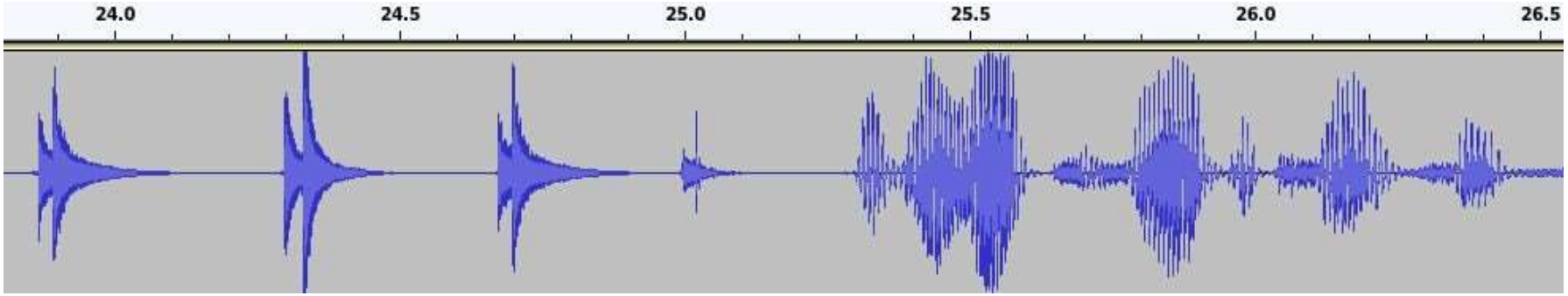
- This TuneUrl-POC project is using AudioContext for playing audio files. The main JavaScript is at src/main/webapp/js/audio-demo.js

- It calls the triggersound file and created the fingerprint and saves locally in JS. Then matches the 100 Milisecond audio with Triggersound.

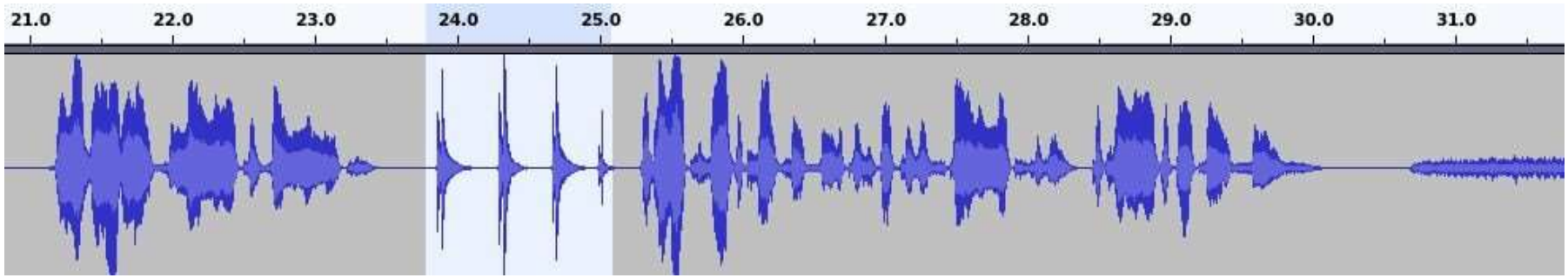
The app continuously monitors the audio stream for the triggersound fingerprint in 1.5-second increments by matching a fingerprint of the audio segment to a "stored" fingerprint of the audio trigger.
1a. During app startup, the app turns the audio trigger into an audio fingerprint
1b. In a loop, the app turns every 1,5-second of streaming audio into an audio fingerprint
1c. The app stitches the new 1.5-second fingerprint to the previous 1.5 fingerprint .

1d. Scan the 3-second stiched audio fingerprint for the triggersound fingerprint
2. Once a triggersound is found, the next 5 seconds of audio are recorded
3. The recorded audio is turned into a fingerprint
4. The fingerprint is sent to the match API
5. Pop-up shows up “Are you interested?” with 2 buttons: Yes + No
a. The pop-up disappears after 7 seconds
6. If the user clicks “No” the pop-up disappears
7. If the user clicks “Yes” the app handles the match API response
8. The app handles the reporting API call(s)



The sample audio above is showing the trigger sound with 1000 milliseconds duration. It is use to mark where the TuneUrl is located in the audio file. The TuneUrl audio segment is an audio after the trigger sound.

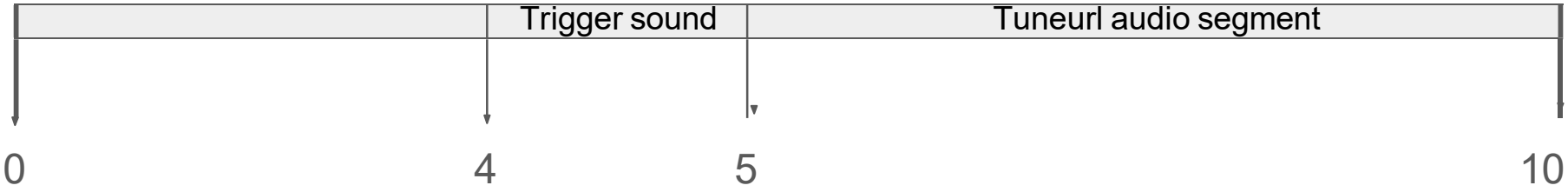
In order to detect the trigger sound in an audio stream, the application is using the audio identification method known as audio fingerprint. Audio fingerprinting is compute-intensive process and currently implemented on the server. comparing two fingerprints is done in locally in JS.



Trigger sound	TuneUrl audio

The trigger sound is an audio segment is currently have a 1000 milliseconds duration while the TuneUrl audio is a least 5000 milliseconds duration.

In order to detect the trigger sound every 100 milliseconds interval, I need to process a 10 seconds audio segment and identify the trigger sound in less than 5 seconds. During initialization, I have made a 10 seconds delay before showing the play button. This enables the scanning of trigger sound to at least be 10 seconds ahead of the playing of the audio.



From 0 to 5 seconds, the trigger sound fingerprint is scan by comparing it with the fingerprints of the 10 seconds audio segment. If the trigger sound is on the 10 seconds audio segment, the 5 seconds audio segment after the trigger sound is converted into a fingerprint for later call to a matching API here.

Fingerprint scanning is done every 100 milliseconds interval.

```
async function startCanvas() {
    console.log('startCanvas');
    initVariables();
    displaySpinner(true);
    initChannelData(0);
    setButtonPlayOrPause(true);
    await doLogin();
    if (hasJWT()) {
        await initTriggerAudio(TRIGGERSOUND_AUDIO_URL);
        if (triggerFingerprintSize > 0) {
            await initLoadFromUrl(LOAD_FROM_THIS_URL);
            if (IF_LOAD_ALL_AUDIO_STREAM) {
                emitTuneUrlInstruction(true, 0);
                await saveAudioEx(initAllTagsEx);
            } else {
                saveAudio();
            }
        }
    }
    displaySpinner(false);
}
```

The JavaScript main process started at function startCanvas().