**MODULE - 8**

**Assignment 15: Add a Timer for Test Sections**

**Objective**

Implement a timer component using react-timer-hook to manage the duration of each test section. Display the timer alongside the test questions to help users track their remaining time effectively.

To tackle the problem of adding a timer for test sections in an IELTS Speaking Test platform using React, we will follow a structured and optimized approach. Below is the step-by-step instruction to solve this problem effectively.

**Step-by-Step Approach**

**Project Setup:**

Ensure that your React environment is set up. If not, create a new React project using create-react-app or any other preferred method.

Install the react-timer-hook library (or an equivalent library) which will be used for the timer. Use npm install react-timer-hook or yarn add react-timer-hook.

**Timer Integration:**

Create a new component, e.g., TimerComponent.js, in your React project.

Import the necessary functions from react-timer-hook.

Configure the timer to start from a predefined duration (e.g., 2 minutes per question).

**Component Design:**

Ensure the timer is displayed prominently alongside the test questions. This can be achieved by placing the timer within the test question component or alongside it in the rendering hierarchy.

Implement visual cues to signal when the time is about to expire. For instance, change the text color to red when there are only 10 seconds remaining, or display a warning message.

**State Management:**

Use React's state management (via useState and useEffect) to handle the timer’s start, pause, and reset functionalities.

Synchronize the timer with the test question progression by updating the state when a new question is presented or when the user interacts with the next/previous question buttons.

**Error Handling:**

Handle edge cases such as:

The timer expiring by triggering an event or callback to handle end-of-time scenarios (e.g., automatically move to the next question or notify the user).

Users navigating away mid-test by using useEffect to listen for page visibility changes or using browser events to pause/continue the timer accordingly.

**Styling:**

Ensure the timer component is visually appealing. Apply styles that align with the rest of the platform’s UI theme. You can use CSS/SCSS modules or styled-components for modular and reusable styles.

Provide accessibility features such as:

Screen reader compatibility by using ARIA attributes.

Clear visual indicators for users with color vision deficiency (e.g., using icons or additional text along with color changes).

**Testing:**

Test the timer functionality extensively:

Check that the timer counts down accurately.

Ensure it starts, pauses, and resets as expected.

Validate that it synchronizes correctly with the test section transitions.

Capture screenshots or screen recordings showing the timer in action alongside the test questions.

Deliverables

**Timer Component File:**

Submit the TimerComponent.js file and any other updated files (e.g., test interface components).

**Setup Instructions:**

Provide a README detailing the setup process, including installation steps, how to run the project, and how to test the timer functionality.

**Testing Evidence:**

Include screenshots or screen recordings demonstrating the timer working in conjunction with the test interface.

Evaluation Criteria

Make sure to address all the evaluation criteria as described:

Timer Functionality (40%):

Ensure the timer accurately counts down, starts, pauses, and resets.

Integration with Test Interface (30%):

Ensure the timer is synchronized with the test sections and is prominently displayed.

Styling and User Experience (20%):

Ensure the timer is visually appealing, provides clear feedback (color changes, warnings), and is accessible.

Submission Completeness (10%):

Include all required files, setup instructions, and testing evidence.

**Key Considerations**

**Reusability and Modularity:**

Design the timer component such that it can be easily reused for different test sections or other timing functionalities within the platform.

**Accessibility:**

Incorporate accessibility features ensuring the timer is usable by individuals with disabilities.