# **Dharmsinh Desai University, Nadiad**

# **Faculty of Technology,**

# **Department of Computer Engineering**

# **B.Tech. CE Semester – VI**

# **Subject: System Design Practice**

# **Project title: News Hunter - Web scraper for latest news**

**By :**

**Aastha Patel, roll no: CE-001**

**Priyavardhan Panchal, roll no: CE-086**

**Mauneel Sorathia , roll no : CE-133**

**Guided by :**

**Prof. Jigar M Pandya**

**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**



CERTIFICATE

This is to certify that the project entitled “**News Hunter - Web Scraper For Latest News**” is a bonafide report of the work carried out by

1) **Aastha D Patel,** Student ID No: **18CEUON120**

2) **Priyavardhan K Panchal,** Student ID No: **18CEUON016**

3) **Mauneel D Sorathia,** Student ID No: **18CEUBS065**

of Department of Computer Engineering, semester VI, under the guidance and supervision for the subject System Design Practice. They were involved in Project training during the academic year 2020-2021.

|  |  |
| --- | --- |
| Prof. Jigar M Pandya  Assistant Professor  Department of Computer Engineering,  Faculty of Technology,  Dharmsinh Desai University, Nadiad  Date: | Dr. C.K Bhensdadia  Prof. and Head,  Department of Computer Engineering,  Faculty of Technology,  Dharmsinh Desai University, Nadiad  Date: |

**Table of Content**

|  |  |  |
| --- | --- | --- |
| Sr No. | Topic | Page no. |
| 1 | Abstract | 4 |
| 2 | Technologies and Tools used | 6 |
| 3 | Software Requirements Specification | 8 |
| 4 | Modules and Work Division | 15 |
| 5 | System Design | 17 |
| 6 | Implementation Details | 20 |
| 7 | Testing | 44 |
| 8 | Challenges faced | 48 |
| 9 | Conclusion | 49 |
| 10 | Limitations and Future Extensions | 52 |
| 11 | References / Bibliography | 51 |

**Abstract**

Our project title is “News Hunter - Web Scraper for latest news”. Before discussing the functionalities of our project, let’s first understand what web scraping is.

Web Scraping is the process of collecting structured web data in an automated fashion. It’s also called web data extraction. In general, web data extraction is used by people and businesses who want to make use of the vast amount of publicly available web data to make smarter decisions.

The basics of web scraping including working in 2 parts : a web crawler and a web scraper.The web crawler is the horse, and the scraper is the chariot.

A web crawler, which we generally call a “spider”, is an artificial intelligence that browses the internet to index and search for content by following links and exploring. Here we first “crawl” the web or one specific website to discover URLs which are then passed on to the scraper.

A web scraper is a specialized tool designed to accurately and quickly extract data from a web page. Web scrapers vary widely in design and complexity, depending on the project. An important part of every scraper is the data locators (or selectors) that are used to find the data that you want to extract from the HTML file - usually xpath, css selectors, regex or a combination of them is applied.

Basically our project is one of the applications of web scraping. We are using web scraping to extract the latest news articles from different websites and show them on our web application.

Here’s a little something about our web application:

* Our web application has no sign in restrictions. It’s open for all.
* The user has an option to categorically sort the articles and read articles according to that particular category.
* Also the user can search the articles by entering any particular keyword and the articles including that keyword will be displayed.
* If the user is kind of busy and just wants to get the highlights of the past 6/12 hours, then he/she also has access to that. Past 6/12 hours news headlines will be displayed to the user and the user can click the ‘read more’ link present below every article if he/she wishes to read the whole article in detail.
* Our news feed will get updated every 20 mins.
* Our web application also shows the weather forecast and keeps it updated.
* Users can report any inappropriate news on the home page and the admin will manage it.
* Users can get push notifications of the latest news.

**Technologies and Tools Used**

Frontend : HTML5, CSS3, Bootstrap, JavaScript

Framework : Django

Database used : MongoDB (NoSQL)

**HTML** : Hyper Text Markup Language is used to create the main structure of a webpage

**CSS** : Cascading Style Sheets is used to define styles of HTML.

**Bootstrap** : It is a free and open source front end library for designing web applications and making it responsive.

**Django** : Django is a Python-based free and open-source web framework that follows the model-template-views architectural pattern.

**MongoDB** : It is a No-SQL database and uses Javascript Object Notation-like documents with schemas

**Libraries Used**

**Beautiful Soup** : It is a Python library for pulling data out of HTML and XML files. It provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree.

## **Scikit-Learn** : It is the most useful and robust library for machine learning . It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python.

**Keras** : Keras is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software](https://en.wikipedia.org/wiki/AI_software) library that provides a [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) interface for [artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network). Keras acts as an interface for the [TensorFlow](https://en.wikipedia.org/wiki/TensorFlow) library.

**Pymongo:** It allows interaction with the [MongoDB](https://www.educative.io/edpresso/what-is-mongodb) database through Python. It is a native Python driver for MongoDB.

**Profanity Filter :** profanity-filter is a Python library for detecting and filtering profanity in the provided text.

**Testing Module**

Django’s ‘unittest’ library module will be used to simulate requests, insert test data, inspect your application’s output and generally to verify the code.

The test cases are to be written in test.py.

**APIs Used**

**OpenWeatherMap API :** OpenWeatherMap is a service that provides weather data, including current weather data, forecasts, and historical data to the developers of web services in JSON format.

**Platforms**

**Version Control Management** : Version controlling helps us to manage the updated versions of the files in a software which is developed by multiple programmers. It is also helpful for peer review by other team members. Github will be used in this web application for version control.

**Software Requirements Specification**

# **Introduction**

## **1.1 Purpose**

A web scraper is a specialized tool designed to accurately and quickly extract data from a web page. Web scrapers vary widely in design and complexity, depending on the project. An important part of every scraper is the data locators (or selectors) that are used to find the data that you want to extract from the HTML file - usually xpath, css selectors, regex or a combination of them is applied. Our project is one of the applications of web scraping. We are using web scraping to extract the latest news articles from different websites and show them on our web application.

## **1.2 Intended Audience and Reading Suggestions**

This document is intended to both the stakeholders as well as the developers of the system.

## **1.3 Product Scope**

This system is intended to:

* Display news by category
* Display news by time
* Allow user to search news by keyword
* Show weather report
* Update news feed every 20 minutes
* Helps user to go through the news without any interrupts of advertisements
* Report news
* Send push notifications to users

## **1.4 References**

The following material was used in creating this document:

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.

# **Overall Description**

## **2.1 Product Perspective**

This web application is an efficient way to keep the users updated with the latest news from various news websites at one destination without any disturbances.

## **2.2 Product Functions**

* Users can sort the articles categorically
* User can filter the news according to the keywords
* User can go through the highlights of past 6/12 hours
* The news feed will be updated every 20 minutes
* System also shows weather report
* System will filter out the advertisements
* User can report a news anonymously if he/she found it offensive
* User can get push notifications
* The web application is accessible to all

## **2.3 Design and Implementation Constraints**

* System is wirelessly networked with an encryption..
* Database is password protected.
* System should use less RAM and processing power.
* The system is accessible to all.

## **2.4 Assumptions and Dependencies**

* Internet connection is mandatory
* Browser in user’s system must support the frameworks

# **External Interface Requirements**

**3.1 User Interfaces**

* Homepage to display the headlines and a short description for each
* A section for weather reports.
* Users can choose the category of the news.
* Users can search news based on a keyword
* Page to display news based on its time duration
* Page to manage user’s push notification subscription

**3.2 Software Interfaces**

Any latest web browser which supports modern technologies such as HTML5,CSS3.

**3.3 Hardware Interfaces**

* Any device on which we can run the web browser.
* The device should have enough RAM(4 GB), capable processor, and Internet connection to run the website smoothly.

**3.4 Communication Interfaces**

* Data sender and receiver should be connected via LAN or WAN in order to communicate.
* Website server and User can be connected through WAN.

1. **System Features**

**4.1 Manage News**

**4.1.1 Display News**

**Description** : Extracting news from news websites and display latest news to the user on the home page.

**Input** : No input

**Output** : Headlines with short descriptions and source of the news are

displayed.

**4.1.2 Update News**

**Description :** The news will be updated after every 20 min.

**Input :** No input

**Output :** Latest news are added on the home page

**4.1.3 Choose category**

**Description :** The user can choose the category of news to be displayed.

**Input :** User Selection

**Output :** The news of that category is displayed on another page.

**4.1.4 Show weather report**

**Description :** The weather report is displayed in a section of the home page.

**Input :** No input

**Output :** Temperature and the type of weather of the day are displayed.

**4.1.5 Filter News based on time**

**Description :** The user can filter out the news according to their convenience based on time.

**Input :** User can choose either 6 hours or 12 hours

**Output :** List of news fetched within last few hours is displayed.

**4.1.6 Search News**

**Description :** The user can filter news based on a keyword. It is a case-insensitive search. Users can search by category, keyword or source of the news.

**Input :** Keyword

**Output :** News related to that keyword is displayed.

**4.1.7 Filter out advertisements**

**Description :** The news is shown without any advertisements.

**Input :** No input.

**Output :** News is displayed free from advertisements.

**4.1.8 Report offensive news**

**Description :** If a user finds any offensive, he/she can report the news anonymously.

**Input :** User selection

**Output :** List of reported news are shown to administrator and the admin can decide whether to keep the news or remove it

**4.1.9 Send notification**

**Description :** When a user visits the web application, he/she is sent a push notification of the latest news. Users can click on the notification to see the news.

**Input :** No input

**Output :** Push notification on desktop

**4.1.10 Delete news**

**Description :** News which were fetched earlier than 36 hours are discarded from the database to avoid cluttering in the database

**Input :** No input

**Output :** News fetched before 36 hours are removed

**4.1.11 Filter news based on profanity**

**Description:** Censoring the headlines and content of the news has to be done to remove any offensive or inappropriate content.

**Input:** No input

**Output:** The news is censored.

**4.1.12 Predicting Category of the news**

**Description:** The category of the news fetched is predicted using Machine Learning.

**Input:** No user input

**Output:** News are categorized among categories: business, technology, sports, entertainment and politics.

**4.2 Manage Administrator**

**4.2.1 Admin login**

**Description :** Admin user can login and manage the news that are shown in the web application

**Input :** User email and password

**Output :** Redirect to admin dashboard

**4.2.2 View all news**

**Description :** Admin users can view all the news with pagination.

**Input :** User input

**Output :** Show all news on dashboard

**4.2.3 View reported news**

**Description :** Admin users can view the news which are reported by an end user anonymously in various pages.

**Input :** User input

**Output :** Show reported news on webpage

**4.2.4 Delete news by admin**

**Description :** Admin can view all the news and delete any is he/she wishes

**Input :** User input

**Output :** News removed from database

**4.2.5 Admin Logout**

**Input :** User input

**Output :** Redirect to login page

**Modules and Work Division**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Module Name** | **Owner** | **Start Date** | | **Completion Date** | | **Status** |
| **Planned** | **Actual** | **Planned** | **Actual** |
| 1 | Project Initialization on GitHub and adding collaborators | Repo owner : Priyavardhan  Collaborators: Aastha, Mauneel | 23/1/2021 | 15/1/2021 | 30/1/2021 | 28/1/2021 | Done |
| 2 | Web Scraping news and store in Database | Priyavardhan | 27/1/2021 | 27/1/2021 | 13/2/2021 | 3/2/2021 | Done |
| 3 | Displaying news on webpage | Priyavardhan | 27/1/2021 | 2/2/2021 | 13/2/2021 | 4/2/2021 | Done |
| 4 | Updating news | Priyavardhan | 27/1/2021 | 2/2/2021 | 13/2/2021 | 4/2/2021 | Done |
| 5 | Deleting old news | Priyavardhan | 27/1/2021 | 11/2/2021 | 13/2/2021 | 12/2/2021 | Done |
| 6 | Show weather report | Priyavardhan | 13/2/2021 | 18/2/2021 | 20/2/2021 | 18/2/2021 | Done |
| 7 | Populating Category webpages | Priyavardhan | 13/2/2021 | 14/2/2021 | 6/3/2021 | 5/3/2021 | Done |
| 8 | Developing classification model to divide news into categories | Aastha | 23/12/2020 | 26/12/2020 | 26/1/2021 | 10/2/2021 | Done |
| 9 | Integrating the classification model into the web application | Aastha | 16/2/2021 | 18/2/2021 | 6/3/2021 | 5/3/2021 | Done |
| 10 | User Interface, Logo Design and Favicon design | Mauneel | 22/1/2021 | 20/1/2021 | 23/1/2021 | 23/1/2021 | Done |
|  |  |  |  |  |  |  |  |
| 11 | Admin Module  - Admin login, logout, display of reported news and every news, deletion of news. | Mauneel | 27/1/2021 | 22/1/2021 | 6/2/2021 | 6/2/2021 | Done |
| 12 | Push Notifications | Mauneel | 8/2/2021 | 8/2/2021 | 13/2/2021 | 13/2/2021 | Done |
| 13 | Search Functionality | Mauneel | 15/2/2021 | 16/2/2021 | 20/2/2021 | 20/2/2021 | Done |
| 14 | Display news of Last 6hrs/12hrs | Mauneel | 22/2/2021 | 22/2/2021 | 25/2/2021 | 25/2/2021 | Done |

**System Design**

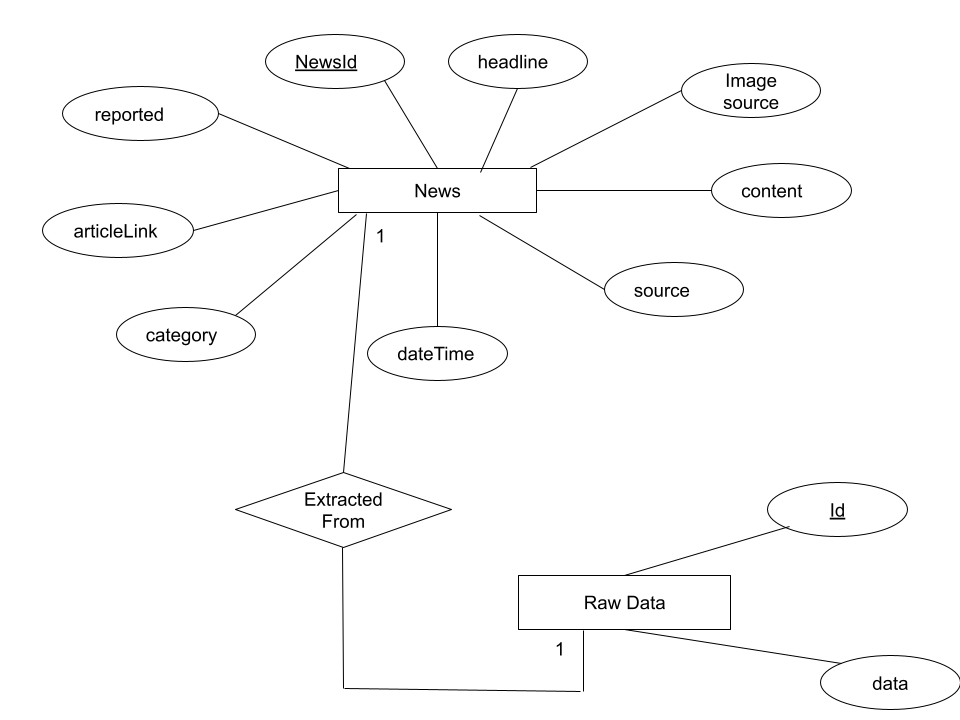
**3-Tier System Architecture**

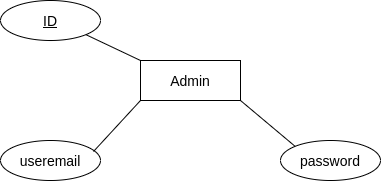
|  |
| --- |
| 1. **Presentation tier** |
| This layer manages the user interface of the  Application. The frameworks and languages used for the user interface are HTML, CSS, Bootstrap. The homepage will display the latest news and wait for user interaction. The user requests will be sent to the 2nd(logic) tier and it will be handled in the following layers. The generated response will be received and displayed to the user. |

|  |
| --- |
| 1. **Application/Middle tier** |
| This layer fetches and processes the data to be displayed on the webpage. It also handles user requests like searching or filtering news. This application uses python-based web framework - Django for interacting with the presentation layer and  data layer for data transfer and data storage respectively. |

|  |
| --- |
| 1. **Data tier** |
| This layer includes the data servers used for storing data  and data access layer which provides APIs to the application layer for managing the data with minimum dependencies. This application uses the NoSQL database - MongoDB for data storage in the form of JSON objects. |

**Entity Relationship Diagram:**

****

****

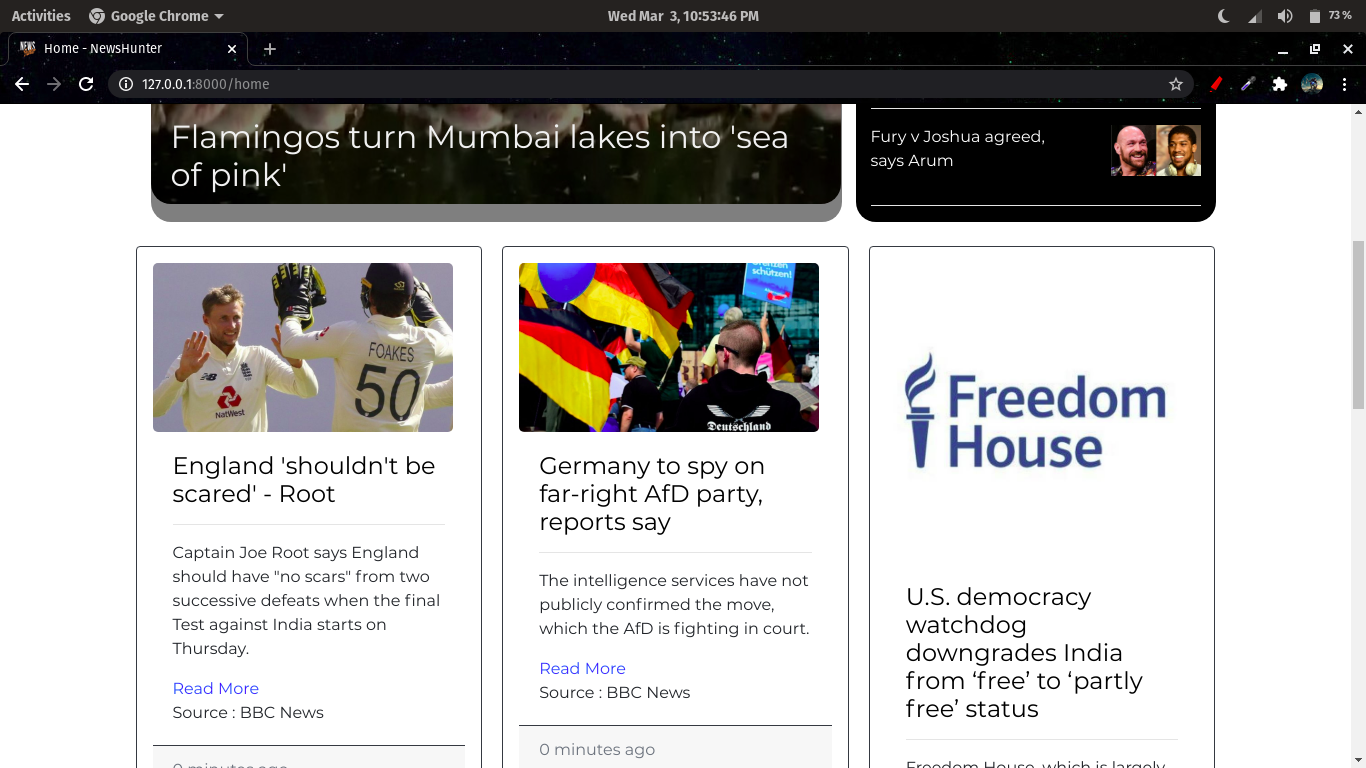
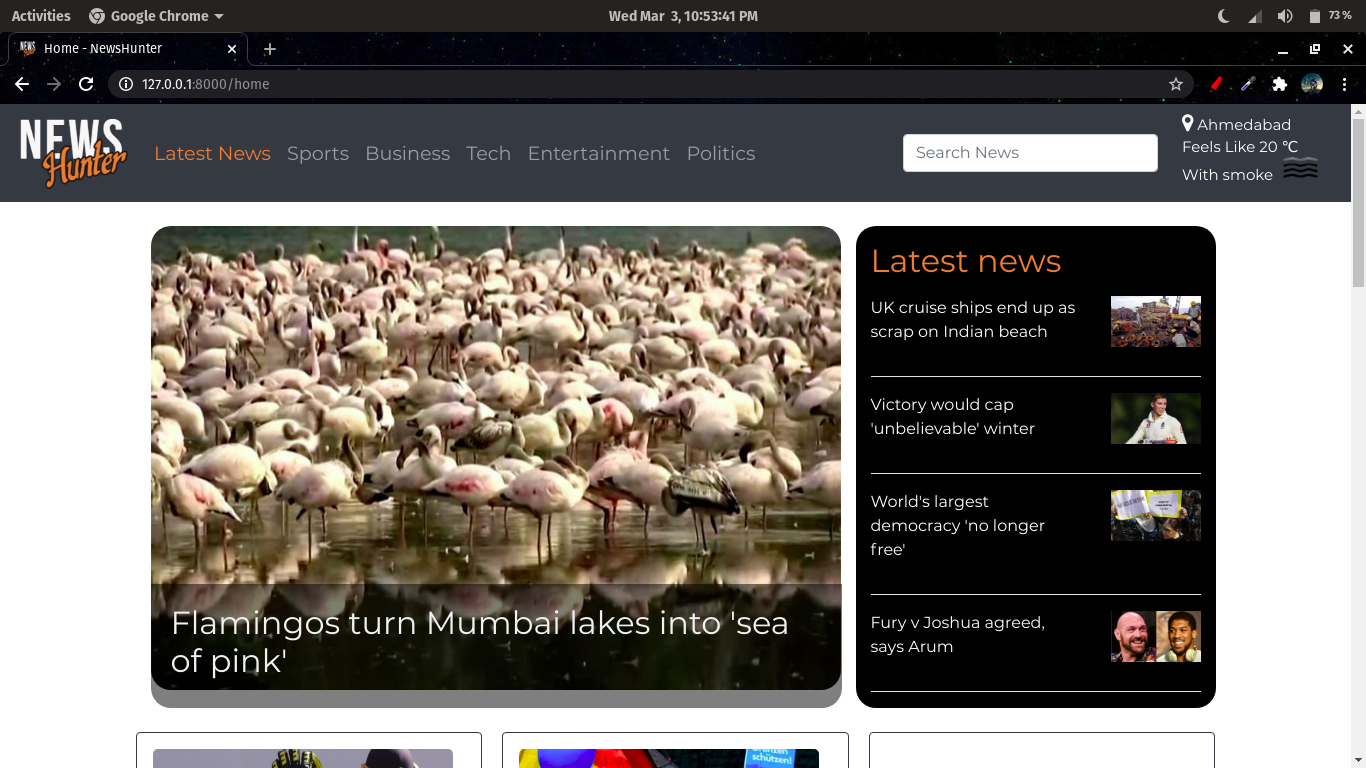
**Data Dictionary:**

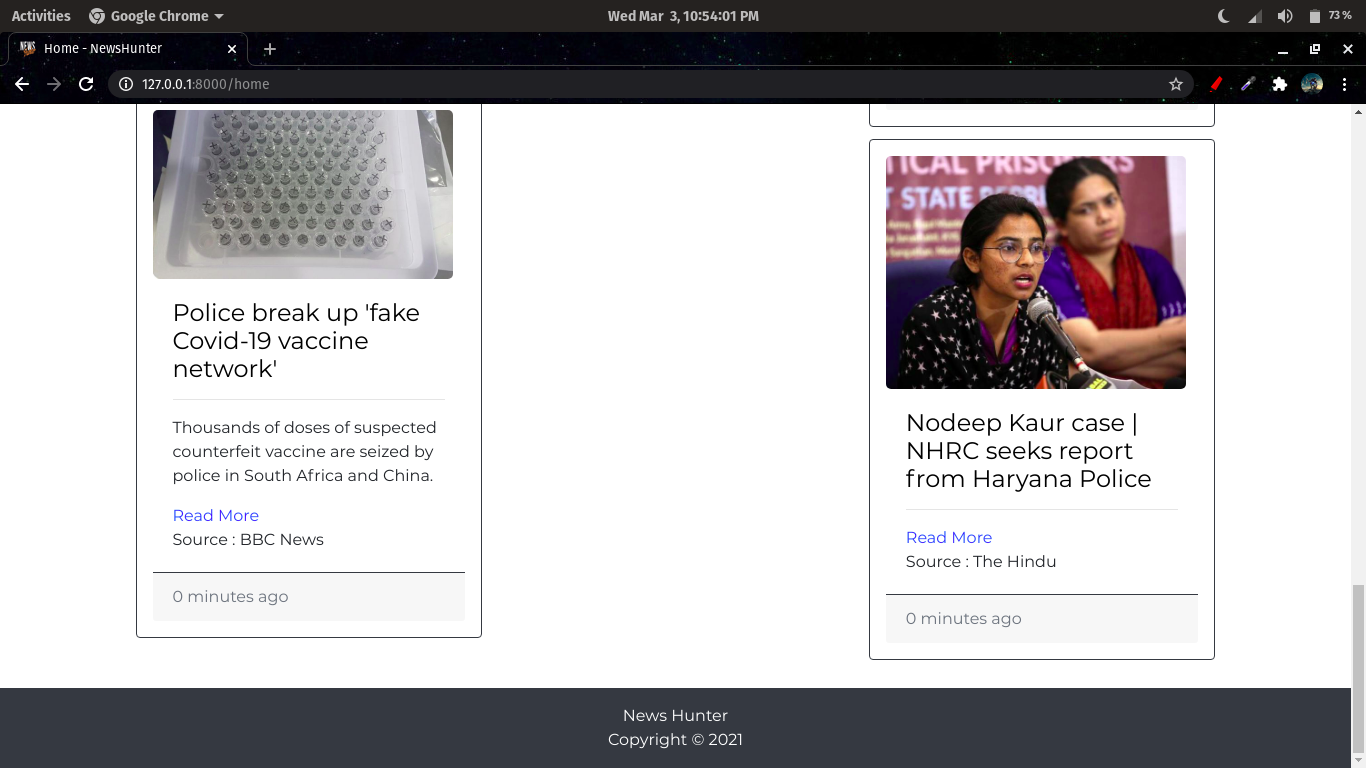
|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data type** | **Constraint** | **Description** |
|  |  |  |  |
| **For News Table** |  |  |  |
| NewsID | Number | Primary key | Identity |
| Headline | varchar | Not null | News Headline |
| Content | varchar | - | News Details |
| Image source | varchar | Not null | Image link |
| Source | varchar | Not null | Source of news |
| DateTime | DateTime | Not null | DateTime of news |
| Category | varchar | - | News Category |
| Article\_link | varchar | Not null | Website Link from which news is extracted |
| Reported | Boolean | - | Whether the news is reported by some user |
| **For RawData Table** |  |  |  |
| Id | Number | Primary Key | Identity |
| Data | varchar | Not null | Html content from which news is extracted |
|  |  |  |  |
| **For Admin Table** |  |  |  |
| Id | Number | Primary Key | Identity |
| useremail | varchar | Not null | Email of admin user |
| password | varchar | Not null | Password of admin user |

**Implementation Details**

**Screenshots and Code snippets**

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

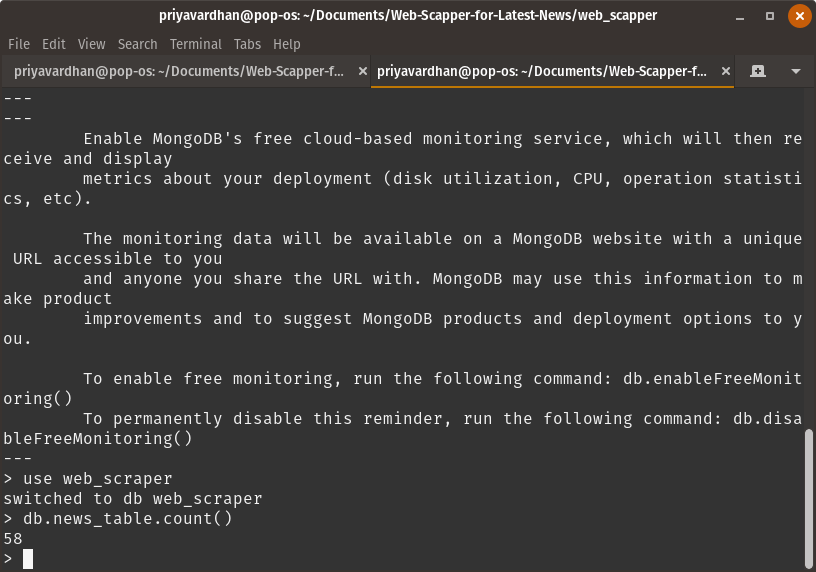
for i in news:

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

if x is None:

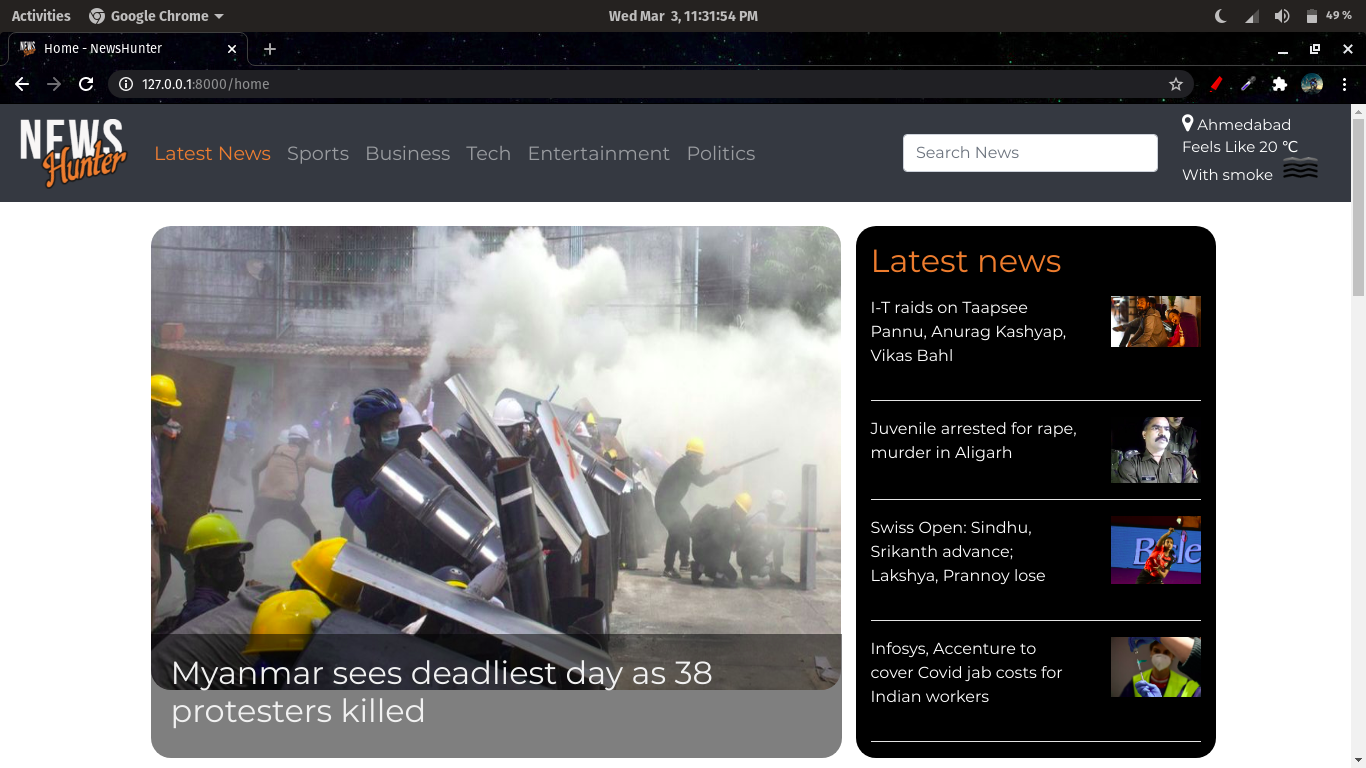
collection.insert\_one( i )

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



* **Updating news**

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.



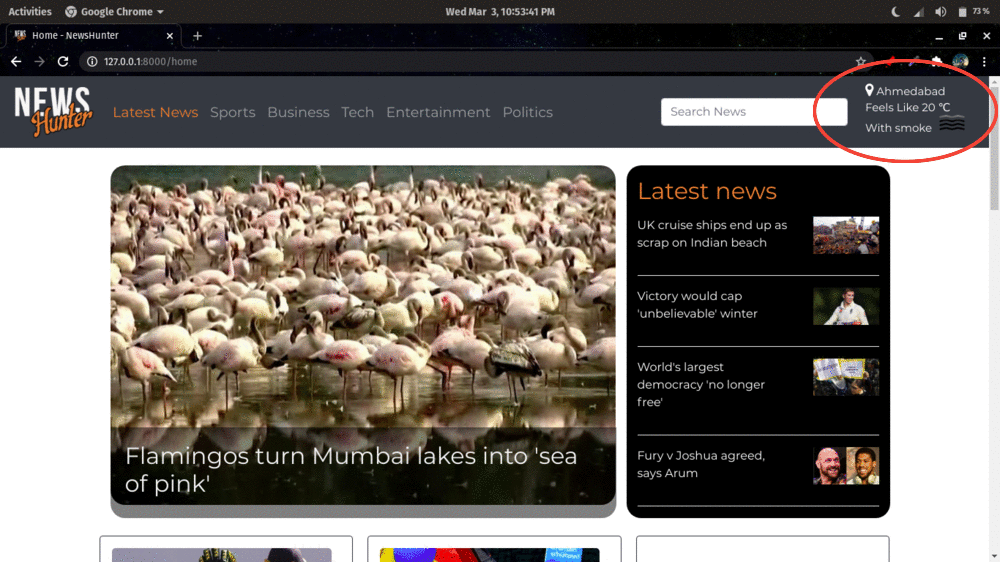
* **All the news older than 36 hours will be deleted from the database**

# 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)

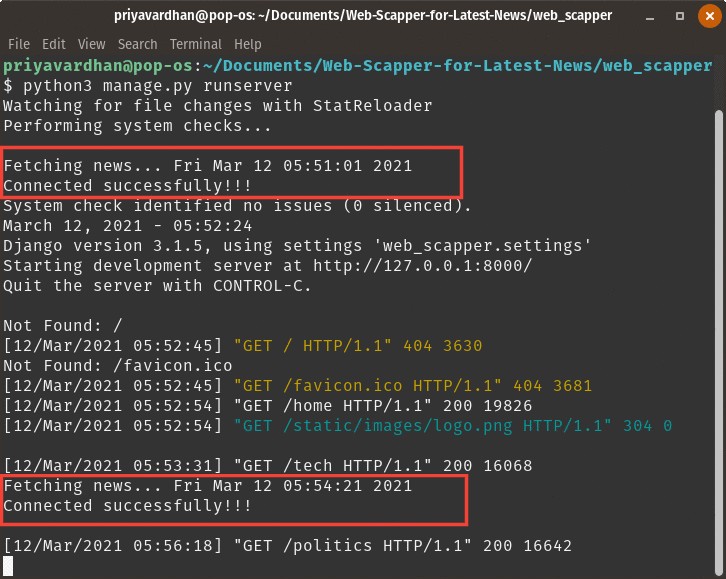
weather\_report['humidity'] = main['humidity']

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

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

return weather\_report

1. **Update news at regular interval of time**



* In django, some files like urls.py, settings.py are executed on startup only once, so to run the script for fetching news, the command is added in the urls.py file to run the script.
* To run the script at a regular interval of time, the concept of threading and timer in python is used which will create a thread and will run the script in background repeatedly after every ‘n’ seconds provided in the parameters. And the parent thread will continue to run the server.

**Code Snippet :**

import time

import threading

import os

def startup():

threading.Timer(200.0, startup).start()

print("Fetching news..." , time.ctime(time.time()))

os.system('python3 latest\_news/fetch\_news.py')

startup()

1. **Predicting the categories of the news according to the headlines**

* For this, we have trained a multi-class classification model using the Logistic Regression algorithm. The model takes news headlines as an input and gives the category as an output. Our model basically divides the news into 5 categories :

1. Politics
2. Business
3. Entertainment
4. Sports
5. Tech

* The training of the model is done as follows :

1. Tokenizing our dataset using Tf-idfVectorizer : It stands for Term Frequency Inverse Document Frequency. In TfidfVectorizer we consider the overall document weightage of a word. It helps us in dealing with most frequent words. Using it we can penalize them. TfidfVectorizer weights the word counts by a measure of how often they appear in the documents.
2. Preparing the features and labels matrix.
3. Fitting the Model.
4. Predicting our outputs using the test dataset.
5. Writing the code which gives us the list of predicted categories.

* Well, the whole code for model training is written in a function called category, which takes a list of headlines as an argument , and that list is used in the above code. Now the above code returns the predicted categories in the form of a list i.e. prediction\_list.
* This category() function is called in the fetch\_news.py file where scrapping work takes place and the output is stored in a variable and then the predicted category is stored in the database along with the news headline.

**Code Snippet :**

import numpy as np

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

import sklearn

from sklearn.feature\_selection import chi2

from sklearn.manifold import TSNE

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

import os

def category(texts):

df = pd.read\_csv("latest\_news/BBC News Train.csv")

df['category\_id'] = df['Category'].factorize()[0]

unique\_category\_df = df[['Category','category\_id']].drop\_duplicates().sort\_values('category\_id')

category\_to\_id = dict(unique\_category\_df.values)

id\_to\_category = dict(unique\_category\_df[['category\_id','Category']].values)

tfidf = TfidfVectorizer(encoding = 'latin-1', stop\_words = 'english', ngram\_range = (1,2), min\_df = 5, norm = 'l2', sublinear\_tf = True)

features = tfidf.fit\_transform(df.Text).toarray()

labels = df.category\_id

for Category, category\_id in sorted(category\_to\_id.items()) :

# do chi analysis for all the items in this category

features\_chi2 = chi2(features, labels == category\_id)

# sorting the indices of features\_chi2[0] - the chi-squared stats of each feature

indices = np.argsort(features\_chi2[0])

# converting the indices to feature names

feature\_names = np.array(tfidf.get\_feature\_names())[indices]

# listing single word features

unigrams = [ v for v in feature\_names if len(v.split(' ')) == 1]

# listing 2-word features

bigrams = [ v for v in feature\_names if len(v.split(' ')) == 2]

sample\_size = int(len(features) \* 0.3)

np.random.seed(0)

# randomly selecting 30% of the sample

indices = np.random.choice(range(len(features)), size = sample\_size, replace= False)

# printing array of all projected features of 30% of the randomly chosen samples

projected\_features = TSNE(n\_components = 2, random\_state = 0).fit\_transform(features[indices])

c\_id = 0 # choosing a category

projected\_features[(labels[indices] == c\_id).values]

model = LogisticRegression(random\_state = 0)

X\_train, X\_test, y\_train, y\_test, indices\_train, indices\_test = train\_test\_split(features, labels, df.index, test\_size = 0.33, random\_state =0)

model.fit(X\_train, y\_train)

y\_pred\_prob = model.predict\_proba(X\_test)

y\_pred = model.predict(X\_test)

model.fit(features, labels)

test\_df = pd.read\_csv("latest\_news/BBC News Test.csv")

test\_features = tfidf.transform(test\_df.Text.tolist())

Y\_pred = model.predict(test\_features)

text\_features = tfidf.transform(texts)

predictions = model.predict(text\_features)

prediction\_list = []

for text, predicted in zip(texts, predictions):

prediction\_list.append((id\_to\_category[predicted]))

return prediction\_list

1. **Profanity filter**

Profanity filters are used to censor out data by hiding or substituting abusive or sexual content with special characters. In this project, a profanity filter is used to censor the headlines and content of the news. Python provides us with a library for implementing the same. It takes the input as the news headline, description and substitutes (if any) abusive words with ‘\*’.

**Code Snippet :**

from profanity\_filter import ProfanityFilter

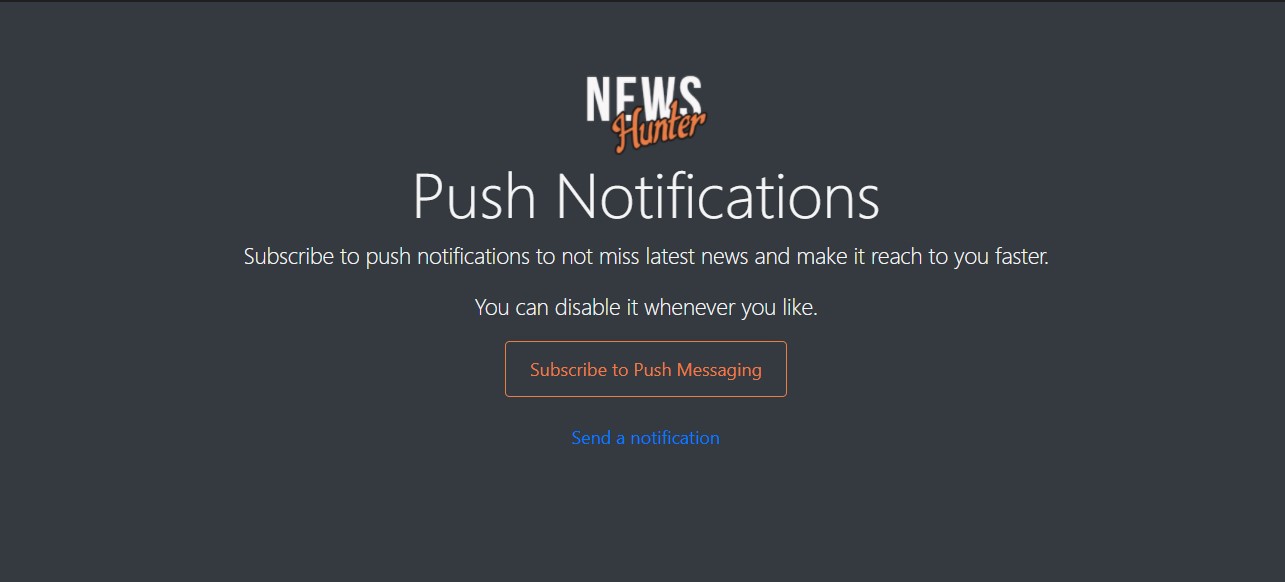
pf = ProfanityFilter()

"Headline" : pf.censor(headline)

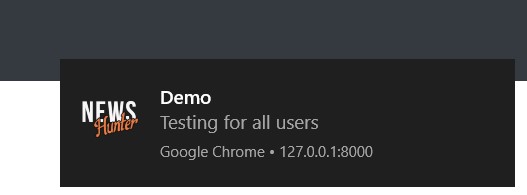
"Content" : pf.censor(content)

1. **Sending Push Notifications**

User can subscribe to notifications:

****

Push Notification:



**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:

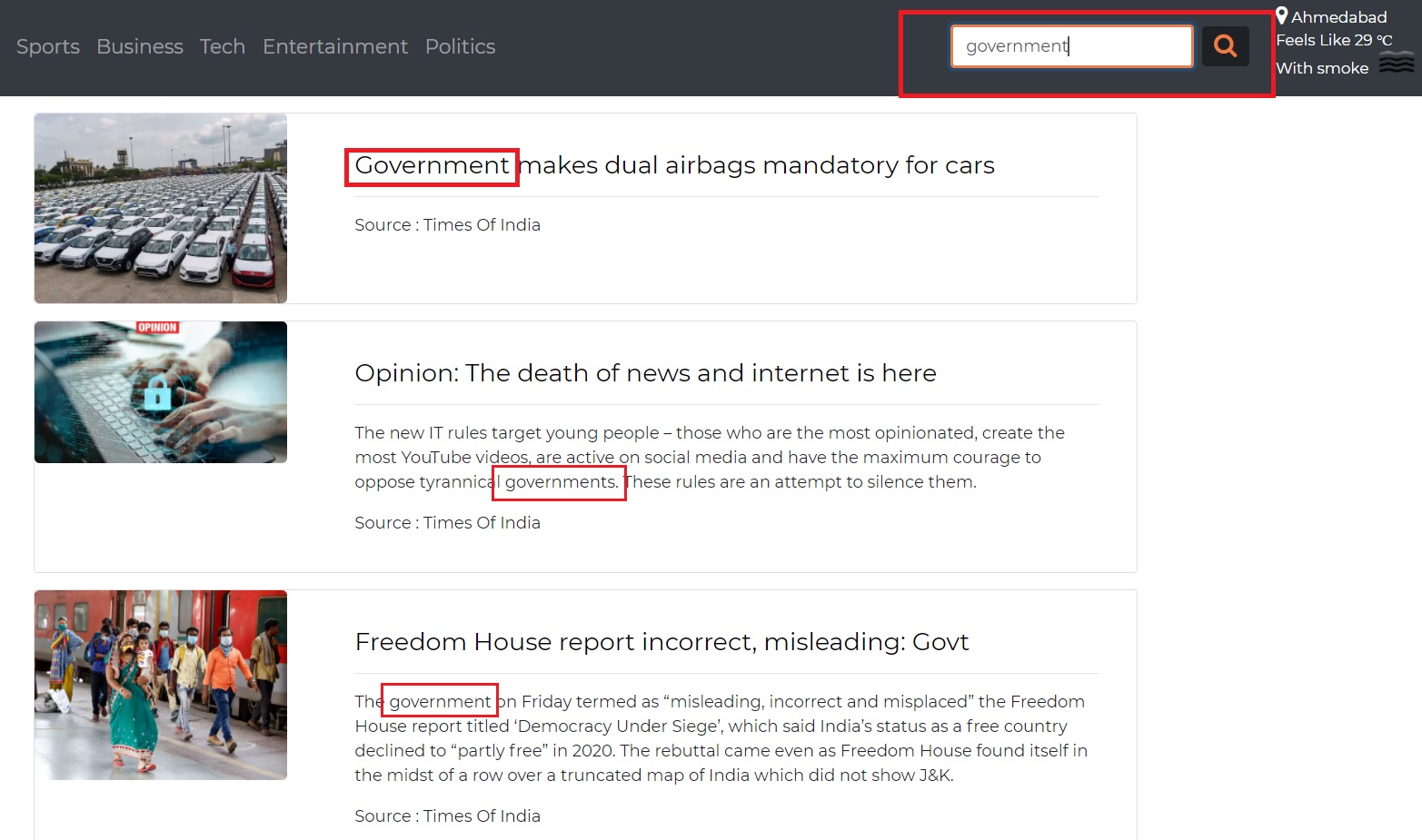
send\_group\_notification(group\_name="all", payload=payload, ttl=1000)

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.

1. **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)

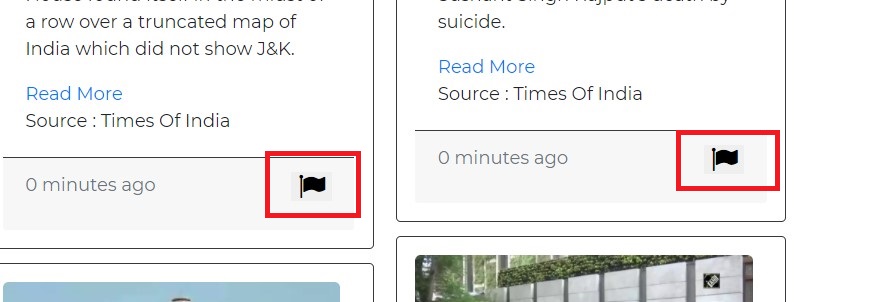
weather\_report = get\_weather()

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

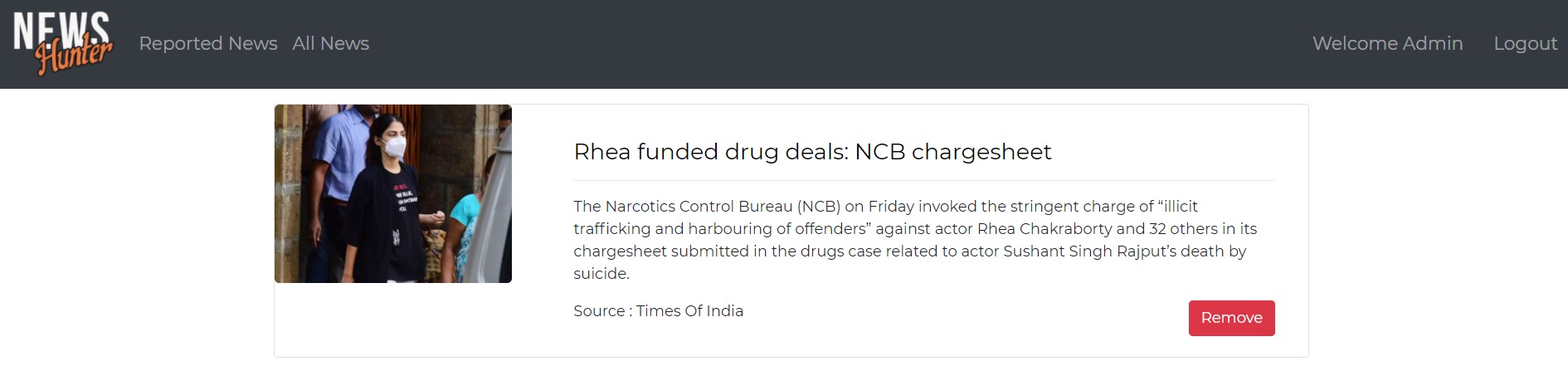
* Here, the text index of table ‘news\_table’ in MongoDB is used for searching a text.
* The search() method gets the query string from the template and fires search query with that string.
* It returns documents having that string.
* It is case-insensitive search and users can search by category, source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

1. **Report News**

UI for reporting a news:

****

News reported to Admin UI:



**Code Snippet:**

@require\_POST

def report(request):

# 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)

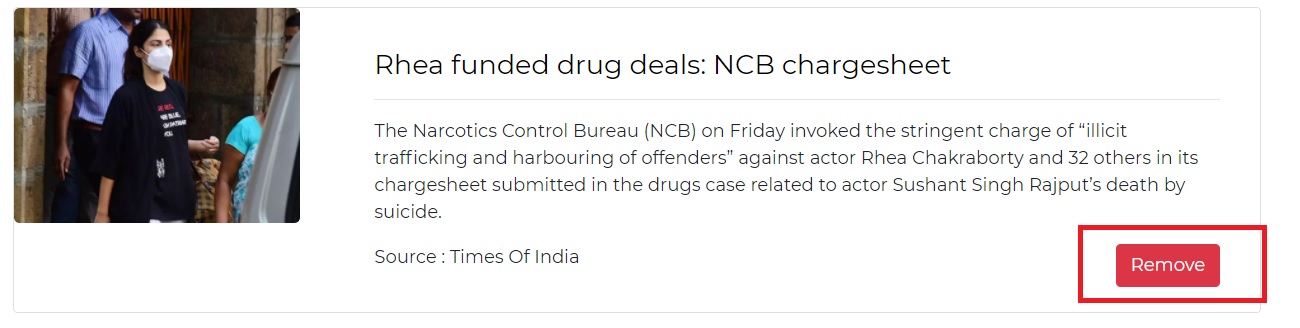
print("\n-------News Id : ",id, "reported successfully!--------\n")

return HttpResponseRedirect("/home")

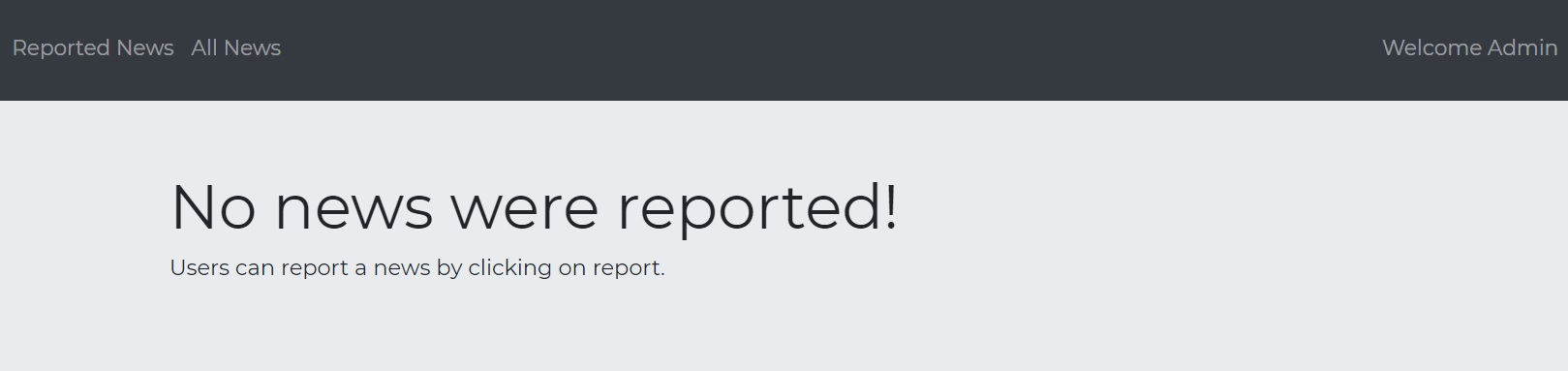
* Users can report news by clicking on the ‘flag’ icon.
* Admin will be able to see the reported news and can delete it.
* So, the report() method will get id of the news from the template and will update the news with “Reported” field equals to true.
* All news having “Reported” true will be shown to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

1. **News Deletion by Admin**

UI to delete a news:



News deleted:



**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")

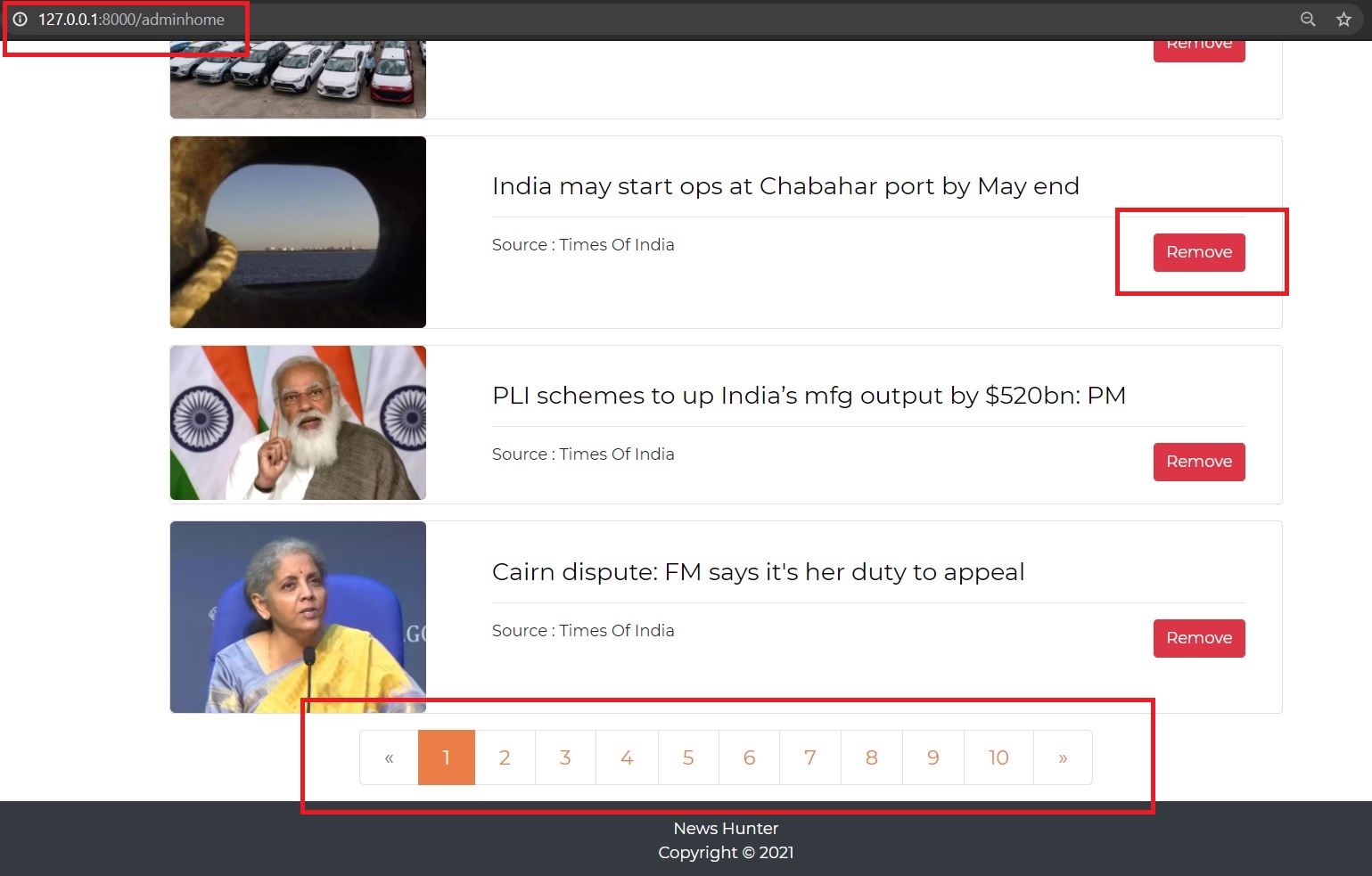
return HttpResponseRedirect('/adminhome')

● The admin can delete news by clicking on the ‘Remove’ button.

● The delete\_news() method gets the id of the news and deletes it from the database.

1. **Display news to Admin**

UI for managing news:



**Code Snippet:**

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})

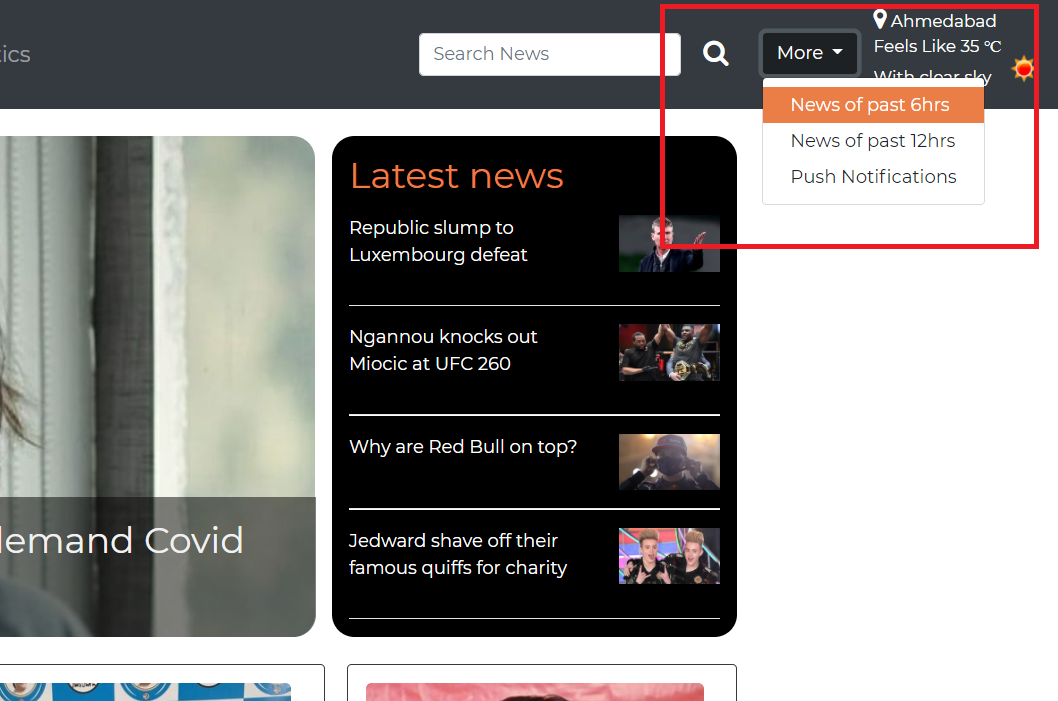
● Admin can view all news and can remove them.

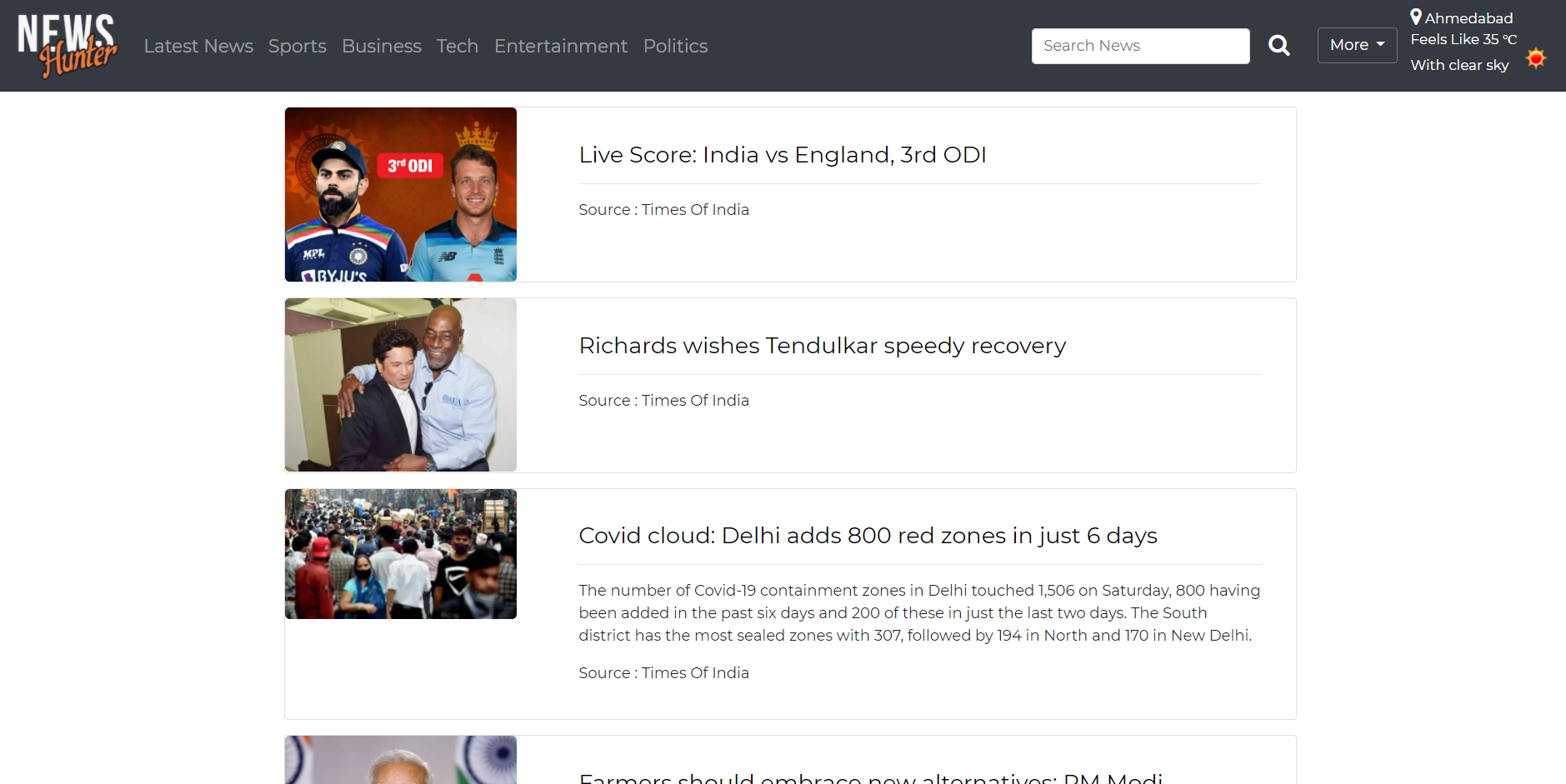
● The adminhome() view returns the news list from the database.

● Pagination provided by django is used to paginate the news list.

1. **Display news of last 6hrs/12hrs:**

UI to view news of last 6hrs and 12hrs:



News of past 6hrs:

**Code Snippet:**

# Showing news of past x hrs

def oldnews(request, hrs):

#Getting news list from database

c = {}

c.update(csrf(request))

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']

# Keeping news which were latest by 6 hrs (21600 sec)

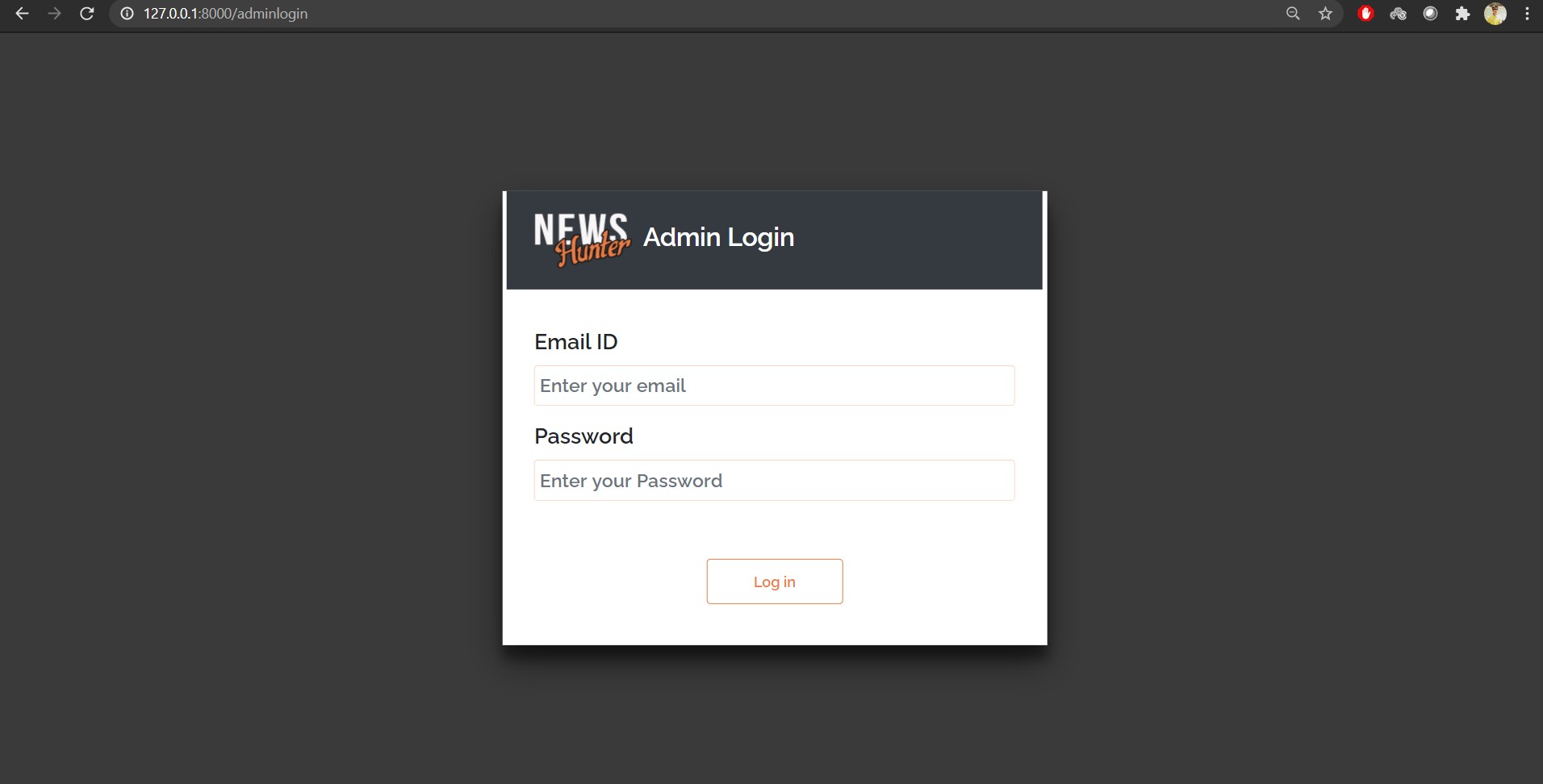
if(duration < (hrs \* 3600)):

news\_list.append(i)

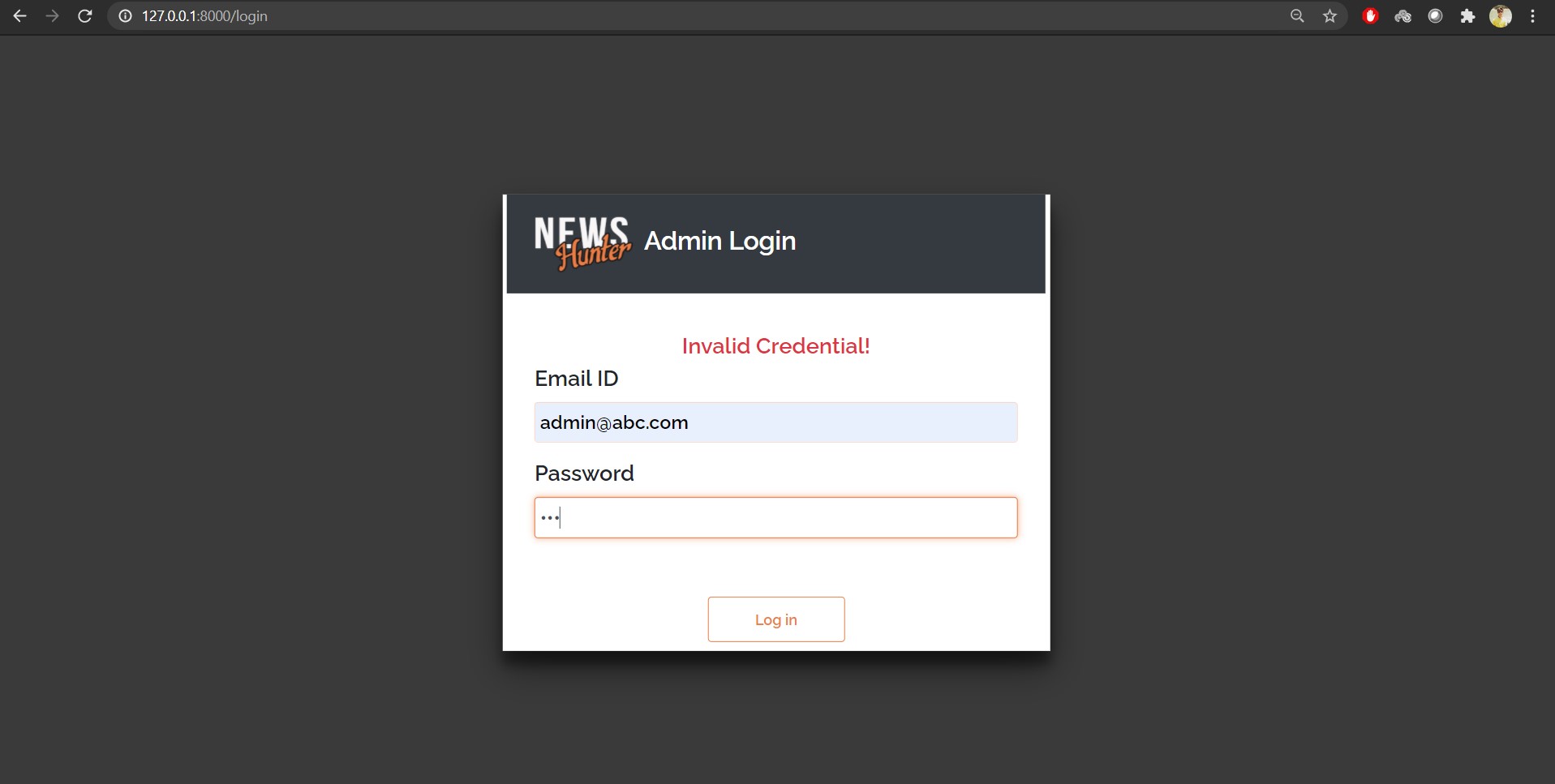
* The view gives a list of news with duration less than 6 hrs or 12 hrs fetched from the database.

1. **Admin authentication and authorization**

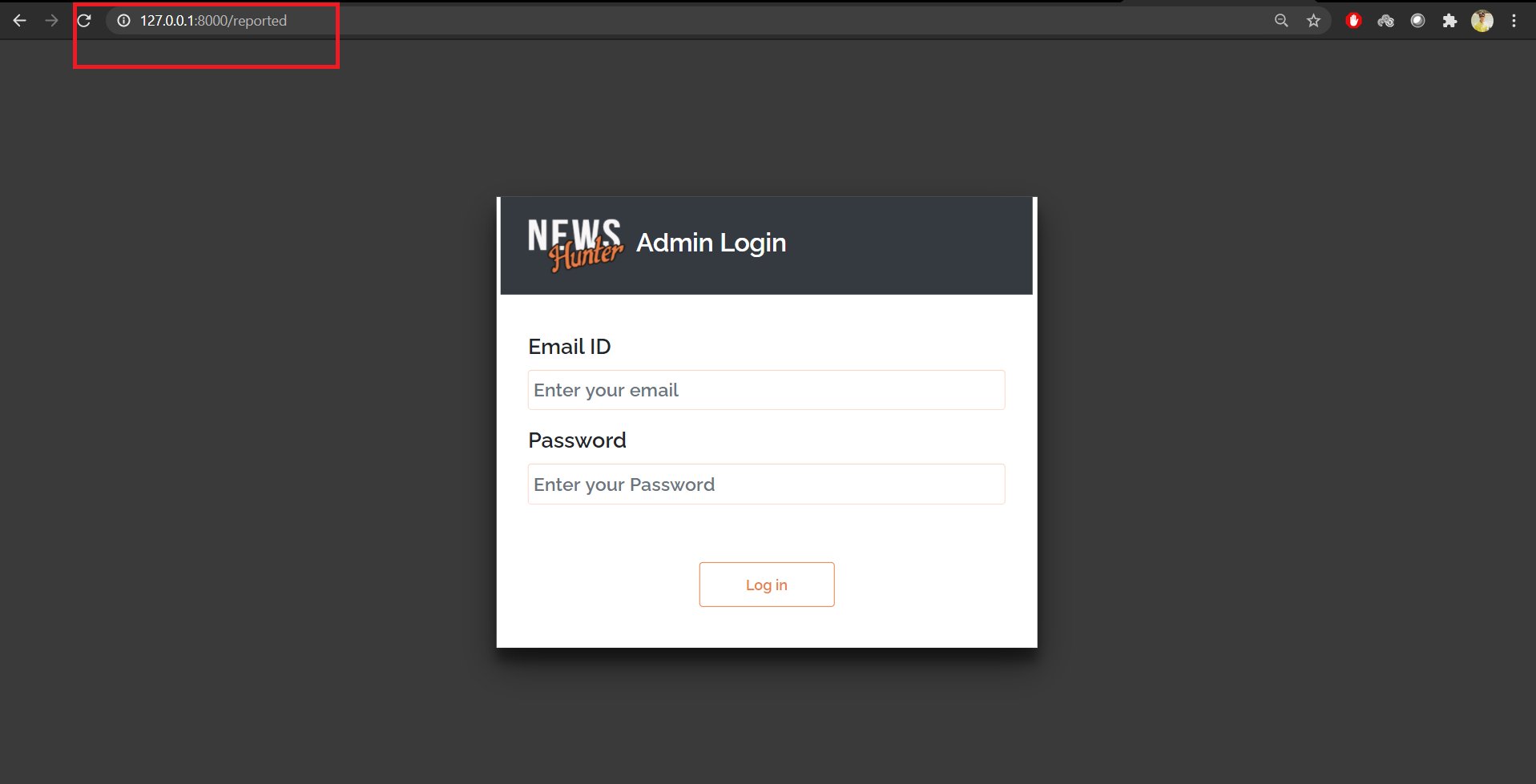
UI for Admin to log in:



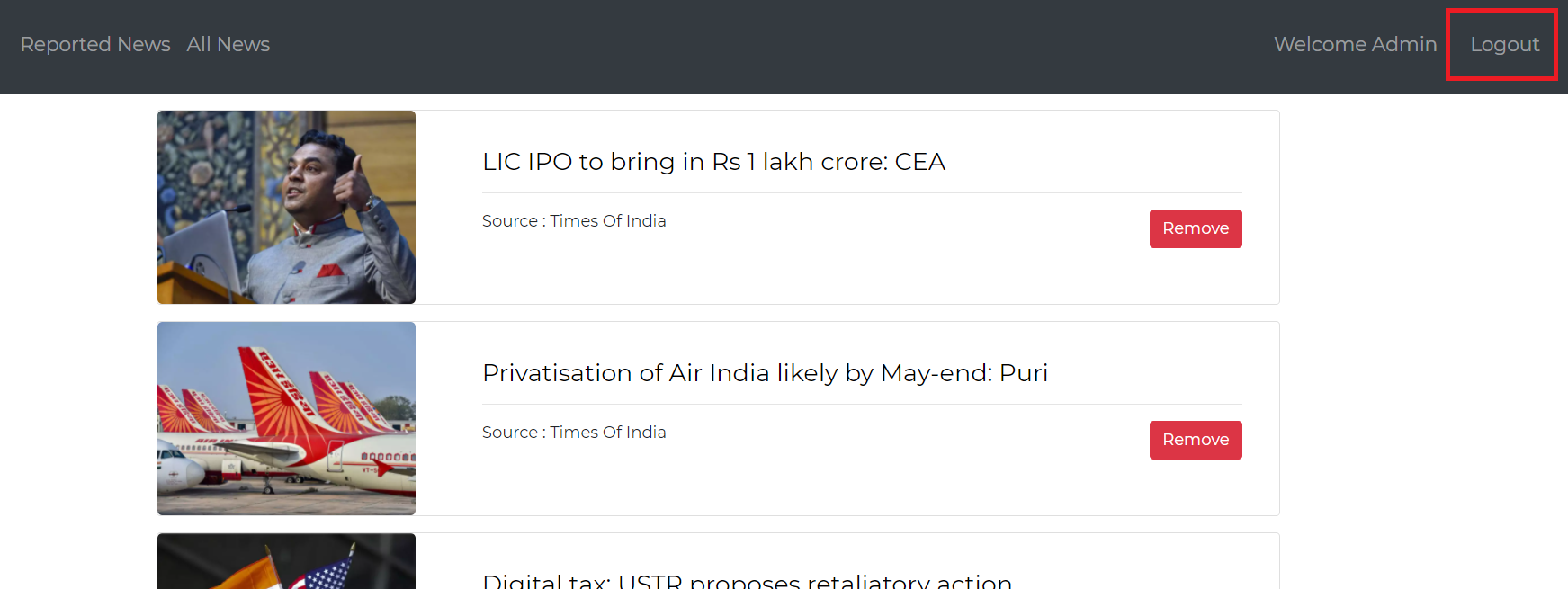
Message is displayed on Invalid Credentials:



Admin is authorized before accessing the admin module pages:



UI to logout:



**Code Snippet:**

@require\_POST

def login(request):

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

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

myclient, mydb = get\_MongoClient()

mycol = mydb["admin"]

userobj = mycol.find\_one({"useremail":useremail})

# print(userobj]["password"])

if userobj is None:

return render(request, 'adminlogin.html', {'error':login\_error})

else:

# print(userobj["password"])

if password == userobj["password"]:

request.session["useremail"] = useremail

return HttpResponseRedirect('/adminhome')

else:

return render(request, 'adminlogin.html', {'error':login\_error})

return render(request, 'adminlogin.html')

* The view compares the email and password from MongoDB and redirects the admin home page, if credentials are matched.

def logout(request):

del request.session['useremail']

return HttpResponseRedirect('/adminlogin')

* The view simply deletes the admin session and admin is logged out.

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

print("user is not logged in")

c = {}

c.update(csrf(request))

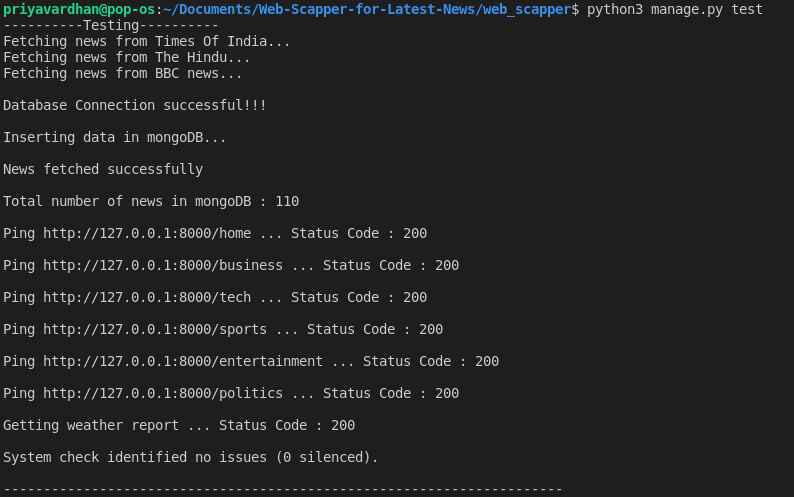
return render(request, 'adminlogin.html', c)

* This code authorizes the admin.
* It checks whether the admin session exists or not. If not, it redirects to the login page.

**Testing**

Testing in Django is done using the test execution framework provided by django itself. The requests are simulated using the file ‘tests.py’ in the application and the respective outputs and status code can be inspected.

Some modules were tested using this test execution framework and some modules were tested manually.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Requirement ID** | **Test Case Description** | **Expected Output** | **Actual Output** | **Test Case Status** |
| 1 | R 4.1.1 | Extracting the HTML code from the website and storing it in dataframe | The HTML code extracted successfully and stored in the data frame | As expected | Pass |
| 2 | R 4.1.1,  R 4.1.2 | Displaying the latest news and updating the site at regular intervals | The news stream updated if there are any changes | As expected | Pass |
| 4 | R 4.1.3 | User can select a category according to their choice | News related to the category selected by the user displayed | As expected | Pass |
| 5 | R 4.1.1 | User can read the full article by clicking the image, headline or “Read More” button | User is redirected to the original article link and can read the full article | As expected | Pass |
| 6 | R 4.1.12 | After any new news is extracted , the headline from the HTML code is extracted and fed to the text classification model to select it’s category | News category determined according to the type of news and is expected to be placed under the right category | As expected | Pass |
| 7 | R 4.1.6 | User can search for any news based on a keyword | All the news containing the search keyword in its headline or description | As expected | Pass |
| 8 | R 4.1.9 | Users get Push notifications after subscribing. | Push Notification is sent to the user. | As expected | Pass |
| 9 | R 4.1.9 | Users won't get Push notifications after unsubscribing. | Push Notifications are not sent to users who have not subscribed. | As expected | Pass |
| 10 | R 4.1.9 | All users who have subscribed should get the notification. | Push Notifications are sent to all users who have subscribed. | As expected | Pass |
| 11 | R 4.1.4 | Show weather report | The temperature and climate description of his/her nearest city is display to the user | As expected | Pass |
| 12 | R 4.1.8 | User can report offensive news anonymously | The news is marked as reported and awaits the administrator decision to remove the news or keep it | As expected | Pass |
| 13 | R 4.2.1 | Administrator login with incorrect credentials | Invalid credentials message and redirect to login page | As expected | Pass |
| 14 | R 4.2.2,  R 4.2.3 | Administrator login with correct credentials | Redirect to dashboard where admin can view the reported and all the news | As expected | Pass |
| 15 | R 4.2.5 | News deletion by administrator | If admin feels that any of the news is a misfit or offensive, he/she can remove the news from database | As expected | Pass |

**Challenges faced:**

1. Difficulties in finding patterns for scraping news from the sources.

* In the web application, news is fetched from different sources and source code is different for all. So it needed some micro-observation to figure out the patterns in the html code of the website and extract respective news headline, description, image and article link.

1. Model training difficulties

* It was very difficult to find the dataset which had a variety of categories.
* With a very large dataset, the accuracy was not obtained up to the mark. It was difficult to train the model with such a large dataset. It consumed a lot of time and effort.

1. Difficulties in search functionality

* We were thinking of creating an algorithm for search like to implement binary search algorithm. But it was taking more time in searching as it fetches data from Database and applies algorithm on that. But we came to know that instead of using such an algorithm, we can use MongoDB’s text index to perform search.
* So, we created a text index on fields: ‘Category’, ‘Headline’, ‘Content’ and ‘Source’ of ‘news’ collection and we were firing search queries which were using this index. It was faster than the algorithm we thought.

**Conclusion:**

* All the functionalities mentioned in the Requirements document have been completed. Users can easily view the latest news along with a weather report of the user's nearby city on our web application.
* Also they get notified when the news feed is updated and can monitor our web application by reporting any malicious or offensive news .
* Users are also provided with an efficient and effort free keyword search. As per their convenience , the user can also filter out the last 6/12 hours news.
* The news extracted from the news websites are divided automatically into their categories by our Multiclass-classification model, which uses a logistic function to classify the news into categories. Users can view news based on these categories.

**Limitations:**

1. The news are divided into only 5 categories. No further category identification takes place.
2. Also the dataset for the ML Model is quite small. So the accuracy obtained is quite high. In future whenever the dataset is increased, it might be the case that the accuracy decreases.

**Future Extensions**

1. Admin can see filtered news using AI and ML for reported news.

* Currently, we have displayed all the reported news to the admin.
* As the list of reported news can be very long, we can use AI and ML to show news which were most reported or are genuinely inappropriate news.

1. News categories can have different levels.

* Currently, we have kept only one level of category of news.
* There can be multiple levels of categories or sub-categories.

1. Searching can be done for substrings also.

* The search we have used, searches the keywords as whole.
* We can improve it by searching keywords in the substring too.

1. Improvement in the weather report.

* We can use an api for locating the user's exact location for a weather report.

1. Improvement in sending push notifications.

* The push notification can be sent to users when they are not accessing our website.

1. Increasing the existing dataset.

* We can add the news extracted from the news sites into our dataset and hence increase it’s size.
* Also with the new dataset we can re-train our ML model and thereby increase it’s accuracy and efficiency.

**References / Bibliography**

[stackoverflow.com](https://stackoverflow.com/) : Problem Solving and debugging

[docs.djangoproject.com/en/3.1/](https://docs.djangoproject.com/): Django Documentation

[docs.mongodb.com/manual/](https://docs.mongodb.com/manual/) : MongoDB Docs

[openweathermap.org/](https://openweathermap.org/) : OpenWeatherMap API

[realpython.com/beautiful-soup-web-scraper-python/](https://realpython.com/beautiful-soup-web-scraper-python/) : Beautiful Soup Docs

[pypi.org/](https://pypi.org/) : Various Python Packages Docs

[www.w3schools.com/](http://www.w3schools.com/) : HTML, CSS, Python Docs

[getbootstrap.com/docs/5.0/getting-started/introduction/](https://getbootstrap.com/docs/5.0/getting-started/introduction/) : Bootstrap Docs

[www.kaggle.com](http://www.kaggle.com) : Dataset

<https://towardsdatascience.com> : For learning ML basics