# Important livestream related notes

## User To Do list for creating stream on the website (YouTube oriented)

1. Setup OBS studio and stream
2. Set youtube stream to "unlisted" for security
3. Get stream key from youtube for OBS to use [WILL BE THE SAME STREAM KEY]
4. Share URL [URL changes]
5. Post URL into manage-livestream form

## Notes from 25th Jan

Stream embed code can be taken from "Share" however the website needs to be able to make this code by stripping away from the original "share URL". I also need to create the stream embed code to be UI friendly for SafeGaze.

User can view stream through youtube for as long as they want (until stream closes) but will only have access to it through SafeGaze because the stream would be "unlisted".

## YouTube (security) flaws

User can access the youtube stream through the link when they have access to it in SafeGaze and then have access to the stream ON youtube until stream ends/pauses.

Facility might have to input a new URL every time they want to stream (EACH DAY!) which can be very repetitive and tedious.

## Notes from 26th Jan

There might be a possibility to keep stream running but stopping it in OBS [does not work – YouTube stops stream if no data comes in after a few minutes] – however there may be a possibility for YouTube to be properly implemented for members to be able to rewatch footage earlier that day.

There might be a possibility to stream directly to the website without a streaming service [does not work – need a video hosting platform]

## Three different approaches for streaming service

1. Using YouTube as a proof of concept, then switch to more professional streaming service in the future if need. YouTube has security flaws (state in 25th Jan notes)
2. Dedicate SafeGaze to one specific streaming service platform
3. Making use of the different services available (facility chooses which service to use)
   1. Different URL input boxes in manage-livestream and user only inputs one box
   2. One input URL but conditional statement gets correct embedded code related to the specific service used
4. (Potentially having a Linux server that hosts the livestream demonstration)