

Team: Sanguine Thirst

Project: reelFeels

COMPSCI-326 / Spring 2018

github.com/mmacquarrie/reelfeels

Zack Bebbington, Megan MacQuarrie, Matt Rubin, Humad Syed

Overview

The idea behind our application is to have a video-sharing platform that allows user emotions to be recorded and viewed for shared videos. Think YouTube but with user emotions replacing the “like” system, allowing for a more accurate representation of whether or not users actually liked the videos

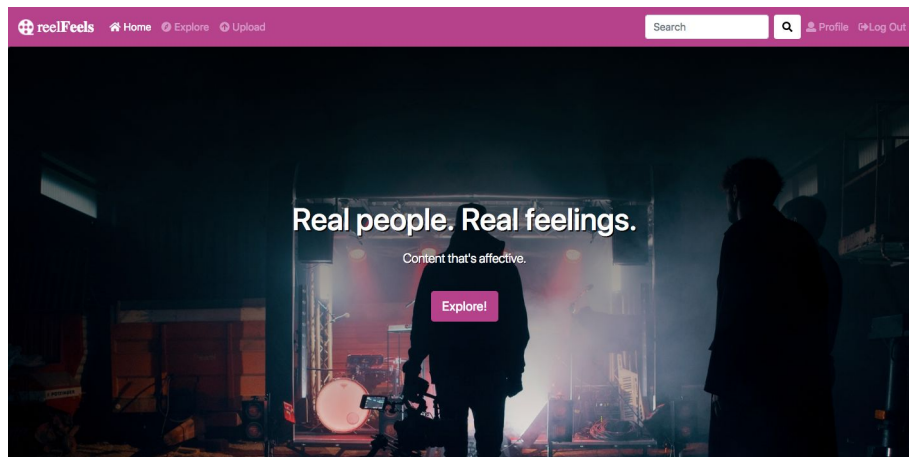
ReelFeels aims to be useful for both content creators and content consumers.

The content creators can benefit from ReelFeels by having access to information about how consumers reacted to their content. For example, a comedian would find it useful to have information about how effective their jokes were, or if their humor offended someone.

Content consumers (i.e viewers) can use ReelFeels to filter videos by emotion. For example, if someone is looking to get cheered up, they can filter using the “Happy” emotion and see videos that brought joy to other people. This is arguably better than just searching “happy videos” as it provides relevant content with minimal user effort.

User Interface

Home: Introduces user to the website; offers information and links to other pages.



Video Content: Allows users to watch videos and see stats; captures user emotions; allows users to read and write comments.

reelFeels
Home
Explore
Upload
Search
Profile
Log Out

zach_bebb
 This is wonderful

Your Stats
General Stats

Your stats

Emotion Probabilities

Joy: 0%
 Sadness: 0%
 Disgust: 0%
 Anger: 0%
 Surprise: 0%

Update Profile: Allows users to update their profile details.

reelFeels
Home
Explore
Upload
Search
Profile
Log Out

Edit Profile Details

Email:

Password:

Profile pic: Currently: [profile_pictures/28167010_905898679571180_6609301780064322076_n_1RN8Tpt.jpg](#) ☐ Clear

Change: No file chosen



Profile: Displays user details such as their name, favorite emotion, shared content, and overall stats.

reelFeels
Home
Explore
Upload
Search
Profile
Log Out

matthew_rubin
 Favorite Emotion: none 😊
[Edit Profile](#)

Feels
Content

Emotion Probabilities

Joy: 100%
 Sadness: 0%
 Disgust: ~20%
 Anger: 0%
 Fear: 0%

Upload: Allows for sharing of videos.

reelFeels

HomeExploreUpload

Search

ProfileLog Out

YouTube URL...

Next



Login: Provides a login interface.

reelFeels

HomeExploreUpload

Search

Log In

Welcome Back!

Username

Password

Log in

Don't have an account?

Sign Up!



Signup: Provides a signup interface.

reelFeels

HomeExploreUpload

Search

Log In

Reel people. Real feelings.

Join a community of people who wear their heart on their screen, and start producing more affective content today!

Create an account

(Optional)

Email

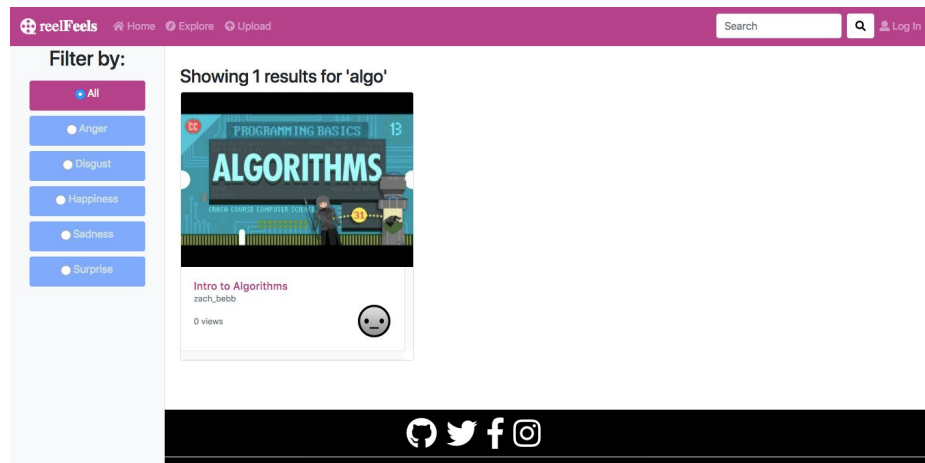
Username

Password

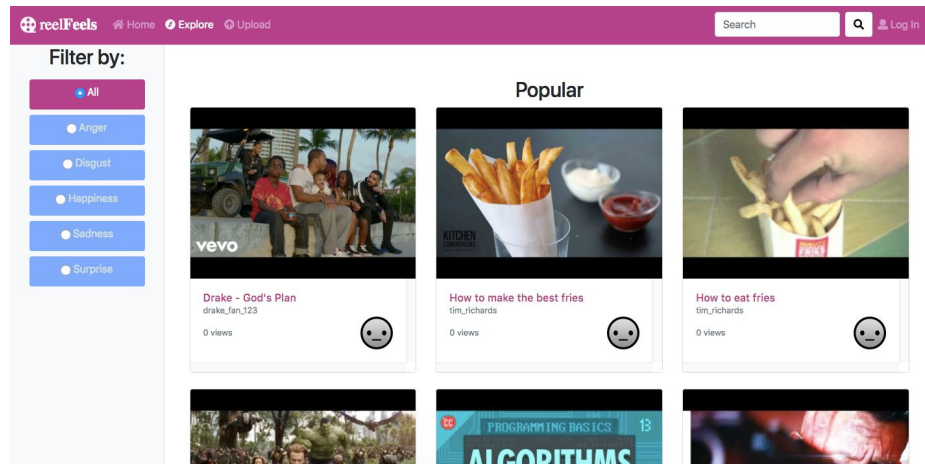
Confirm Password

Sign Up!

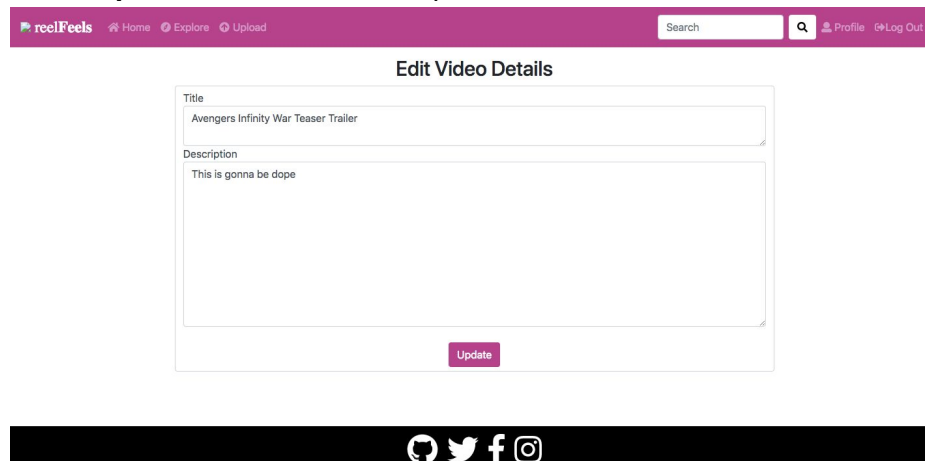
Search: Shows search results




Explore: Shows new and popular content.



Video Update: Allows users to update video details.



Video Delete: Allows users to delete video.

 [Home](#) [Explore](#) [Upload](#) [Profile](#) [Log Out](#)

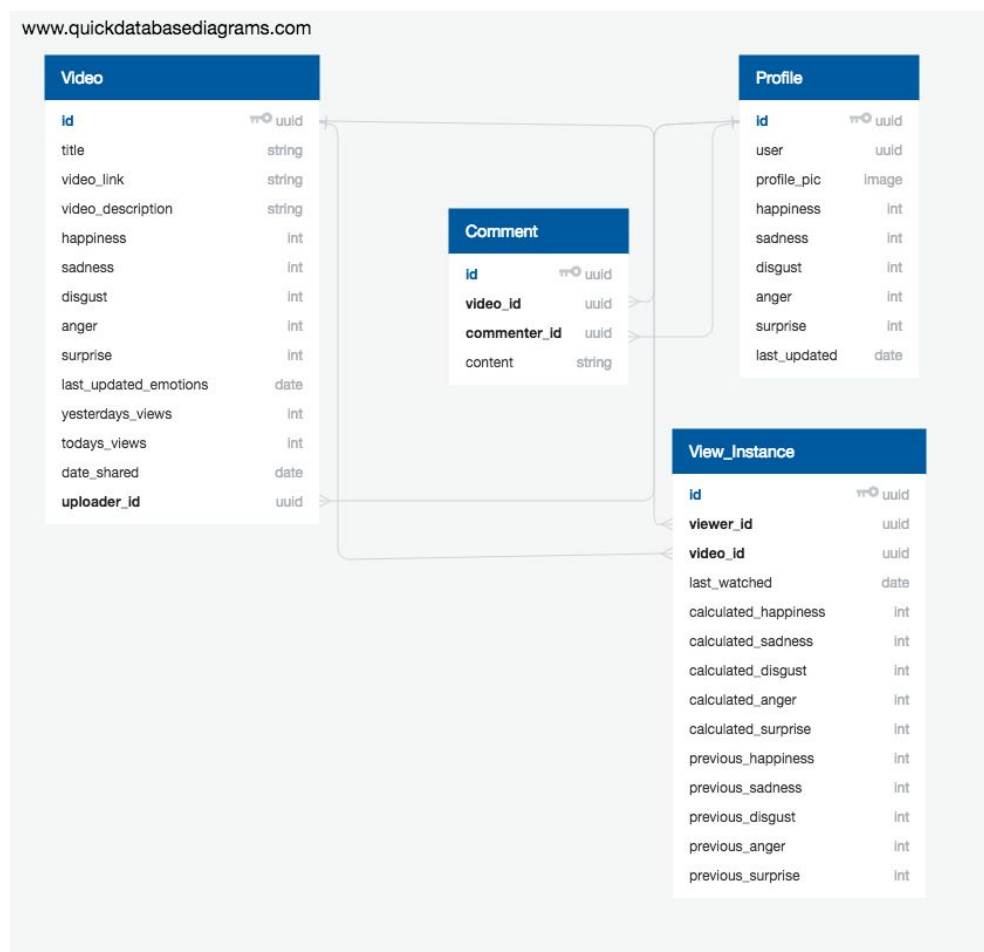
Delete Video

Are you sure you want to delete *Avengers Infinity War Teaser Trailer*?

Confirm



Data Model



The relations between the tables can be seen in the diagram above.

Video table:

This table contains information about the videos that have been shared. The fields themselves are pretty self-explanatory. The fields that are interesting to notice are the 'yesterdays_views' and 'todays_views'. These are needed to figure out if a video is popular. There is also a foreign key to the uploader. The fields for the emotions here store the aggregate of the emotions that users felt when watching this video.

Comment table:

This table stores the comments along with foreign keys to the video and the author.

Profile table:

This table stores information on our users. The emotions here are the aggregate of the total emotions the user has felt while watching any of the content. There is also a user field that links our profiles to the Django user model.

View Instance table:

This table is required because we want to keep track of the emotions users felt each time they watched the video. Each instance of a video being watched by a user is stored here. The fields for calculated emotions and previous emotions are needed for our core algorithm which determines the changes in user's emotions and their intensity.

URL Routes/Mappings

| Path | View | Description | Permissions |
|---------------------|----------------|-------------------------------|---|
| / | index | Displays home page. | None |
| /video/<video_id> | video_content | Displays video page. | None |
| /profile/my-profile | profile | Displays own profile. | Must be logged in. |
| /profile/update | update_profile | Displays profile update page. | Must be logged in and owner of profile. |
| /profile/<user_id> | profile | Displays profile page. | None |
| /upload | upload_page | Displays upload page. | Must be logged in. |
| /login | login_page | Displays login page. | None |
| /signup | signup_page | Displays signup page | None |
| /search | search_page | Displays search results page. | None |

| | | | |
|---|----------------------|---|---|
| /explore | explore_page | Displays explore page. | None |
| /logout | logout_page | Logs user out and redirects to home page. | Must be logged in. |
| /video/<pk>/edit | video_update_form | Displays video update page. | Must be logged in and owner of video. |
| /video/<pk>/delete | video_confirm_delete | Displays video delete page. | Must be logged in and owner of video. |
| /comment/add/<video_id> | video_content | Adds comment to video; redirects to video. | Must be logged in. |
| /comment/delete/<video_id>/<comment_id> | video_content | Deletes comment from video; redirects to video. | Must be logged in and owner of comment. |

Authentication/Authorization

We used Django's session and authentication middleware to log in our users and maintain sessions. In order to check if a user was authorized to see certain options or to perform certain tasks, we checked if the user was authenticated both in our templates and on our server.

In order to do certain things like create comments or upload videos, the user needs to be logged in. A logged in user also sees a different navbar than other users, which allows them to go to their profile and to log out.

Users are allowed to do things like edit or delete an updated video only if the video was uploaded by that user. Other users will not even be able to see the option to edit or delete videos. Similarly, users who authored some comments will see the option to delete them, but the button to delete someone else's comments will not be available. Additionally, if a user tries to send a request to our server to do something that they are not authorized to do, their request will silently fail because we are also checking user authorization on our server.

The rest of our website is publicly visible to everyone.

Team Choice

For our team choice, we decided to implement the [Affectiva API](#). This API uses facial recognition and machine learning to identify the probability of the emotions a user is

experiencing. This enhances our application because emotions are determined and calculated in real time and in the browser itself.

Instead of users being made to tag content with what they felt, we simply use this API to gather that information directly. The API does require camera access but is relatively fast since it doesn't need to communicate with a server to perform facial recognition and analysis.

The API works in the background on the video page. When playing a video, the browser will ask for permission to access the camera. When the camera is loaded, the chart on the right of the video will begin updating itself as it determines changes in the user's facial features.

Conclusion

All members of our team started with varying levels of web development experience. However, all of us learned something new during the course of the semester. Python and Django were both new experiences for everyone. It took a little while to get used to Python but everything came naturally towards the end of the project.

We faced a number of difficulties during the course of our project. One of the more significant ones was trying to link our Profile model to the built-in Django user model. It would have been useful to know that Django already contains a model that can handle much of the core functionality of users.

Another problem we faced was trying to come up with a core algorithm to make our idea work. Although we ended up implementing something that work, there are a lot of redundancies in our code that could possibly be fixed. However, this is a minor problem that doesn't affect the fact that our project works as intended. We also needed to figure out how to send data to our server without overloading it. We ended up sending the data after a set amount of time once enough data had been aggregated.

As a team, we worked very well together. At times, our schedules wouldn't line up properly, but we kept in touch via Slack at all times and helped each other out when necessary. Overall, our project was a success. We learned and applied new skills while working together as a team and ensuring that everyone was on the same page and that work was divided amongst us equally.