Udemy interview - homework assignment

Hello! This assignment is designed to provide a small practical exercise in building applications.

Practical Exercise — Collaborative Hangman

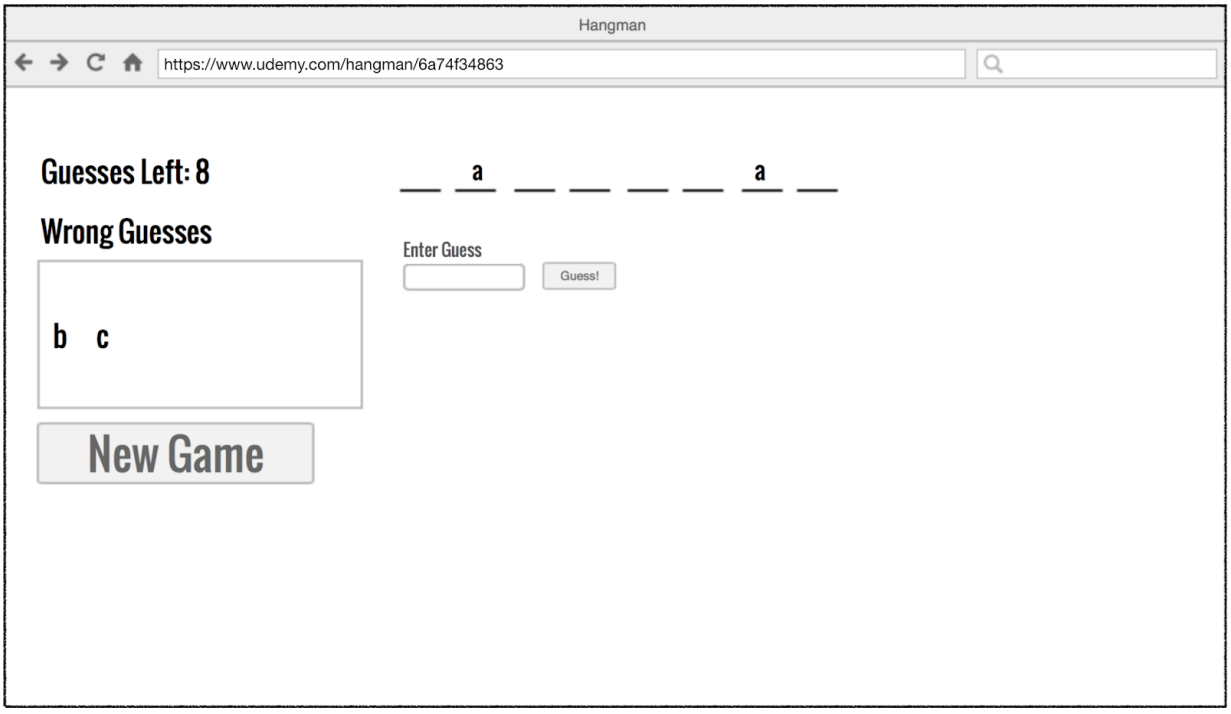
We'd like to implement a backend service that communicates via a well-formed API. That API will power a simple, collaborative game of "hangman".

The rules of "hangman", if you are unfamiliar, are as follows:

- To begin, the player is presented with a "masked word"; that is, a sequence of blank spaces. The player's goal is to guess which word has been selected.
    - The player may begin a new game at any time by clicking "New game". This does not affect other players (see "Collaboration" below)
- They are also given a number of "guesses" (for simplicity, we'll fix this number to 10)
- The game consists of the player *guessing* which letters might appear in the word.
- Each "turn" consists of a player guessing a single letter, e.g. "a".
- If the letter **exists** in the word, then **all of its locations in the word are revealed.**
- If the letter **does not exist**, then the number of "guesses" is reduced by 1.
- Play continues until the word is **completely revealed**, or the number of guesses is reduced to 0.

The UI below represents a game in-progress, where:

- The "secret word" is "hangman"
- The player has guessed: "a", "b", and "c"



Collaboration

This application does not require user accounts; anyone can start a new game. However, a player can collaborate with another person by sharing the URL in their address bar.
The second player will see the game in the exact same state as the other player.
However, the state does not need to be updated in real-time (*no websockets required*), instead, the game state only re-renders when:

- The player makes a guess, in which case the *complete current state should be shown*
- The player completely refreshes the page

Additional Requirements / Assumptions:

1. The player's browser should never receive the “secret” word; instead, only the server is aware of it
2. Your API should expect & handle malformed or malicious requests.
3. You can assume the server has a list of words in a well-known file location
4. 50% of all games are played by multiple players

Please Provide:

Please provide a complete API design (URLs, request/response bodies, error codes, etc.) Include as complete an implementation as possible of the API you designed, in the language & framework you are most comfortable with.

Beyond that, please assume that you will require multiple application servers to meet scale or robustness/availability requirements. Please discuss how you would persist game data (choice of store, schema). You don't need to implement the ha/robust solutions, just discuss them.

We'd recommend just pointing us to a github repo with your answers, but if something else is easier, that's fine.