Rob Kraemer

David Jones

COMP4447, Summer Term

# Video Sharing Platform Summary

## Page Setup:

This platform operates with a main homepage containing a drop-down menu displaying all the video topics available. From there, you can navigate to each page for a specific topic to see the applicable videos. On each video, there's the ability to "like", comment, or reply in addition to a twitter feed showing tweets tagged with that topic and other related videos on the platform.

## Process:

Within each user profile, there's the ability to upload videos to the website. In order to do this, there's a submit function where the user specifies a Video Name, provides the link from YouTube, and selects a topic/tag from a predetermined list to apply to the video. From there, the provided information gets written to a text file (.txt) formatted as a .csv file with all the other information for the other videos. The platform then pulls the information from this file to display.

## Django interaction with JS and HTML:

- The Video Upload Function begins with the user interface that prompts the users to enter the name and video url, as well as select the tag for the video to be associated with. Upon submission, the form information is passed from the browser page through the Django framework to our python file. We then use a function to correct the video url so that it can be embedded, and write the url, along with its id and name to the .txt file containing the video data for the associated tag in a csv format. This video can now be accessed by subject in the video sharing platform.

- The Play Video Function is accessed by the user from the Video Topics drop down located on the header. By selecting the topic, the video player is called, along with the topic selected. The topic is then passed through the Django framework to read/search for the .txt file based on the search/topic, which returns the videos in the associated data .txt file. These related videos are then displayed in the sidebar and the first video is primed to watch. The user can then navigate the sidebar and select the titles of those videos to watch, which are displayed on click.

- In order to get the tweets to display, we pass the selected topic to our video player alongside the video url itself, where the topic is then utilized as a twitter handle. We then append this handle to the twitter api call, which returns an embedded timeline of the most recent tweets from the handle in a widget format using Javascript. We use this script code to embed the final widget into our sidebar with the scroll feature.

- The comment section currently works by storing the comments in an innerHTML, providing user functionality to add, reply, like, and delete comments. Our ultimate goal is to move the comments to a data file, where they can then be accessed and viewed based on subject/video, creating a memory for the comments. Additionally, we can also store user metadata such as click/comment rates and time stamps.

- We currently have a simple Like/Dislike button with limited functionality that stores no data. Our goal is to introduce this functionality in our next phase, where the likes are limited to one per user and counted. We can then store the metadata, to learn more about the videos uploaded/shared and how users have reacted/respond to each video.

**Next Steps:**

- The next steps for this would be to add the framework to process the data generated from the site.

- The first thing would be to write the likes and comments from each page to a database to help process the data easier.

- Building out the main page with a full search capability, that will read data, and return videos by title/subject.

- Natural language processing could be added to this to help understand what the users are saying and conversing about for each video. From this, we can see which topics generate the most traffic and engagement.

- We are also working on building the functionality to access the timestamp of each initial comment and correlate it to a certain spot in the video. This would allow us to potentially understand which parts of videos drive the most engagement. We could apply similar circumstances to other videos to help drive more conversation amongst the users and promote collaboration.

- We also plan to build functionality which would analyze which videos from the “Related Videos” section are most utilized and whether or not they are deemed useful. From this, we could generate new suggestions based on what is most useful to other users.

- We plan to also implement natural language processing techniques that will further enhance how tweets are displayed around the video player by collecting user and video metadata that can then be used to scrape the twitter feeds and hashtags through the Twitter API, ultimately displaying more relevant tweets.

- Lastly, we are eager to flesh out functionality of the website, so part of future development would be to play with the aesthetics, design, and layout. Doing so would introduce new opportunities for analyzing different data points such as effective ad placements, user friendliness, and user engagement, which we could do by tracking cursor movement over the website.