**MODULE - 8**

**Assignment 16: Enhance UI with Modals**

**Objective**

Create a reusable modal component to display test instructions, results, or other important information. Integrate this modal into the test dashboard for displaying key test-related content dynamically.

To approach the problem of enhancing the UI with modals, follow these step-by-step instructions:

**Step-by-Step Approach:**

**1. Understand Requirements:**

Modal Component:

Create a reusable modal component that accepts content via props.

Dynamic content support.

Features:

Include an overlay for emphasis on modal content.

Smooth open/close transitions.

Keyboard navigation accessibility.

Integration:

Show test instructions before starting the test.

Display results after completing the test.

Styling:

Align with platform’s UI.

Customization options (modal size, button styles).

**2. Design Modal Component:**

Props Design:

Determine necessary props:

content (the dynamic content to display),

isOpen (boolean to manage visibility),

onClose (function to handle closing).

Include additional props for customization (e.g., size, button styles).

Structure & Elements:

Modal container.

Overlay/background.

Close button.

Content area.

**3. Implement Features:**

Overlay:

Create a semi-transparent layer to highlight modal content.

Position it to cover the entire viewport.

Open/Close Functionality:

Implement smooth transition animations (e.g., fade, slide).

Use state management for controlling the modal’s visibility.

Accessibility:

Ensure the modal can be navigated via keyboard:

Focus management when modal opens.

Close modal with ESC key.

Tab through interactive elements inside modal.

**4. Integrate Modal into Test Dashboard:**

Before Starting Test:

Add a button (e.g., “View Instructions”) to trigger the modal with test instructions content.

After Completing Test:

On test completion, set the modal content with test results and trigger it.

**5. Styling:**

Align with Existing UI:

Follow the current design language and styles of the platform.

Customization Options:

Allow adjustments like modal size, button styles via props.

**6. Testing:**

Modal Triggering:

Click button to open the modal.

Verify modal displays correctly upon opening.

Dynamic Content Rendering:

Pass different contents (instructions and results) to modal via props.

Ensure the modal dynamically updates content as required.

Accessibility:

Navigate the modal with keyboard:

Ensure elements are reachable and focusable.

Validate escape key functionality to close the modal.

Check tab navigation within the modal.

**7. Submission:**

Files and Updates:

Submit the modal component file.

Provide updates made to the test dashboard integration.

Evidence:

Include screenshots or screen recordings demonstrating modal functionality.

**8. Evaluation Criteria:**

Reusability:

Confirm modal handles various content and conforms to different use cases (test instructions, results).

Integration:

Validate smooth operation within the test dashboard.

Styling and User Experience:

Ensure modal is visually appealing and user-friendly.

**Submission Completeness:**

Verify all required files, evidence, and functionalities are included.

By following these structured and detailed steps, you ensure an optimized solution that meets the requirements and is suitable for competitive programming test cases.