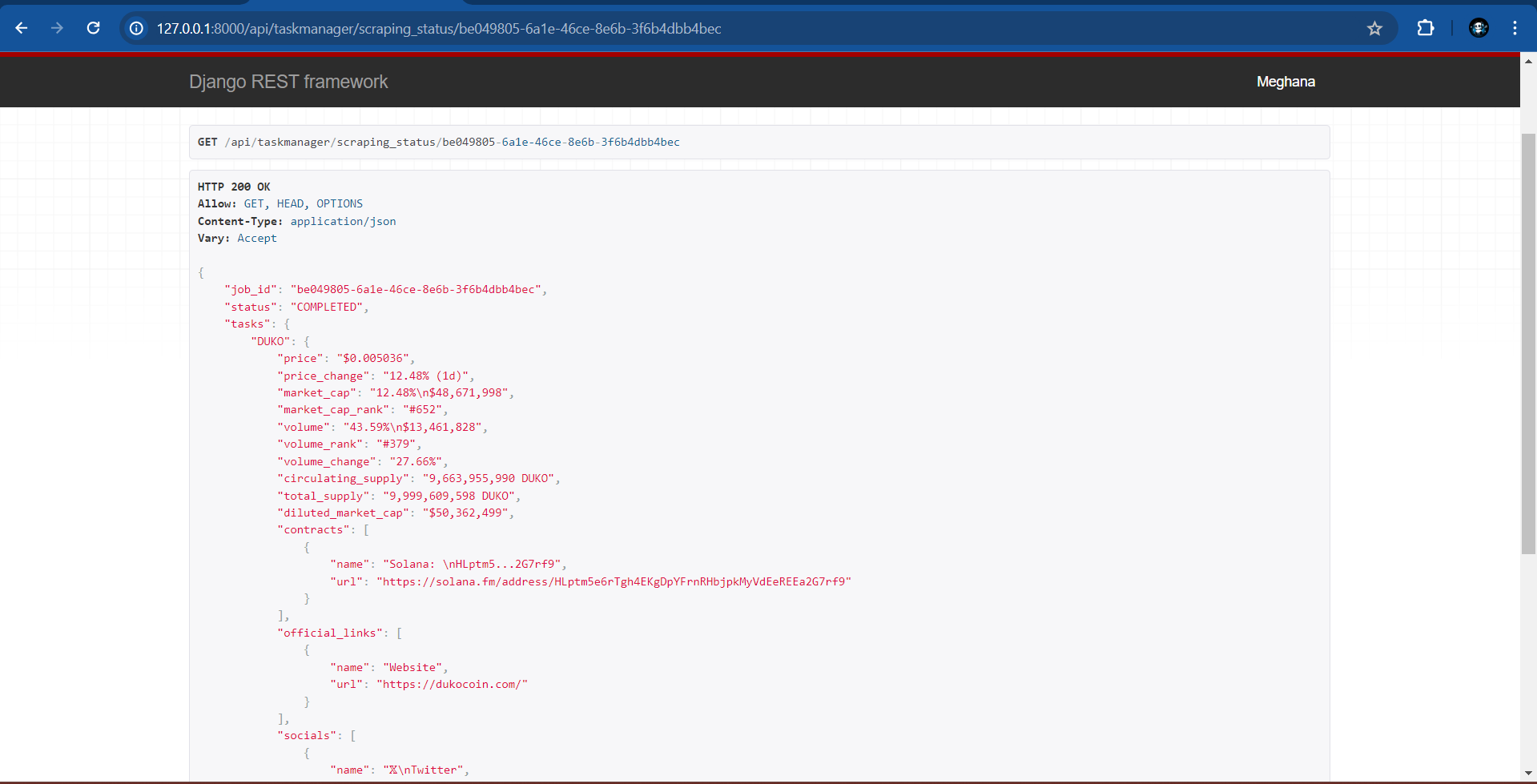
Then it will give some JOB\_ID:



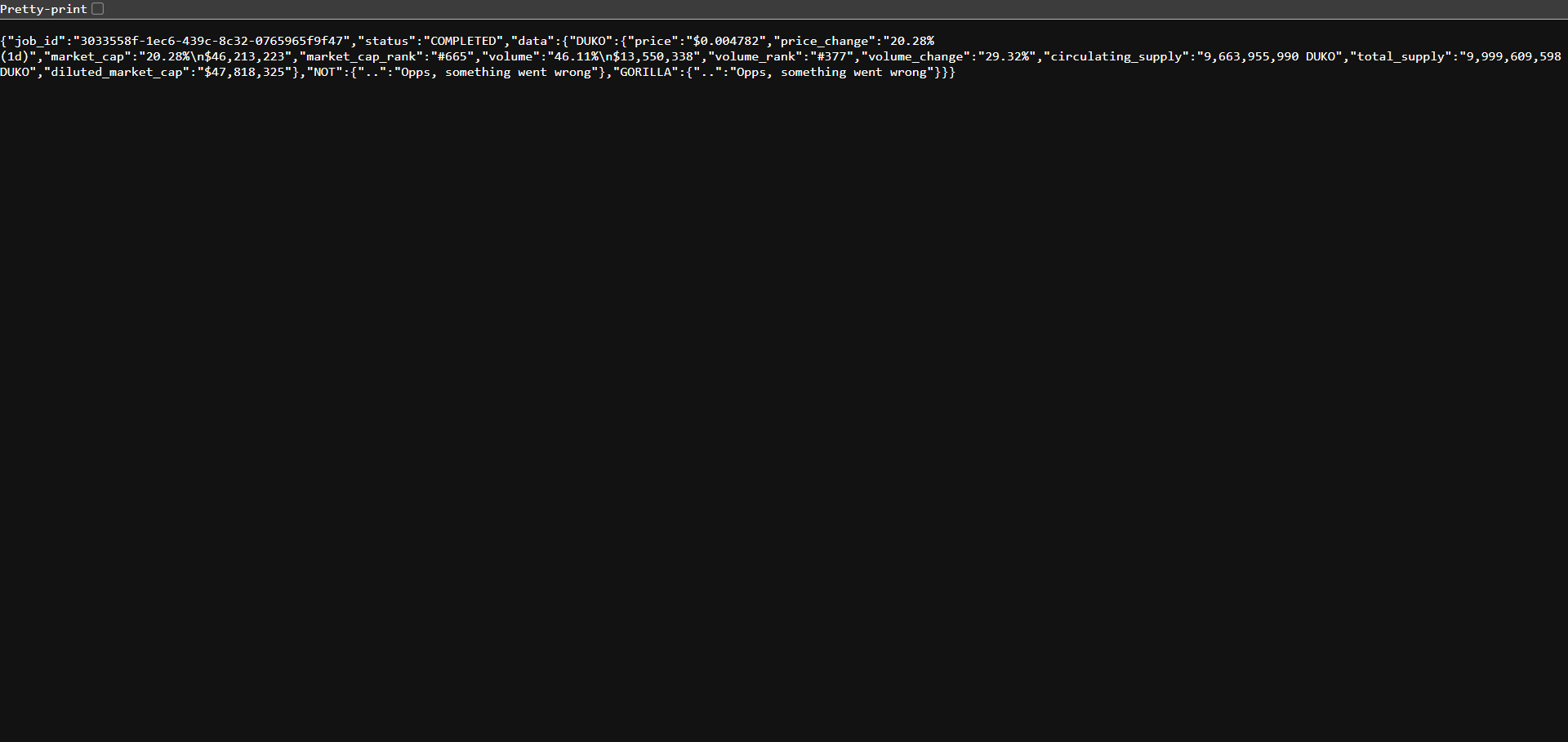
"job\_id": "be049805-6a1e-46ce-8e6b-3f6b4dbb4bec"

Now copy the JOB\_ID and paste it in the second path in the URLs of your application in the code you mentioned:

* path('taskmanager/scraping\_status/<uuid:job\_id>', ScrapingStatusView.as\_view(), name='scraping\_status'),



This is the API format and in the GET, you can find JSON to have a view in JSON format

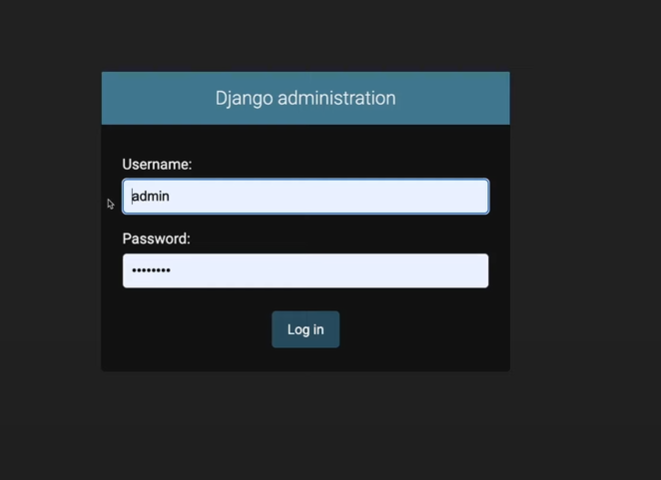


So, this is the process of getting the data from a website.

You can also create your super user in the django admin to view the details:

When you open the link and /admin

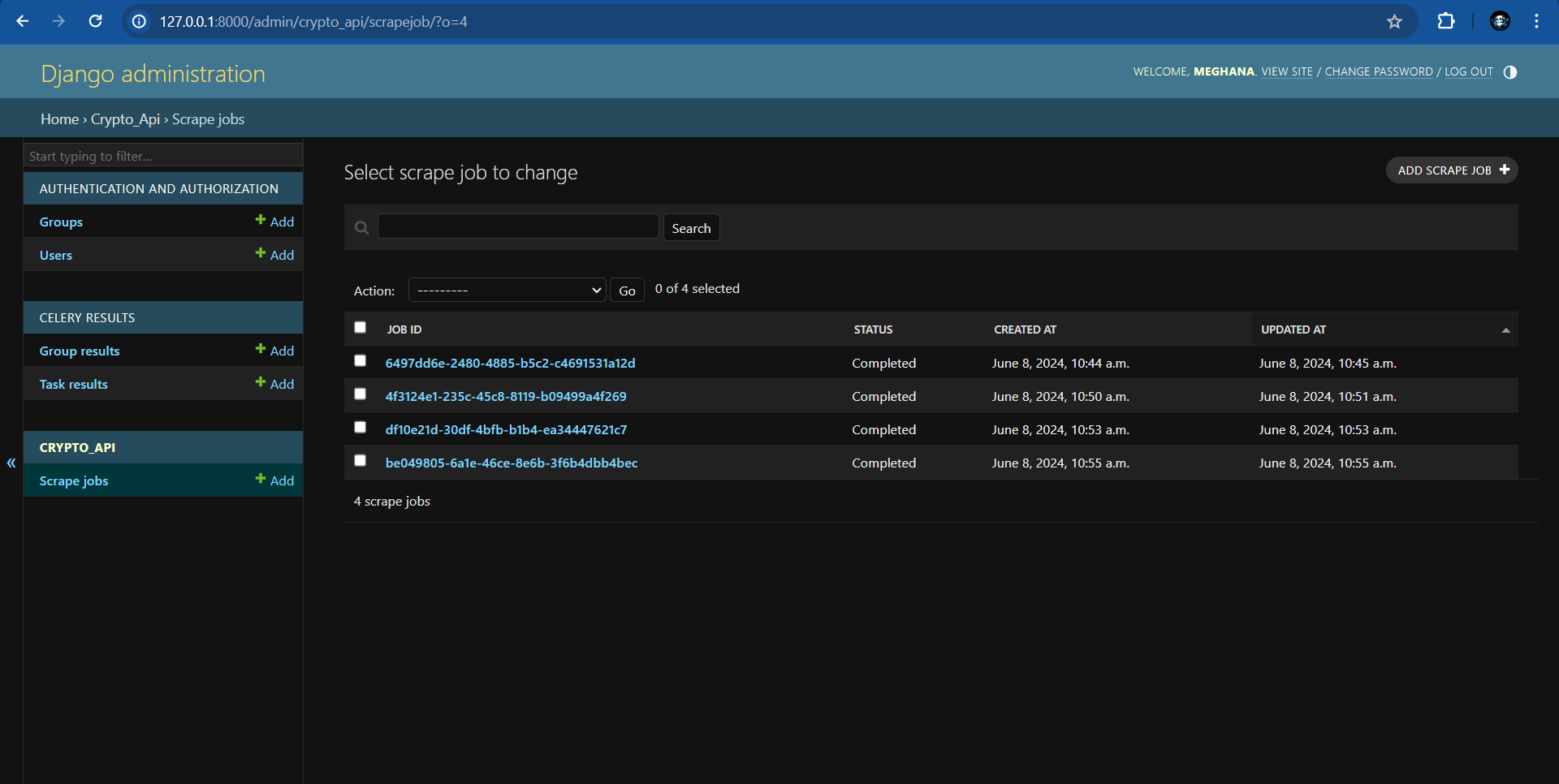
http://127.0.0.1:8000/admin then you can see:



You can set username and password by the following commands in cmd:

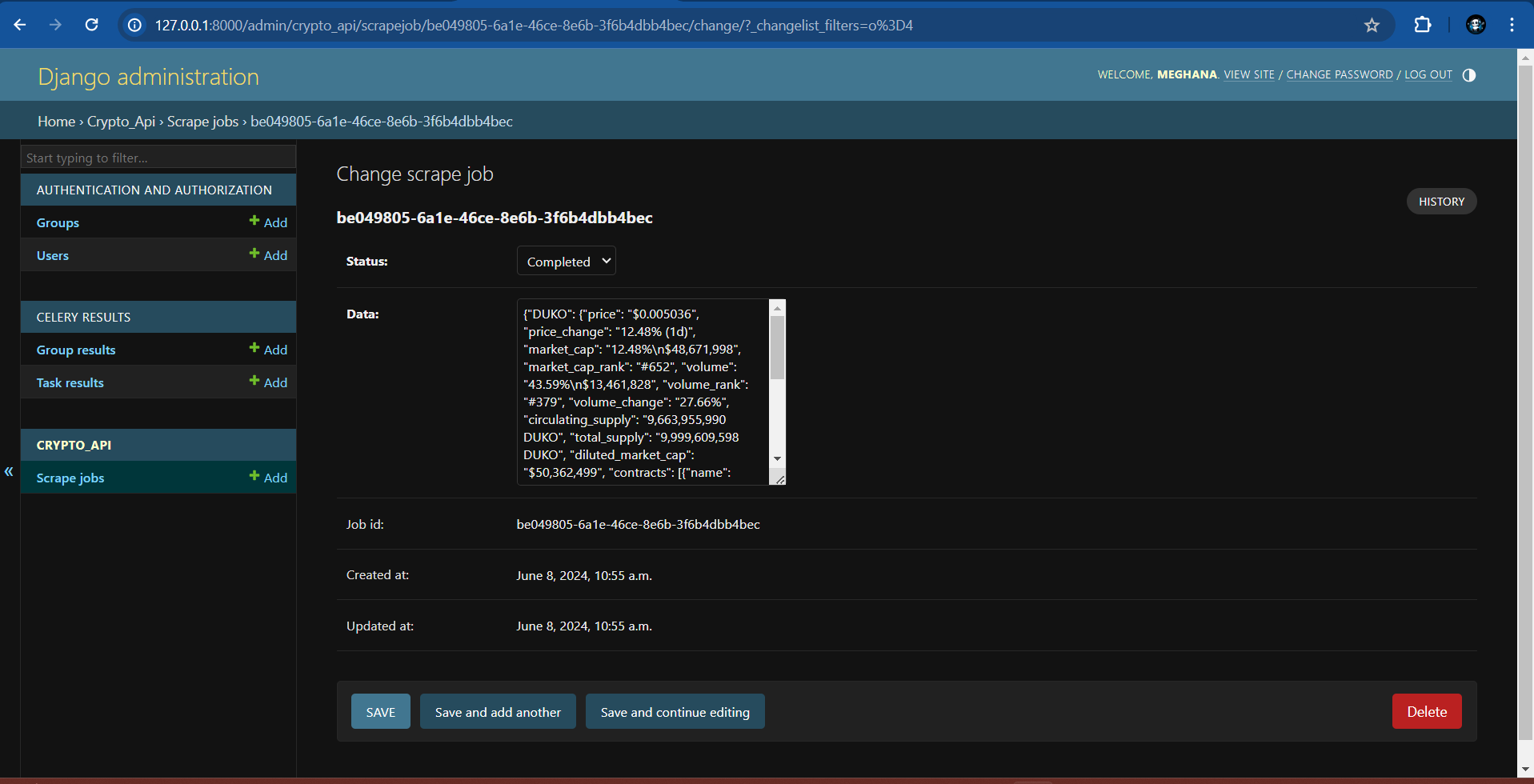
* python manage.py createsuperuser
* set a username
* give your email
* set password

Then again run the server and go to admin and login to:



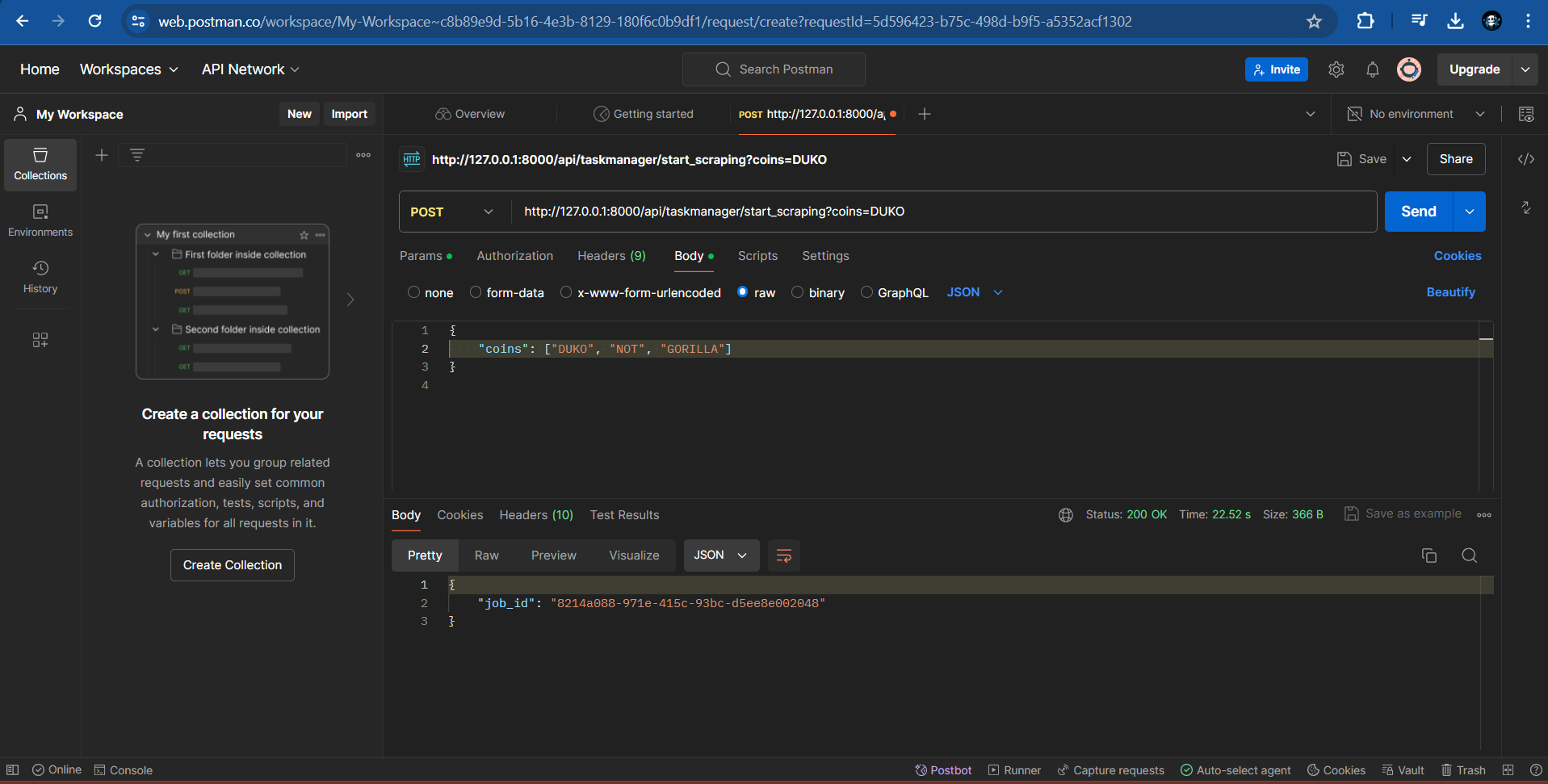
So that it will show the jobs that had been run previously

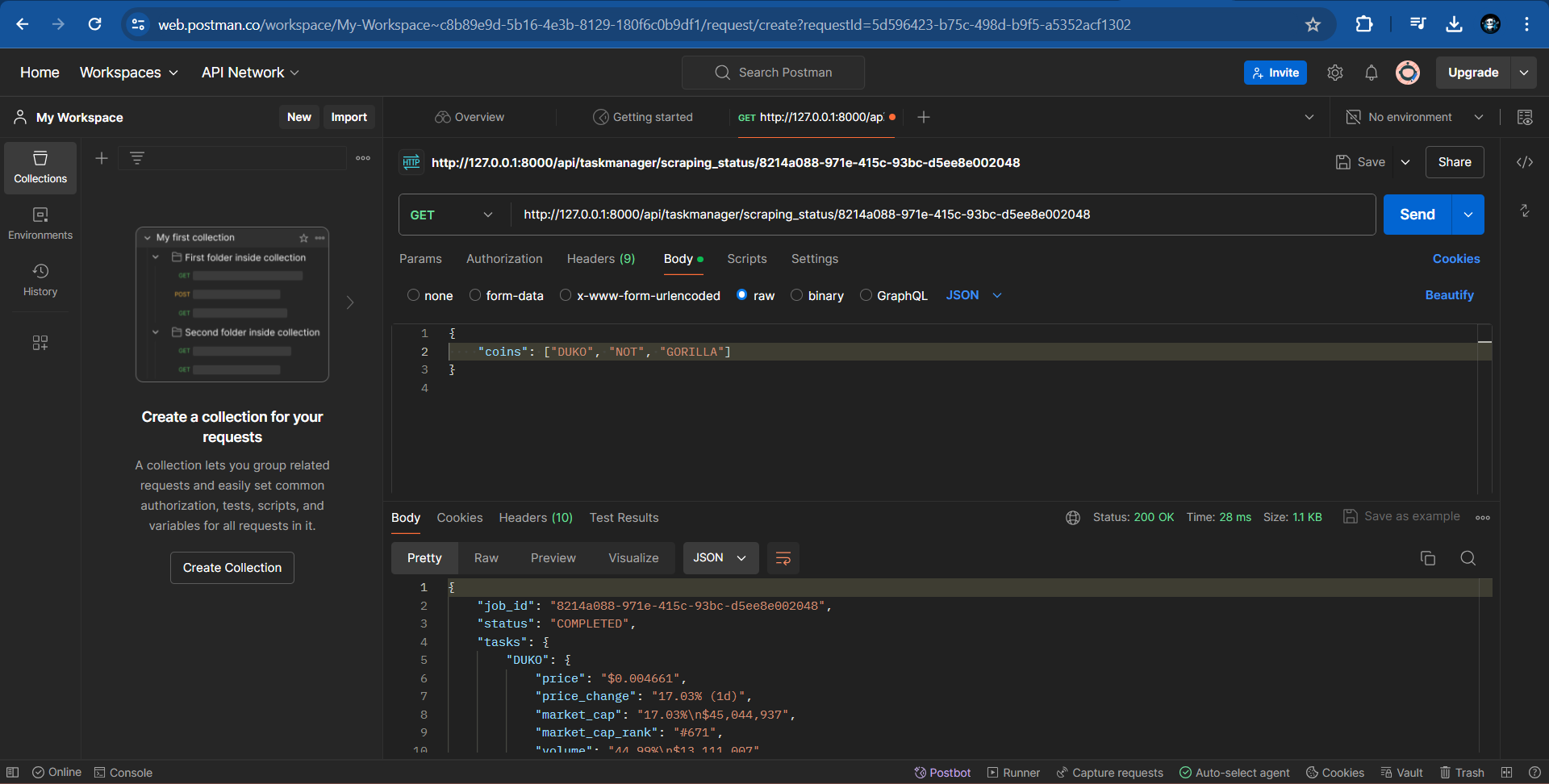
And also, can see the details of it:



Thus, the admin creation is done and can view your job\_ids saved.

Postman results:





GitHub link: https://github.com/Meghanayedla32/web\_scraping