PyCK Project Report

COVID SCRAPER

Introduction

Problem Statement:

The project acts as a tool to collect reliable and official data from different websites and project them at a single place for efficient and easier access.

Importance:

The Covid-19 pandemic took the whole world with a hit. With the chaotic atmosphere during the second wave, and the constant fight for oxygen cylinders and hospital beds, it is evident that knowledge of information is everything. While there are resources on the web for every need, a common being needs easy and organized access to it. Thus, the project caters to scraping the content of reliable and official webpages and presenting them together in a single place for easier access.

Motivation:

The website is a work in the Data Science field, catering to statistics regarding the Covid19 situation in India/world. I have been into web development for quite some time, and have developed web pages using PHP as a backend language. Having learnt about the immense power of Python and its libraries, it was only the best thing to try its capabilities in web development. Web scraping has always been in my mind, since it's incredible how easily data can be fetched and utilized, and there could be no better area of development than Covid 19, as there is such a vast collection of information available, plus a dire need of organizing the same.

Implementation Details

Languages and Frameworks used:

Front-end:

- HTML for the basic skeleton of the webpage.
- Base CSS for designing and appearance.
- Bootstrap 5 framework for a quick design and customizations.
- FontAwesome for icons and images.

Back-end:

- Django (a Python-based web framework) for rapid development and clean, pragmatic design in the model-template-views architectural pattern.
- BeautifulSoup is a Python library that is used for pulling data out of HTML and XML files.
- Requests is a Python library used for making HTTP requests.
- Html5lib, lxml, html.parser are python libraries used for parsing the scraped HTML content.
- Maps JavaScript API provided by Google for incorporating an interactive world-map.

Features:

NavBar

The navbar is a pure implementation of Bootstrap 5 with the icons included from FontAwesome for better appearance. There are in-page hyperlinks to id elements through each icon. Also, there is an attached Modal to the bell-icon, for recent news of Covid 19 pandemic in India which is scraped from the news sections of the Google Search “Coronavirus India”.
(https://www.google.com/search?q=Coronavirus+india&sxsrf=ALeKk00ZaqgrW0sY9HyDTHtZOkND-0wRYw:1626259516219&source=lnms&tbm=nws&sa=X&ved=2ahUKEwiYiJvmsOLxAhU34HMBHa0JAhsQ_AUoAXoECAEQAw&biw=1366&bih=625)

Header

The header is a pure implementation of base CSS, with a linear gradient on top of the background image. The cards are an implementation of BootStrap 5 with column

specifications. The data displayed on the cards is further scraped from Covid 19 world information provided by (https://www.worldometers.info/coronavirus/)

World Map

I always had an intention of portraying the total cases data on a world map. This is fulfilled by Google Maps API. The backend of the map runs on JavaScript, while the inclusion to the webpage has been done using a mere tag. An API key was generated for Maps JavaScript API, which helps in including Google Maps for web-related projects. The JavaScript code is very flexible with easy access to any region world-wide using particular attributes. The total cases for each country is being scraped from (https://www.worldometers.info/coronavirus/)

India (State) Information

The end table is dynamically created by iterating through the list of data scraped from (https://www.mygov.in/corona-data/covid19-statewise-status/) with a proper display of state/UT name, confirmed, recovered and deceased numbers. The designing has been done via base CSS.

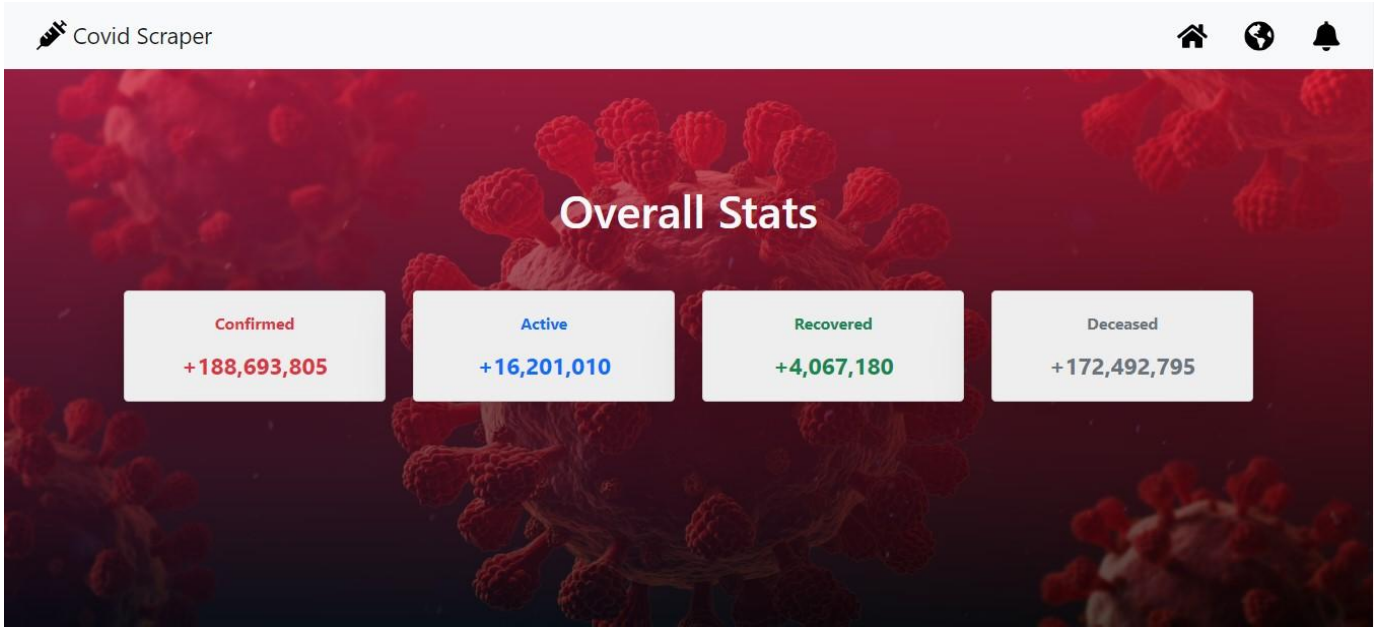
Problems Faced:

As both Django and BeautifulSoup are completely new libraries for me, there were many problems I faced during the development period, and work-arounds I had to implement. Some of these includes:

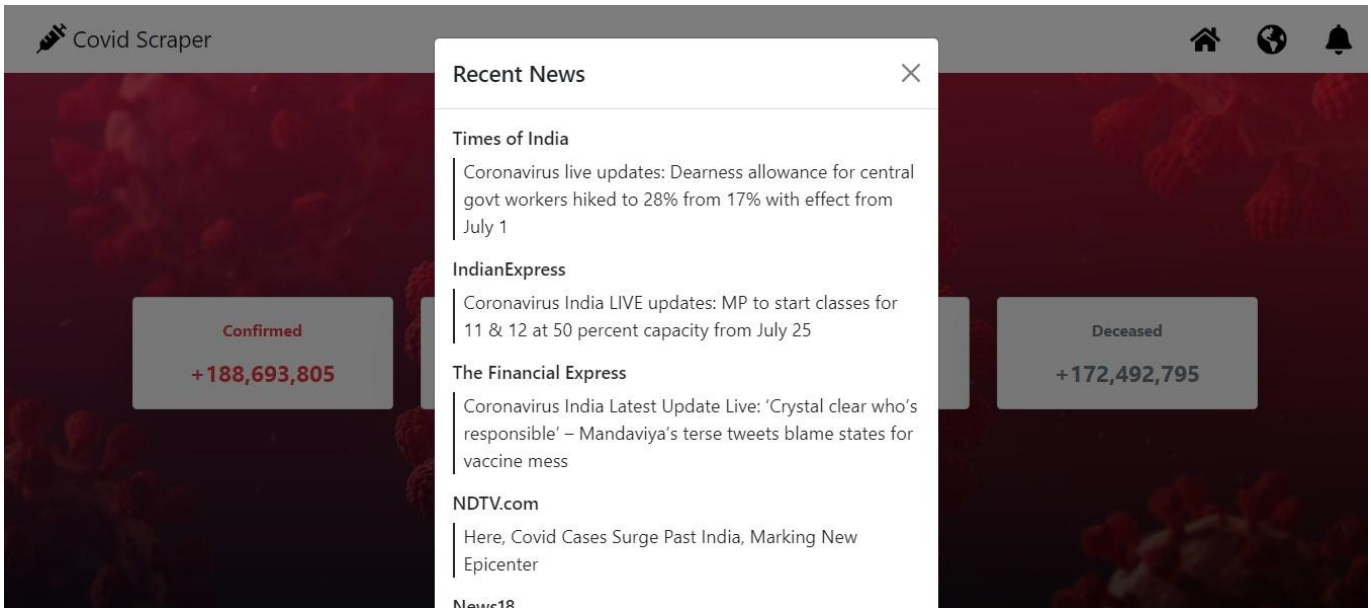
- Django itself is a completely different backend language as compared to PHP, with a different work environment and a model-template-view architectural pattern. So, there was an initial hurdle of understanding the whole system. However, it turns out that it's a very organized method for building large web applications.
- While watching BeautifulSoup tutorials I was under the impression that the scraped HTML content would be identical to the content which can be viewed by Inspect Element. However, that's not the case. Most frequently, the divs and classes visible in Inspect were not scraped as it is, and therefore I had to work with what had been scraped rather than what could be seen on Inspect.
- I later found that while it's very easy to scrape basic HTML webpages, good websites, nowadays, use JavaScript for activating different parts of their page, and hence, many a times, the scraped HTML content would contain the but not its text content.

- While scraping data using BeautifulSoup and creating a list of it in Python, there are elements with “None” content, and in the loop it’s an Attribute Error to access their text using .get_text attribute. Hence, I had to include a try...except block for such cases.
- The backend code for GeoChart based on Google was written by-default in JavaScript, of which I have scarce knowledge. So the data injected to it, the list in Python, had to be converted to a JSON file. This file then had to be appropriately read into the script which was present in a .html file using interpolation of a Django variable.

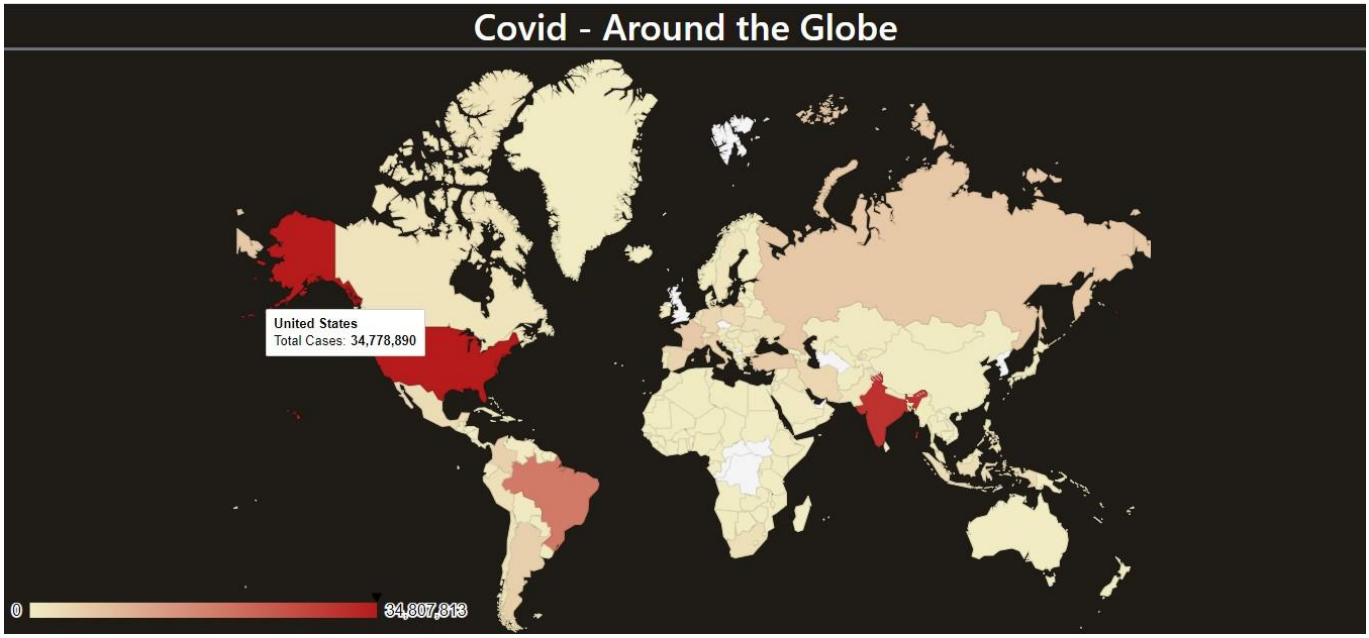
Screenshots



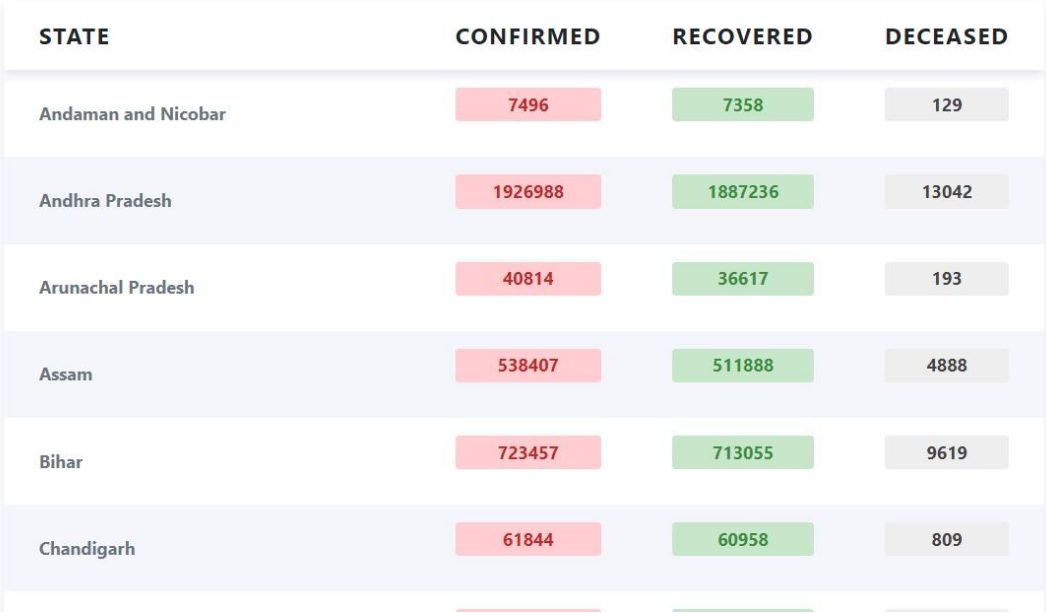
Navbar along with the header with cards layout



Modal (BootStrap 5) for display of recent news on clicking the bell-icon.



A display of Covid cases around the globe using Maps JavaScript API provided by Google on an interactive map.



STATE	CONFIRMED	RECOVERED	DECEASED
Andaman and Nicobar	7496	7358	129
Andhra Pradesh	1926988	1887236	13042
Arunachal Pradesh	40814	36617	193
Assam	538407	511888	4888
Bihar	723457	713055	9619
Chandigarh	61844	60958	809

A display of the state-wise information in a tabular form with an odd-even color coding scheme.

Conclusion

The project in all was a great learning experience, as it acted as an introduction to Django which I would now continue to use as a backend language, and a glimpse into the world of Data Science via web scraping using Beautiful Soup. The project has infinite scope of expanding, to the length of becoming a full portal which could act as a go-to page for any and all queries. I also had the intention of using the CoWin API for displaying vaccine availability in different regions, but as it turned out, the API key procurement required a lot of registration and company affiliation. All-in-all, given the 2-3 weeks time frame, I was able to sync my knowledge in web development with the strength of Python and its libraries, to build this web scraped website, portraying Covid-19 related information of India and the World.