Practical Coding Assessment - Front End

You're going to build a website backed by a simple backend API. This website will show all the users of an app, and if a user is selected, it will show the user's full profile page. You will be able to pull both the list of users and user details from the backend API, which will be detailed later.

This project is expected to take approximately 2 hours to complete. Completion of these initial phases is required, as there will be a follow-up session to expand its functionality and discuss your code and thought process.

Please ask questions if you have any while working on this project. We expect the final version to either be runnable locally or hosted somewhere publicly accessible before our conversation. Provide instructions in a README file on how to set up and run the app from scratch or how to access your hosted version. Feel free to include any other relevant notes that might be useful in the README. Assume that the person reading the README is a software developer but not an expert in this platform.

Prerequisites

1. No specific framework is required for this project, but we do suggest using Javascript or Typescript. During the collaboration session, we'll add more features on top of what you built. Choose a tech stack that you're comfortable with and that is also appropriate for this type of application.
2. While we aren't expecting a production level of polish, there should be some styling applied to your project.
3. The end result should be runnable and incorporate automated tests.
4. In the README, provide clear instructions on how to run the project or how to access the hosted version, and how to execute the automated tests.

Backend API Description

- Base API URL: https://interviews-accounts.elevateapp.com/api/ui
- User List Endpoint URL: /users
    - This endpoint takes two parameters:
        - authentication_user_id : Your user id
        - authentication_token : Your auth token
    - Output: a list of user ids:

```
{
   "user_ids": [integer]
}
```

- User Details Endpoint URL: /users/<user_id>, where <user_id> is the user id of the user for which we want to retrieve the details
    - This endpoint takes two parameters:
        - authentication_user_id : Your user id
        - authentication_token : Your auth token
    - Output:
        - If the user is not found, the endpoint will return 404
        - If the user is found, then the endpoint will return in JSON the user details:

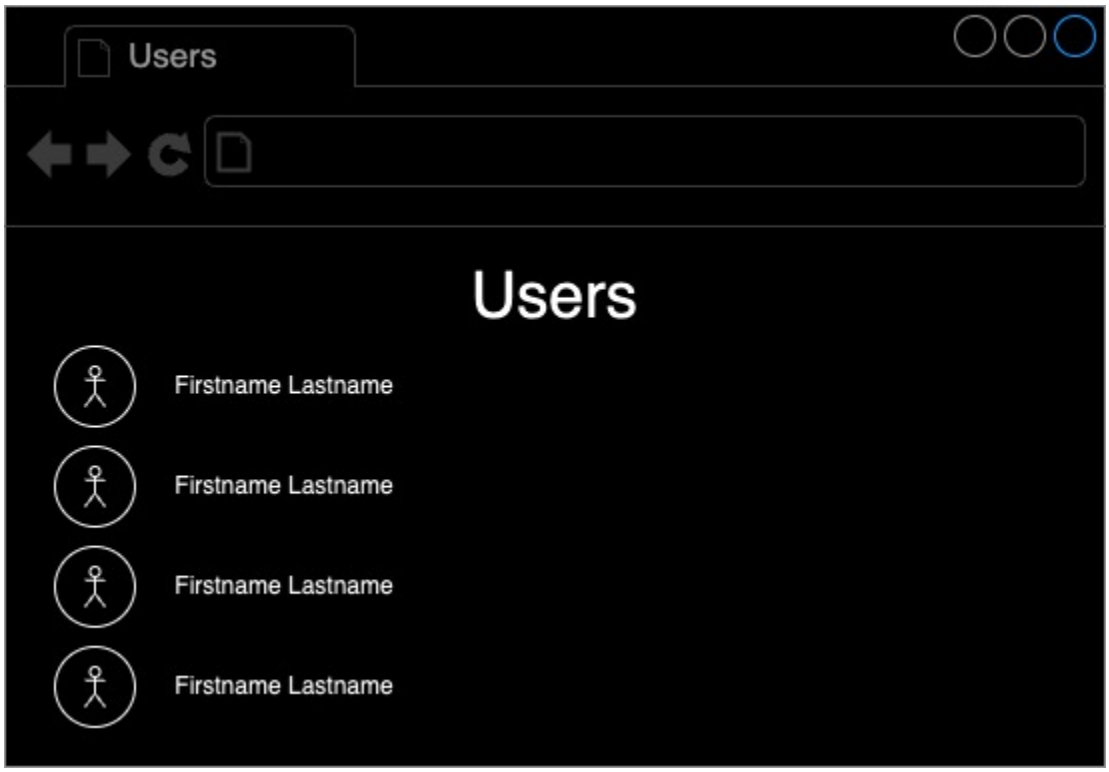
```
{
   "first_name": string,
   "last_name": string,
   "image": base64 encoded jpg image,
   "stats": {
      "current_streak_in_days": integer,
      "skills": {
         "math": {
            "current": integer,
            "level": string,
            "max": integer
         },
         "reading": {
            "current": integer,
            "level": string,
            "max": integer
         },
         "speaking": {
            "current": integer,
            "level": string,
            "max": integer
         },
         "writing": {
            "current": integer,
            "level": string,
            "max": integer
         }
      },
      "total_sessions_played": integer
   }
}
```

Phase 1 - User List

A user of the website should be able to view a list of all users of the application. This list should include:

- Each user's full name
- Each user's profile picture

Sample Mockup:



Phase 2 - User Profiles

After selecting a user from the list implemented above, we should be presented with the profile for that user. The profile should display:

- The user's full name
- The user's profile picture
- The user's current streak
- The user's total sessions played
- A visualization of the user's skill levels
    - Design is up to you but something that visualizes the user's current skill level vs the maximum level for the skill, e.g. a progress bar.

Sample Mockup: