Review scraper from scratch till deployment

Table of Contents

Preface **1**

Introduction **2**

Application Architecture **5**

Python Implementation **6**

Local system output **11**

Deployment **12**

Preface

It is a step by step guide for creating a web scraper, in this case, a review scrapper right from scratch and then deploying it to the heroku cloud platform. Text scrappers are extensively used in the industry today for competitive pricing, market studies, customer sentiment analysis, etc. This book takes a simple example of an online cell phone purchase and tries to explain the concepts simply, extensively, and thoroughly to create a review scrapper right from scratch and then its deployment to a cloud environment.

Web Scraping(Text)

1. Introduction:

Web scraping is a technique using which the webpages from the internet are fetched and parsed to understand and extract specific information similar to a human being. Web scrapping consists of two parts:

- Web Crawling➐ Accessing the webpages over the internet and pulling data from them.
- HTML Parsing➐ Parsing the HTML content of the webpages obtained through web crawling and then extracting specific information from it.

Hence, web scrappers are applications/bots, which automatically send requests to websites and then extract the desired information from the website output.

Let's take an example:

how do we buy a phone online?

1. We first look for a phone with good reviews
2. We see on which website it's available at the lowest price 3. We check whether it's delivered in our area or not
4. If everything looks good, then we buy the phone.

What if there is a computer program that can do all of these for us? That's what web scrappers necessarily do. They try to understand the webpage content as a human would do.

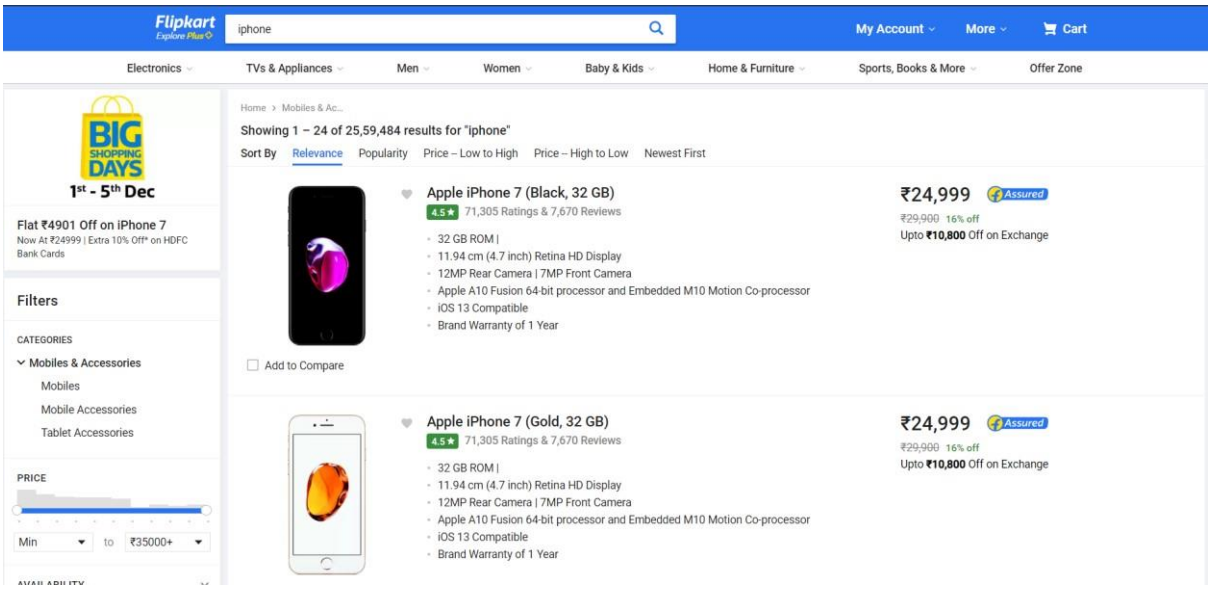
Other examples of the applications of web scrapping are:

- Competitive pricing.
- Manufacturers monitor the market, whether the retailer is maintaining a minimum price or not.
- Sentiment analysis of the consumers, whether they are happy with the services and products or not.
- To aggregate news articles.
- To aggregate Marketing data.
- To gain financial insights from the market.

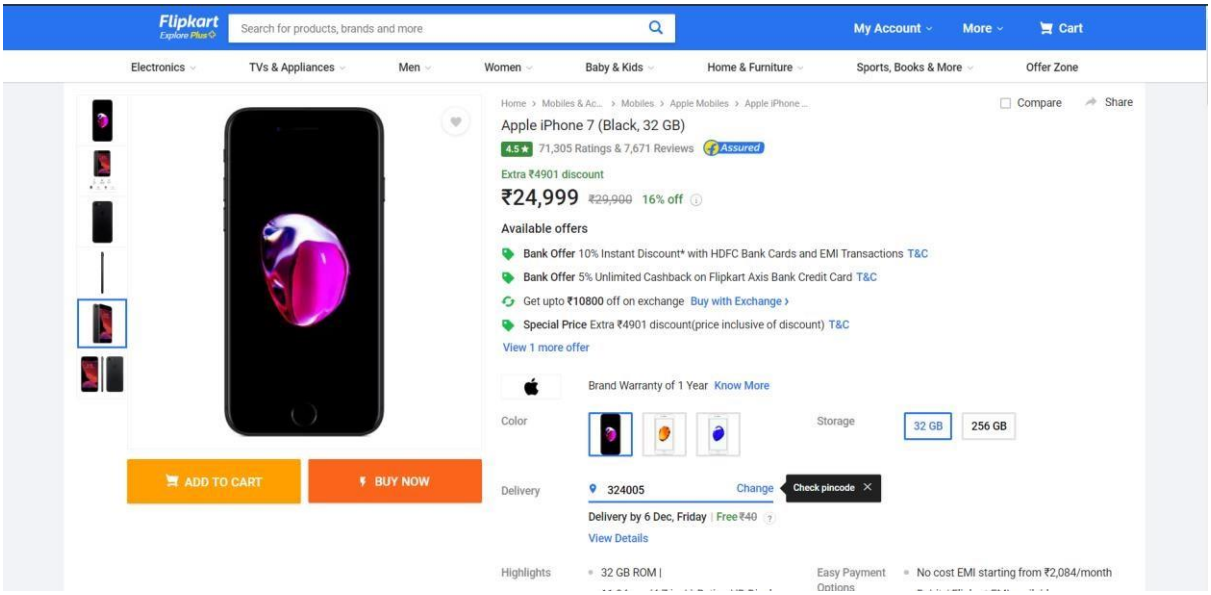
- To gather data for research.
- To generate marketing leads.
- To collect trending topics by media houses. And, the list goes on.

In this document, we'll take the example of buying a phone online further and try to scrap the reviews from the website about the phone that we are planning to buy.

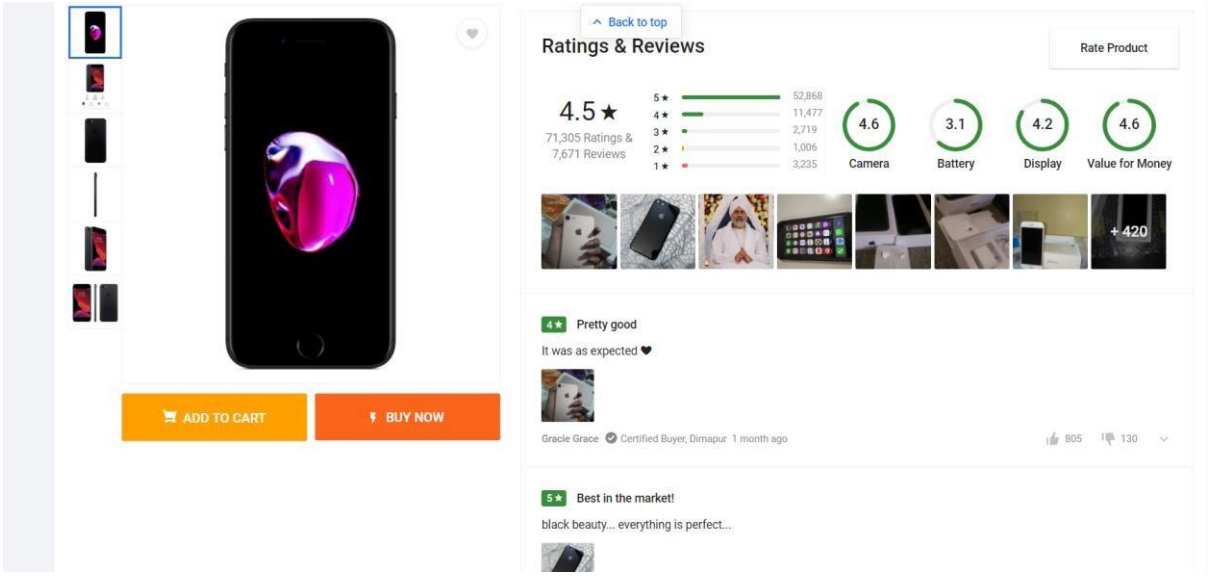
For example, if we open filpkart.com and search for 'iPhone', the search result will be as follows:



Then if we click on a product link, it will take us to to the following page:



If we scroll down on this page, we'll get to see the comments posted by the customers:



Our end goal is to build a web scraper that collects the reviews of a product from the internet.

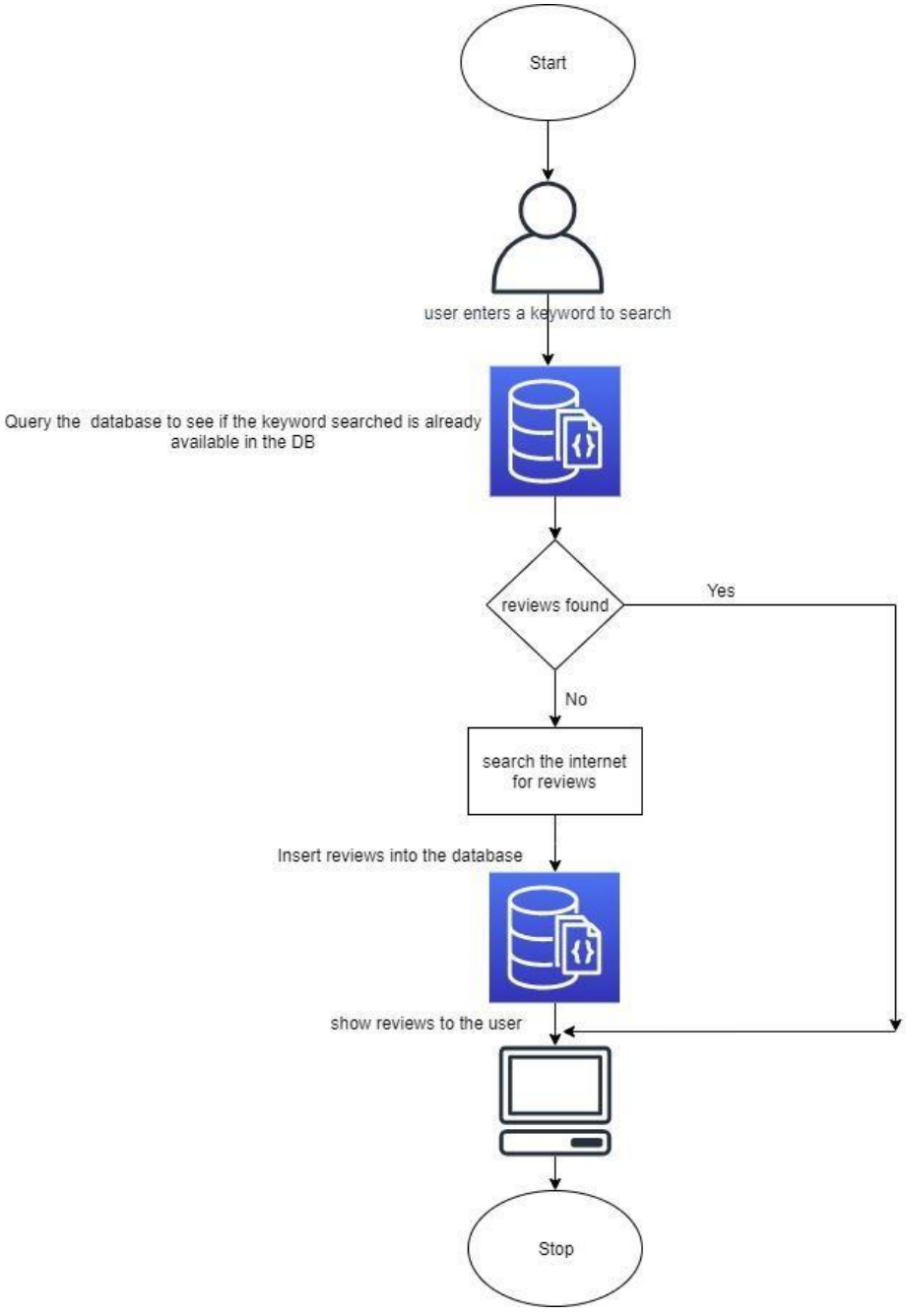
2. Prerequisites:

The things needed before we start building a python based web scraper are:

- Python installed.
- A Python IDE (Integrated Development Environment): like PyCharm, Spyder, or any other IDE of choice (Explained Later)
- Flask Installed. (A simple command: pip install flask)
- MongoDB installed (Explained Later).
- Basic understanding of Python and HTML.
- Basic understanding of Git (download Git CLI from https://gitforwindows.org/

Application Architecture

1. The architecture of the application is:



Python Implementation

Note: I have used PyCharm as an IDE for this project.

1. Let's create a folder called 'reviewScrapper' on our local machines.
2. Inside that folder, let's create two more folders called 'static' and 'templates' to hold the code for the UI of our application. Inside 'static', let's create a folder called 'css' for keeping the stylesheets for our UI.
3. Let's create a file called 'flask_app.py' inside the 'reviewScrapper' folder.
4. Inside the folder 'css', create the files: 'main.css' and 'style.css'. The files are attached here for reference.


main.css


style.css

5. Inside the folder 'templates', create three HTML files called: 'base.html','index.html', and 'results.html'. The files are attached here for reference.

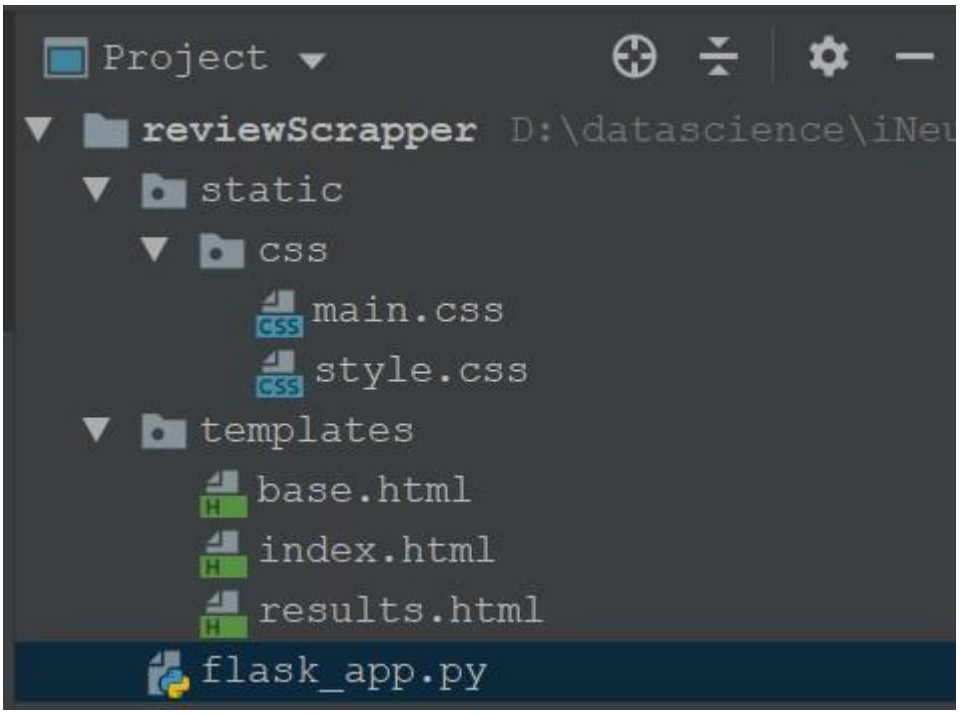

base.html


index.html


results.html

   - base.html❼ It acts as the common building block for the other two HTML pages.
   - index.html❼ Home page of our application.
   - results.html❼ Page to show the reviews for the searched keyword.

6. Now, the folder structure should look like:



7. Now, let's understand the flow:
   a) When the application starts, the user sees the page called 'index.html'.
   b) The user enters the search keyword into the search box and presses the submit button.
   c) The application now searches for reviews and shows the result on the 'results.html' page.
8. Understanding flask_app.py.

   a) Import the necessary libraries:

```
from flask import Flask, render_template,
request,jsonify from flask_cors import CORS,cross_origin
import requests
from bs4 import BeautifulSoup as bs from
urllib.request import urlopen as uReq
import pymongo
```

   b) Initialize the flask app

```
app = Flask(__name__)  #
initialising the flask app with the
name
'app'
```

c) Creating the routes to redirect the control inside the application itself. Based on the route path, the control gets transferred inside the application.

```
@app.route('/',methods=['POST','GET']) # route with allowed
methods as POST and GET
```

d) Now let's understand the 'index()' function.

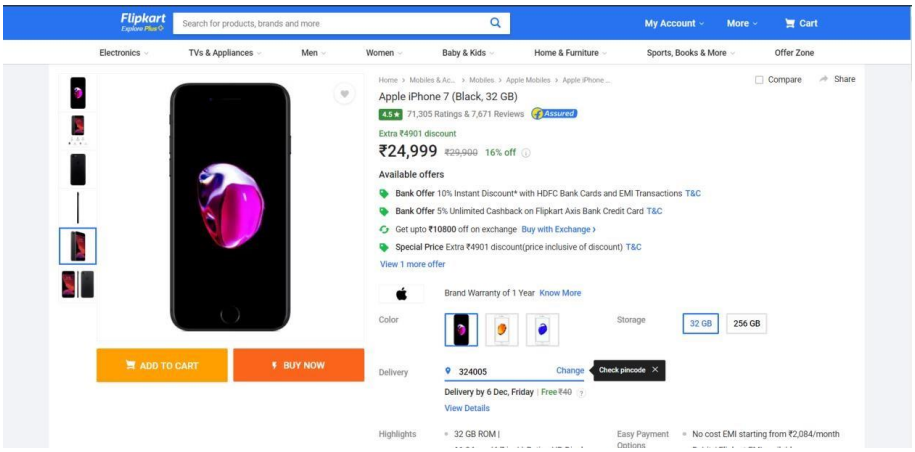
i. If the HTTP request method is POST(which is defined in index.html at form submit action), then first check if the records for the searched keyword is already present in the database or not. If present, show that to the user.

```
dbConn = pymongo.MongoClient("mongodb://localhost:27017/")
# opening a connection to Mongo
db = dbConn['crawlerDB'] # connecting to the database
called crawlerDB
reviews = db[searchString].find({}) # searching the
collection with the name same as the keyword if
reviews.count() > 0: # if there is a collection with
searched keyword and it has records in it
    return render_template('results.html',reviews=reviews)
# show the results to user
```

ii. If the searched keyword doesn't have a database entry, then the application tries to fetch the details from the internet, as shown below:

```
flipkart_url = "https://www.flipkart.com/search?q=" +
searchString # preparing the URL to search the product on
Flipkart
uClient = uReq(flipkart_url) # requesting the webpage from
the internet
flipkartPage = uClient.read() # reading the webpage
uClient.close() # closing the connection to the web server
flipkart_html = bs(flipkartPage, "html.parser") # parsing
the webpage as HTML
```

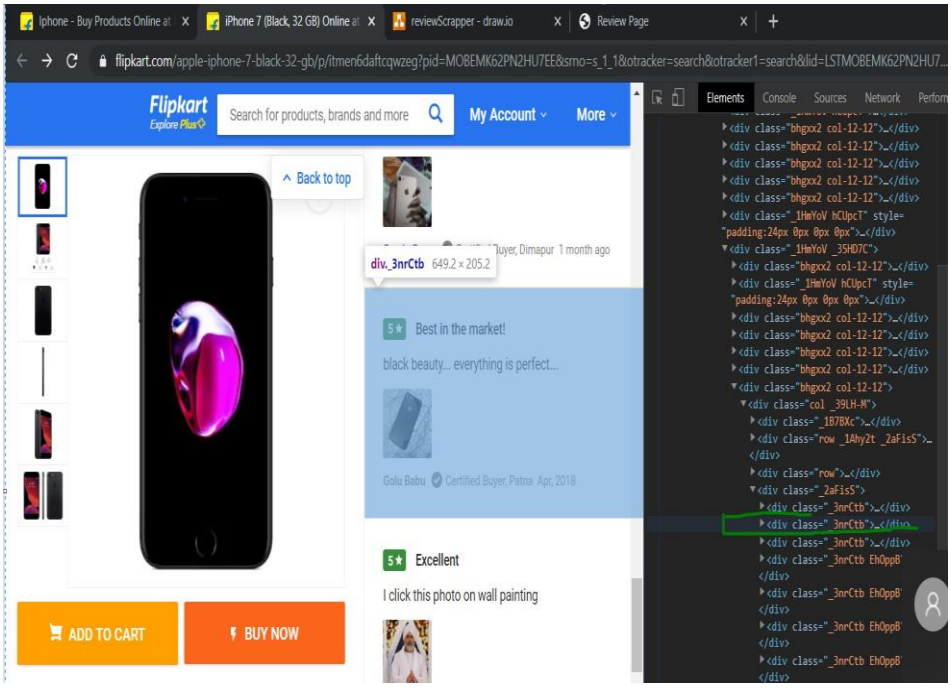
iii. Once we have the entire HTML page, we try to get the product URL and then jump to the product page. It is similar to redirecting to the following page:



The equivalent Python code is:

```
productLink = "https://www.flipkart.com" +
box.div.div.div.a['href'] # extracting the actual product
link
prodRes = requests.get(productLink) # getting the product
page from server
prod_html = bs(prodRes.text, "html.parser") # parsing the
product page as HTML
```

iv. On the product page, we need to find which HTML section contains the customer comments. Let's do inspect element(ctrl+shift+i) on the page first to open the element-wise view of the HTML page. There we find the tag which corresponds to the customer comments as shown below:



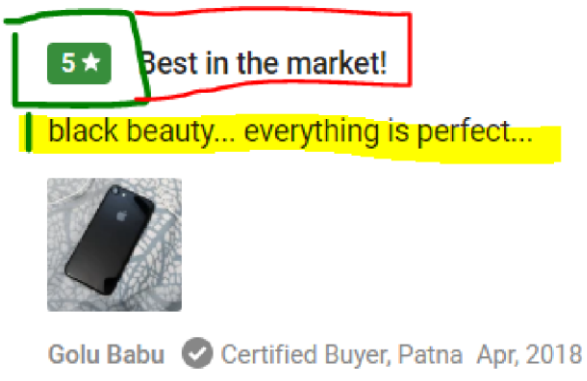
Python code for implementing the same is:

```
commentboxes = prod_html.find_all('div', {'class':
"_3nrCtb"}) # finding the HTML section containing the
customer comments
```

v. Once we have the list of all the comments, we now shall extract the customer name(in grey), the rating(in green), comment heading(marked in red), and the customer comment(highlighted in yellow) from the tag.

The Python code for the same is:

```
reviews = [] # initializing an empty list for reviews #
iterating over the comment section to get the details of
the customer and their comments for commentbox in
commentboxes:
```



```
    try:          name = commentbox.div.div.find_all('p',
{'class':
'_3LYOAd _3sxSiS'})[0].text
     except:
name = 'No Name'
     try:          rating =
commentbox.div.div.div.div.text
     except:          rating
= 'No Rating'
     try:                              commentHead   =
commentbox.div.div.div.p.text                 except:
commentHead  =  'No  Comment  Heading'          try:
comtag = commentbox.div.div.find_all('div',
{'class': ''})
        custComment = comtag[0].div.text
except:          custComment = 'No Customer
Comment'
```

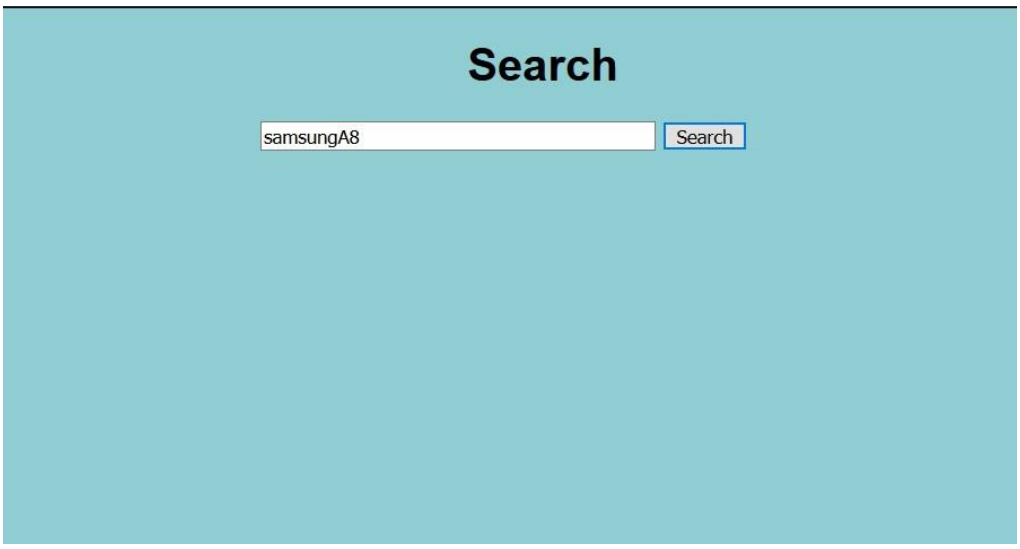
If you notice, the parsing is done using the try-except blocks. It is done to handle the exception cases. If there is an exception in parsing the tag, we'll insert a default string in that place.

vi. Once we have the details, we'll insert them into MongoDB. After that, we'll return the 'results.html' page as the response to the user containing all the reviews. The python code for that is:

```
mydict = {"Product": searchString, "Name": name, "Rating":
rating, "CommentHead": commentHead,
              "Comment": custComment} # saving that detail
to a dictionary
    x = table.insert_one(mydict) #insertig the dictionary
containing the rview comments to the collection
reviews.append(mydict) #  appending the comments to the
review list
return render_template('results.html', reviews=reviews) #
showing the review to the user
```

e) After this, we'll just run our python app on our local system, and it'll start scraping for reviews as shown below:

Home Page:



Search Results:

REVIEWS

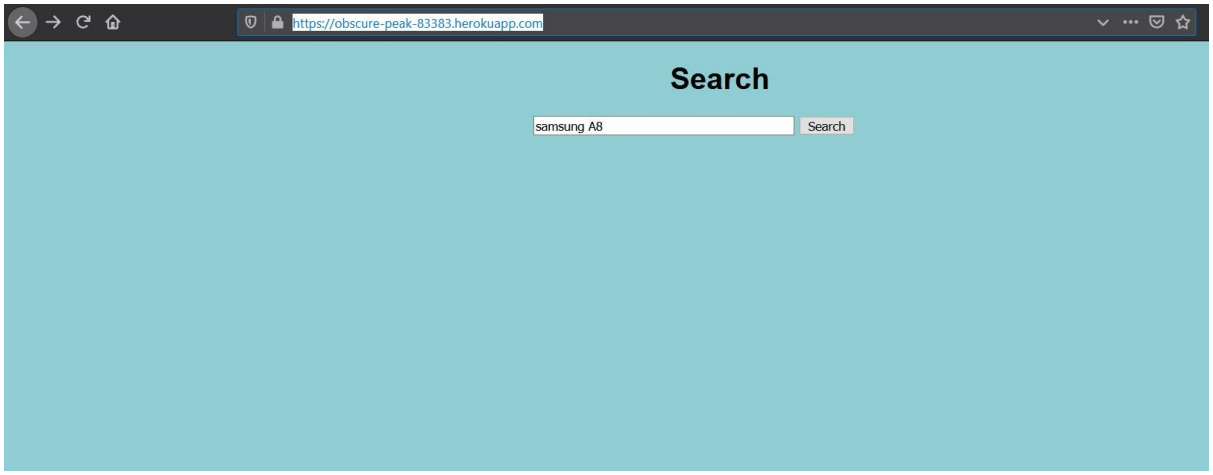
Product	Name	Rating	Comment Heading	Comments
samsungA8	Soumya Guha Roy	4	Beauty & the beast combined.	I have been using the device since last 3 days. Here is my review Pros: 1. Excellent display 2. Fast focus and fast capture camera 3. Nice UI 4. Decent battery backup (after 24 hours, battery remains ~30% with medium usage. 5. Good performance. 6. Unread notification count is spot on, like we have in iOS. 7. The metallic design is superb, and feels premium. Go for the gold one, than the white. The white one looks plast...
samsungA8	deepak pagar nashik	5	superb ,wesome,stylish, powerful ,metal finishing with complete security handset by samsung	i bought this stunning metal body slim phone before 15 days ,and i m really happy to use it ,this cell having 32 gb internal memory due to this not i used my memory card . its hybrid sim phone but no problem because u have more internal capacity . Samsung Galaxy A8 is a awesome device again from Samsung. A8 could have had Dual SIM along with SD card (u can

Deployment

The Python app that we have developed is residing on our local machine. But to make it available to end-users, we need to deploy it to either an on-premise server or to a cloud service. Heroku is one such cloud service provider. It is free to use(till 5 applications).
We'll deploy this application to the pivotal cloud, and then anybody with the URL can then consume our app.

URL: https://reviewscrapper-fantastic-raven-rj.cfapps.io/

Final Result:



Thank You!