Project Documentation: Two-Player AsyncTriviaEngine

Project Overview:

Two-Player AsyncTriviaEngine is a two-player trivia game designed to test JavaScript skills, specifically focusing on game logic, turn-based functionality, and API integration. The game involves fetching trivia questions from an external API, alternating turns between two players, and scoring based on the difficulty of questions answered. The project emphasizes smooth user flow, score management, and real-time feedback.

Key Features

- **Two-Player Turn-Based Gameplay:** Players take alternate turns answering trivia questions from three categories: Easy, Medium, and Hard.

- **Trivia API Integration:** Trivia questions are fetched from an external trivia API, dynamically displayed for each player.

- **Scoring System**: Players earn different points based on the difficulty of the question:

  - Easy: 10 points

  - Medium: 15 points

  - Hard: 20 points

- **Responsive UI & Notifications**: Player progress, scores, and end-game results are displayed in real time. A notification announces the winning player at the end of the game.

- Score Saving: At the end of the game, players can save their scores locally for future reference.

Project Plan

Problem Breakdown:

The game involves several smaller tasks that need to be handled step by step. These tasks include:

1. Player Setup: Allow two players to input their names before starting the game.

2. Category and Difficulty Selection: Fetch and filter questions based on difficulty levels from the API.

3. Turn Management: Alternate between players while displaying questions to each player in a turn-based system.

4. Score Management: Update and display player scores after each round based on correct answers.

5. Game Flow Management: Control the flow from the start to the end of the game, ensuring a smooth transition between different game phases.

6. End-Game Handling: Display results based on player scores and provide options to restart the game or save scores.

Approach for Each Task

1. Player Setup

Goal: Allow two players to enter their names and start the game.

Approach:

- Create an initial HTML form that takes the names of two players.

- On form submission, store the player names and redirect to the main game page.

- Validate that both names are provided before proceeding.

2. Category and Difficulty Selection

Goal: Fetch questions from The Trivia API based on difficulty.

Approach:

- Make an API call using JavaScript's `fetch () ` method to retrieve 6 questions per game (2 easy, 2 medium, 2 hard).

- Store the questions in an array for use during the game.

- Shuffle the array to randomize the order of questions.

3. Turn Management

Goal: Manage the alternate turn system for both players.

Approach:

- Use a variable to track the current player (e.g., `currentPlayer = 1` or `currentPlayer = 2`).

- Toggle the player after each question is answered, ensuring Player 1 always answers the odd-numbered questions and Player 2 answers the even-numbered ones.

- Display whose turn it is above each question.

4. Score Management

Goal: Track and display player scores based on the correctness of their answers.

Approach:

- Initialize a score variable for each player (e.g., `scorePlayer1 = 0`, `scorePlayer2 = 0`).
- After each question is answered, update the score based on whether the answer was correct:
  - Easy: +10 points
  - Medium: +15 points
  - Hard: +20 points
- Display the current scores at the top of the game screen.

5. Game Flow Management

Goal: Control the flow of the game and ensure smooth transitions.

Approach:

- Start the game with player setup, move to question display, and alternate turns until all questions are answered.
- Once all questions have been answered, transition to the end screen.
- Provide real-time feedback after each turn (e.g., correct or incorrect).

6. End-Game Handling

Goal: Display the winner and provide options to restart the game or save scores.

Approach:

- Compare player scores after all questions have been answered to determine the winner.
- Display the winning player’s name and score.
- Provide buttons for restarting the game or saving the final scores locally.

Code Structure

HTML Files:

- index.html: The initial form where players enter their names.
- game.html: The main game interface where questions are displayed, and players take turns answering.
- end.html: The end-game screen that shows the final scores and the winner.

JavaScript Files:

- index.js: Handles player setup and redirects to the main game.
- game.js: Manages turn-taking, scoring, question fetching, and displaying real-time notifications.
- end.js: Displays the winner and provides options to save scores or restart the game.

CSS Files:

- app.css: General styling for the entire game.
- game.css: Specific styling for the question display, turn indicators, and score tracker.
- end.css: Styling for the end screen and notification popups.

Steps for Implementation

1. Set Up HTML & CSS:

- Build the basic structure for player input (index.html), the game interface (game.html), and the end-game screen (end.html).
- Apply responsive CSS styles for a clean, user-friendly interface.

2. Fetch Trivia Questions:

- Use `fetch () ` to retrieve 6 questions (2 easy, 2 medium, 2 hard) from The Trivia API.
- Shuffle the questions and store them in a variable for easy access.

3. Implement Turn-Based Logic:

- Write a function to toggle between Player 1 and Player 2 after each question.

- Display whose turn it is using DOM manipulation.

4. Score Tracking & Display:

- Create functions to update the scores based on the player's correct/incorrect answer.
- Display scores in real-time and update them on the screen after each question.

5. Handle Game End:

- Check when all questions are answered and trigger the end-game sequence.
- Calculate and compare scores to determine the winner.
- Redirect to `end.html` with a display of the final scores and the option to restart.

Conclusion:

Two-Player AsyncTriviaEngine is a robust two-player trivia game that combines real-time API integration, turn-based gameplay, and intuitive user flow. The project makes effective use of JavaScript for fetching data, managing state, and ensuring a smooth user experience. Through structured game logic and real-time feedback, the game provides an interactive and engaging trivia challenge for two players.