Youtube Stats Technical Report

Our group and website name is Youtube Stats. Our members include Hassan Sheikh, Jordy Tello, Brennan Wilkins, Tristan Hsieh, and Chris Pulicken.

Problems

Website content problems:

One of the problems we ran into while creating the website was finding a third entity type. We already knew that we wanted to include videos and channels on the website, but we were unsure about what other entity type would have enough attribute statistics to include on the website. We originally came up with four ideas: Youtube topics, categories, comment threads, or playlists. We eventually decided to use playlists as our third entity type, since topics and categories didn't have any channels directly associated with them, and comment threads did not have a lot of associated attribute statistics. Playlists worked well as a third entity type because it had a single channel as its creator, and multiple videos associated with it. Another problem we ran into while creating this website was finding playlists that had the appropriate relationship with channels and videos. For example, a lot of the more well known channels only had playlists of their own videos. This problem was solved by moving to more obscure videos and using playlists created there instead. A third problem we ran into was avoiding outdated statistics. For example, statistics like the number of views or the date last modified are likely to change after the creation of an entity type. In order to solve this problem, we plan to impliment an application programming interface to collect the data from Youtube directly, instead of manually typing the information into the static html pages.

Team coordination problem:

One more problem we encountered was the problem of keeping formatting consistent between every member of the group. Since it was likely that multiple people would end up writing code for this project, we had to ensure that there would be no sudden switch in coding style. In order to fix this problem, we collectively agreed that we would use four spaces as opposed to tabs. This especially helped our code when using Python, since Python is relatively strict about indentation. Another issue that challenged our team was balancing our obligations to complete our assignments in the middle of spring break. This resulted in logistical challenges wherein the majority of our work had to be done remotely, and members had to juggle being with their families while making sure to accomplish the workload this first milestone had.

Use Cases

This website can be used to access the statistics of Youtube videos, channels, and playlists, providing an efficient way to collect Youtube data. The statistics will include the date of publication or the date last edited as well as user statistics such as number of likes or number of subscribers. Each page will also have a link to its corresponding Youtube webpage. All the video pages will link to its creator's channel as well as any playlist it was on. The channel pages

will link to the playlists and videos that were created by the channel. The playlist pages will link to the channel of the playlist's creator and any videos on the playlist. This website's accessibility will be aided by a splash page to help users navigate the website, an about page that contains every group member's bio as well as Git Lab statistics and a list of sources, and model pages. The model pages will include sortable tables that contain five attributes and three rows representing three entity types. These features could be of use for advertisers to collect data on the best channels or videos to display their ads on, in order to decrease their cost-per-click and increase clicks-per-ad. If further scaled, this could also be used by Youtube as a tool to increase their ad revenue by attracting more advertisers.

Tools

Front-end:

For the front-end we typed out the titles and information for every video, playlist, and channel into html files. For example, each html file for a video contained the name, the date published, number of views, number of comments, number of likes, and number of dislikes. Each html page also included links to the other two entity types. Plain HTML was used to determine which order the text was in, which text would serve as a heading, and where the links would be located. We also implemented CSS to change the aesthetic of all the web pages. For example, CSS was used to change the color of the links, the background color, and the text color. We also used CSS to change the formatting and style of the text as well as the way the links respond to users clicking them. Javascript will be used as a way of gathering data. We will use an application programming interface to collect data to put into our HTML pages. This wwill be to ensure that the data on the HTML pages were are not outdated. For example, if a video gains views after the pages are published, the application programming interface will update the information. In order to access the application programing interface, we will connect to the endpoint of the API. This will allow us to retrieve data on Youtube videos, channels, and playlists in the form of an array. We will then take this data and display it on the HTML page. Most of the HTML, CSS, and Javascript for the web-pages were implemented using Bootstrap. In order to do this, we inserted a Bootstrap CSS URL into a link tag at the beginning of our HTML files. This allowed us to use design features of Bootstrap.

Back-end:

For the back-end we referred to the splash.html file with Python code. This was done in a virtual environment, which set up a local host. Within the python file, we created a Flask object, and then under a function called "index," we used the Jinja Template Engine to render a template of a pre-existing html file called "splash.html". We then used Flask to publish the html files to the local host, allowing them to be viewed in a browser.

Embedding-media services:

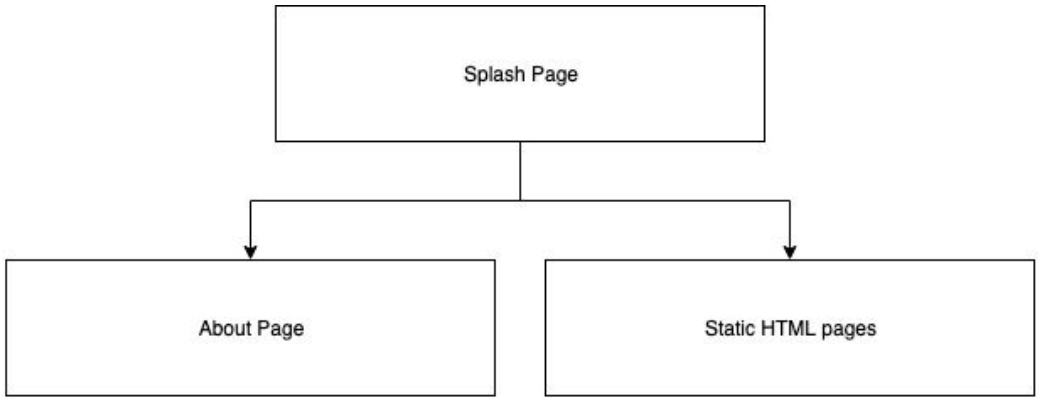
For embedding-media services, we will use an application programming interface to collect images and videos based on the content from youtube and publish them on our HTML pages.

Hosting

In order to set up Google Cloud Project, we had to download the Google Cloud software development kit. After downloading the kit, we went to the Google Cloud Project console and created a project with our website's name. We then established a billing account in order to successfully complete the project's creation. After that, we inserted a yaml file into the folder with the nine static html pages to help publish them. After inserting the yaml file, we cloned the folder with the yaml and html files onto our local directories to ensure that they functioned correctly. Then we used terminal to deploy the project to the internet.

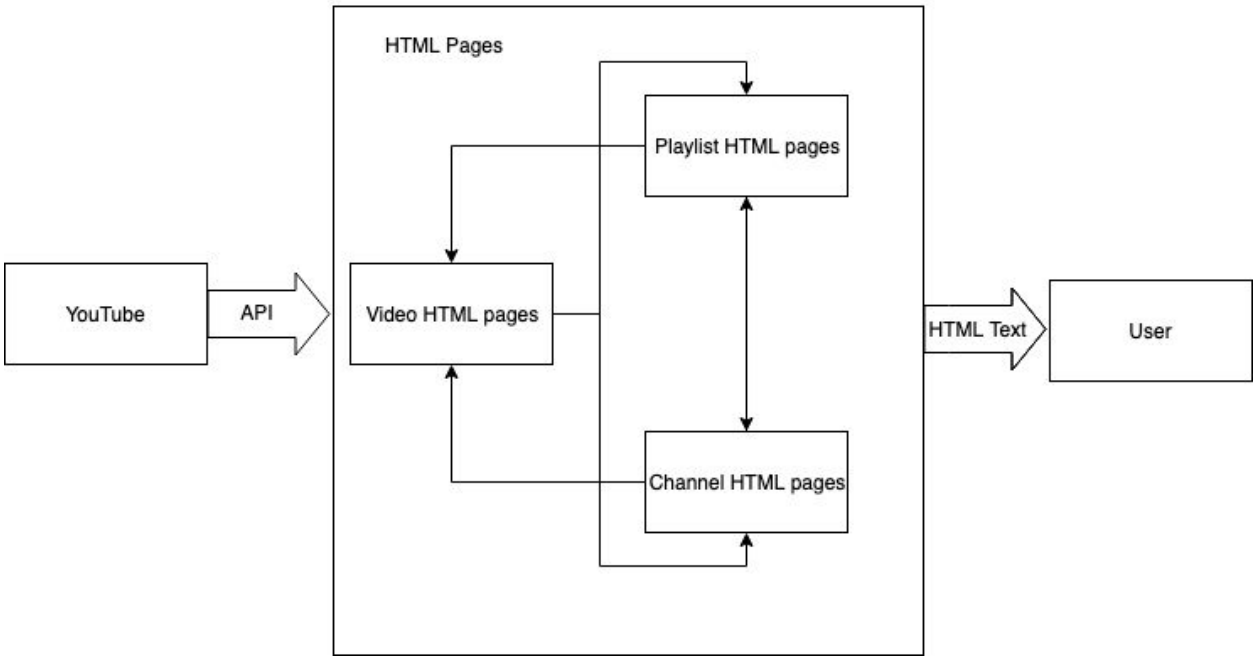
Diagrams

Site layout diagram:



This diagram displays how the site will be laid out. The user will be greeted by a Splash page, which will allow them to navigate to the about page or the static HTML pages. The about page will contain the user bios and photos, and the static HTML pages will contain Youtube statistics.

User case diagram:



This diagram shows how the site can be used. The user can navigate through the html pages for different entity types via links. Each html page will draw data from Youtube using an API, and the data will be converted into text on an HTML page for the user to read directly.