

## **ABSTRACT**

A news aggregator which is also known as a feed aggregator or a news reader or simply an aggregator, is a client software or a web application that aggregates syndicated web content such as online newspapers, blogs, podcasts and video blogs (vlogs) in one location for easy viewing. It is widely used because the internet is filled with information so it becomes difficult to find relevant news or information, that's where news aggregator comes into play. You can personalize your news aggregator according to the news that you would like to read. For example – some people would prefer reading international news while the others might be interested in financial or sports news.

The news aggregator is a python application which aggregates news headlines and news via several electronic news publishers. The system collects news from HTML and RSS web documents by using source- specific information extraction programs (wrappers) and parsers such as Urllib, Requests and BeautifulSoup Modules of python, organizes them in database (SQLite3) which updates on availability of new news headlines and display them to pre-defined news categories and constructs personalized views via a web-based interface. Adaptive personalization is performed, based on the individual user interaction.

# **INDEX**

<b>CHAPTER 1</b>	<b>INTRODUCTION TO THE PROJECT</b>
<b>CHAPTER 2</b>	<b>ABOUT PROJECT</b>
	<b>2.1 OBJECTIVES OF THE PROJECT</b>
	<b>2.2 SOFTWARE REQUIREMENTS</b>
	<b>2.3 BLOCK DIAGRAMS AND FLOW</b>
<b>CHAPTER 3</b>	<b>PROJECT</b>
	<b>3.1 PROJECT SOURCE CODE</b>
	<b>3.2 PROJECT SOURCE CODE</b>
	<b>3.3 PROJECT RESULT &amp; SNAPSHOTS</b>
<b>CHAPTER 5</b>	<b>CONCLUSION</b>
<b>REFERENCES</b>	

# CHAPTER 1

## INTRODUCTION TO NEWSSCRAPER

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages. Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. We should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus.

Web scraping is a term used to describe the use of a program or algorithm to extract and process large amounts of data from the web. Whether you are a data scientist, engineer, or anybody who analyzes large amounts of datasets, the ability to scrape data from the web is a useful skill to have. Let's say you find data from the web, and there is no direct way to download it, web scraping using Python is a skill you can use to extract the data into a useful form that can be imported.

In this project we construct a web application called News Scraper. It is an application that aggregates syndicated web content such as online newspapers, blogs, podcasts and video blogs in one location for easy viewing. It is widely used because the internet is filled with information so it becomes difficult to find relevant news or information, that's where news aggregator comes into play. You can personalize your news aggregator according to the news that you would like to read. For example – some people would prefer reading international news while the others might be interested in financial or sports news.

The news aggregator is a python application which aggregates news headlines and news via several electronic news publishers. The system collects news from HTML and RSS web documents by using source- specific information extraction programs (wrappers) and parsers such as Urllib, Requests and BeautifulSoup Modules of python, organizes them in database (SQLite3) which updates on availability of new news headlines and display them to pre-defined news categories and constructs personalized views via a web-based interface. Adaptive personalization is performed, based on the individual user interaction.

## CHAPTER 2

### 2.1 Objectives of the Project

The main objectives of this project are to extract the news posted in different news publishing site, organized them in one site and provide the user. The main objectives can be summarized as:

- To develop a dynamic news site.
- To allow the user to find all the best news.
- To provide all news in one place.

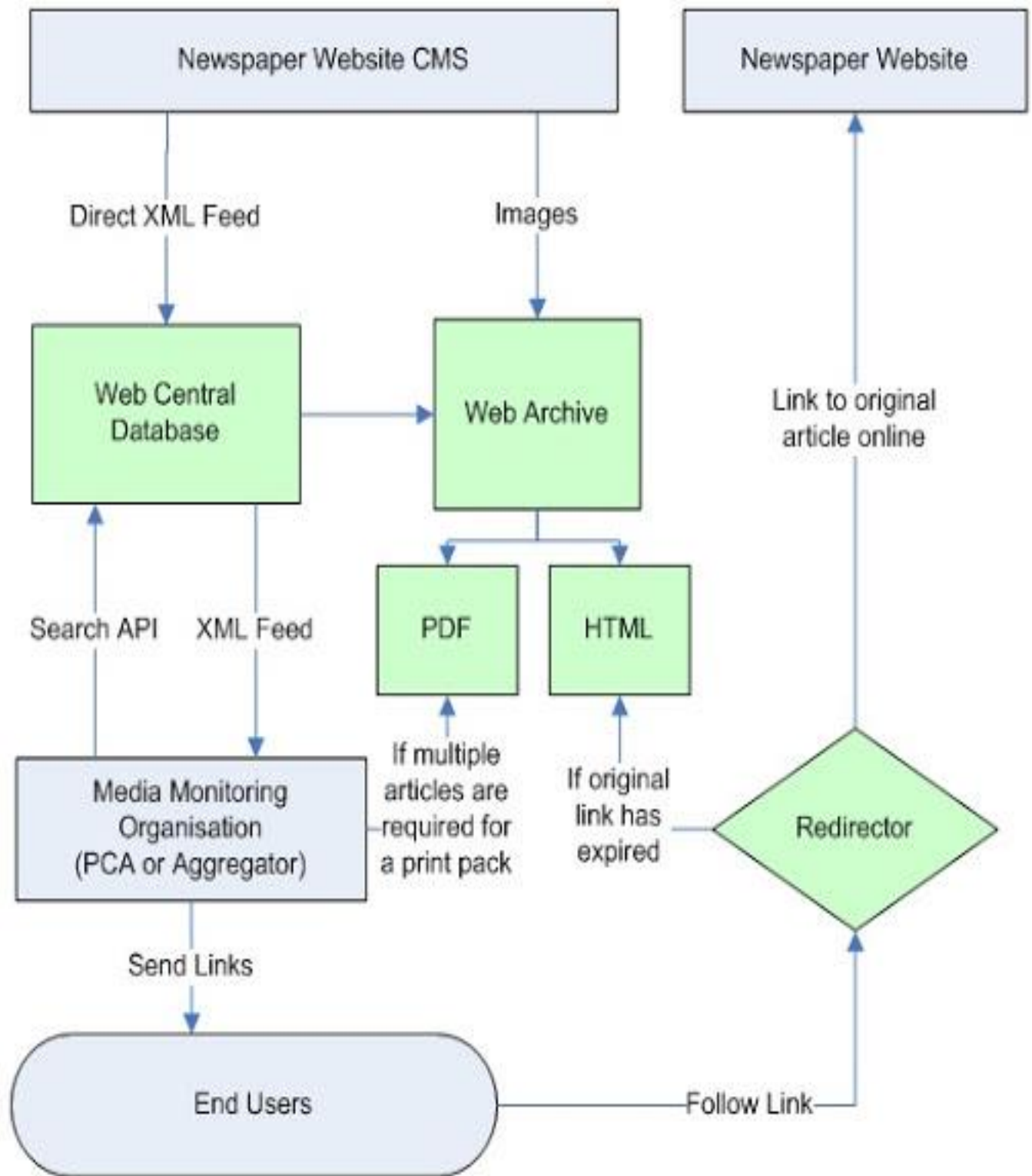
### Limitations

The limitations of this project is that it just lists the news posted in other news publishing sites so there is no option for posting news. It's hard for consumers to get the exact information about news of their interests. This online aggregator system is unable to retrieve information that has been updated by the news sites instantly. Some instantly changed information can be missed by the system.

### 2.2 Software Requirement of the Project

- **PyCharm** - PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science with Anaconda.
- **Web browser** - A web browser is a software application for accessing information on the World Wide Web. When a user requests a particular website, the web browser retrieves the necessary content from a web server and then displays the resulting web page on the user's device.
- **Internet connection** – A stable and steady internet connection is advised in order to access the web browser to access data and to work on various web applications. Without a stable internet connection accessing the web application wouldn't be possible.

## 2.3 Block Diagram of Project



## CHAPTER 3

### MODULES, SOURCE CODE & PROJECT SNAPSHOTS

#### URLLIB

Urllib module is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the *urlopen* function and is able to fetch URLs using a variety of different protocols.

Urllib is a package that collects several modules for working with URLs, such as:

- `urllib.request` for opening and reading.
- `urllib.parse` for parsing URLs
- `urllib.error` for the exceptions raised
- `urllib.robotparser` for parsing robot.txt files

Let's see these in details.

- **urllib.request:** This module helps to define functions and classes to open URLs (mostly HTTP). One of the most simple ways to open such URLs is :  
`urllib.request.urlopen(url)`
- **urllib.parse:** This module helps to define functions to manipulate URLs and their components parts, to build or break them. It usually focuses on splitting a URL into small components; or joining different URL components into URL string.
- **urllib.error:** This module defines the classes for exception raised by `urllib.request`. Whenever there is an error in fetching a URL, this module helps in raising exceptions.

Main Code Used in Project using this Module

```
import urllib.request, urllib.parse, urllib.error
req = urllib.request.Request(f"https://www.hindustantimes.com/latest-
news/?pageno={i}", headers={'User-Agent': 'Mozilla/5.0'})
res = urllib.request.urlopen(req).read()
```

#### BEAUTIFUL SOUP -

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. Beautiful Soup provides a few simple methods and Python idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application

2. BeautifulSoup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and BeautifulSoup can't detect one. Then you just have to specify the original encoding.
3. BeautifulSoup sits on top of popular Python parsers like lxml and html5lib, allowing you to try out different parsing strategies or trade speed for flexibility.

BeautifulSoup parses anything you give it, and does the tree traversal stuff for you. You can tell it "Find all the links", or "Find all the links of class external Link", or "Find all the links whose urls match "foo.com", or "Find the table heading that's got bold text, then give me that text."

Valuable data that was once locked up in poorly-designed websites is now within your reach. Projects that would have taken hours take only minutes with BeautifulSoup.

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(res, "html.parser")

news_box = soup.find('ul', {'class': 'latest-news-bx more-latest-news more-separate'})
all_news = news_box.find_all('a')

for i in range(len(all_news)):
    title = (str(all_news[i].get('title')))
    link = (str(all_news[i].get('href')))
    image = (str(all_news[i].get('src')))
```

## SQLITE3 –

SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

### Some Important Methods in SQLite:

#### sqlite3.connect(database):

This API opens a connection to the SQLite database file. If database is opened successfully, it returns a connection object.

When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed.

**connection.cursor([cursorClass]):**

This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor.

**cursor.execute(sql [, optional parameters]):**

This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks and named placeholders (named style).

**For example** – cursor.execute("insert into people values (?, ?)", (who, age))

**connection.close():**

This method closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost!

**cursor.fetchone():**

This method fetches the next row of a query result set, returning a single sequence, or None when no more data is available.

**cursor.fetchall():**

This routine fetches all (remaining) rows of a query result, returning a list. An empty list is returned when no rows are available.

**SQLite DB Source Code:**

```
import sqlite3

conn = sqlite3.connect('news_db.sqlite')
cur = conn.cursor()
def create_connection():

    cur.execute('DROP TABLE IF EXISTS indiatoday')
    cur.execute('DROP TABLE IF EXISTS dainikjagran')
    cur.execute('DROP TABLE IF EXISTS hindustantimes')

    cur.execute('''CREATE TABLE IF NOT EXISTS dainikjagran(
                    title TEXT, link TEXT PRIMARY KEY, imagelink TEXT)''')

    cur.execute('''CREATE TABLE IF NOT EXISTS indiatoday(
                    title TEXT, link TEXT PRIMARY KEY, imagelink TEXT)''')

    cur.execute('''CREATE TABLE IF NOT EXISTS hindustantimes(
                    title TEXT, link TEXT PRIMARY KEY, imagelink TEXT)''')
```



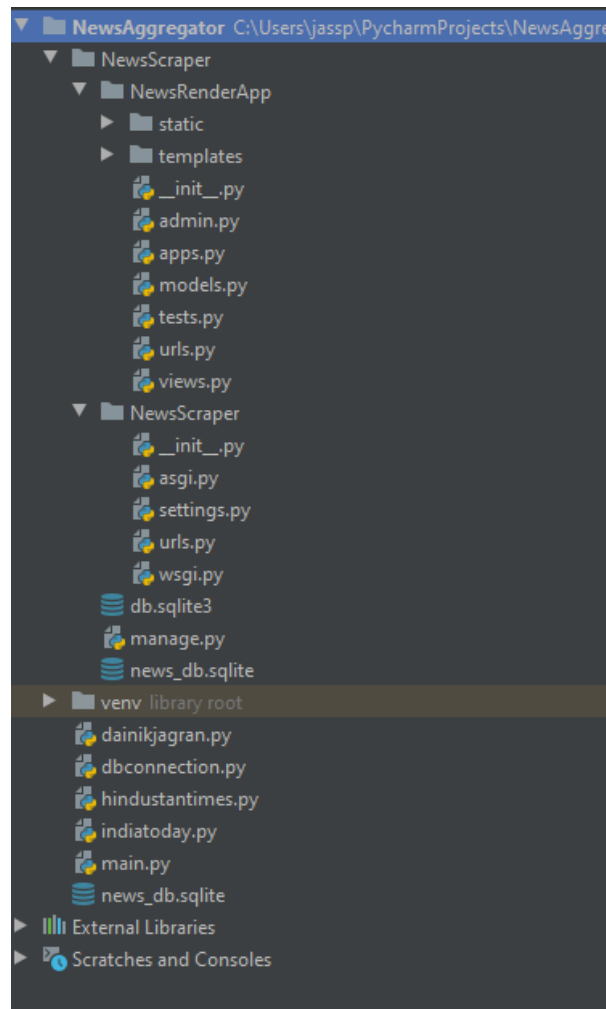
## DJANGO –

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

**Django** is a Python-based free and open source web framework, which follows the model-template-view (MTV) architectural pattern. It is maintained by the Django Software Foundation (DSF), an independent organization established as a non-profit.

Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

### Our File Architecture to Scrape, Store, Display & Run the Website



## Some Important Django File Required to Create & Run a WebApp:

1. **Models.py:** This File Consists of Model Required to Fetch News From out Databases and Display our Content.

```
from django.db import models

class IndiaTodayNews(models.Model):
    title = models.CharField(max_length=200)
    image = models.URLField(null=True, blank=True)
    url = models.TextField()

    def __str__(self):
        return self.title

class DainikJagranNews(models.Model):
    title = models.CharField(max_length=200)
    image = models.URLField(null=True, blank=True)
    url = models.TextField()

    def __str__(self):
        return self.title

class HindustanTimesNews(models.Model):
    title = models.CharField(max_length=200)
    image = models.URLField(null=True, blank=True)
    url = models.TextField()

    def __str__(self):
        return self.title
```

## 2. Url.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
    path('indiatoday_scrape/', views.indiatoday_scrape, name="indiatodayscrape"),
    path('news/indiatoday', views.news_list_indiatoday, name="indiatoday"),
    path('dainikjagran_scrape/', views.dainikjagran_scrape, name="dainikjagranscrape"),
    path('news/dainikjagran', views.news_list_dainikjagran, name="dainikjagran"),
    path('ht_scrape/', views.hindustantimes_scrape, name="hindustantimesscrape"),
    path('news/hindustantimes', views.news_list_hindustantimes, name="htimes"),]
```

**3. Manage.py:** This is the Main Page Which Controls the Execution of the Program and Help in Managing all Resources.

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'NewsScraper.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

## Code For our Views Which are to Be Displayed on Site:

The following image shows us the code & output which scrape the news & updates database as news is requested as the news is updated. We are using three Newspapers to scrape the news and display in our customized views.

## 4. Views.py

```
import urllib.request
from django.shortcuts import render, redirect
from bs4 import BeautifulSoup
from .models import *
import sqlite3

def index(request):

    return render(request, 'index.html')

def news_list_indiatoday(request):
```

```

    headlines1 = IndiaTodayNews.objects.all()[:]
    context = {
        'object_list': headlines1,
    }
    return render(request, "news/indiatoday/indiatoday.html", context)

def news_list_dainikjagran(request):
    headlines2 = DainikJagranNews.objects.all()[:]
    context = {
        'object_list': headlines2,
    }
    return render(request, "news/dainikjagran/dainikjagran.html", context)

def news_list_hindustantimes(request):
    headlines3 = HindustanTimesNews.objects.all()[:]
    context = {
        'object_list': headlines3,
    }
    return render(request, "news/hindustantimes/hindustantimes.html", context)

def indiatoday_scrape(request):
    conn = sqlite3.connect('db.sqlite3')
    cur = conn.cursor()
    cur.execute('DELETE FROM NewsRenderApp_indiatodaynews')
    conn.commit()
    res = urllib.request.urlopen("https://www.indiatoday.in/top-stories").read()

    soup = BeautifulSoup(res, "html.parser")
    news_box = soup.find('div', {'class': 'view-content'})
    all_news = news_box.find_all('h2')
    all_news_link = news_box.find_all('a')
    all_news_image = news_box.find_all('img')
    for i in range(len(all_news)):
        title = (str(all_news[i].get('title')))
        link = (str("https://www.indiatoday.in" +
str(all_news_link[i].get('href'))))
        image = (str(all_news_image[i].get('src')))
        new_headline = IndiaTodayNews()
        new_headline.title = title
        new_headline.url = link
        new_headline.image = image
        new_headline.save()

    return redirect("../")

def dainikjagran_scrape(request):
    conn = sqlite3.connect('db.sqlite3')

```

```

cur = conn.cursor()
cur.execute('DELETE FROM NewsRenderApp_dainikjagrannews')
conn.commit()
res = urllib.request.urlopen(f"https://english.jagran.com/latest-
news").read()

soup = BeautifulSoup(res, "html.parser")
news_box = soup.find('ul', {'class': 'topicList'})
all_news_link = news_box.find_all('a')
all_news_image = news_box.find_all('img')

for i in range(len(all_news_link)):
    title = (str(all_news_image[i].get('alt')))
    link = (str("https://english.jagran.com" +
str(all_news_link[i].get('href'))))
    image = (str(all_news_image[i].get('data-src')))
    new_headline = DainikJagranNews()
    new_headline.title = title
    new_headline.url = link
    new_headline.image = image
    new_headline.save()
return redirect("../")

def hindustantimes_scrape(request):
conn = sqlite3.connect('db.sqlite3')
cur = conn.cursor()
cur.execute('DELETE FROM NewsRenderApp_hindustantimesnews')
conn.commit()
req = urllib.request.Request(f"https://www.hindustantimes.com/latest-news",
headers={'User-Agent': 'Mozilla/5.0'})
res = urllib.request.urlopen(req).read()

soup = BeautifulSoup(res, "html.parser")
print(res)
news_box = soup.find('ul', {'class': 'latest-news-bx more-latest-news more-
separate'})
all_news = news_box.find_all('a')

for i in range(len(all_news)):
    title = (str(all_news[i].get('title')))
    link = (str(all_news[i].get('href')))
    image = (str(all_news[i].get('src')))
    new_headline = HindustanTimesNews()
    new_headline.title = title
    new_headline.url = link
    new_headline.image = image
    new_headline.save()

return redirect("../")

```

## BOOTSTRAP –

Bootstrap is a CSS framework that makes it easier to create website and web application user interfaces. Bootstrap is especially useful as a base layer of CSS to build sites with responsive web design.

Bootstrap 3 was out for almost five years before Bootstrap 4 was finally released at the start of 2018. When learning Bootstrap you will likely come across many tutorials that cover Bootstrap 3 but not version 4.

If you are completely new to Bootstrap and CSS frameworks in general then I recommend learning Bootstrap 4. If you have already been working with Bootstrap 3 then there is no major rush to upgrade to the latest version. Bootstrap 4 is more complicated than version 3 because it has a lot more features so the learning curve is a bit steeper.

Full Stack Python is actually built with an early version of Bootstrap 3. However, this site is so heavily customized with my own CSS that I likely will never upgrade to Bootstrap 4 because there are no new features that I feel will be useful in my specific situation.

### Bootstrap Code for Displaying news in News Views:

```
{% load static %}
<!DOCTYPE html>
<html>
<head>
    <title>IndiaToday News</title>

    <!-- Favicons -->
    <link href="{% static 'assets/img/favicon.png' %}" rel="icon">
    <link href="{% static 'assets/img/apple-touch-icon' %}" rel="apple-touch-icon">

    <!-- Google Fonts -->
    <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Raleway:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">

    <!-- Vendor CSS Files -->
    <link href="{% static 'assets/vendor/bootstrap/css/bootstrap.css' %}"
rel="stylesheet">
    <link href="{% static 'assets/vendor/icomfont/icomfont.css' %}" rel="stylesheet">
    <link href="{% static 'assets/vendor/boxicons/css/boxicons.css' %}"
rel="stylesheet">
    <link href="{% static 'assets/vendor/animate.css/animate.css' %}"
rel="stylesheet">
    <link href="{% static 'assets/vendor/owl.carousel/owl.carousel.css' %}"
rel="stylesheet">
    <link href="{% static 'assets/vendor/venobox/venobox.css' %}" rel="stylesheet">

    <!-- Template Main CSS File -->
    <link href="{% static 'assets/css/style.css' %}" rel="stylesheet">
```

```

</head>
<body>

<!-- ===== Header ===== -->
<header id="header">
  <div class="container d-flex">

    <div class="logo mr-auto">
      <h1 class="text-light"><a href="#"><span>NewsScraper</span></a></h1>

    </div>

    <nav class="nav-menu d-none d-lg-block">
      <ul>
        <li class="active"><a href="index.html">Home</a></li>
        <li><a href="services.html">About Project</a></li>
        <li><a href="portfolio.html">Resources</a></li>
        <li><a href="contact.html">Developers</a></li>

      </ul>
    </nav><!-- .nav-menu -->

  </div>
</header><!-- End Header -->
<br>
<section id="services" class="services">
  <div class="container">

    <div class="section-title" data-aos="fade-up">
      <h2>India Today</h2>
      <form method="POST" action="{% url 'indiatodayscrape' %}"
target="_self">
        {% csrf_token %}
        <button type="submit" class="btn btn-success">Fetch News</button>
      </form>
    </div>

    <div class="row">
{% for object in object_list %}
      <div class="col-lg-4 col-md-6 d-flex align-items-stretch">
        <div class="icon-box">
          <div class="icon" style="background-color: white; height:128px;
width: 128px;"> <img class="card-img-top" src = "{{ object.image }}"></div>
          <h4><a href="{{object.url}}"><h5 class="card-
title">{{object.title}}</h5></a></h4>
        </div>

      </div>
    {% endfor %}
  </div>

</div>

```

```

</section>

<!-- ===== Footer ===== -->
<footer id="footer">

    <div class="footer-top">
        <div class="container">
            <div class="row">

                <div class="col-lg-3 col-md-6 footer-links">
                    <h4>Useful Links</h4>
                    <ul>
                        <li><i class="bx bx-chevron-right"></i> <a href="#">Home</a></li>
                        <li><i class="bx bx-chevron-right"></i> <a href="#">About
Project</a></li>
                        <li><i class="bx bx-chevron-right"></i> <a href="#">Our
Resouces</a></li>
                        <li><i class="bx bx-chevron-right"></i> <a
href="#">Developers</a></li>
                    </ul>
                </div>

                <div class="col-lg-3 col-md-6 footer-links" style="text-align:
center;">
                    <h4>Developed & Designed By: </h4>
                    
                    <div class="social-links mt-3">
                        <a href="#" class="twitter"><i class="bx bxl-twitter"></i></a>
                        <a href="#" class="facebook"><i class="bx bxl-facebook"></i></a>
                        <a href="#" class="instagram"><i class="bx bxl-instagram"></i></a>
                        <a href="#" class="google-plus"><i class="bx bxl-skype"></i></a>
                        <a href="#" class="linkedin"><i class="bx bxl-linkedin"></i></a>
                    </div>
                </div>

                <div class="col-lg-3 col-md-6 footer-contact" style="text-align:
center;">
                    <h4 >Developed & Designed By: </h4>
                    
                    <div class="social-links mt-3">
                        <a href="#" class="twitter"><i class="bx bxl-twitter"></i></a>
                        <a href="#" class="facebook"><i class="bx bxl-facebook"></i></a>
                        <a href="#" class="instagram"><i class="bx bxl-instagram"></i></a>
                        <a href="#" class="google-plus"><i class="bx bxl-skype"></i></a>
                        <a href="#" class="linkedin"><i class="bx bxl-linkedin"></i></a>

```



```

        </div>
    </div>

    <div class="col-lg-3 col-md-6 footer-info">
        <h3>About NewsScaper</h3>
        <p>The News Aggregator Is a Python Application Which Aggregates News
Headlines via Several Electronic News Publishers. the System Collects News from
HTML by Using Source- Specific Information Extraction Programs Such as Urllib,
Requests and BeautifulSoup Modules</p>

    </div>

</div>
</div>
</div>

<div class="container">
    <div class="copyright">
        Python Project <strong><span>NewsScraper</span></strong>. SRMIST
    </div>
    <div class="credits">

        Designed by <a href="#">Jasmeet Singh & Riya Sharma Using Bootstrap</a>
    </div>
</div>
</footer><!-- End Footer -->

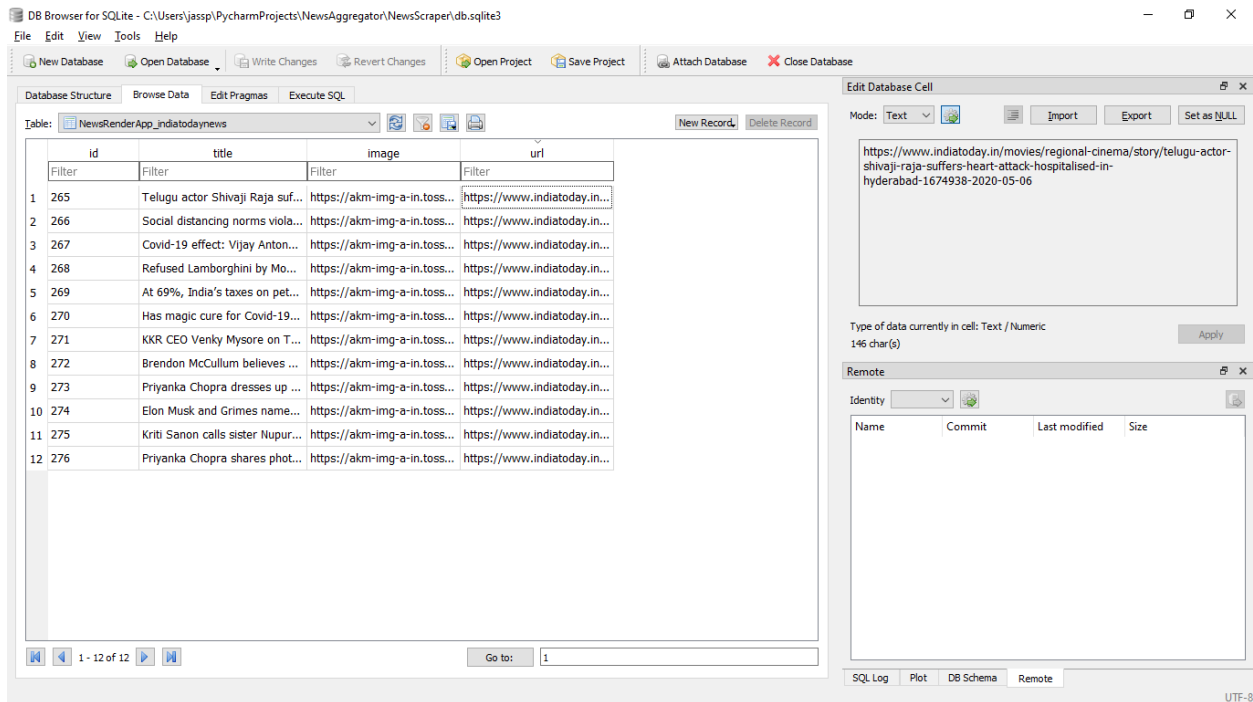
<a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>

<!-- Vendor JS Files -->
<script src="{% static 'assets/vendor/jquery/jquery.min.js' %}"></script>
<script src="{% static 'assets/vendor/bootstrap/js/bootstrap.bundle.min.js'
%}"></script>
<script src="{% static 'assets/vendor/jquery.easing/jquery.easing.min.js'
%}"></script>
<script src="{% static 'assets/vendor/php-email-form/validate.js' %}"></script>
<script src="{% static 'assets/vendor/jquery-sticky/jquery.sticky.js'
%}"></script>
<script src="{% static 'assets/vendor/owl.carousel/owl.carousel.min.js'
%}"></script>
<script src="{% static 'assets/vendor/waypoints/jquery.waypoints.min.js'
%}"></script>
<script src="{% static 'assets/vendor/counterup/counterup.min.js' %}"></script>
<script src="{% static 'assets/vendor/isotope-layout/isotope.pkgd.min.js'
%}"></script>
<script src="{% static 'assets/vendor/venobox/venobox.min.js' %}"></script>

<script src="{% static 'assets/js/main.js' %}"></script>
</body>
</html>

```

## Output:



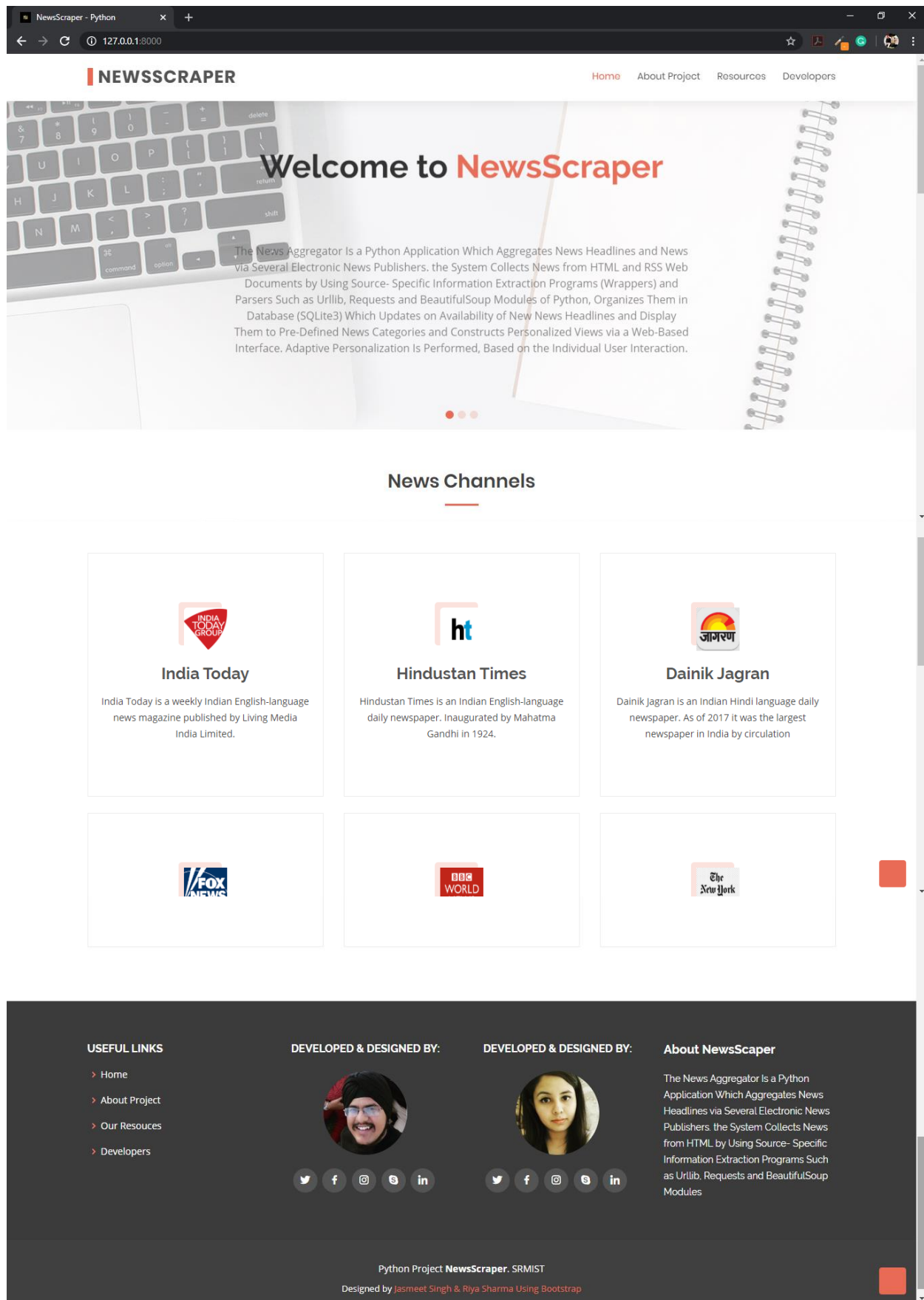
**News on Top of IndiaToday Website is fetched and saved in our database.**



**As seen in the red box the top news are also with link and image link in our database.**

	id	title	image	url
	Filter	Filter	Filter	Filter
1	265	Telugu actor Shivaji Raja suf...	https://akm-img-a-in.toss...	https://www.indiatoday.in...
2	266	Social distancing norms viola...	https://akm-img-a-in.toss...	https://www.indiatoday.in...

## Our WebApp UI:



## News Channel WebApp View:

IndiaToday News

Project Structure — django-proj

127.0.0.1:8000/news/indiatoday


☆

NEWSSCRAPER


HomeAbout ProjectResourcesDevelopers

India Today


Fetch News




Telugu actor Shivaji Raja suffers heart attack, hospitalised in Hyderabad




Social distancing norms violated: BJP MP writes to UP CM Yogi Adityanath to close liquor shops in red zone



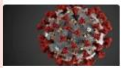
Covid-19 effect: Vijay Antony reduces his salary by 25 per cent to help producers




Refused Lamborghini by Mom, 5-yr-old caught driving to Cali to buy one. With \$3




At 69%, India's taxes on petrol, diesel highest in world




Has magic cure for Covid-19 already been found?




KKR CEO Venky Mysore on The Hundred investment: Will evaluate if they approach us




Brendon McCullum believes New Zealand should have its own BBL team




Priyanka Chopra dresses up in Rs 18k dress and tiara for Met Monday with niece at home



Elon Musk and Grimes name their son X Æ A-12. What does it mean?



Kriti Sanon calls sister Nupur her squishy, mushy quarantine. See cute pics



Priyanka Chopra shares photo with pet dog: I promise Gino loves my cuddles

We Have to Create a Index Page and Page for Each News Channel So We Used Bootstrap to Design & Make our UI and Make use of Database and Most Importantly Django to Host Website and Create a CGI Between Scraping, Database and Our Application.

On Clicking on the News Views We Get Redirected to the News Website:



## **CHAPTER 5**

### **CONCLUSION**

Online Aggregation System has been developed as a prototype to fetch & display news through the news publishing sites. The main thing that we want is that our website must be user friendly, any user who is capable to use web can visit to our website and perform their shopping, we also put some effort towards making our website as light as possible with use of Python Django Framework so that it could load easily in slow internet connection.

The Django framework gives us a simple and reliable way to create the News Scraping System. It provides powerful functionalities and concise syntax to help programmers deal with the database, Web Scraping and Making Comfortable for user to Use Other Modules, the web page and the inner logic. The experience of developing the group component in the system also helped me learning a lot of website development with Django. Within the Django framework, we have successfully accomplished the requirements of the system. The only limitation for this system is that it cannot just update anytime at runtime it may require user permission to Enter new News Feeds, it may still encounter problems during real time use. However, even if that happens, the flexibility of Django would provide a simple way to fix the problem, as well as add new features into the system.

## REFERENCES

### **Python Modules Documentation:**

<https://www.djangoproject.com/start/>

<https://pypi.org/project/beautifulsoup4/>

<https://pypi.org/project/urllib3/>

### **Bootstrap Modules Documentation:**

<https://getbootstrap.com/docs/4.4/getting-started/introduction/>

### **News Aggregators:**

[https://en.wikipedia.org/wiki/News\\_aggregator](https://en.wikipedia.org/wiki/News_aggregator)

[https://link.springer.com/chapter/10.1007/978-3-540-77471-6\\_10](https://link.springer.com/chapter/10.1007/978-3-540-77471-6_10)