Test: Full-Stack Web Engineer

A test project for full-stack engineers applying to Slang

Overview

This document describes a programming challenge for full-stack engineers looking to work with Slang. We expect that you spend at least a few hours over the next week to develop a solution that meets as many of the specifications listed below as possible.

Feel free to submit your work quickly if you feel that you've adequately demonstrated your abilities, or to take up to as much time as you'd like over the next week if you really feel like fleshing it out a bit more. Be sure to balance quality and productivity, and keep in mind that we take code quality, UI/UX, and best practices very seriously!

The task is based on a real-world task that we've had to solve, and while it's been boiled down into specifications appropriate for a short development assessment, it is still representative of some of the work you'd be exposed to in this role. We suggest that you read through everything here — it's a quick read — before starting. And, as always, let us know if you have any questions or comments! I will be available to answer any questions at `kamran@slangapp.com` at any point throughout the day/night.

Specifications

The descriptions here are intentionally a bit free-form — we'd like you to make most of the design and development decisions yourself.

Functionality overview

We'd like you to build an application which presents users with English spelling exercises. The user will be presented with the pronunciation of an English term and a pool of scrambled letters from that term's spelling, and the user must provide the correct spelling of the term. The application should give the user feedback about his or her spelling before continuing on to the next exercise.

What we're expecting

This simple application is designed to mimic one of the exercises we present to our real-world users to help them learn English. The exercise has two main halves: the user input, and the feedback output.

Your back end should start with a dictionary of English terms. This can be hard-coded, imported from a dictionary file, or fetched from an API — it's up to you. When rendering an exercise, one of these English terms should be selected, its letters scrambled and displayed to the user, and a pronunciation should be provided. The audio of the pronunciation could be fetched from a text-to-speech API (there are many free and easy-to-implement APIs online), produced by some local tool, or produced via the Web Speech API (search for the relatively new `SpeechSynthesis` interface).

The input side of the exercise involves playing the audio to the user somehow (this might involve a custom audio-player component, for example), displaying the pool of letters, and allowing the user to input and submit his or her answer (like dragging and dropping letters from the pool or using keyboard input).

The output side of the exercise involves displaying to the user how he or she did. For a correct submission, this might be a simple check mark, but for an incorrect submission, you might want to intelligently display the difference between the user's spelling and the correct spelling. If the user just transposed a pair letters, for example, could the application display that in a useful way?

The user should be able to continue receiving and submitting these exercises one after another. You might want to include some overall tracking of progress across the exercises. You don't need to worry about authentication or persisting the user's progress across sessions, though.

As a company that highly values the importance of good user experience, we'd like to see a thoughtful interface. We understand that this is not necessarily a design-oriented role — don't worry too much about aesthetics if that's not your thing — but we recognize the value of a user-oriented mindset and we think it would be great if your application is exceptionally usable. In the same vein, we're interested in how you might implement interface animations and transitions, feedback and empty states, and so on. In a discussion after the test, we'll talk about any additional features you might include if you had more time, so keep track of any ideas that come up while you're working.

Finally, we definitely respect that everyone's got their own opinions on testing. If you feel strongly that testing is important and you would like to demonstrate your test-writing abilities, you might want to include a bit of testing — unit and/or integration — with your app if you've got time. It is not necessary, however.

Technologies

Our current front-end web UI is a React-based application written in TypeScript. We use MobX for state management, Webpack for module bundling, PostCSS and CSS Modules for styling, and Yarn for package management. Our codebase makes heavy use of modern JavaScript constructs like async/await, and thoroughly utilizes TypeScript's type system (our web UI is 100% TypeScript/TSX).

Our current back end is a separate Rails 5 API application. We use Bundler for package management, ActiveModelSerializers for our view layer, and PostgreSQL for our database.

Our UI and API are completely independent applications, but you are welcome to integrate your front and back ends into a single application if that is easier for you.

While you certainly aren't required to use this exact same stack, we will be looking out for candidates that can demonstrate a mastery of the technologies we use. That said, if you don't have, e.g., TypeScript experience, but are otherwise comfortable with modern JavaScript, we would love to get you up to speed on our TypeScript codebase!

At the very least, we ask that your application target modern desktop browsers. If you're feeling up for it, you may choose to implement a responsive design to support mobile browsers as well.

Feel free to use — and cite — any open-source solutions you can find. We'll be keeping an eye out for best practices. We'll also be on the lookout for documentation, code standards and quality, and application structure and organization.

Submission

You should submit your application as a Git repository on either GitHub or BitBucket. You should share your repository with me (username `kamkha` on either one), and you should use the repository as you would for any other project — commit early and often! We're interested in seeing the history for your work, too.

If it's easy for you, feel free to deploy the application and submit a link to the deployed version alongside the repository.

All right — that's it. Thanks for reading through this, and let me know if you have any questions or comments. We're excited to see what you come up with!