Take-Home Practical Coding Session - Backend

We're going to build a RESTful backend for an existing mobile app. The mobile app allows a user to log in, submit game completion information, and see some basic summary statistics. Assume that the mobile app is already built and we need to match with its expectations for the API.

This project is expected to take approximately 4 hours to complete, though completion is not the primary goal. Please ask questions if you have any in the course of working on this project. We will have a short follow-up discussion about your code as a wrap-up to this session, so be prepared to talk about your choices and thought process.

We expect the final version to be runnable prior to our conversation. Please provide instructions in a README file on how to set up and run the app from scratch. Feel free to include any other relevant notes that might be useful in the README. It would be good to assume that the person consuming the README may not be a “Rails native”.

Prerequisites

1. The project must be completed in Ruby on Rails. Our backend systems are all predominately Rails and we want to evaluate your abilities on that platform.
2. The API will be using JSON as the preferred content type.
3. The end result should be runnable. If there are special instructions then those should be included in the project’s README.
4. Take time to test.
5. We will provide a Rubocop config file that matches our internal coding style guides.

Phase 1 - Signup and Authentication

The client should be able to signup a new user and, once signed up, provide a user’s credentials to the backend and have that user become “logged-in”. We would like this to follow best practices for security.

The signup endpoint will be hosted at “/api/user” using a POST verb. A user should have an email, username, full_name, and password. The API will receive the new user’s information in JSON format and reply with a “201 Created” on success.

For login the path should be “/api/sessions” and the request will arrive as a POST with the credentials in JSON format. Return a token for the user.

Subsequent API requests should be restricted to logged-in users.

Phase 2 - Game Listing and Completion Ingestion

There will be a list of games that are always being added to. This set of resources will also be the basis for reporting a user's completion events back to the backend.

Each game has an id, a name, a URL, and a category (Math, Reading, Speaking, Writing). The client will get a list of the current games and where to pull them from every time the app is opened. The endpoint will live at "/api/games" and should respond to a GET request. The response should be an array of games nested under the "games" key in JSON. Each game has an id, name, url, and category.

The client will post a completion event when a game is completed. This will be a POST to the endpoint /api/user/game_events. The request JSON should be under the "game_event" key and include the fields type, occured_at, and game_id. The type field will always be "COMPLETED". The game_id will match with the id provided in the games index response.

Phase 3 - User Details and Stats

The client will request the user details after login and each time the application opens to get the latest stats and any changes to the user details. This will be a simple GET request to /api/user. The response should have the user object in JSON format nested under the "user" key. The object should include at least the user's id, username, email, and full_name. Additionally, the response should have the user's summary statistics about their completed games. The client is built with the following as a sample response:

```
{
  "user": {
    "id": "54321",
    "username": "myname",
    "email": "test@example.com",
    "full_name": "My Name",
    "stats": {
      "total_games_played": 5,
      "total_math_games_played": 3,
      "total_reading_games_played": 1,
      "total_speaking_games_played": 0,
      "total_writing_games_played": 1
    }
  }
}
```

Phase 4 - Add streak to response

```
{
   ...
   "stats": {
      ...
      "current_streak_in_days": integer
   }
}
```

Add “current_streak_in_days” to response

Streak definition:

Streak is the number of days in a row where the user has played any games starting from the current day.

Note that the streak isn’t lost if the user has not played a game on the current day. For example, if the user played 2 days ago and yesterday but not today, their streak should still be 2. If they also played today, their streak should be 3.