The Testing Process

Amara is a web-based subtitling software that is built using Python. Because of the fact that it is web based, our team will likely run view tests, feature tests, and functionality tests in order to ensure that all of the individual parts of Amara are running correctly. We'll run these tests through a driver written in Python that will execute the tests, printing out either a "." for a passed test or an "F" for a failed test. At the end of the entire test suite, the driver will then print out the error information for all the failed tests, as well as the file location of the tests that failed.

Requirements Traceability

The software allows users to connect to a video on the internet and allows them to create a subtitle file specifically for that video that other users can view, as well as allowing those other users to comment on the video/subtitles page. It needs to have system for handling users, a way to input video embedding links, a subtitle file generator, and a rudimentary commenting system.

Tested Items

Test ID: 1

Requirement Being Tested:
We are testing if the user data(first name, last name, password) matches what is stored in the database

Component Being Tested:
check_user_data(self, user, data, orig_user_data=None)

Test inputs:
jettiejr@g.cofc.edu, josh, jettie

Expected outcomes:
True, True, True
because it is asserting true if the email, first and last name all match the provided data in the database.

Path to file:
unisubs/apps/api/tests/test_users.py

Test ID: 2

Requirement Being Tested:
instantiates a blank user. A user with no information.

Component Being Tested:
test_create_user_blank_data(self):

Test inputs:
There is not really imputs, we just dont input any information

Expected outcomes:
```
  'username': 'test-user',
          'email': 'test@example.com',
          'first_name': '',
          'last_name': '',
          'full_name': '',
          'bio': '',
          'homepage': ''
```

By not entering in information we are expecting that all the data to be empty strings.

Path to file:
unisubs/apps/api/tests/test_users.py

Test ID: 3

Requirement Being Tested:
We are testing to see if the function user_can_edit_subtitles() gives the correct output of false when we give it a user who is not staff

Component Being Tested:
user_can_edit_subtitles(user, video, language_code)

Test inputs:
User object who is not staff and for whom the user_can_edit_subtitles() function for workflow objects returns false, a video withe the url: https://www.youtube.com/watch?v=dQw4w9WgXcQ, en

Expected Outcomes:
False, as the supplied user should not have permission to edit subtitles.

Path to File:
apps/subtitles/permissions.py

Test ID: 4

Requirement Being Tested:
updating the comments section when given a POST request

Component Being Tested:
update_comments(request)

Test inputs:
A Django request object with a POST command for a comment object

Expected Outcomes:
The displayed view should contain the new comment object at the bottom of the screen.

Path to File:
apps/comments/views.py

Test ID: 4

Requirement Being Tested:
updating the comments section when given a POST request

Component Being Tested:
update_comments(request)

Test inputs:
A Django request object with a POST command for a comment object

Expected Outcomes:
The displayed view should contain the new comment object at the bottom of the screen.

Path to File:
apps/comments/views.py

Test ID: 5

Requirement Being Tested:
We are testing if we can create a video from a specific URL

Component Being Tested:
test_create_videos(self):

Test inputs:
"url": "http://www.example.com/sdf.mp4","langs"

Expected outcomes:
True which means the video was created and ran from testhelpers.views

Path to file:
unisubs/apps/testhelpers/tests.py

Testing Schedule

- Sep 27 - Choose 5 test cases to run
- Oct 18 - Create the automated testing framework
- Nov 10 - Write the remaining tests needed
- Nov 22 - Create fault injection tests
- Nov 29 – Finish writing final report

Test Recording Procedures

The driver for our testing suite will save the results of every run, including the pass/fail responses and any errors given by failing tests, to a test file, along with the date it was ran, and an ID attached to the test.

Hardware and Software

The only requirements to have Amara up and running are having Docker installed, (which for our Linux distro required Ubuntu 3.10 or later), and adding unisubs.example.com to the hosts file, pointing at 127.0.0.1

Constraints

Because of the difficulty of trying to make differing schedules mesh together, meeting with each other so that we can more effectively tackle the problem could be a large obstacle.

System Tests

The suite should not require rigorous system testing, as it is very small.