Test Product Manager Dashboard & API - 2020

Please write your answer in English in any document format. You can either share your code directly in a file or upload it to Github. **Please don't use the name Streamroot anywhere as it is a registered trademark.**

We will pay attention to:

- The correctness of your English and the way you present your work
- Your ability to understand other people's problems
- Your attention to details
- Your ability to dig into an unfamiliar and complex subject and find answers

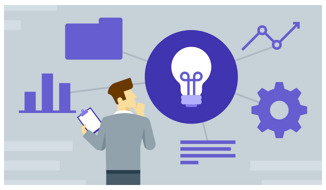
We will not hold against you:

- Imperfect solutions if the attempt is sensible
- Code quality if it works and is easily runnable (exercice 2)

As a reminder, you should submit your answers a week after you received this document but you can submit them earlier if you'd like. Don't hesitate to send questions our way if something isn't clear. We provide recommended time to spend on each exercise but you can spend more time if you need/want to.

Good luck!
Streamroot Product Team

<u>Exercice 1: Product Management (2 hours)</u>



Streamroot Dashboard stakeholders have expressed interests in two problems this quarter and we need you to design features to solve them.
For one of the two problems below, you will need to:

1. Propose a design **draft** including:
    a. A list of functional requirements for the feature. You can categorize them as *Nice to have / Should have / Must have*. These requirements will be submitted to the engineering team to evaluate their feasibility and complexity and will then be used to write down a technical specification. You can think big here but complexity/feasibility/cost considerations are to take into account.
    b. A mockup or wireframe of the dashboard UI and explanation on how to read/use it if need be. You can send it in any format you want as long as it's easily readable.
    c. A very high level architecture diagram or paragraph describing where informations should be passed between client, server and dashboard
    d. An answer to "How does it solve the problem?"
2. Provide a short product description of the feature (that would be shared on the public documentation).

- **Problem A: Monitoring of Streamroot Client Release**
  Because Streamroot Dashboard aggregates traffic volume and viewers count for all the versions of Streamroot clients that are in production, customers and support teams have no way of monitoring when our client team deploys a new version in production and how many of our viewers are updating their apps to the new version.
  Assuming that Streamroot client code can send the information of their version to our servers, propose a design for a dashboard feature that solves this lack of visibility.

  <u>Extra context informations</u>:
  Streamroot dashboard already shows for a given customer:
    - Volume, bandwidth, viewers count over time
    - Filters on platform and content allowing customers to see those metrics for a specific platform or content.

- **Problem B: Geographic activation of Streamroot**
  Working with international broadcasters comes with a perk: those customers can have very different legal and content concerns depending on the region of the world/the country where

their viewers are watching. For this reason, they would like to have control over the activation of Streamroot technology based on the country/region where the viewer is watching. Assuming that our servers can easily detect the locations of our viewers, propose design for a dashboard feature that gives international broadcasters full control over where the tech is activated.

Extra context informations:

- You can focus on country based activation and not worry about bigger/smaller regions
- Broadcasters won't need to update this list very often (a couple times a year)
- Activating streamroot is a matter of passing a value "on" or "off" to the client app.

You can access a trial account of our dashboard here (beware it doesn't have a lot of traffic data associated with it but you should see a few in July 2019):

- **URL:** https://dashboard.streamroot.io
- **Login:** charles.sonigo+productcandidates@streamroot.io
- **Password:** y!fC#t2/vR2(#e"

Exercice 2: Basic Streamroot hacking (0.5 - 1 hour)

Ressources:

- Streamroot documentation: https://support.streamroot.io
- Level3 HLS stream: http://vod-l3.delivery.streamroot.io/vodOrigin/tos1500.mp4/playlist.m3u8
- Cloudfront HLS stream: http://wowza-test-cloudfront.streamroot.io/vodOrigin/tos1500.mp4/playlist.m3u8
- Streamroot Account:
    - **URL:** https://dashboard.streamroot.io
    - **Login:** charles.sonigo+productcandidates@streamroot.io
    - **Password:** y!fC#t2/vR2(#e"

Questions:

1. Create a test page with Video.js player. It should include the streamroot Graph library (you can find it in Streamroot Documentation), and you should be able to generate some P2P by opening 2 tabs at the same time.

2. Create 2 test pages, one with the Cloudfront stream, the other one using the Level3 stream. Generate some P2P between these 2 pages using the contentIdGenerator option.
3. *Non mandatory*: Add a button in the page to enable/disable P2P upload.
4. *Non mandatory*: Give 2 improvement suggestions on our documentation.
5. Give 2 improvement suggestions on our dashboard.