**CS 407 Project Backlog**

# Mercury – Team 11

Kai Tinkess (ktinkess@purdue.edu)
Kris Leungwattanakij (kleungwa@purdue.edu)
Alex Hunton (ahunton@purdue.edu)
Christopher Lee (lee3880@purdue.edu)
Leonard Pan (pan353@purdue.edu)

## Problem Statement:

Online services such as Twitch and YouTube provide video streaming and hosting to users. However, these are closed-gate applications that are slow, rely on corporate servers, and are designed in favor of large-scale communities that can be monetized. A free and open-source software (FOSS) tool that is decentralized and self-hosted offering the same utility to small groups would be able to prioritize speed, efficiency, and ease of use. A peer-to-peer approach provides security and stability, especially on a LAN, which would prevent exposure to the wider internet.

## Background Information:

Video streaming is a field that exploded in probability over the last decade. After YouTube first took off with video hosting, live streaming entered the scene and exposed an additional part of the market. Live streaming tends to support younger users congregating around a particular niche or community. Video hosting also attracts those users, but also allows for tutorials and educational content.

The existing offerings of this domain have become progressively corporate and anti-consumer as they mature. Despite this, not many systems have been developed that can be self-hosted. The largest open-source competitor is Owncast, which allows for users to connect their own servers to video streams. However, it still routes through a configured website and serves web traffic. This prevents true privacy.

Mercury will solve all of the above problems by allowing for a self-hosted stream through a desktop application to send data application to application to another client. Because it will be FOSS, there will be no profit incentive or

design made around principles beyond efficient streaming. Additionally, the open-source nature and direct communication ensures privacy.

**Functional Requirements:**

## 1. Installation & Setup

1. As a user, I would like to download Mercury from the Github page from the releases sidebar.
2. As a user, I would like to be able to read an intuitive and concise tutorial on how to use Mercury.
3. As a user, I would like to be able to read a tutorial on how to set up cloud tunneling or port forwarding to permit incoming traffic.
4. As a user, I would like to have to accept an ethics agreement before using the application.

---

## 2. User Profile & Client Settings

5. As a user, I would like to set user information, such as display name, for the Mercury client.
6. As a user, I would like to set client settings, such as baud rate or quality, for the Mercury client.
7. As a user, I would like for my settings and information to be non-volatile and persist between sessions.
8. As a user, I would like to be able to toggle dark/light mode.

---

## 3. Discovering & Joining Streams

9. As a user, I would like to be able to connect to a stream given an IP address and see the username of the host.
10. As a user, I would like to be able to see a list of available streams on the Mercury client in a stream browser, and connect to one with a click.
11. As a user, I would like to be able to connect to a stream via a http link that opens via the Mercury client and connects me to that stream.
12. As a user, I would like to be able to connect to a stream given a domain name that automatically resolves to an IP.

13. As a user, watching a stream, I would like to be able to enter a known global password to start watching the stream if one is required.

---

## 4. Hosting a Stream

14. As a user, before hosting a stream of my screen, I would like to be able to select the video source/screen/application to be streamed from a list of options.
15. As a user, I would like to be able to start hosting a stream on my computer that takes some live video input.
16. As a user, I would like to be able to optionally add my stream to the stream browser.
17. As a user, I would like to see my public/private IP displayed somewhere to send other people for them to connect with.
18. As a user, I would like to be able to optionally generate a http link that will allow other users to easily connect to my stream.
19. As a user, I would like to be able to end a stream I am hosting.
20. As a user, I would like to be able to add a delay to my stream.
21. As a user, after hosting a stream, I would like to set a global password that is required to be given before a user can connect to the stream.
22. As a user, hosting a stream, I would like to be able to disable annotations on my stream.
23. As a user, hosting a stream, I would like to be able to mute audio of the entire stream at any point.
24. As a user, hosting a stream, I would like to be able to mute microphone audio, with notification showing up if I am speaking despite being muted..
25. As a user, when hosting a stream, I would like the option to select a mp4/mov (video) instead of my desktop, and stream that video.
26. As a user, when hosting a stream, I would like the option to select a mp3 (audio) instead of my desktop, and stream that video.
27. As a user, when hosting a stream (that is a mp4), I would like to be able to pause and start the video.
28. As a user, when hosting a stream, I would like to be notified when my network connection is unstable.
29. As a user, when hosting a stream, I would like my window to display some marker to show that my stream is live.
30. As a user, before hosting a stream of my screen, I would like to be able to specify the audio source to be streamed, and only have that source be streamed.

## 5. In-Stream Experience (For Viewers and Hosts)

31. As a user, after connecting to a stream, I would like to see the video stream the host is broadcasting.
32. As a user, after connecting to a stream, I would like to hear the audio stream the host is broadcasting.
33. As a user, after connecting to a stream, I would like to send messages in a chat room.
34. As a user, after hosting a stream, I would like to send messages in a chat room.
35. As a user, after connecting to a stream, I would like to see messages other people have sent in the chat room.
36. As a user, after hosting a stream, I would like to see messages other people have sent in the chat room.
37. As a user, after connecting to a stream, I would like to be able to send reactions to the stream.
38. As a user, after connecting to a stream, I would like to be able to see when the stream was started and how long it has been going for.
39. As a user, after connecting to a stream, I would like to be able to full screen the steam I’m watching.
40. As a user, after connecting to a stream, I would like to be able to adjust the quality of the stream.
41. As a user, after connecting to a stream, I would like to be able to adjust the volume of the stream.
42. As a user, I would like to be able to annotate a stream using a “marker” tool.
43. As a user I would like to be able to select a color for the annotation “marker”.

**Non-Functional Requirements:**

## Clean Code

We should ensure that the code we write is clean, easy to understand, and extensible by adhering to the following list of requirements.

1. Meaningful names should be used for variables, files, and functions.
2. Each file/function should be documented with information including but not limited to:
    a. Its function
    b. Its parameters (type and purpose)
    c. What it returns (if not void)
3. Consistent indentation
4. Code styling should remain consistent through the use of a linter.
5. The client UI should be intuitive and lightweight to operate
6. Classes should follow a similar pattern to one another and make sense in the context of the project as a whole.

## Security

We should ensure that users of our apps have their information securely transferred between one another. We will accomplish this by following the CIA triad and using best practices for our design.

- Confidentiality - We will ensure that information broadcasted from our users is encrypted such that receivers of the broadcast have proper methods to decrypt it. This will ensure that man-in-the-middle attacks are mitigated.
- Integrity - We will once again use proper security protocols to ensure that end users of broadcasts can ensure that the data received is from the correct source and any data modified is thrown out.
- Availability - By having our app decentralized FOSS, users will be protected from centralized attacks and remain in control of the availability of their content.

We will achieve these goals through proper choice and implementation of networking protocols such as TCP/UDP and SSL/TLS.

## Networking Performance

We should aim to minimize network load wherever possible. We will do this through the use of various networking benchmarking features within our code. Furthermore, we will take key notice to use and implementation of particular networking protocols to allow for best outcomes for our networking.