Oscar Wong

CNIT 581 Milestone 4

Inline Youtube annotations for Democratized Fact Checking

1. Project Context and Original Wireframes

This project's main use case is to implement an inline annotation system for Long From Youtube videos to help combat fake news and misinformation. The main goal is for users to be able to create and rate on annotations to democratize the information exchange on the Internet. Users should be able to log in to a Youtube account and add Annotations to videos at specific timestamps to add extra context to inform other people. The entire site is designed with the default Youtube UI in mind to integrate this function to viewers' normal Youtube UX to minimize the need to acclimate.

The Django prototype in Milestone 4 is based primarily on the wireframe proposed in Milestone 1 and later implemented in Milestone 2 and Milestone 3. The document will explain the changes that were made in Milestone 4 and how it correlates with the requirements in Sections 2 and 3. Instructions on how to set up the Django server can be found in Section 4.

a. The "Your Annotations" Page

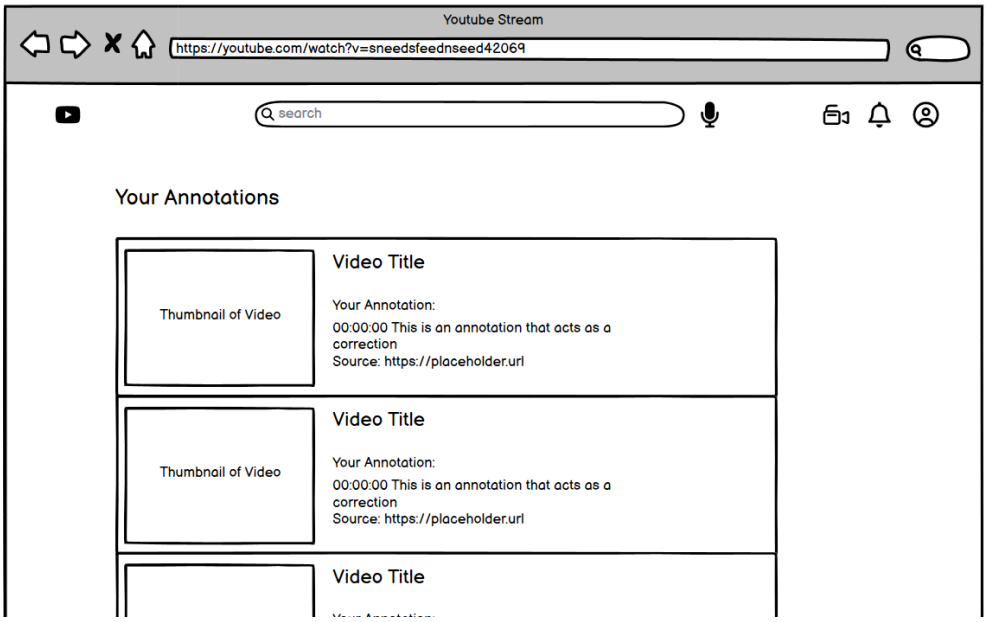

Figure 1: UI Mockup for showing all of a user's submitted annotations on multiple videos.

This Page acts as a dashboard that shows all the annotations a user has submitted across multiple videos. This can be accessed via the route **`/annotationList`**

b. The “Video” Page

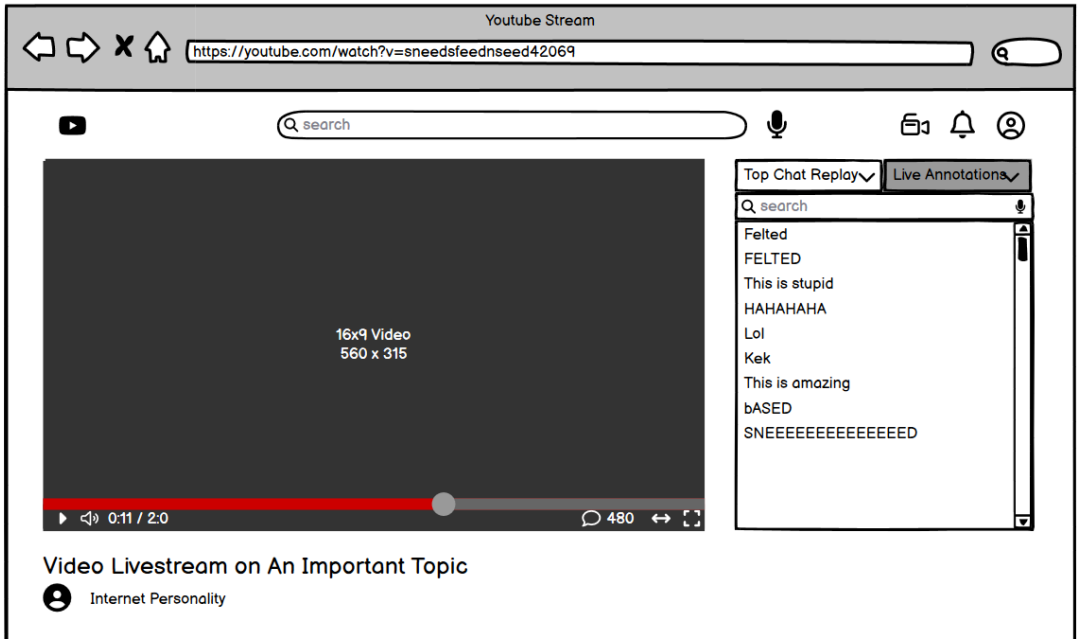

Figure 2: UI Mockup for showing chat replay on a video.

This page mimics the typical layout one would see on a Youtube Livestream. Upon entering the page, the right panel is by default set to the “Chat Replay” tab. Users can then toggle across different pages using a drop down menu to show different annotations, including “Live Annotations”, “All Annotations”, and “Add Annotations” This can be accessed via the default route or the routes **`/video`** and **`/CNIT581-048-Milestone4`**

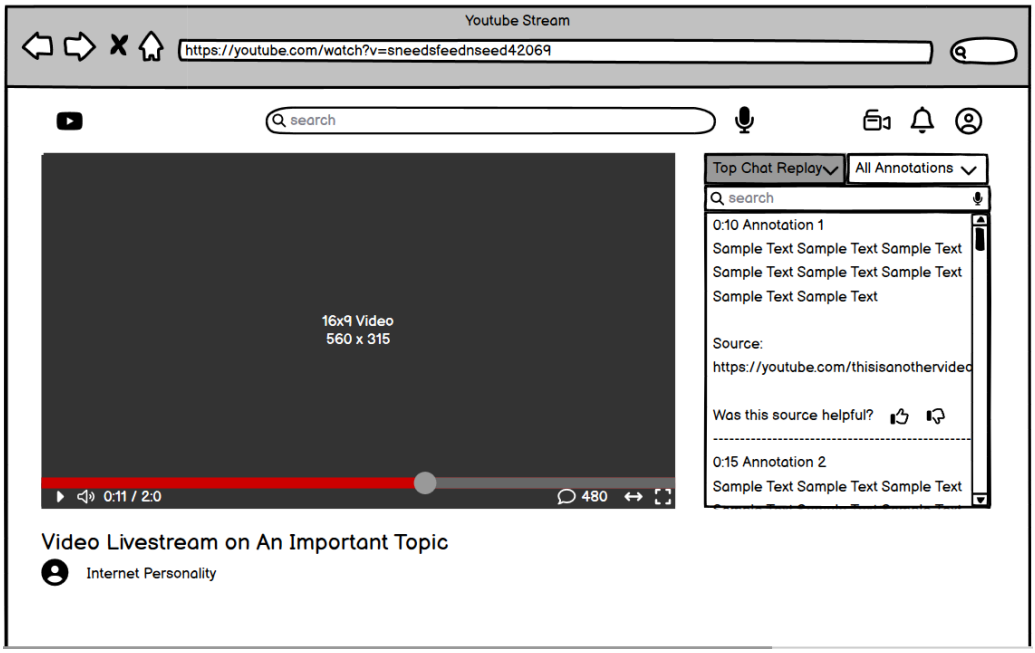

Figure 3: UI Mockup for showing all annotations on a video.

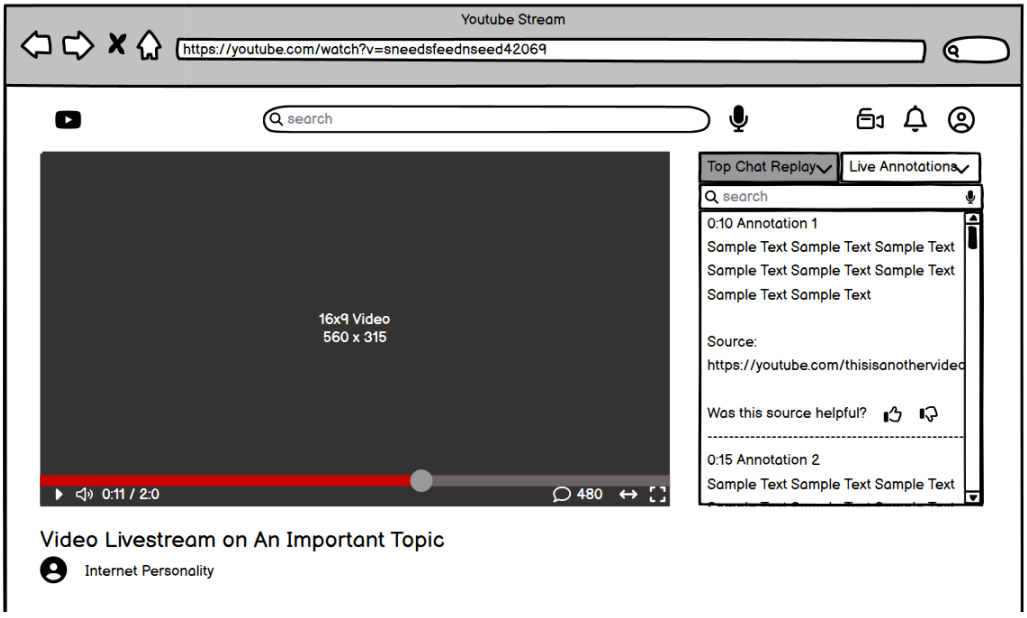

Figure 4: UI Mockup for showing live annotations on a video.

c. The “Add Annotations” Page

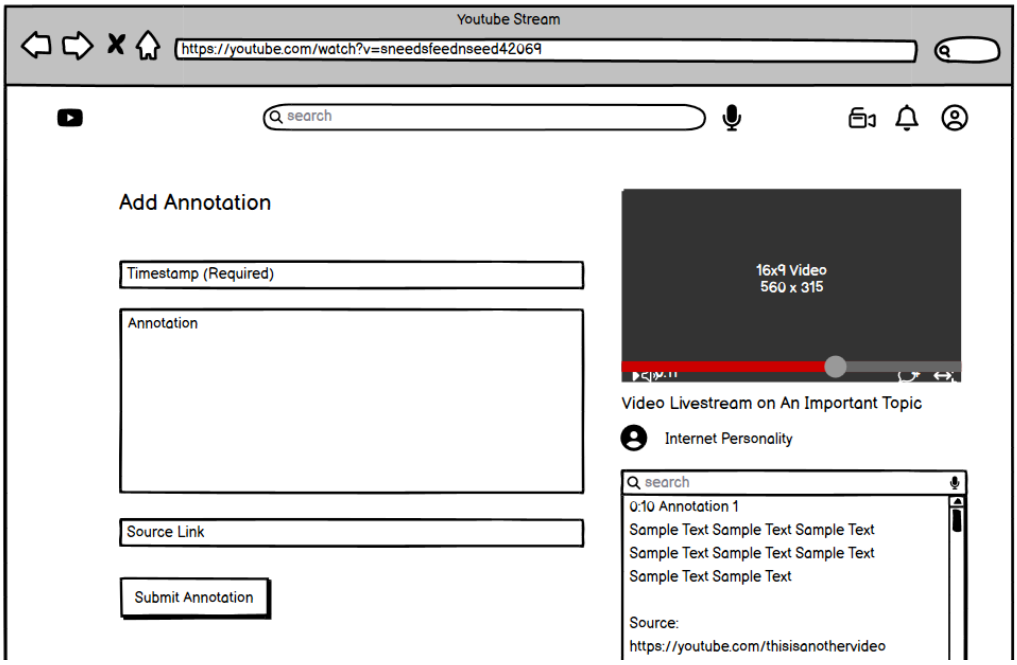

Figure 5: UI Mockup for adding annotations on a video.

The “Add Annotations” Page can be accessed via the drop down menu on the “Video” Page under “Annotations” > “Add Annotations”. This page is a form that allows users to submit their own annotations on a specific point in the video with the appropriate time stamp, annotation text, and the resources that support the annotation. The video that will be annotated along with a list of existing annotations is available on the right of the page for reference when writing annotations. This can be accessed via the route **`/addAnnotation`**

Additionally here is a list of urls that were added in previous Milestones:

- **`/viewAnnotation`** - in which an annotation is displayed in detail. Available through the **`/annotationList`** page.
- **`/editAnnotation`** - in which an annotation is displayed in detail. Available through the **`/viewAnnotation`** page.
- **`/accounts/login`** - for logging in
- **`/accounts/logout`** - for logging out

2. Changes that were made in the prototype in Milestone 4

For Milestone 4, A database component was added to the Django Project in the form of a Sqlite Database with proper models that sync with Django.

The three models in models.py are defined as follows:

```
# A class that syncs with existing Django Users

class newUsers (models.Model):

    user = models.OneToOneField(User, on_delete=models.CASCADE,
null=True, default=None)

    userid = models.CharField(max_length=255)

    username = models.CharField(max_length=255)

# A class for annotations. Is a Child of the newUsers class

class annotation (models.Model):

    timestamp = models.CharField(max_length=8)

    annotation = models.TextField(blank=False)

    citation = models.CharField(max_length=255)

    author = models.ForeignKey(newUsers, on_delete=models.CASCADE)

# A class for live chat messages. Is a Child of the newUsers class

class chat (models.Model):

    posted = models.DateTimeField(auto_now_add=True)

    comment = models.CharField(max_length=255, blank=False)

    author = models.ForeignKey(newUsers, on_delete=models.CASCADE)
```

The Corresponding Entity Relationship Diagram is as follows:

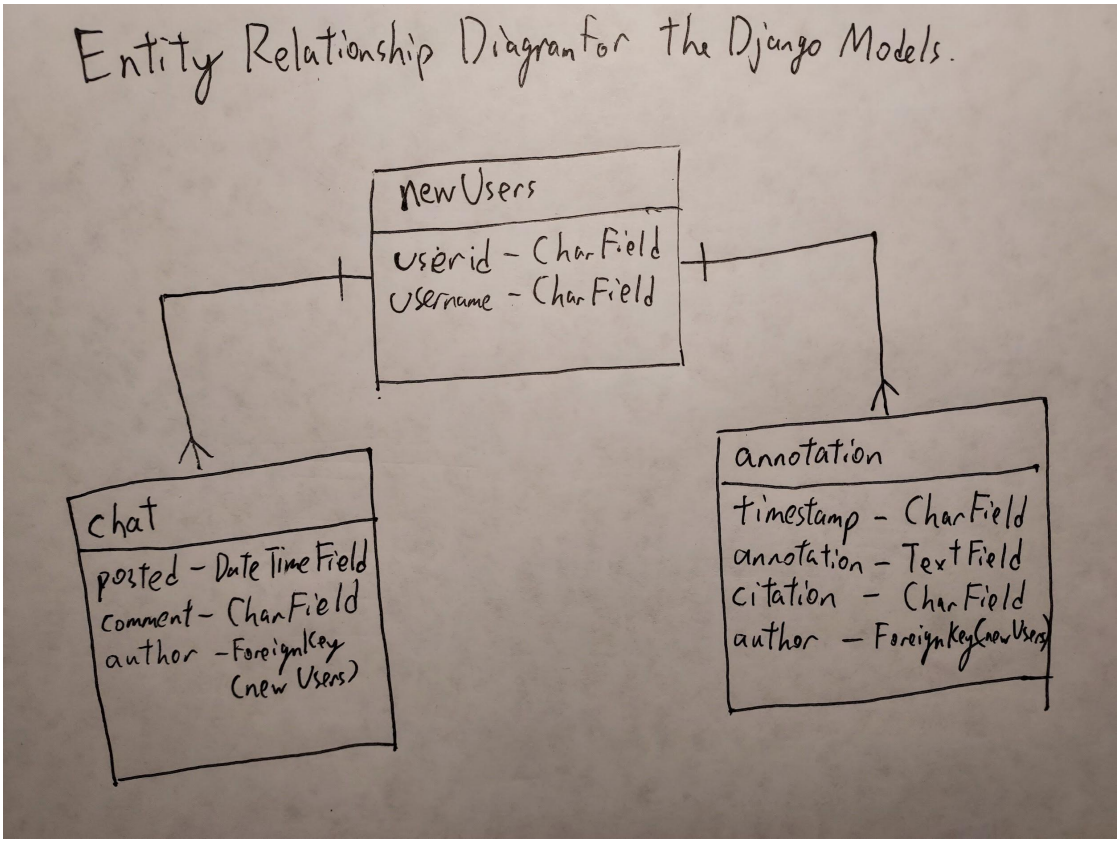

Figure 6: Entity Relationship Diagram for the Django Models.

In which there are three tables, where newUsers is a parent table to annotation and chat. As per the Milestone requirements, the entire model has three tables, each table having more than three fields (SQlite by default assigns a separate Primary Key attribute to all tables), and three field types were used overall (Charfield, Textfield, and DateTimeField). Furthermore, the parent-child relationship between the tables properly enforces Third Normal Form in the database. This schema should fulfill the necessary requirements of Milestone 4.

In Milestone 3, all dummy data was written in a dictionary located in data.py in the project. As of Milestone 4, all data is properly stored in the sqlite3 database located in the `/annotationsite` folder. All transactions that were previously done with the dummy data (i.e. add Annotations, edit annotations, retrieve a list of annotations, see a specific annotation in detail) now works with a proper database Backend.

3. Things that were added in Milestone 4.

Aside from the above, the following changes were made in Milestone 4.

- The video in **`/video`** now shows the views, like, and dislikes of the featured video on the website. This information is obtained via returnyoutubedislike.com, a third party website that provides video metrics that are now made unavailable by Youtube on specific videos.

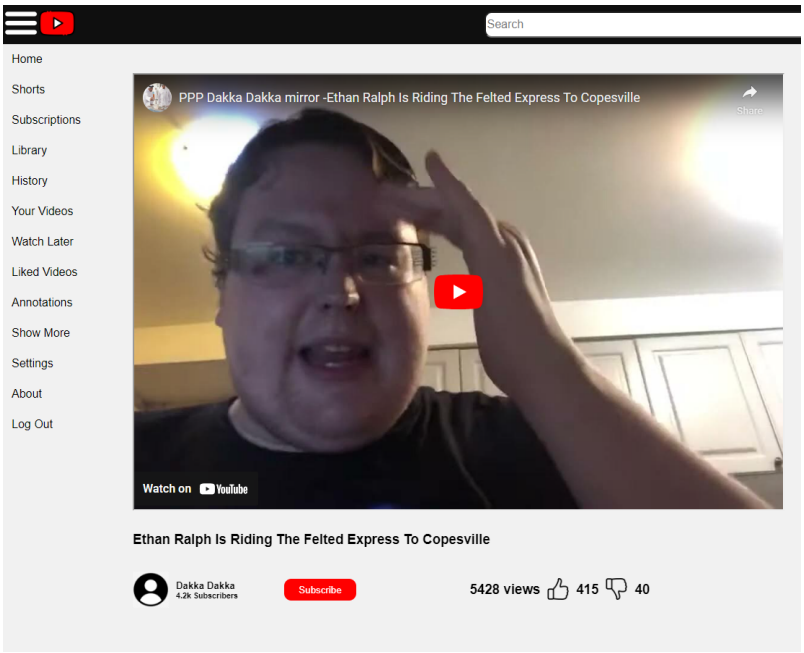

Figure 7: video.html now shows the view, like, and dislike counts of the video via external API

- The right panel in **`/video`** is working with a proper database back end. Toggling between “Live Chat” and “All Annotations” will send an AJAX GET request to the database to retrieve the appropriate data to populate in the panel view. “Live Annotations” is still unimplemented due to problems with the iframe Youtube API. Currently the option acts the same as the “All Annotations” option.
- The live search function in **`/addAnnotations`** and **`/editAnnotations`** is not working. When users type into the Search Bar, the webpage will send an AJAX POST request to the back end to filter all available annotations and dynamically return ones that match the substring.
- Three routes were added in urls.py. These include
    - **`/getAnnotation`** - where all annotations in the database is retrieved in JSON syntax listed by timeframe in ascending order
    - **`/getChat`** - where all chats in the database is retrieved in JSON syntax listed by

posted in ascending order
- **`/liveSearch`** - in which a filtered list of annotations is returned. This page only works with POST functions, so accessing it directly via a browser will cause an error.

4. References, Libraries, and Instructions

The website by default does not log users in. To see the contents of the site, the following credentials are available as a tet user via the login page under /accounts/login

Username	**`sneed`**
password	**`formerlychuck`**

A logout button is added to the bottom of the side menu bar on the left of the page. This button is available on all pages within the project.

Users will have to install the requests python module to properly handle external API calls. Once in a python environment shell, enter **`pip install requests`** to install the library. Failure to do so might brick the site. (For more information about the requests module please see here: https://www.w3schools.com/python/module_requests.asp)

No external libraries or frameworks are used for Milestone 4. Everything was written in HTML, CSS, Javascript, Django, and SQLite. The site should be available once initialized with **`python manage.py runserver`** in the **`annotationsite`** folder. (Tested in Brave/Chrome only)

Once the server has started, the site should be available at **`http://127.0.0.1:8000/`**, though depending on your Django settings that might be different. Please see the terminal for the most up to date url.

A complete list of all the image's sources used in the website can be found in the text file named "citations.txt" inside the image folder. A list of sources for specific functions that were implemented were listed in views.py. Bing AI was used when implementing the AJAX post request since there was barely any documentation on how to parse QueryDict objects with Vanilla Javascript on the internet.

Note: Hi! I don’t know if the Django project is exporting properly. I’m just dumping the entire project file directly from my computer on github. If it does not run for you or if the user credentials are not working properly, please email me ASAP. I will try and update the Github repository ASAP. Thanks!