OVERVIEW

Before we meet, we would like you to complete the below exercise. It is OK if the code isn't

perfect, but it should run. Please do not spend more than four hours working through this

exercise. Our goal is to learn a bit about your thought processes and how you approach

problems while still being respectful of your time.

We offer this activity in lieu of a technically-focused interview or test. When we meet, we may

ask you some technical questions or opinions about platforms and we almost certainly will ask

you about some of the decisions you make in your code.

You don't have to stick to a particular style guide, but your code should be easy to read.

Performance and “time to paint” matters greatly to us, but this exercise doesn’t afford you the

time to make that a primary consideration. Your comments and thought process are as

important, if not more important, than the code itself.

PART 1:

Using the React framework, create a styled test HTML page that uses Google Publisher Tags (GPT) and JavaScript to request two different sized ads from our ad server, set targeting data in the form of key values, and then render the ads to the page from the ad response.

Please use React 16.3. Include appropriate import statements (including a package.json is not necessary). The code should be contained in a single file (including js, jsx, html and css as appropriate.)

**GPT library documentation:** https://support.google.com/admanager/topic/28788

**GPT API reference:** https://developers.google.com/doubleclick-gpt/reference

**Lotame Audience Extraction API reference**: see PDF attached.

1) When you define the two ad slots on your test page, please use the following ad unit path for both: '344101295/SI/www.silive.com/news/index.ssf'. Ad sizes should be 300x250 and 728x90.

2) Before making the ad request, set slot-level targeting for each ad slot with size as the key value (ex: size=300x250 and size=728x90 respectively).

3) Before making the ad request, use JavaScript to make a call to Lotame’s Audience Extraction API, parse the results, store some values and pass them to the ad call:  
  
 a) First, include Lotame’s script in the source of your test page within the head, so you can be tracked as a Lotame user.   
  
 <script async type="text/javascript" src="//tags.crwdcntrl.net/c/931/cc.js?ns=\_cc931" id="LOTCC\_931"></script>  
  
 b) Within your js file, make a call to the Lotame Audience Extraction API, and implement a   
 1-second timeout for a response to return.

c) If the Lotame API call returns within 1 second, parse the JSON response and store the tpid value in localStorage if one exists.  
  
 d) Set page-level targeting for the two ad slots with the tpid value returned from the JSON response. (ex: tpid=1266bec41f02b8b6a37d42a977233ff1)

d) If the Lotame API call doesn't return after 1 second or no tpid value exists within the returned response, implement logic to fall back to the tpid value stored in localStorage and set the value as the page-level targeting key-value.   
  
 e) If the Lotame API doesn’t return in 1-second AND there is no pre-existing tpid value already stored in localStorage, then just send in a single page-level key value (tpid='none') as the default.

f) Render the two ads on the test page, with both slot-level and page-level key values being passed on the ad calls.

PART 2:

Building upon the page created in the first part, write a React component that renders the two ads when the user presses a button on the page. (Defines the ad slots, sets slot-level, page-level targeting and requests, renders the ads).

Each time the button is pressed, you should generate a random key/value pair (anything you want) to set as slot-level targeting on each of the ad slots.

Thank you for your time, we look forward to meeting you.