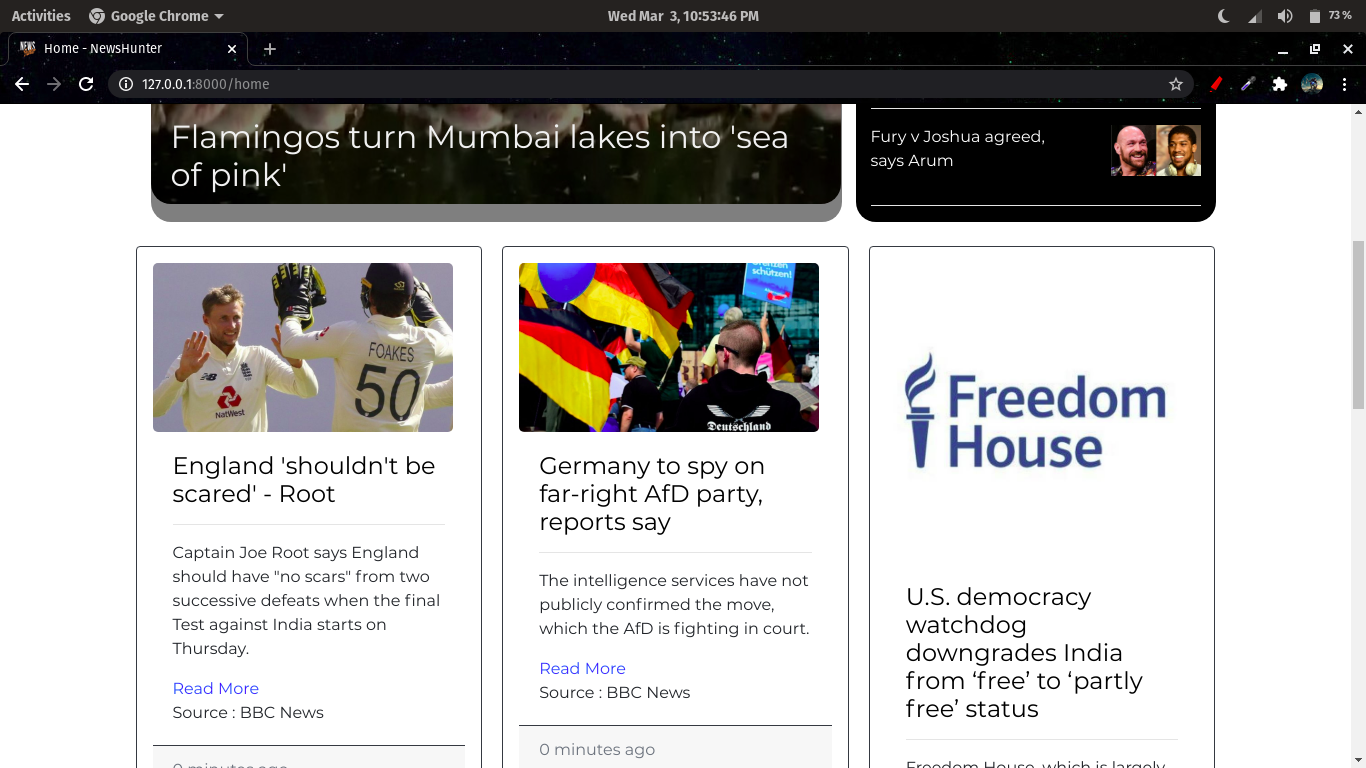
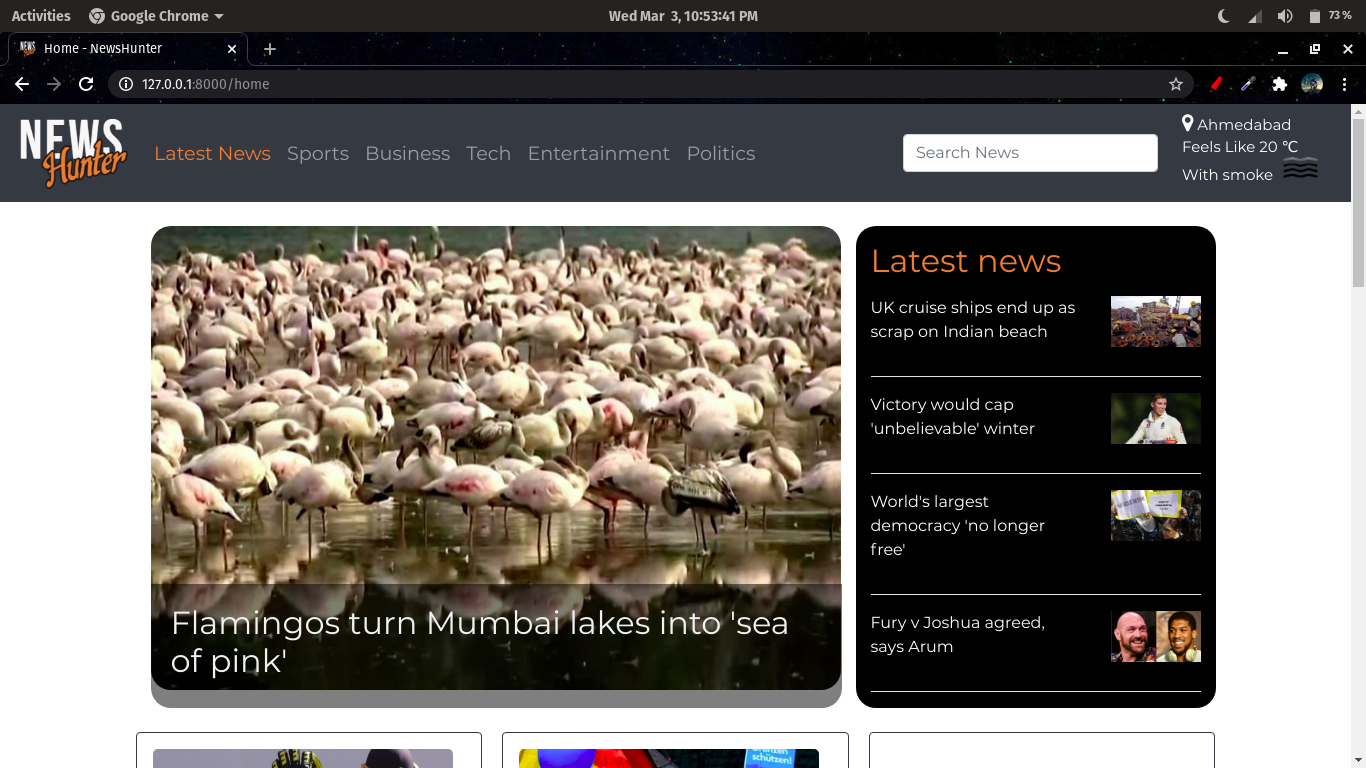
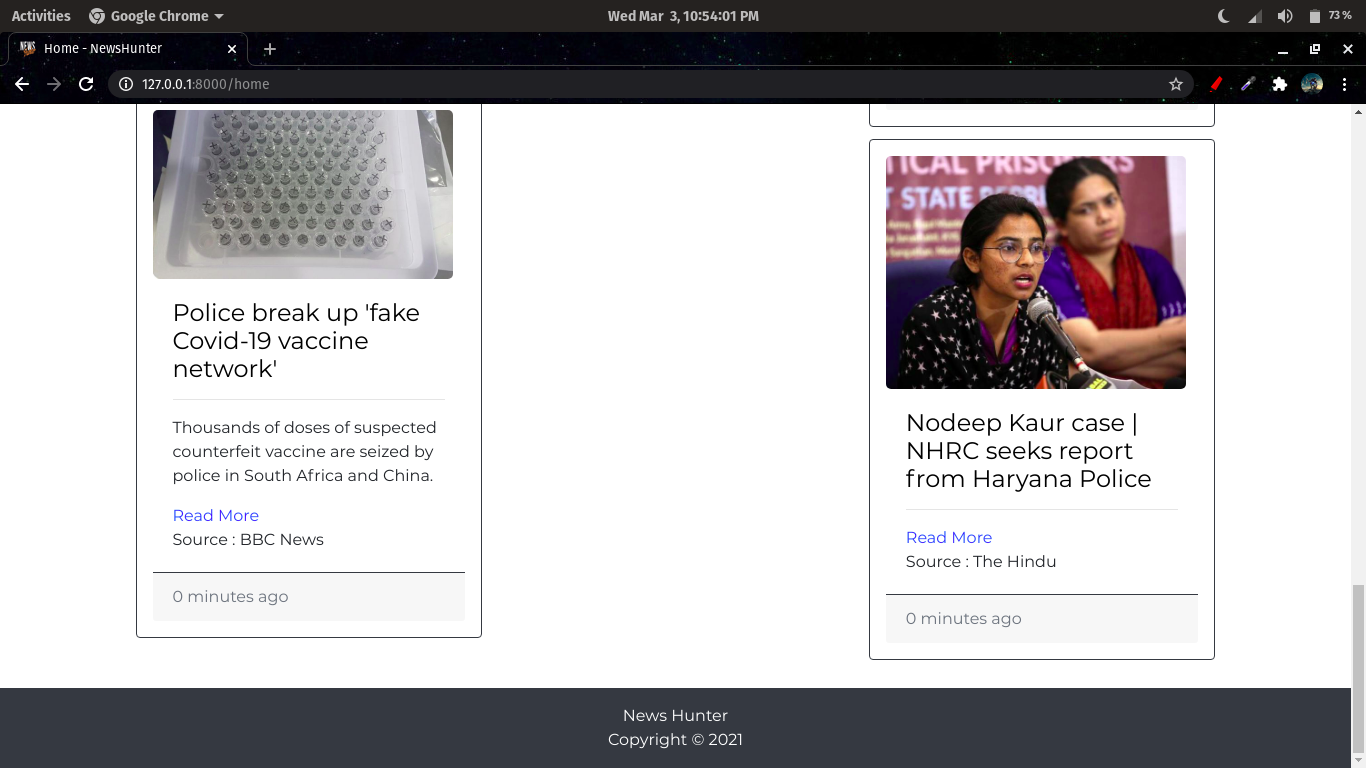
**Project Title : News Hunter**

# **Actual Screenshots with brief code snippet**

1. **Web scraping news from websites and displaying on webpage**

****

****

**Code snippets :**

* **Web scraping news**

import requests

from bs4 import BeautifulSoup

import json

from datetime import datetime

from pymongo import MongoClient

url1 = 'https://timesofindia.indiatimes.com/news'

url2 = 'https://www.thehindu.com/news/'

url3 = 'https://www.thehindu.com/'

url4 = 'https://www.bbc.com'

page1 = requests.get(url1)

page2 = requests.get(url2)

page3 = requests.get(url3)

page4 = requests.get(url4)

news = []

soup1 = BeautifulSoup(page1.content, 'lxml')

soup2 = BeautifulSoup(page2.content, 'lxml')

soup3 = BeautifulSoup(page3.content, 'lxml')

soup4 = BeautifulSoup(page4.content, 'lxml')

In our web-application, we are fetching news from 4 urls as shown in above code and content is scraped from the pages and is stored in respective variables.

news\_format =

{

"Img\_src" : "",

"Headline" : "",

"Article\_link" : "",

"Category" : "",

"Content" : "",

"Source" : "",

"DateTime" : datetime.now(),

"Reported" : False

}

Further, the headline, image source, article link, source, Datetime is extracted from the html source code and stored in a python dictionary.

Likewise, news from all the websites are stored in the given format and add in a list - “news”.

* **Adding news to database**

from pymongo import MongoClient

try:

conn = MongoClient()

print("Connected successfully!!!")

except:

print("Could not connect to MongoDB")

client = MongoClient("mongodb://localhost:27017/")

web\_scraper = client['web\_scraper']

db = conn.web\_scraper

collection = db.news\_table

# print("Insert data in mongoDB")

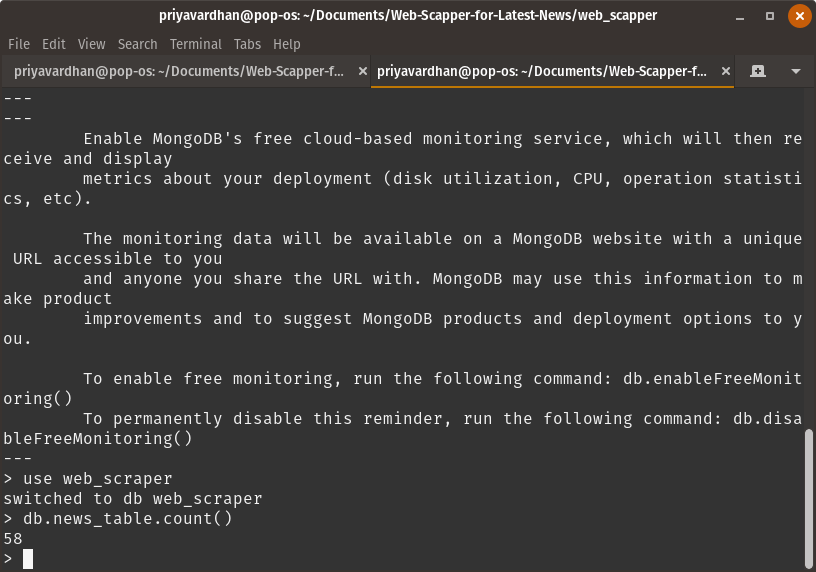
for i in news:

x = collection.find\_one({'Headline' : i['Headline']})

if x is None:

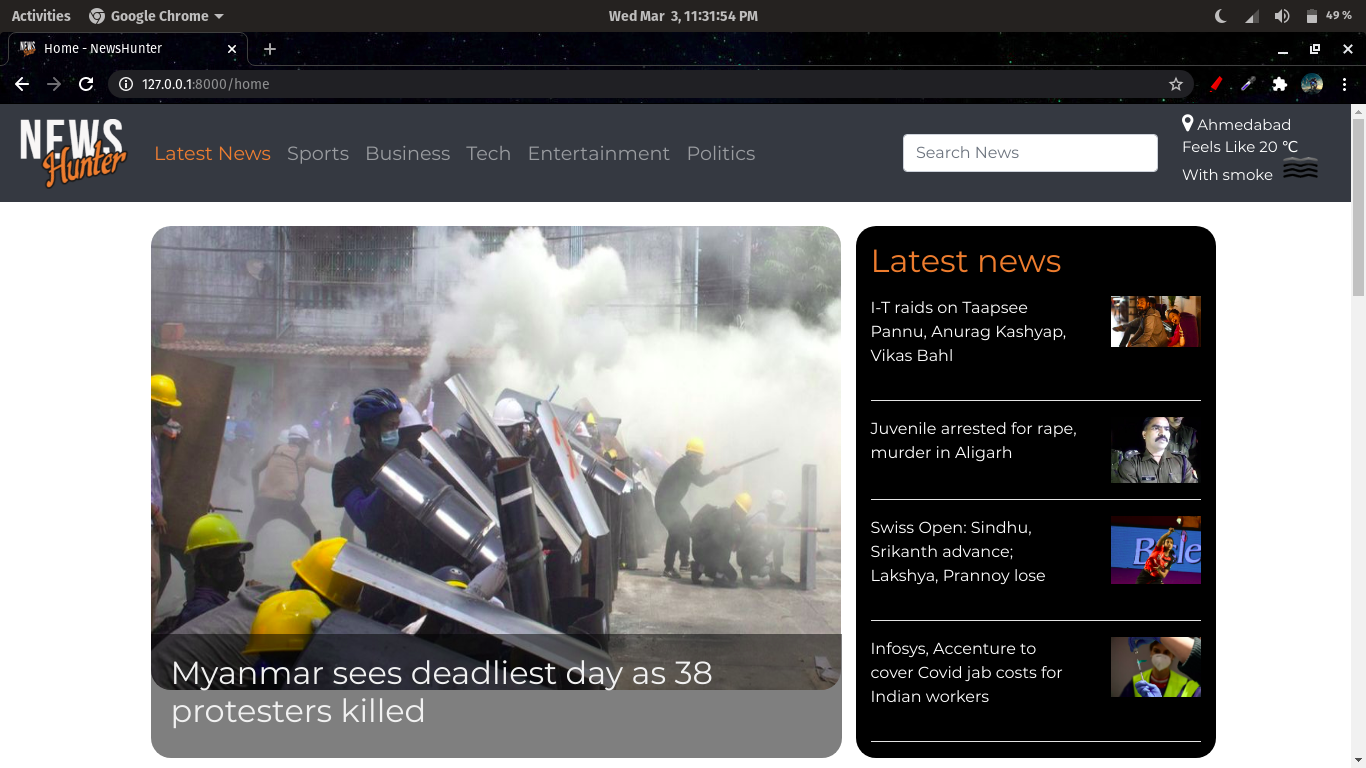
collection.insert\_one( i )

Next, connection is established with mongodb and the list of news is stored in the database.



* **Updating news to database**

Also, on every run of the python script, 58 news are fetched from the websites and the unique news are only added in the database to avoid redundancy.

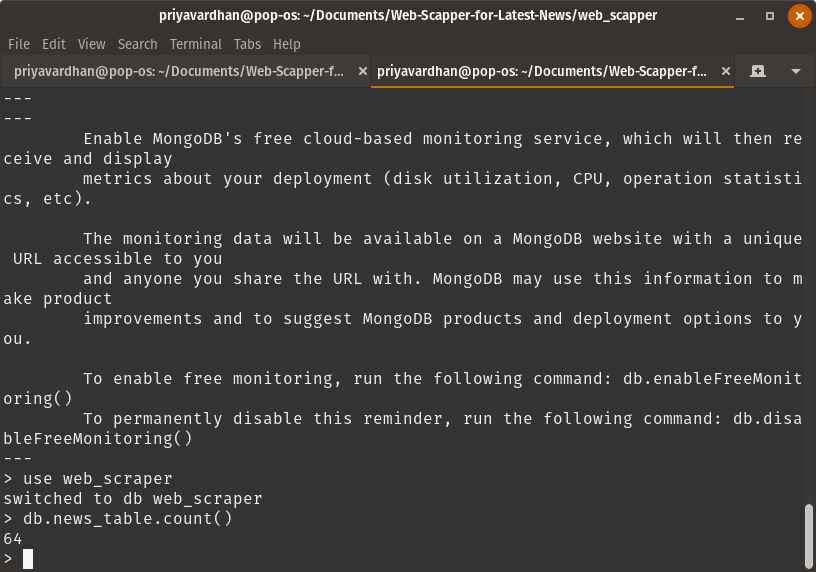


* **All the news older than 36 hours will be deleted from the database**

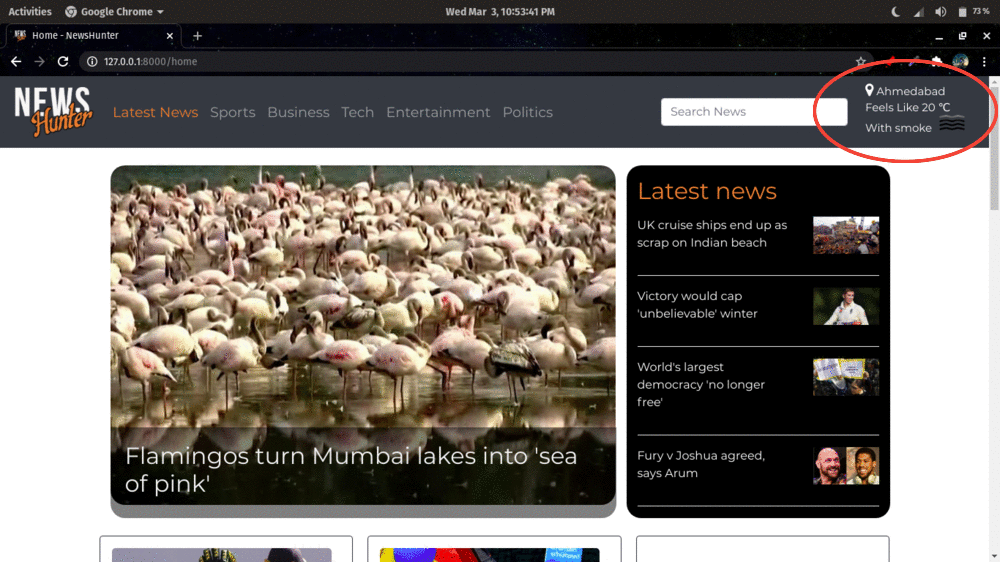
# Delete news with duration more than 36 hrs(129600 seconds)

if duration>129600:

mycol.delete\_one({'Headline' : i['Headline']})

****

1. **Show weather report**

****

On the homepage, a weather report is shown using the OpenWeatherMap api which takes the location as input and gives the temperature(in Kelvin), humidity, weather and icon which is displayed in the top right corner of the webpage.

**Code snippet :**

import requests

response = requests.get('http://api.openweathermap.org/data/2.5/weather?q='+city+'&APPID=46c58e3eded12aaeb86c8287b19e4de5')

weather\_report = {}

if response.status\_code == 200:

data = response.json()

main = data['main']

weather\_report['city'] = city

weather\_report['temp'] = int(main['feels\_like'] - 273.15)

weather\_report['humidity'] = main['humidity']

weather\_report['weather'] = data['weather'][0]['description']

weather\_report['icon'] = data['weather'][0]['icon']

return weather\_report

1. **Predicting category of the news**

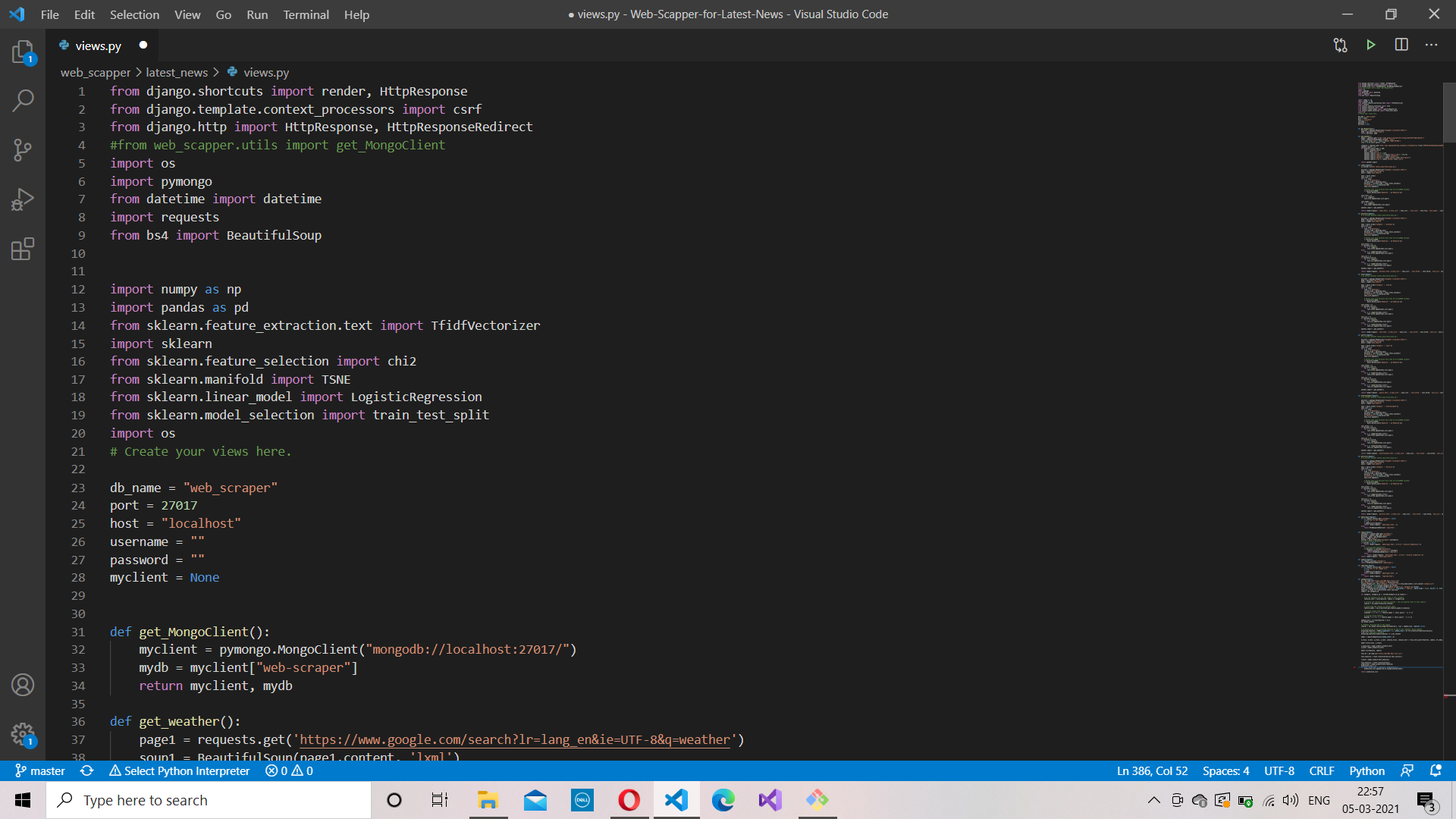
For predicting the category , a multiclass classification model is trained with the following details :

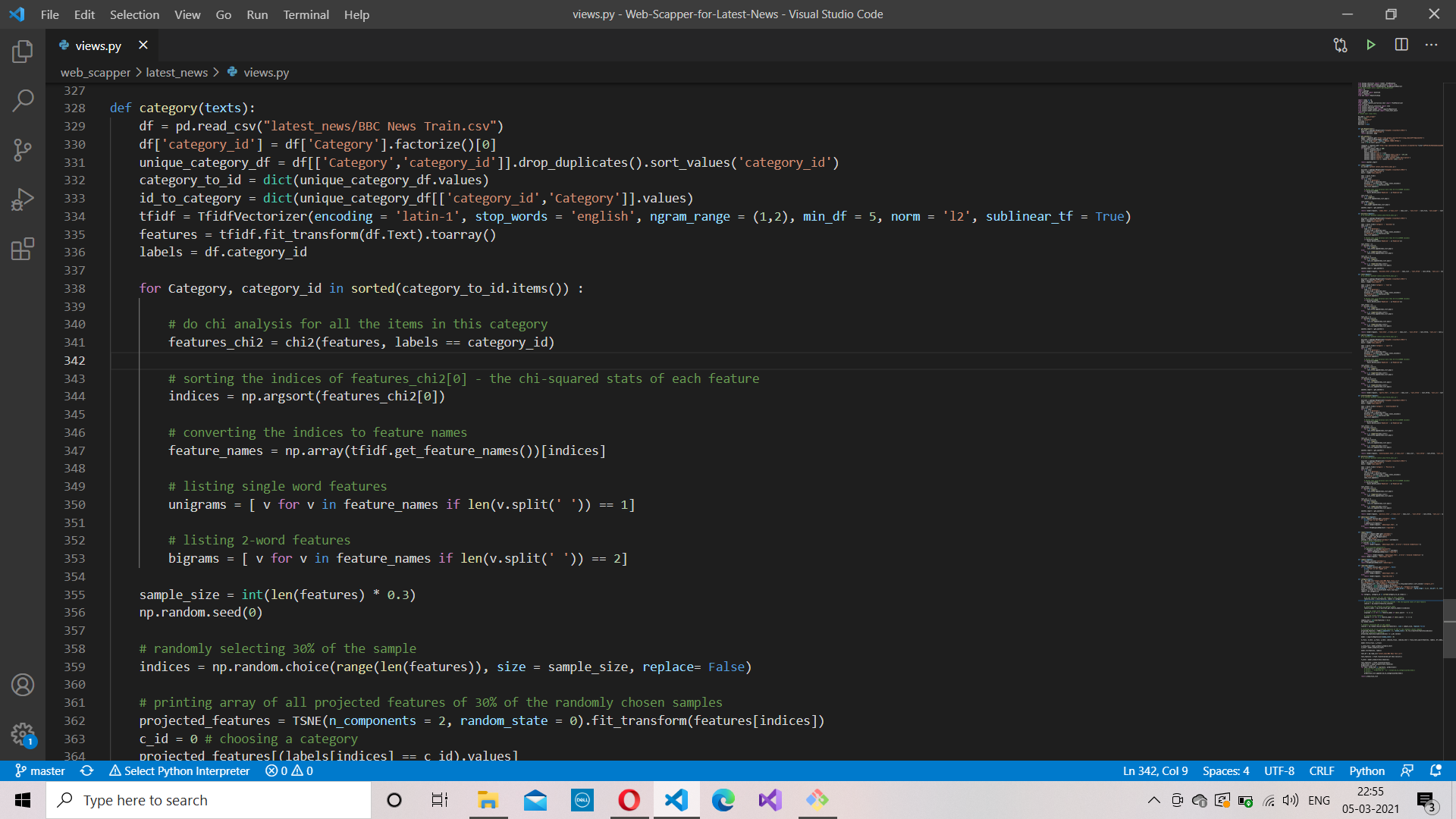
1. Algorithm : Logistic Regression
2. Tokenizer : TfdifVectorizer
3. Dataset : BBC news dataset

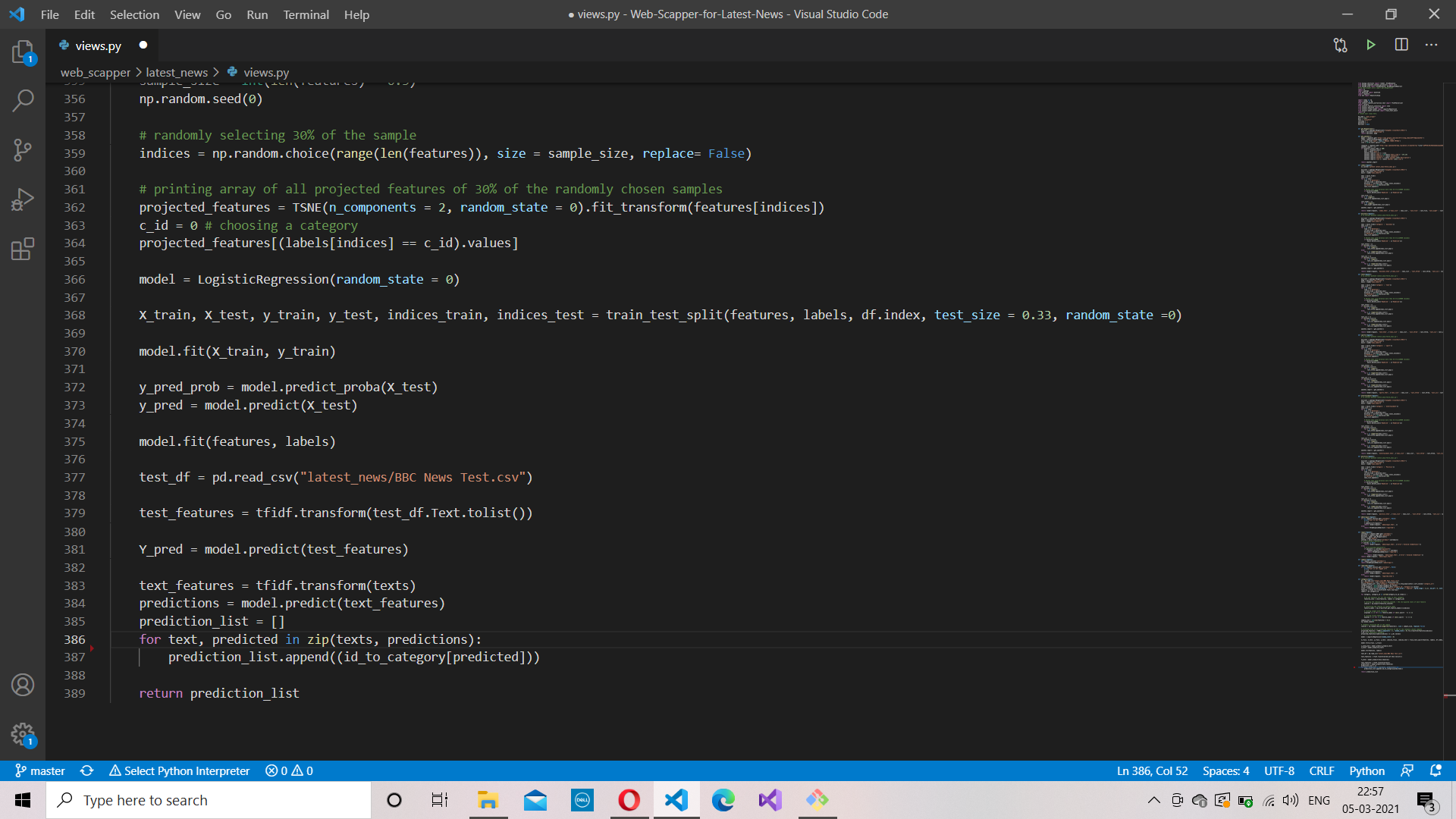
Now as far as the integration of the model is concerned, a method called category is present in the views.py which takes the list of headlines as the input and gives the list of predicted categories for the headlines.

Now ,this category() method is called in the fetch\_news.py file where the headlines are passed as an argument.

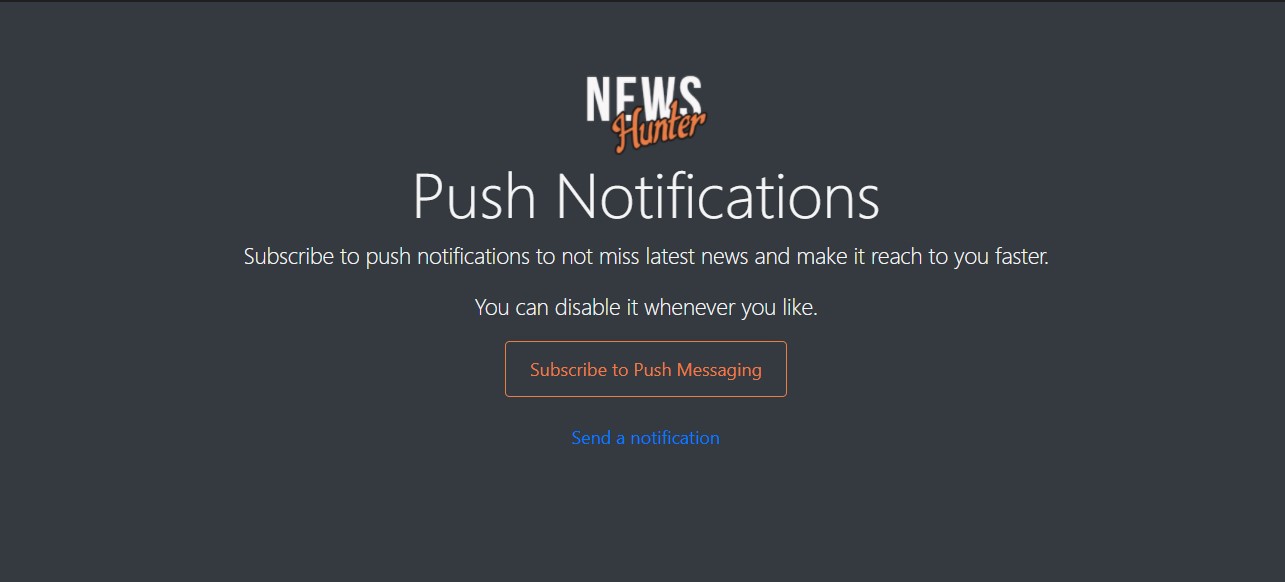
**Code Snippet :**

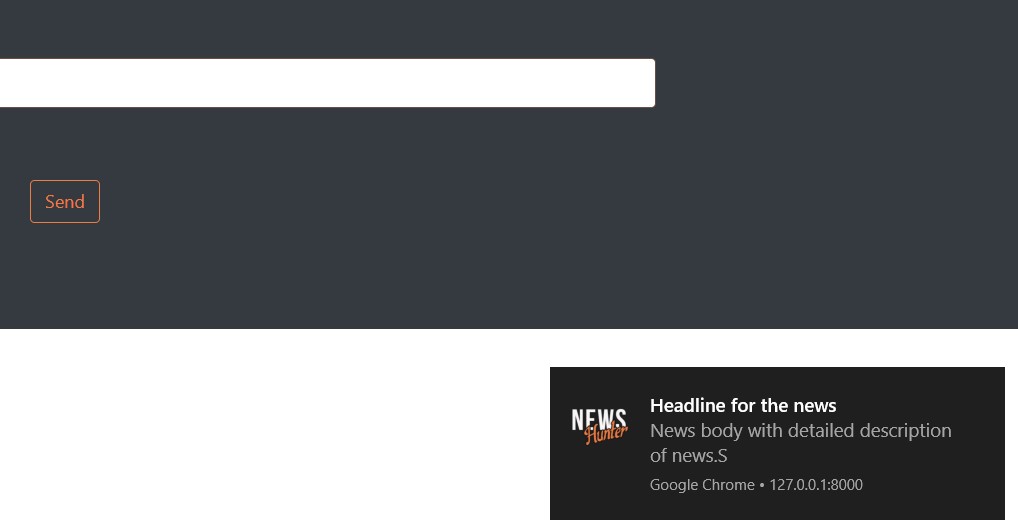
****

****

****

**4. Sending Push Notifications**

****

****

**Code Snippet:**

@require\_POST

def send\_notif(request):

head = request.POST.get('head')

body = request.POST.get('body')

send\_pn(head, body)

return HttpResponseRedirect('/demo\_notif')

#Sending push notification

def send\_pn(head, body):

payload = {"head": head, "body": body, "icon": "/static/images/notification.png", "url": "http://127.0.0.1:8000/home"}

try:

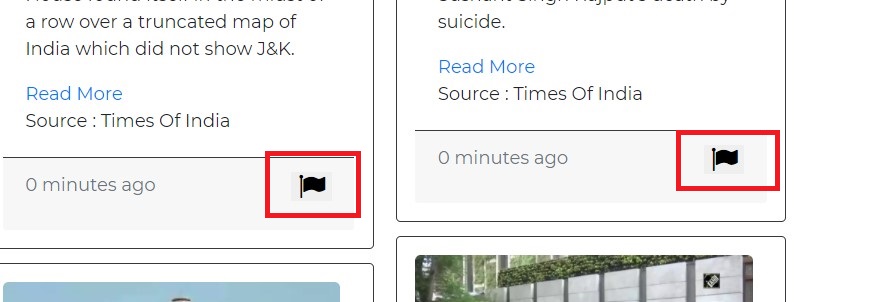
send\_group\_notification(group\_name="all", payload=payload, ttl=1000)

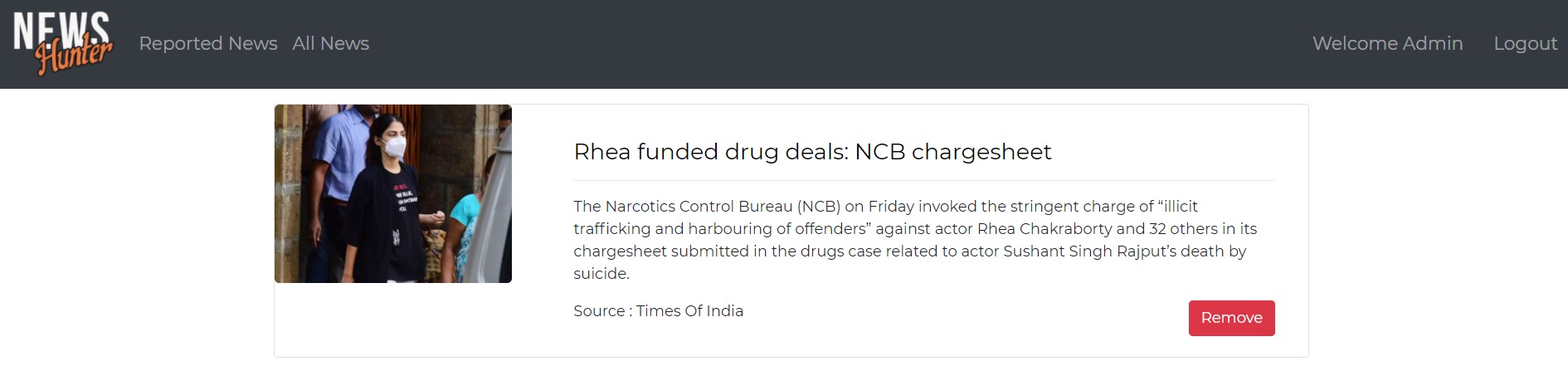
except Exception as e:

print(e)

* It uses “django-webpush” package for sending push notification
* send\_pn() method forms a payload of header, body, icon and url of the notification.
* It uses send\_group\_notification() method of “webpush” library to send notification to users of group “all” with payload and ttl is time for which this notification will be stored by web push server.

**5. Report a news**

****

****

**Code Snippet:**

@require\_POST

def report(request):

if not request.session.get('useremail', None):

print("user is not logged in")

c = {}

c.update(csrf(request))

return render(request, 'adminlogin.html', c)

# Updating 'reported' field of news in database

myclient, mydb = get\_MongoClient()

mycol = mydb["news\_table"]

id = request.POST.get('newsid')

myquery = { "\_id": ObjectId(id)}

newvalues = { "$set": { "Reported": True } }

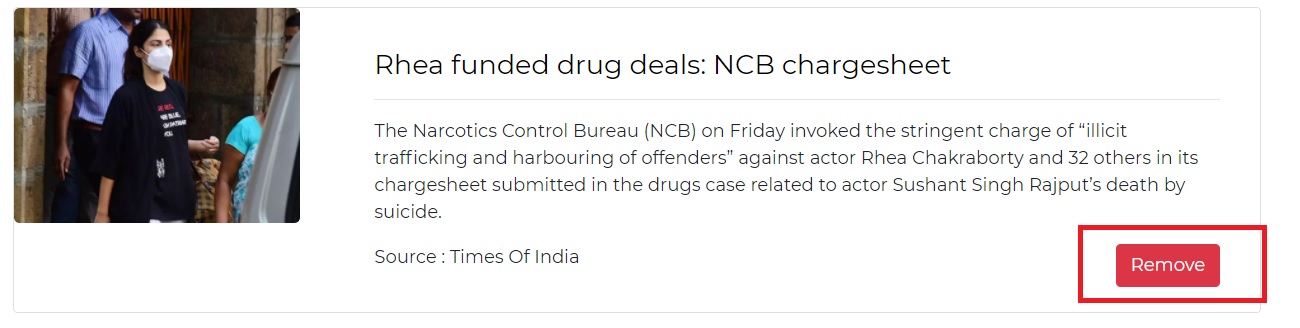
mycol.update\_one(myquery, newvalues)

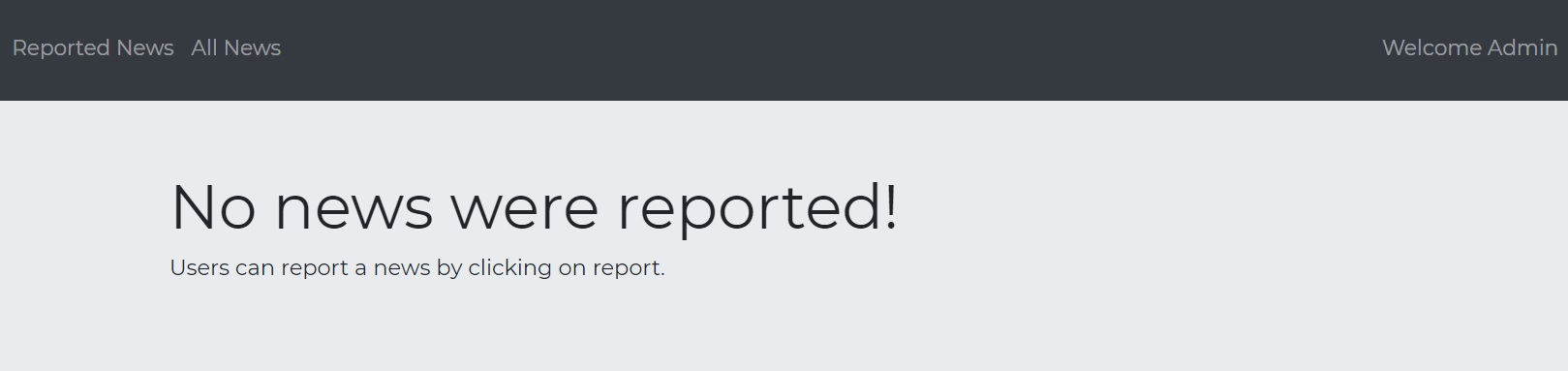
print("\n-------News Id : ",id, "reported successfully!--------\n")

return HttpResponseRedirect("/home")

* User can report a news by clicking on ‘flag’ icon
* Admin will be able to see the reported news and can delete it.
* So, the report() method will get id of the news from the template and will update the news with “Reported” field equals to true.
* All news having “Reported” true will be shown to the admin.

**6. Deletion of news by admin**

****

****

**Code Snippet:**

@require\_POST

def delete\_news(request):

if not request.session.get('useremail', None):

print("user is not logged in")

c = {}

c.update(csrf(request))

return render(request, 'adminlogin.html', c)

#Deleting news from database

myclient, mydb = get\_MongoClient()

mycol = mydb["news\_table"]

id = request.POST.get('newsid')

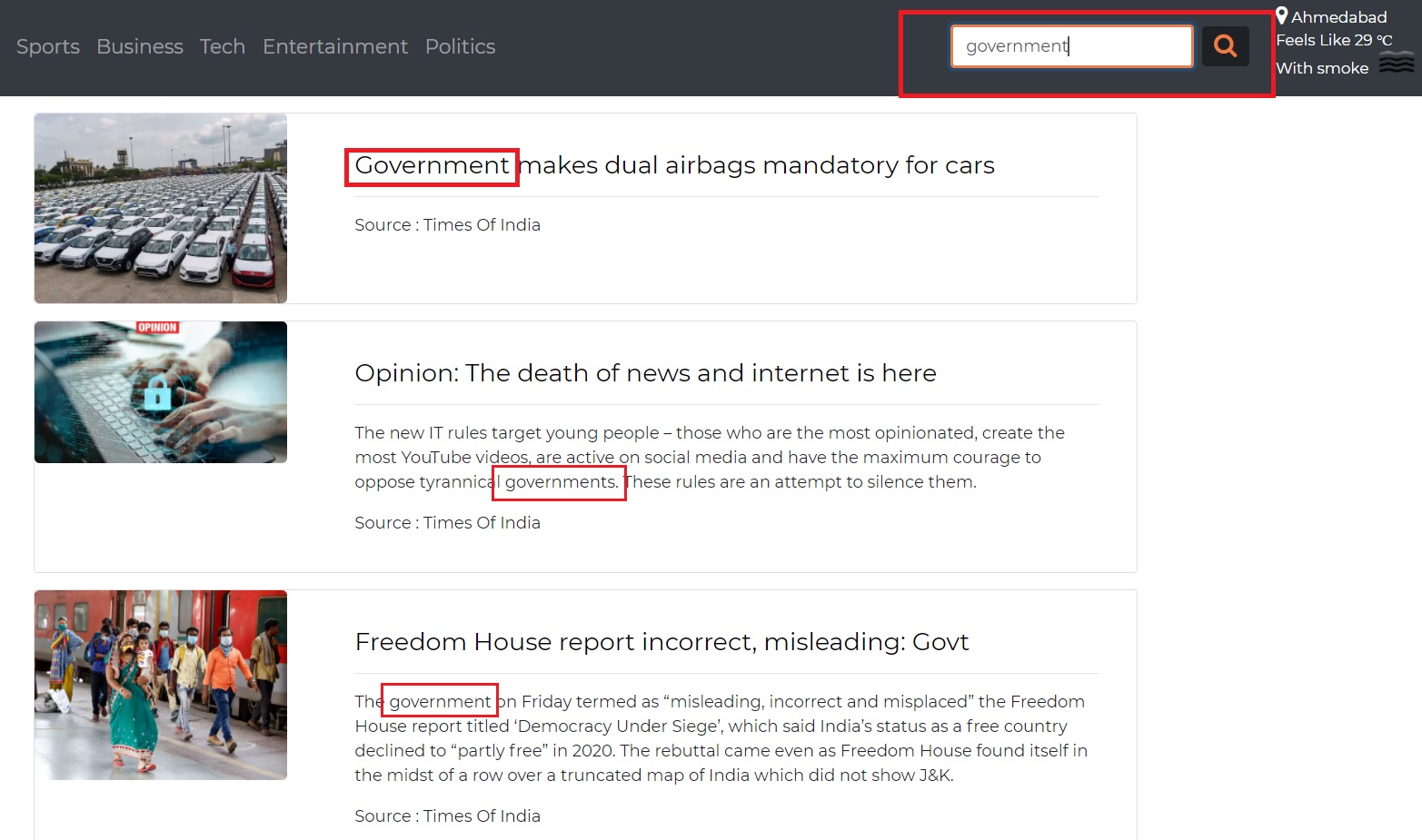
mycol.delete\_one({'\_id' : ObjectId(id)})

print("\n-------News Id : ",id, "deleted successfully!--------\n")

return HttpResponseRedirect('/adminhome')

* The admin can delete news by clicking on ‘Remove’ button.
* The delete\_news() method gets the id of the news and deletes it from the database

**7. Searching news**

****

**Code Snippet:**

# Searching News

@require\_POST

def search(request):

# Getting Search query

query = request.POST.get('squery')

# For null query

if query is None:

return HttpResponseRedirect("/home")

myclient, mydb = get\_MongoClient()

mycol = mydb["news\_table"]

# Using MongoDB's Text index for searching on Headline, Content, Category and Source fields

news = mycol.find({"$text" : {"$search" : query}})

news\_list = []

for i in news:

news\_list.append(i)

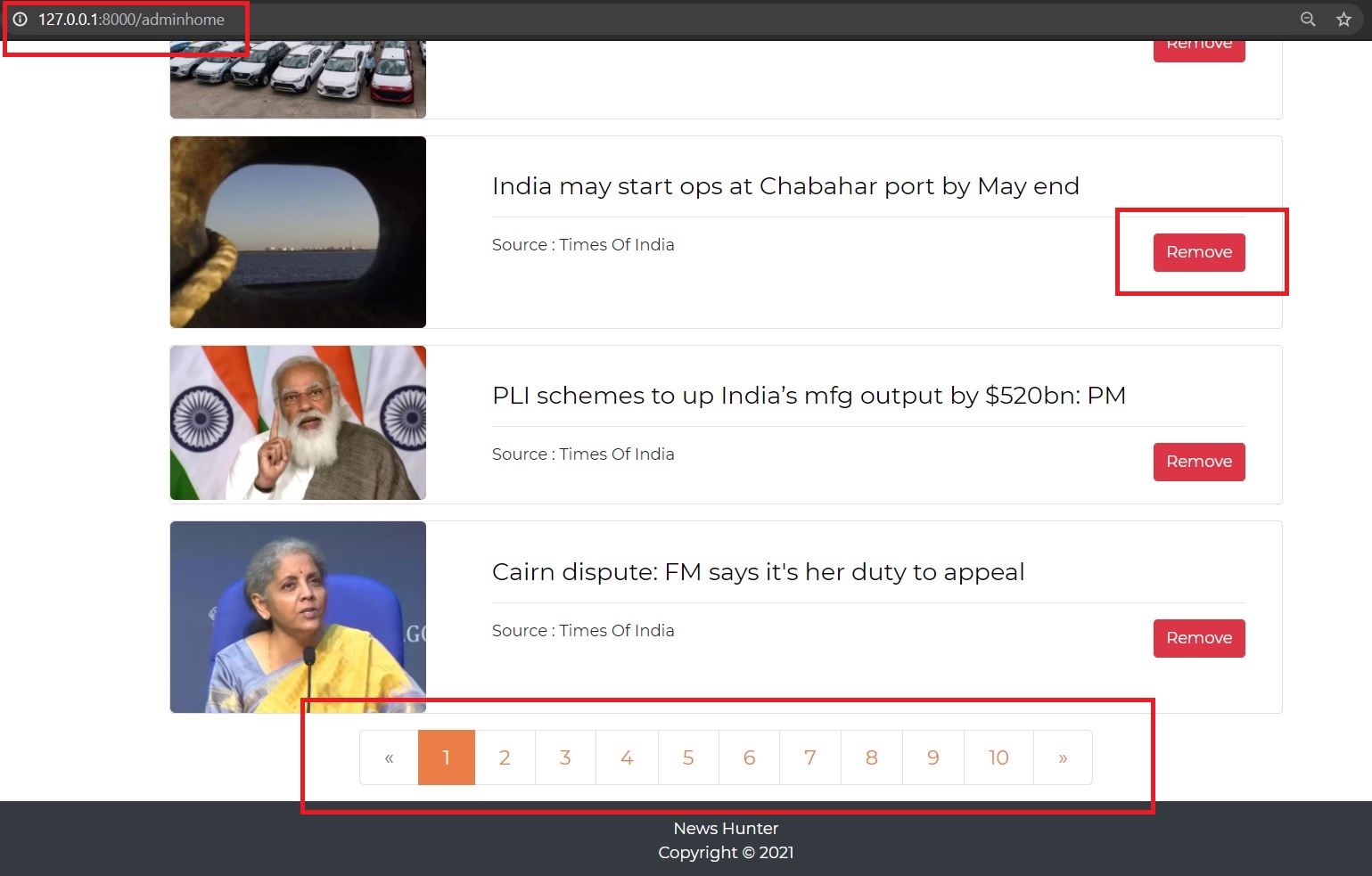
nl\_len = len(news\_list)

weather\_report = get\_weather()

return render(request, 'search.html', {"news" : news\_list, "nl\_len" : nl\_len, 'weather\_report' : weather\_report})

* Here, text index of table ‘news\_table’ in MongoDB is used for searching a text.
* The search() method gets the query string from the template and fires search query with that string.
* It returns documents having that string.

**8. Admin can view all news**

****

**Code Snippet:**

# All news page for Admin

def adminhome(request):

if not request.session.get('useremail', None):

print("user is not logged in")

c = {}

c.update(csrf(request))

return render(request, 'adminlogin.html', c)

c = {}

c.update(csrf(request))

#Getting news list from database

myclient, mydb = get\_MongoClient()

mycol = mydb["news\_table"]

news = mycol.find()

news\_list = []

for i in news:

time = i['DateTime']

current\_time = datetime.now()

#Setting the duration

duration = (current\_time - time).total\_seconds()

i['DateTime'] = int(duration//60)

i['id'] = i['\_id']

news\_list.append(i)

page = request.GET.get('page', 1)

#Django Pagination

paginator = Paginator(news\_list, 10)

try:

news = paginator.page(page)

#For first page

except PageNotAnInteger:

news = paginator.page(1)

#For last page

except EmptyPage:

news = paginator.page(paginator.num\_pages)

return render(request, 'adminhome.html', {"news" : news})

* Admin can view all news and can remove them
* The adminhome() view returns the news list from the database.
* Pagination provided by django is used to paginate the news list.