Assignment T3: First Iteration

Team members:
Ping-Feng Lin, pl2730
Yuan-Hsi Lai, yl4305
Yen-Min Hsu, yh3328
Shao-Chi Wu, sw3525

Team name:TaiOne
IA: Asif Mallik

Part 1:

Submit the link to the team github repository containing your code.

https://github.com/Jefflin413/ASE-team-project

Part 2:

`s00`: As **a live streamer**, I want to show my exciting life in a cool way to my subscribers/audience so that I can be popular and probably earn some money.

My conditions of satisfaction are:

1. I can sign up with my own account.
2. Start streaming
3. See my own live-streaming

Error/exception:

1. If a streamer re-submit a "build streaming pipeline" request again, the website shows a warning

`v00`: As **a subscriber**, I want to watch quality streaming so that I will feel a little bit better about my miserable life.

My conditions of satisfaction are:

1. I can either sign up without much effort or watch live-streaming in anonymous mode.

Error/exception:

1. If a subscriber uses a browser our system does not support such as IE, there should be a message saying that “your browser does not support this, please change to firefox.”

Basic Flow Acceptance Testing

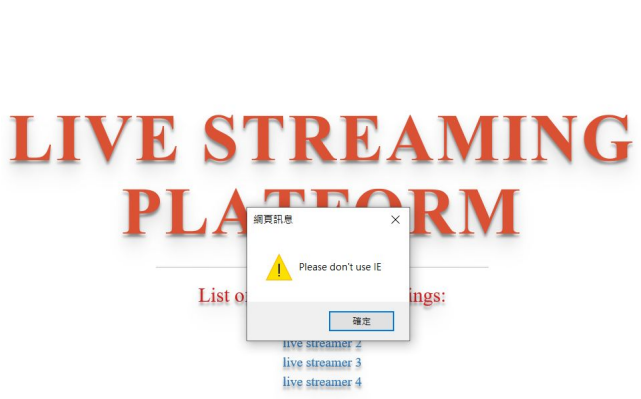
1. **User 1**, representing a **streamer**, gets a key and an url generated by the system. User 1 enters them into OBS and opens the camera to start streaming.
2. **User 2**, representing a **viewer**, connects to the website and uses an account to login. User 2 clicks on one of the streamers (not user 1) shown on a customized homepage and finds that the streaming is not started yet. User 2 goes back to the home page and clicks on the channel of user 1 and user 2 can watch the live stream.

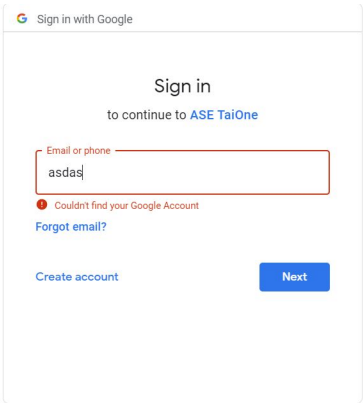
Valid and Invalid input during the test flow:

Valid flow is the same as described above, every step is executed correctly.
Invalid input can be discussed separately:

1. User 1(streamer) enters the wrong account and/or password, causing login failure. In this situation there's prompt in the login page notifying User 1 that he/she fails to login
2. User 1 starts building the streaming pipeline (POST request is sent to the backend), but he accidentally closes the website/re-submit the request/refresh the page. In these cases, the website shows "told you not to do that"
3. User 1 input wrong information to the OBS. In this case we are not responsible for checking the correctness of input to OBS while the OBS itself has the function which is able to examine it. On our website it will give you a running video user that shows nothing.
4. User 1 who uses a web browser that does not support our video user won't be able to see his/her own live-streaming.
5. User 2 (audience) can have the same authentication problem as User 1 described in 1. of this section.
6. User 2 uses a web browser that is not supported by our platform, he receives an alert suggesting he change a browser.

Some Testing Screenshots





Discussion:

The page that the streamers use to create their streaming pipeline and see their live-streaming will be modified in the future. Our idea is to iteratively call from the front-end using AJAX to the backend to see if the current streaming pipeline exists/under construction/is completed instead of asking the streamers to wait until the pipeline is fully completed. In that way we won't have to prevent users from closing/refreshing/re-submitting the request since those things won't again trigger the construction process that will raise an error.

One bug occurred in the stream page: when the video user is loaded before the streamer starts streaming using the OBS, the video won't be displayed normally. Only if the streamer has already started streaming then the video user on the website can work normally. We will fix it by removing the video user on the stream endpoint and then provide a link to the viewer page of the streamer's own live-streaming.

Part 3:

Submit the link to the folder(s) within your github repository containing all the test cases that are automatically invoked by your unit testing tool.

https://github.com/Jefflin413/ASE-team-project/tree/master/tests

Submit the link to the file(s) in your repository that configure the build tool and/or package manager and the automated unit testing tool.

Package manager configuration
requirement.txt
Automated unit testing tool: unittest (a python package) with PyCharm

Part 4:

We use a PyCharm utility, Code inspections, to find bugs and check style (at the same time). It generates reports in the form of a set of html, css, and js files.

https://github.com/Jefflin413/ASE-team-project/tree/master/report

The structure of this directory is the following:

```
report
├─── iteration1_after
│    ├─── index.html
│    ├─── script.js
│    └─── styles.css
└─── iteration1_before
     ├─── index.html
     ├─── script.js
     └─── styles.css
```

After downloading these files, please open report/iteration1_before/index.html to read the report with errors, and open report/iteration1_after/index.html for the one that is cleaner.

Explanation for unsolved warnings:

PEP 8 naming convention violation inspection 3 weak warnings
    file app.py 3 weak warnings
    WARNING Variable in function should be lowercase

These warnings arise because we are using the function `render_template` from the third party package `flask`, and the argument names of `render_template` are set in the HTML templates which is the first argument of this function. And we think that variable names that are consistent with our HTML DOM tree design will be less confusing.

PEP 8 coding style violation inspection 1 weak warning
    file test_app.py 1 weak warning
    WEAK WARNING PEP 8: E402 module level import not at top of file

We can't import `app.py` at the top of the file because we need to replace some modules imported by `app.py` with mocks such as `unittest.mock.MagicMock` before the import statement.