Stepan Manaenko

manaeste@fel.cvut.cz

Quiz App Documentation

June 1, 2023

Project Goal

The goal of this project is to create a web-based quiz application that tests the user's knowledge in a particular topic. The app should present multiple-choice and true/false questions to the user and provide feedback on their answers. At the end of the quiz, the user should receive their score and have the option to repeat the quiz.

Implementation Steps

1. Design the HTML structure for the quiz app, including the main container, question display area, progress indicator, and buttons container.

2. Style the app using CSS to create an appealing and user-friendly interface. Define variables for primary and secondary colors, background color, text color, and other necessary styles.

3. Create the JavaScript classes for the quiz, questions, and true/false questions. Implement methods for retrieving the current question, checking the user's answer, and determining if the quiz has ended. Use local storage to save and retrieve the quiz state, including the current question index, score, and remaining time.

4. Write JavaScript functions to handle the display of questions, choices, and progress. Update the UI dynamically based on the current quiz state. Implement a function to show the final score and provide an option to repeat the quiz.
5. Add event listeners to the buttons and handle user interaction. When a user selects an answer, check if it is correct and update the score accordingly. Advance to the next question and update the UI.
6. Implement a countdown timer that limits the time available for each question. Display the remaining time and end the quiz when the time runs out.
7. Create an SVG image based on the user's score and display it on the UI. Add animation effects to the image for visual appeal.
8. Set up a service worker to cache the app files for offline usage. Define cache files and handle fetch requests to respond with cached files when available.

Functionality Description

1. Upon loading the quiz app, the user is presented with a title indicating the topic of the quiz and the first question.
2. The app retrieves the user's previous progress from local storage. If there is no saved progress, a new quiz is initialized.
3. The app displays the current question, along with the available choices. For multiple-choice questions, all options are presented as buttons. For true/false questions, only the "True" and "False" buttons are displayed.
4. The user selects an answer by clicking on the corresponding button. The app checks if the answer is correct and updates the score accordingly. The next question is then displayed.

5. The app shows the progress of the quiz, indicating the current question number and the total number of questions.

6. A countdown timer starts at the beginning of the quiz, displaying the remaining time. When the timer reaches zero, the quiz ends and the final score is shown.

7. At the end of the quiz, the app determines if the user succeeded or failed based on the score. Minimum score is 5/7. The final score, along with a success/failure message, is displayed. An SVG image corresponding to the quiz result is shown, with animation effects upon clicking.

8. The user has the option to repeat the quiz by clicking the provided button. Clicking the button resets the quiz, including the score, question index, and remaining time. The quiz starts again from the beginning.

9. The quiz should be working even if the server connection failed. It uses Service Worker API for caching.

10. The questions and the quiz title can be directly changed if needed in the source code (file quiz.js).

Conclusion

This web-based quiz app offers an effective and enjoyable way for users to assess their knowledge in a particular topic while providing a seamless and interactive experience.