**Single-Player Trivia Game**
**Version 1 Review**

**Overview**

This web app is a basic, single-player trivia game that uses the Open Trivia Database API to serve up multiple-choice questions and associated answer options. As developed to date, it is hard-coded to present users with a mix of 13 general knowledge questions of medium difficulty during every game session. It takes players through three key user journey steps:

1) visit homepage and select button to start the game;
2) play the game by answering each question using radio buttons; and
3) see final results once game is over.

In the background, the game fetches the Q&A data from the API, cleans it up, evaluates whether each selected answer is correct or incorrect, and calculates a running score.

**Demonstration**

Please visit the walkthrough video here: https://share.descript.com/view/byIQnPnaelJ

**Issues**

In hindsight, my original scope of work—a real-time multiplayer game with user accounts and custom game creation—was unrealistic. Even if I had not been sick for three weeks, I now believe that it wasn't feasible to complete such a large build on my own in that amount of time. I'm thankful that you offered up the alternative of doing a single-player game. While I have experience with building out Q&A assessments quickly and could have created a custom test using my own content, I think it was a valuable experience for me to practice fetching API data to drive the game engine.

Beyond this, I didn't encounter too many issues other than a general sense that it was a lot more difficult to create than I expected, and that I am still very much in research and learning mode with Python overall. I noticed this was especially true when I'd complete one element one day and then have to wait several hours or days before doing more—when I returned, I felt like I had to make sense of everything all over again. This was especially true of dealing with the game sessions in Flask, and I stumbled over the "write" functions needed before a new question was loaded.

**Self Reflection**

In all honesty, I'm happy this first phase is done and the game is functional, if ugly and a bit clunky. I'm disappointed that I initially planned something that was unattainable, as it feels like I wasted a lot of time early on.

Even so, this was much more difficult and time-intensive for me to work through than I had anticipated, even with the pared-down scope, and my initial optimism in my baseline knowledge fell by the wayside. I had far too many "*I know that I know this, so why don't I remember what I know?*" moments. (Being terribly sick for over three weeks didn't help.)

Because I was feeling so horribly and didn't want to fall behind, I also probably leveraged AI more than I needed to to help troubleshoot solutions when I encountered errors (instead of mostly working through things using the VS Code debugger), and for non-Python stuff I know how to do, like writing HTML. While I understand the changes I made and selected whether to implement its suggestions on a line-by-line basis, in hindsight, I wish I hadn't felt pressed for time and had worked through things systematically on my own for longer before doing so.

**<u>Phase 2 Milestones</u>**

July 14:

1. Apply page design
    - Add all the necessary styling (Tailwind CSS) and JavaScript

July 21:

2. Adjust the display of user messages through styling or JavaScript
    - The user messages, particularly the feedback messages between questions, currently make for a clunky experience; at minimum, they need better styling to feel more visually dynamic. This is required before milestone 3.
3. Adjust /answer → /question (or /results) page redirects for a cleaner and less clunky transition
    - Primarily, this is about automatically redirecting away from the separate '/answer' interface as soon as the answer is graded, rather than using it to display the user message, which feels clunky. However, this will require a solution to #2 above that allows for a temporary message to be displayed on the question view, which may require JavaScript; I need to investigate

July 28:

4. Add correct answer to user feedback message when incorrect message is submitted
    - Currently, the user gets a binary "you were correct/incorrect" type message after they submit their to a question; it would be a better UX if they didn't just see "you were wrong" but also "XYZ was the correct answer."
5. Revise any remaining hard-coded items to be flexible
    - This is mostly a final review to make sure nothing is locked in unnecessarily, like hard-coding 13 as the number of questions in score calculations, which would need to be hunted down later and fixed if the code were to be reused or new capabilities were to be added

August 4:

6. Add any final graphics or visual enhancements as needed
7. Launch on PythonAnywhere for live testing/demo

These two items are not official milestones but are product enhancements I'll try to tackle if the milestones take less time than expected:

- [Pending time] Display countdown timer after user selects start button
    - Currently, the user selects the play button on the homepage and they see a static message that the game is about to begin; it's a similar issue to what I'm addressing in #3. A better game experience would be to see a timer visually counting down to the first question (builds anticipation); this will likely require JS.
- [Pending time] Allow user to choose from more than one game option
    - For instance, easy vs difficult, or long or short; add the option either at the beginning on the homepage or as an alternative next step when the initial game ends.