Dashboard UI Engineer Assignment

Callstats.io collects WebRTC metrics from the beginning to the end of an audio/video conference session. In this task, we ask you to aggregate and visualise these metrics on a dashboard.

Data source

You are given an SQLite database (as well as an CSV file) with aggregated data about conference sessions from a small subset of callstats.io database. The first line of the file contains names of metrics available for the conference session. Below is the explanation of each metric. There are 51 metrics, but please don't be scared, as you don't need to use most of them in this task.

Please note that due to the probabilistic nature of data we collect, not all metrics are available for every session. For example, some sessions are monitored only on the sending endpoint, thus we don't have any metrics measured by the receiving endpoint.

- appID → unique identifier of the application (e.g., Skype)
- localID → unique identifier of the session sender
- remoteID → unique identifier of the session receiver
- osName → name of sender's operating system (e.g., Windows)
- osVer → version of sender's operating system (e.g., 10)
- buildName → name of sender's browser (e.g., Firefox)
- buildVer → version of sender's browser (e.g., 54.0.1)
- ip → version of IP used (0 if IPv4, 1 if IPv6)
- udp → transport protocol used (0 if TCP, 1 if UDP)
- longestDisruptionMs → duration of longest connectivity disruption during the session in milliseconds
- numDisruptions → number of connectivity disruptions during the session
- senderCity → name of sender's city
- senderCountry → name of sender's country
- senderIsp → name of sender's ISP
- senderLatitude → geographical latitude coordinate of the sender
- senderLongitude → geographical longitude coordinate of the sender
- senderAs → Autonomous system (AS) number of the sender
- recvCity → name of receiver's city
- recvCountry → name of receiver's country
- recvIsp → name of receiver's ISP
- recvLatitude → geographical latitude of the receiver
- recvLongitude → geographical longitude of the receiver
- recvAs → Autonomous system number of the receiver
- meanSendingRateKbps → average sending rate
- codec → media codec used
- mediaType → type of session (1 for "video", 2 for "audio", 0 for "unknown")
- meanSentFrameHeight → average frame height sent by the sender
- meanSentFrameWidth → average frame width sent by the sender
- medianSentFrameHeight → median frame height sent by the sender
- medianSentFrameWidth → median frame width sent by the sender
- 95thPercentileSentFrameHeight → 95th percentile of frame height sent by the sender

- 95thPercentileSentFrameWidth → 95th percentile of frame width sent by the sender
- medianSentFrameRate → median frame rate sent by the sender
- 95thPercentileSentFrameRate → 95th percentile of frame rate sent by the sender
- meanSentFrameRate → average frame rate sent by the sender
- 95thPercentileRtt → 95th percentile of session RTT
- meanReceiveRateKbps → average receive rate
- meanRecvFrameHeight → average frame height received by the receiver
- meanRecvFrameWidth → average frame width received by the receiver
- medianRecvFrameHeight → median frame height received by the receiver
- medianRecvFrameWidth → median frame width received by the receiver
- 95thPercentileRecvFrameHeight → 95th percentile of frame height received by the receiver
- 95thPercentileRecvFrameWidth → 95th percentile of frame width received by the receiver
- 95thPercentileRecvFrameRate → 95th percentile of frame rate received by the receiver
- meanRecvFrameRate → average frame rate received by the receiver
- medianRecvFrameRate → median frame rate received by the receiver
- fractionalLost → fraction of packets lost during the session
- 95thPercentileLosses → 95th percentile of packets lost during the session
- 95thPercentileJitter → 95th percentile of jitter during the session
- sentQuality → perceived quality of sent data
- recvQuality → perceived quality of received data

Implementation goals

The implementation goal is to visualise **the Sending rate per AppID**. Obviously the assignment outcomes don't need to be limited to just this, and if you want to play more with the data, you're encouraged to do so and try to find other meaningful visualisations (*).

Sending rate per appID should cover:
- Distribution of sending rate within the appID selected by the user
- Distribution of average application sending rate across all appIDs

Please make your implementation easy to run for us, i.e. we don't want to install anything extra on our machines to compile or run your application. Either host your app for us, only provide static files, or write a simple Node.js server that could be run by something like `npm start`.

** Some other parameters you could explore include e.g*

Sending rate per buildName, buildVer
- *Distribution of average sending rate for the combination of buildName and buildVer selected by the user*

Media type per appID
- *Distribution of media types with the appID selected by the user*