**Title: Web Scraping**

**Team member details: PES1202101188 - Abhinav R Bharadwaj**

**PES1202101449 – Anshul Dasyam**

**Abstract :**

This web application helps content creators to view the statistics of YouTube channels and view comparisons between subreddits/submissions on Reddit. We used Selenium and BeautifulSoup for extracting data from YouTube and PRAW for extracting data from Reddit and Plotly served as a great tool to visualize the data in form of various graphs. We used Dash to provide a beautiful UI for users to interact with. Dask links the front-end and back-end (Python3). As of now, we are performing data analytics on 2 social platforms.

**Table Of Contents:**

[Introduction 4](#_Toc94357605)

[Design/Implementation 5](#_Toc94357606)

[Testing 6-8](#_Toc94357607)

[Result and Analysis 9-14](#_Toc94357608)

[Conclusions & future enhancements 15](#_Toc94357609)

[References 16](#_Toc94357610)

**Introduction:**

Web scraping[[1]](#endnote-1) is a process of automating the extraction of data in an efficient and fast way. With the help of web scraping, you can extract data from any website, no matter how large is the data, on your computer. When we extract data with the help of a web scraping tool, we can save that data in the format of ‘CSV’ or ‘JSON’.

Web Scraping has a variety of applications in industries such as Data Science, E-Commerce, Media, Finance, Research. Our project helps employees in the Entertainment Industry.

Our program allows users to extract data from popular social media sites, namely Reddit and YouTube. Using this data, we can gather information about YouTube channels (views of their videos, like/dislike ratio, date of upload, comments, etc.) or Reddit subreddits (Daily posts, users, comments, menu, bio, etc.). Our project can be used by content creators on YouTube to compare themselves to rival channels.

Extracting data from Reddit is made possible by using a python wrapper for Reddit API known as PRAW[[2]](#endnote-2). Extracting data from YouTube was done using Selenium[[3]](#endnote-3) and BeautifulSoup[[4]](#endnote-4).

After extraction of data, we used Plotly[[5]](#endnote-5) to represent the data in a graph format (Bar, Scatter, Pie, etc.). We feel that Plotly is better than matplotlib in this scenario because it gives us a better interactive experience. We used Dash for front-end purposes rather than Tkinter[[6]](#endnote-6).

**Design/Implementation:**

**File Structure:**

- app.py

- index.py

- apps

|-- \_\_init\_\_.py

|-- app1.py

|-- app2.py

-assets

|-- style.css

Index.py --- <https://github.com/Anshul1563/Webscraping/blob/main/index.py>

app.py --- <https://github.com/Anshul1563/Webscraping/blob/main/app.py>

app1.py --- <https://github.com/Anshul1563/Webscraping/blob/main/app1.py>

app2.py --- <https://github.com/Anshul1563/Webscraping/blob/main/app2.py>

style.css --- <https://github.com/Anshul1563/Webscraping/blob/main/style.css>

\_\_init\_\_.py is empty

**Testing:**

In the initial stages of our project, we were using Matplotlib to display our graphs but we decided to shift to Plotly because we found that customization with Plotly was much easier to do and it was more interactive as well. The graphs we have got using matplotlib also had a really low resolution and we couldn’t see some of our labels and numbers. Working with Plotly was much easier and was also compatible with Dash[[7]](#endnote-7), which brings us to the next topic.

In Tkinter, we cannot develop web applications, but we can easily do so with Dash. Dash is a python framework created by Plotly for creating interactive web applications. Dash, you don’t have to learn HTML, CSS, and JavaScript to create interactive dashboards, you only need python. Dash is open source and the applications built using this framework are viewed on the web browser. We can use bootstrap components[[8]](#endnote-8) which are made especially for Dash to customize our applications.

Coming to our project, We have found that our main problem was that our APIs were slow in extracting and parsing information from the respective sites. So we had to optimize our code, try out various APIs to reduce the time taken to execute our code.

Initially, we had planned to use a module called Requests to retrieve the HTML source code but we were able to retrieve only 10-20 YouTube videos as YouTube employs a lazy loading feature (Doesn’t load the whole page unless we scroll to the end of the page). We had to imitate the scrolling action using Selenium to reach the end of the page and then retrieve the HTML.

Using Selenium, it took a lot of time to parse through the HTML source code because it had to parse through each line, but after integrating BeautifulSoup with Selenium, we were working with only part of the source which was useful for our application, therefore optimizing our application.

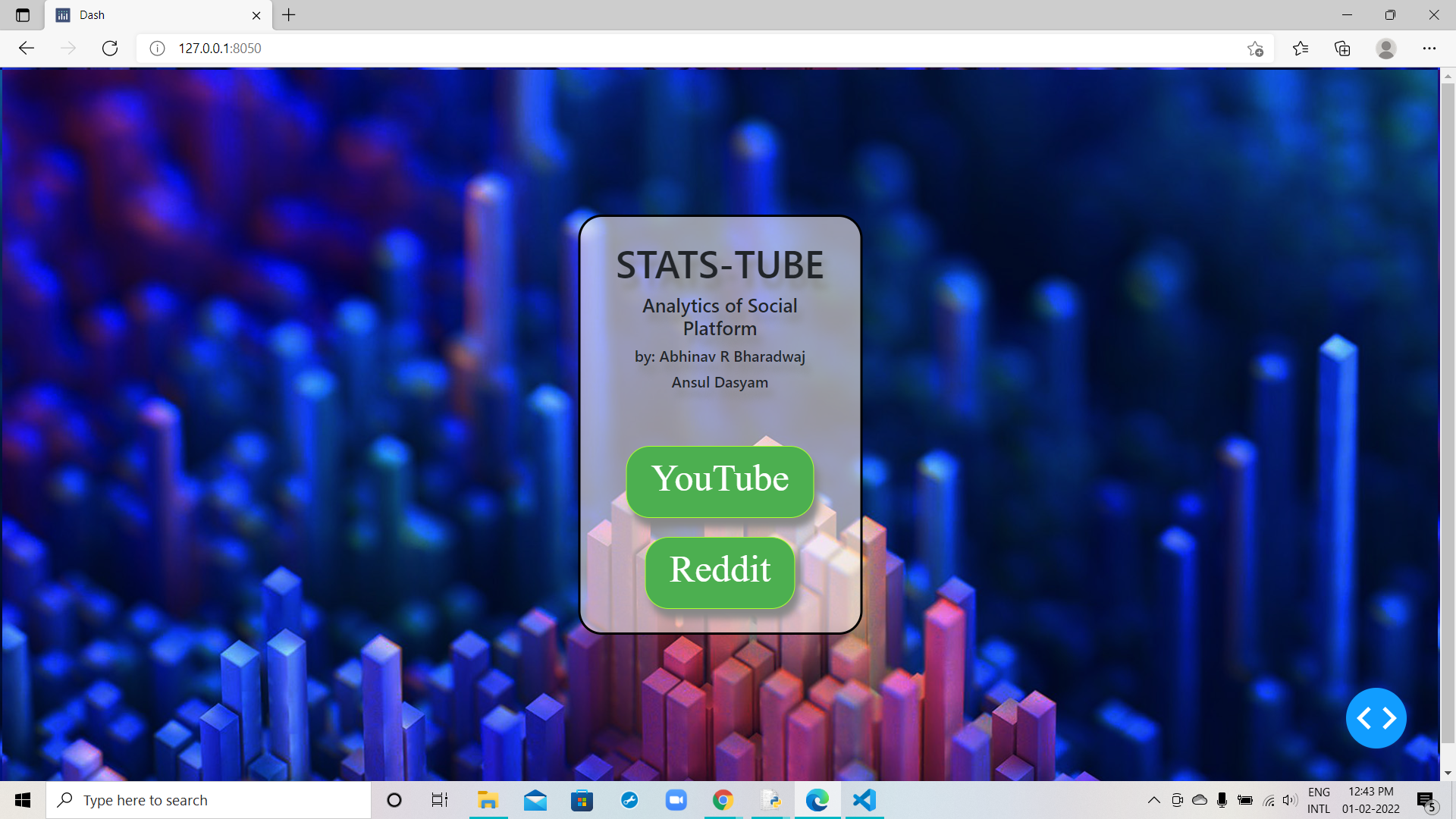
With Reddit’s PRAW API it could only return 1000 posts per subreddit, therefore we had to switch to a different API (Pushshift, PSAW) to work with a larger dataset. But Pushshift’s limitation was that it could handle only 100 requests per second thereby increasing the time taken to run our application. In the end, we decided to use a combination of both PRAW and Pushshift for certain parts of our application.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.No | Input | Test Case | Expected Output | Result |
| 1 | <https://www.youtube.com/c/veritasium/videos> | The number of videos extracted is equal to the actual number of videos | 328 Videos | 328 Videos  Pass |
| 2 | <https://www.youtube.com/watch?v=IgF3OX8nT0w>  The Most Powerful Computers You've Never Heard Of | Views of the video extracted are equal to the actual number of views of the video | 6.1M | 6,114,673 views  Pass |
| 3 | <https://www.youtube.com/c/veritasium/videos>  These are the asteroids you are worried about | Timeframe extracted is equal to the actual Timeframe the video was published in. | 1 year ago | 1 year ago  Pass |
| 4 | <https://www.reddit.com/r/programming/comments/sh4835/a_ux_designer_walks_into_a_tesla_bar/> | The number of upvotes extracted is equal to the number of upvotes of the post | 1326 upvotes | 1326 upvotes  Pass |
| 5 | <https://www.reddit.com/r/programming/comments/sh4835/a_ux_designer_walks_into_a_tesla_bar/> | The number of comments extracted is equal to the number of comments on the post | 406 comments | 406 comments  Pass |
| 6 | <https://www.reddit.com/r/science/comments/sh6g3f/new_research_suggests_that_ancient_trees_possess/> | The flair of the post extracted is the same as the actual flair of the post | Environment | Environment  Pass |

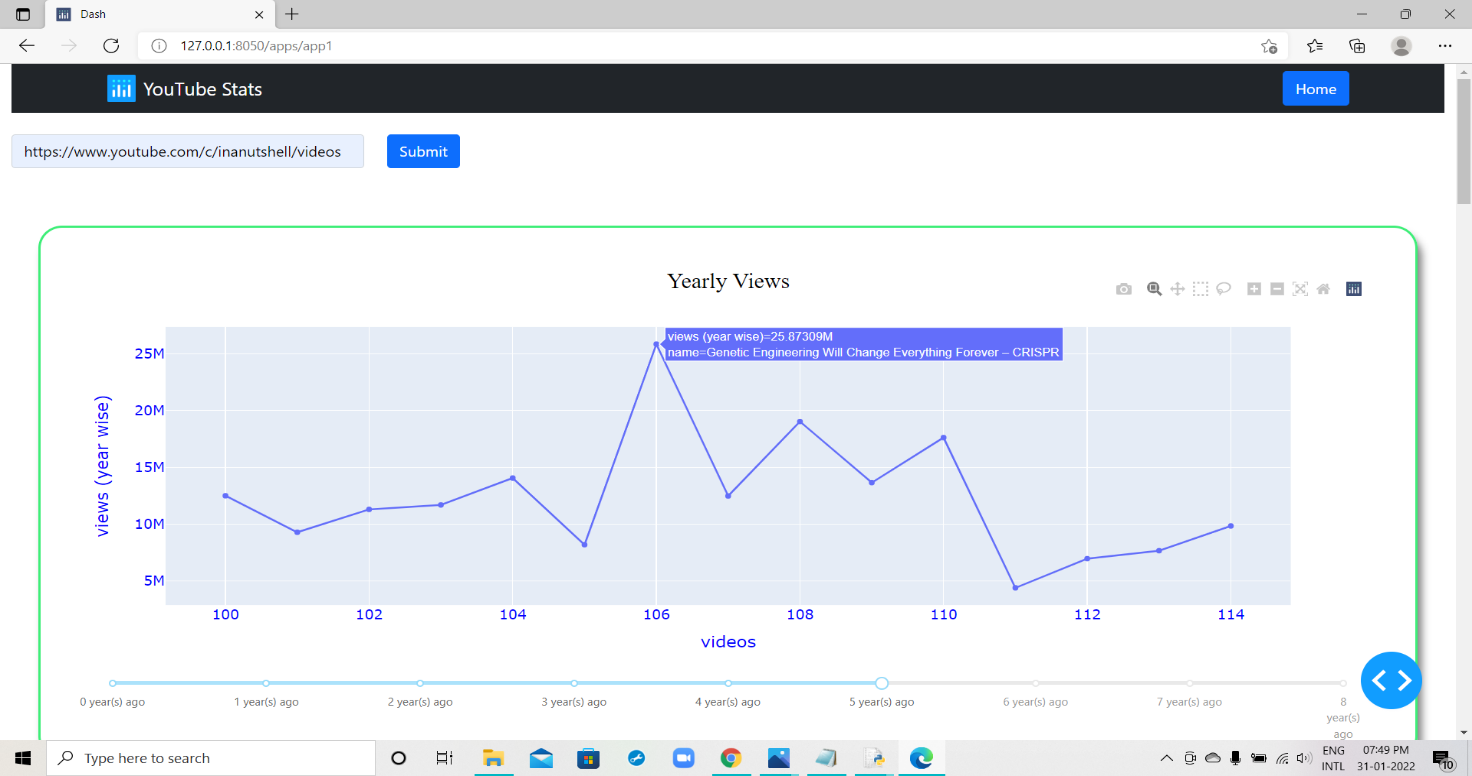
**Result and Analysis:**



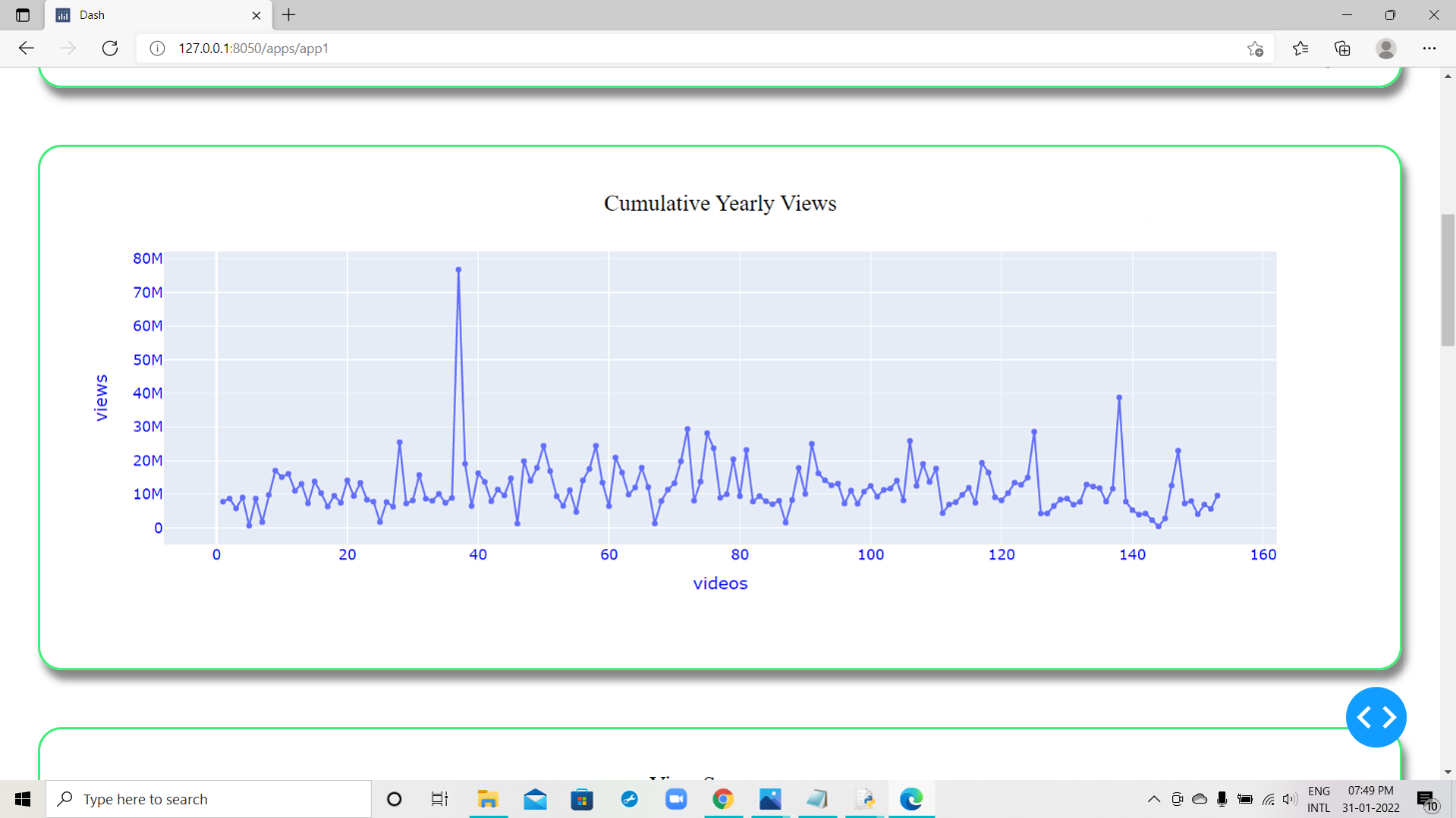
****

Home Page

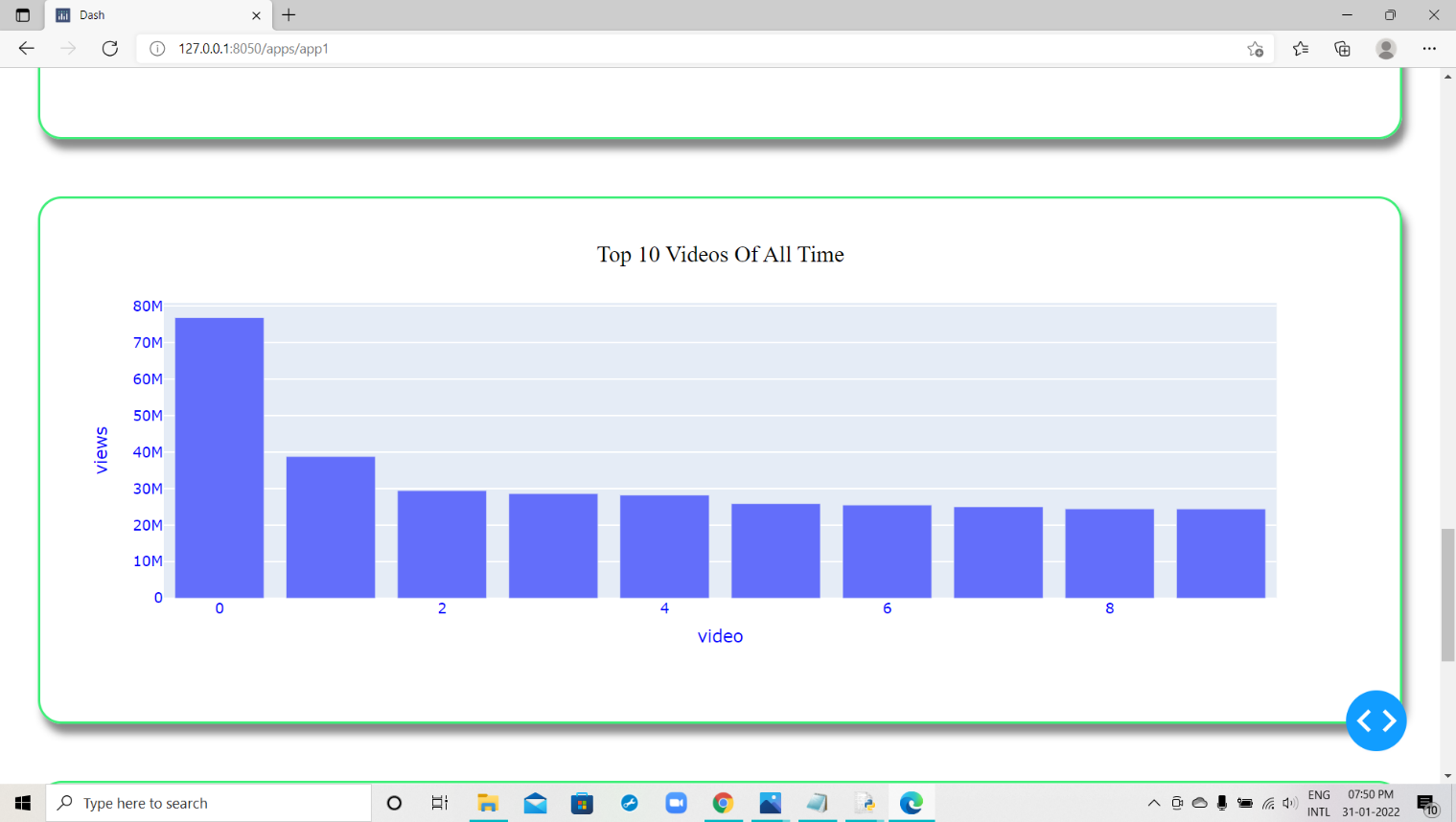




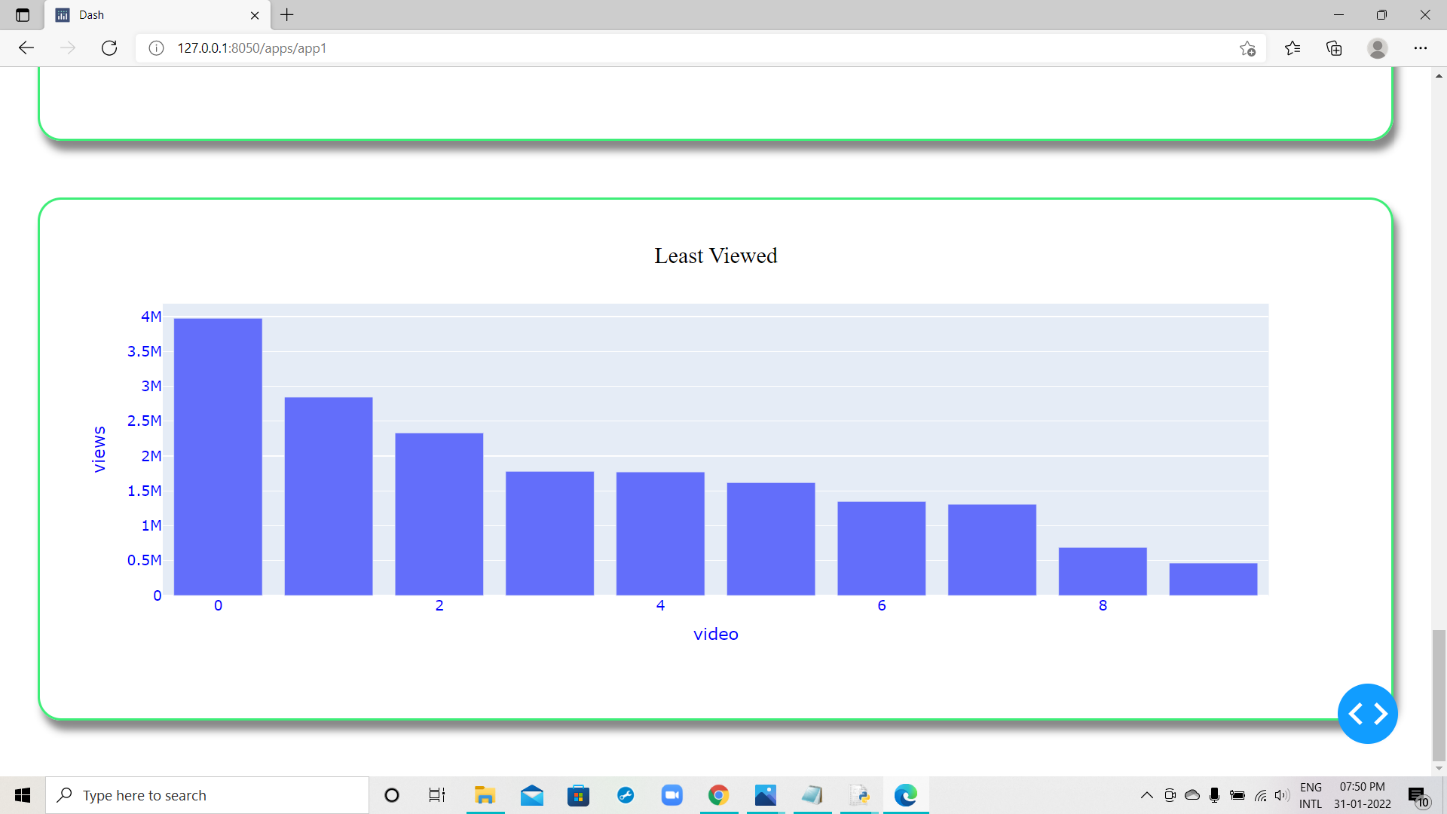
Total views of videos corresponding to the year it was released



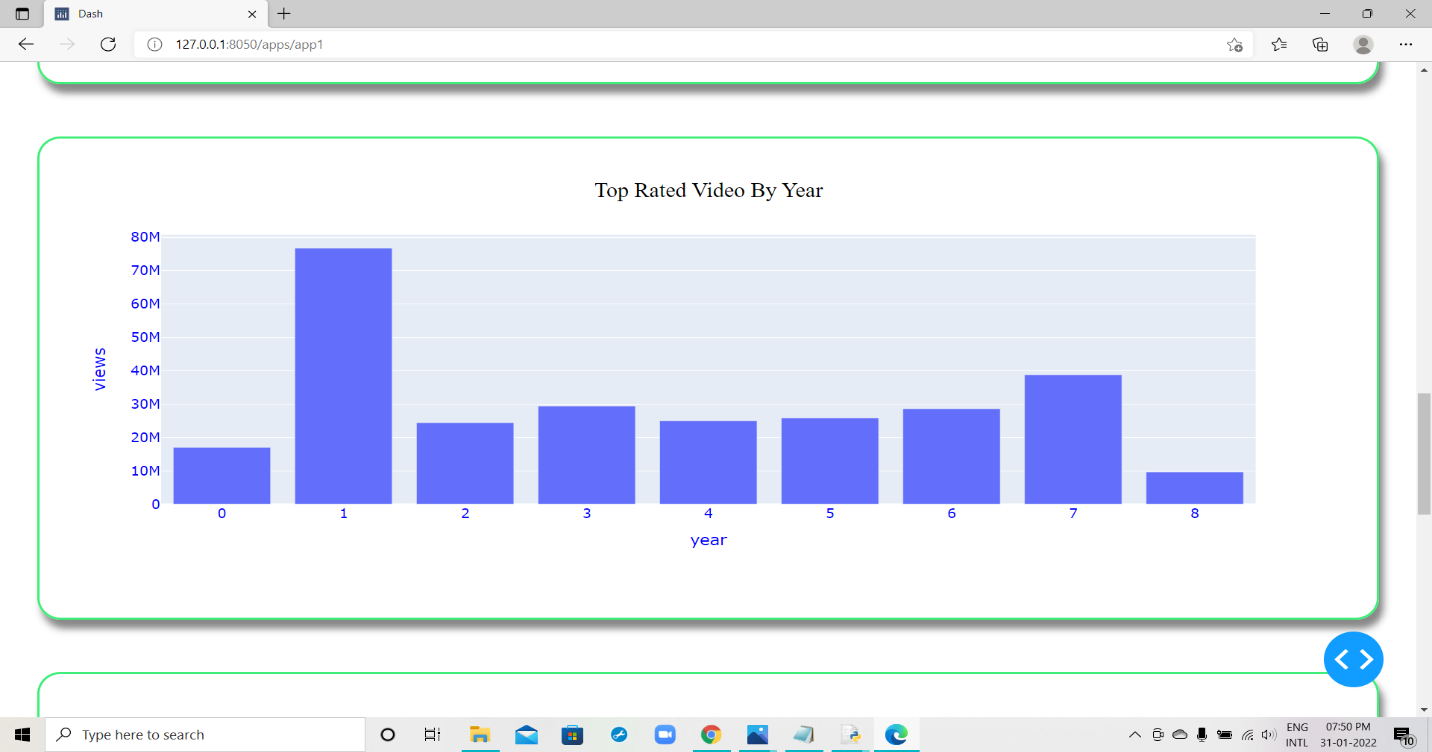
Total views of videos (Compact View)



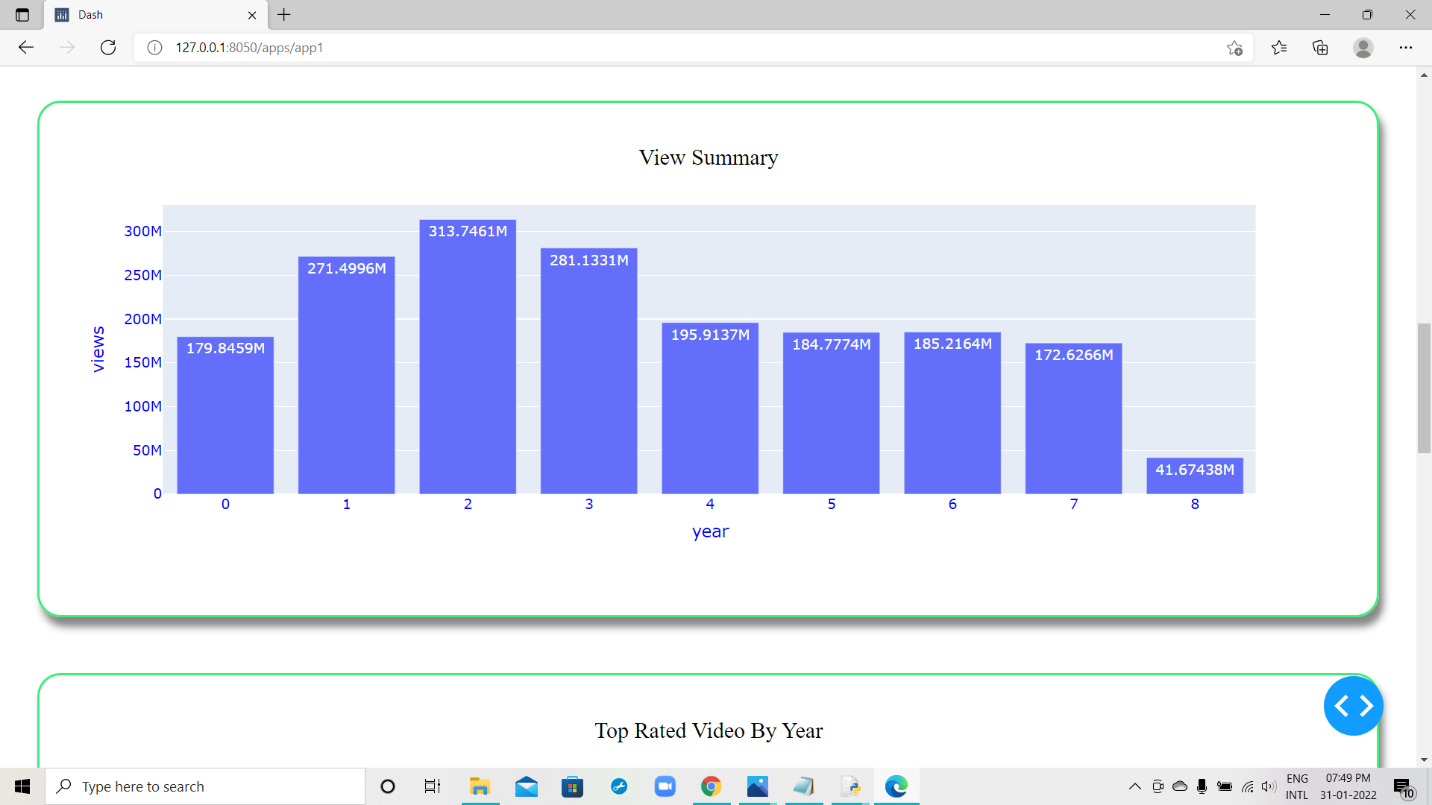
Best performing videos of the channel



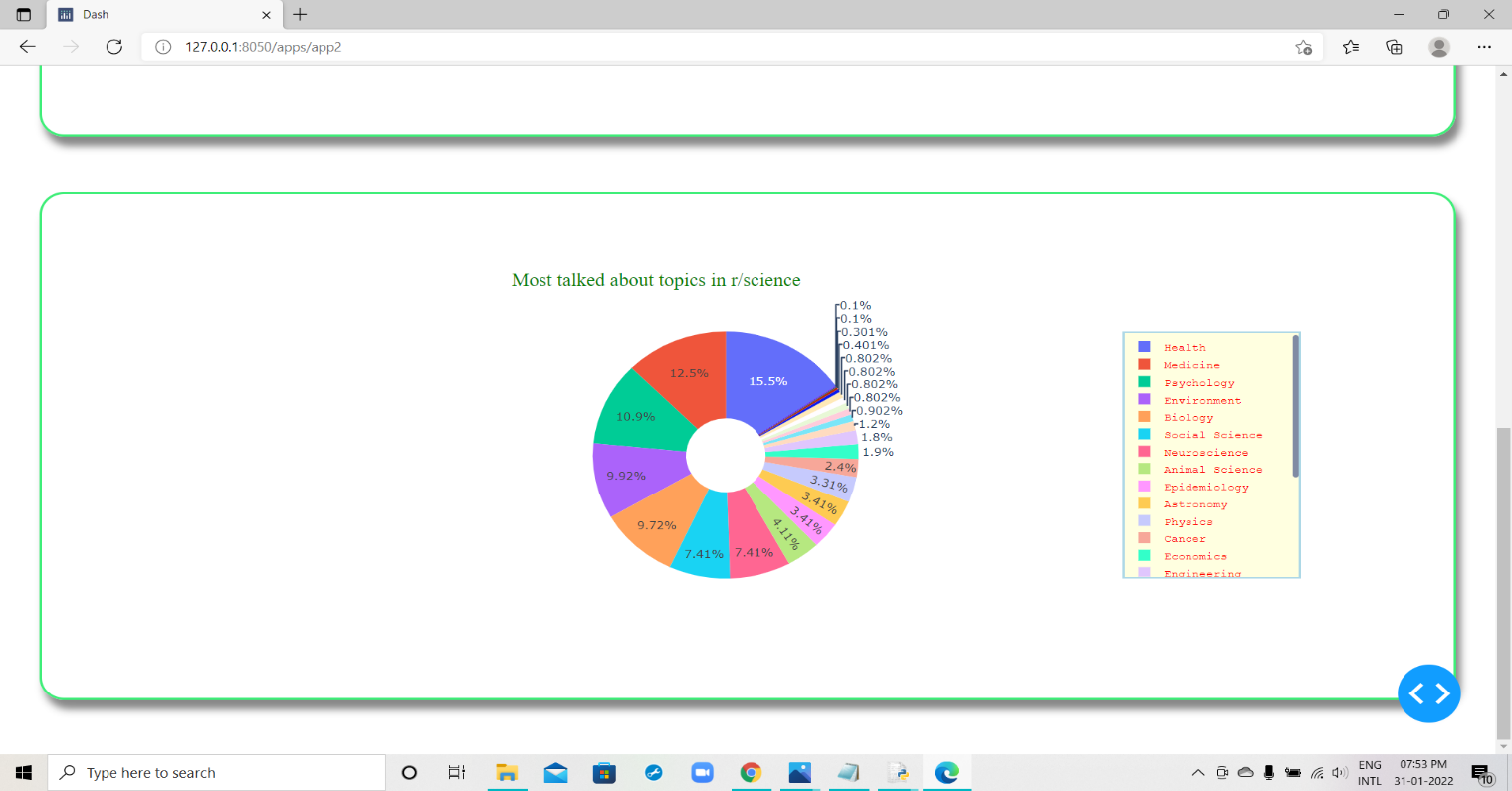
Worst performing videos of the channel



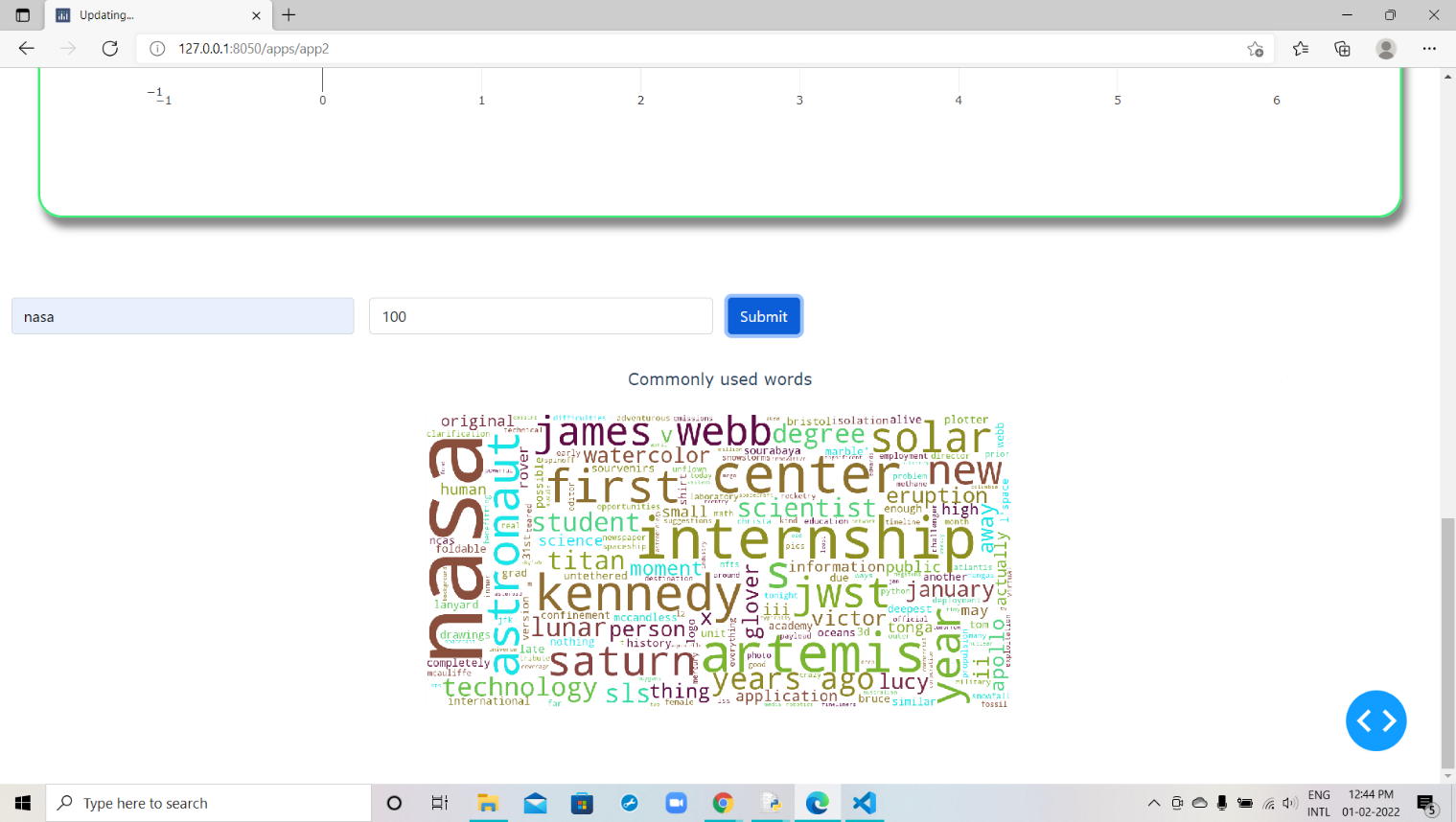
Best video of each year



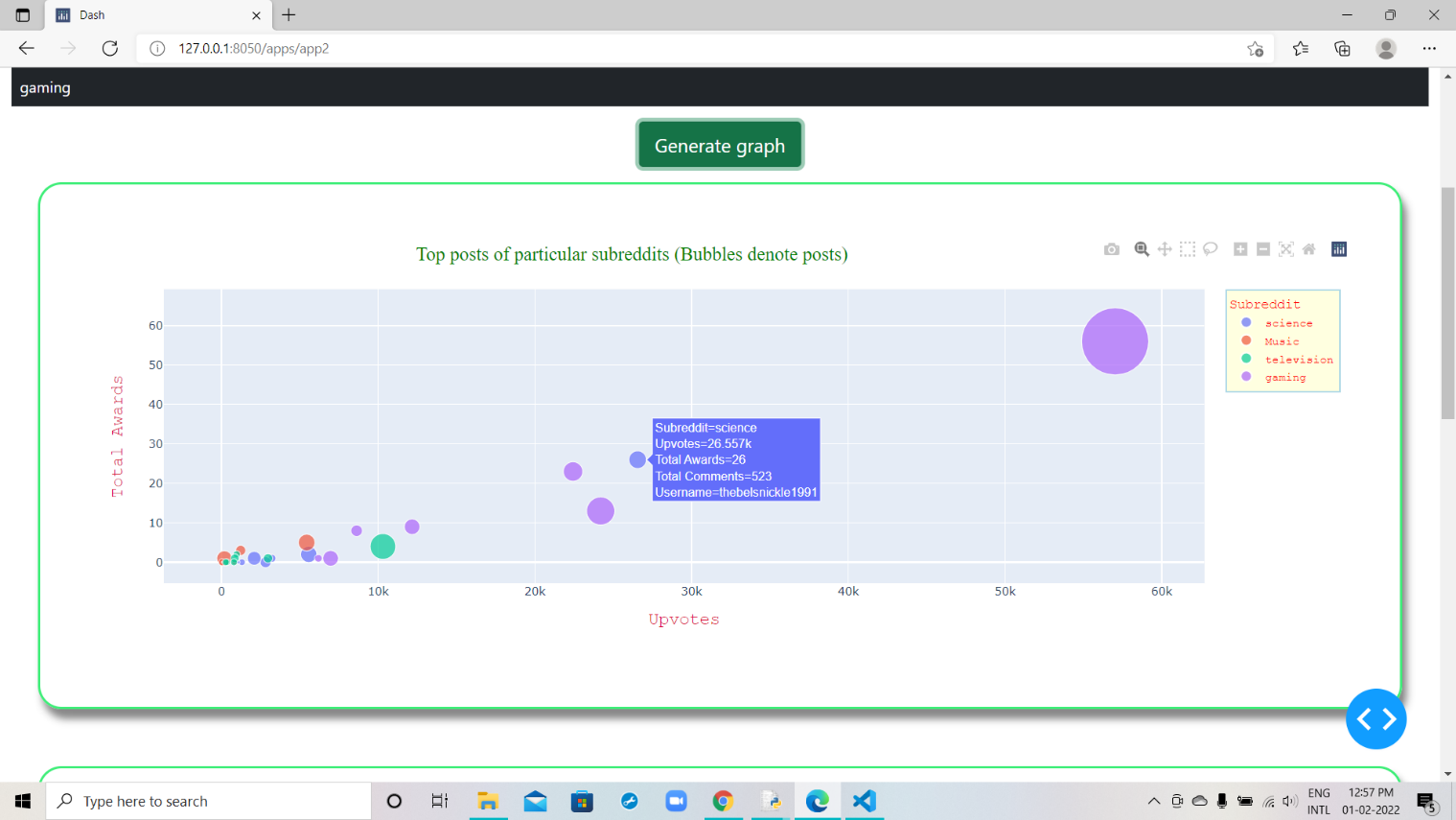
Cumulative views of the channel per year



Pie Chart on different Flairs in r/science



Wordcloud representation of keyword frequency in a given subreddit



Scatter Plot of different posts in subreddits

**Conclusions & future enhancements:**

With this web application, we can get an insight into different channels on YouTube and subreddits on Reddit.

With our YouTube counterpart, we have the the following statistics:

* Number of views vs content posted, year wise
* Year wise histogram of total number of content as well as views
* Top 10 content (based on views) from the start of the channel
* Bottom 10 content (based on views) from the start of the channel
* Best performing video of each year

With our Reddit counterpart, we have the following statistics:

* Top posts of each subreddit
* Most frequent topics of each subreddit
* Topics discussed in r/science

**Possible upgrades to our application:**

* Better UX and UI , supporting more social media platforms
* Hiding the pop-up (YouTube website) when running the application.
* Display comparisons between various YouTube channels
* Working with bigger datasets in Reddit.
* Optimising our application by implementing threading.

**References:**

1. Web Scraping - <https://en.wikipedia.org/wiki/Web_scraping> [↑](#endnote-ref-1)
2. Praw - <https://praw.readthedocs.io/> [↑](#endnote-ref-2)
3. Selenium - <https://selenium-python.readthedocs.io/> [↑](#endnote-ref-3)
4. BeautifulSoup - <https://beautiful-soup-4.readthedocs.io/en/latest/> [↑](#endnote-ref-4)
5. Plotly -  [https://plotly.com/python/](%20https://plotly.com/python/) [↑](#endnote-ref-5)
6. Tkinter - <https://docs.python.org/3/library/tkinter.html> [↑](#endnote-ref-6)
7. Dash - <https://dash.plotly.com/> [↑](#endnote-ref-7)
8. Bootstrap Component - <https://dash-bootstrap-components.opensource.faculty.ai/docs/quickstart/> [↑](#endnote-ref-8)