Take-Home Assignment

Senior Frontend Engineer - Royalty Processing

Introduction

We're excited to have you as part of our interview process!

This take-home assignment is designed to evaluate your technical skills, system design approach, and overall software craftsmanship.

Context

Imagine you are part of the team building a music distribution system for a streaming platform akin to Spotify or Apple Music, that serves millions of users worldwide. Its main purpose is to generate invoices to pay authors for the music being streamed there. The system consumes calculation results for authors and their songs from another subdomain. Our final goal here is to allow users to send acceptance of calculation. They do that by checking progress of royalty calculations and (when they determine it's the best moment) by issuing an invoice. Which happens in another system after user notifies it via UI interaction.

Task Requirements

Server-Side Application (Backend)

We need a server application with an endpoint, that serves list of 10 songs, each song containing identifier, name, author, and progress value.

Example data structure (based on popular songs of 2024):

ID	Song Name	Author	Progress
1	"Flowers"	Miley Cyrus	0.15
2	"Anti-Hero"	Taylor Swift	0.27
3	"As It Was"	Harry Styles	0.12
4	"Heat Waves"	Glass Animals	0.38
5	"Unholy"	Sam Smith ft. Kim Petras	0.03
6	"Calm Down"	Rema & Selena Gomez	0.10
7	"Bad Habit"	Steve Lacy	0.35
8	"I'm Good (Blue)"	David Guetta & Bebe Rexha	0.58
9	"Lavender Haze"	Taylor Swift	0.41
10	"Creepin'"	Metro Boomin, The Weeknd, 21 Savage	0.32

Context on "Progress" field: Progress represents the calculation progress from another subdomain. Technical details are not important for this task, but in short: progress means the percentage of calculated royalties for the author of a given song (value between 0 and 1, where 1 = 100% calculated).

Client Application (Frontend)

We need a client application with a user interface containing at least two components:

- Songs table component:
  - Displays a table with columns: ID | Song name | Author | Progress | "Issue Invoice" Button
  - Fetches data from the server application
  - **"Issue Invoice" button functionality:**
    - After clicking, additional information should appear in the table row:
      - Date of last click
      - Progress value from that click moment
    - This information should be visible next to the button
- Invoice history component:
  - Displays a list of all "Issue Invoice" button clicks
  - Each list item contains: date, author, song name, progress value from the moment of issuance

Interaction logic:

When user clicks "Issue Invoice" button, we need to:

- update table component - add information about date and progress in the corresponding row
- update history component - add new entry to the list of issued invoices
- apply appropriate state management between components

Extra Information

- **Language**: The frontend part should be implemented using React with TypeScript, the server must be hand-crafted (please do not use packages like `npx http-server` or similar ready-made solutions), we suggest Express.js
- **Communication**: The communication between the server and client are expected to be done via HTTP protocol.
- **State Management**: We will be looking for application of appropriate state management pattern.
- **Freedom of Choice**: You are free to choose everything else you want to use: the build tools, libraries, etc.
- **Keep it simple**: You don't need to dedicate days to this; simply demonstrate your ability to craft excellent software.
- **Functional**: Please ensure that the code in the submission is fully functional on a local machine and include instructions for building and running it.
- **Dependencies**: Avoid any external dependencies (remote cloud databases, remote search engines, etc.) to solve the task. This exercise is intended to assess your problem-solving skills rather than creating a system reliant on cloud connections.
- **MVP**: Use this exercise as a guide for design decisions, considering it as the initial prototype of a Minimum Viable Product (MVP) that will evolve into production ready deliverable. Be prepared for further discussions regarding how to transition and scale the prototype for future deployment. We understand that "production-ready" may have different interpretations, and we look forward to discussing what it means to you during the review.

Deliverables

- Please submit your solution either as a link to a public repository or as a zip file with all the necessary components for local building and running.
- If there are additional artifacts (e.g., slides, Miro-board, etc.) that you'll use during the presentation, please include them in your submission.
- Be prepared to present and explain your design and implementation, and to implement small new feature for the application during the next stage of interview process (you are free to use any tool, editor, documentation, or other source of knowledge, which you are using in your daily work with a screen share).

If you have any questions or require further clarification, feel free to reach out to us. We are looking forward to receiving your solution!