**MODULE - 2**

**Assignment 4: Create Responsive Layouts**

**Objective**

**Build a responsive layout for the "Test Overview" and "Results" sections of the IELTS Speaking Test platform using CSS Flexbox and Grid. The layout should adapt seamlessly to mobile, tablet, and desktop devices.**

**Scenario**

**The IELTS Speaking Test platform requires a dashboard interface that displays an overview of the test and results. Your task is to design a visually appealing and responsive layout for these sections.**

To create a responsive layout for the "Test Overview" and "Results" sections of the IELTS Speaking Test platform using CSS Flexbox and Grid, follow these optimized step-by-step instructions. This approach ensures that the layout adapts seamlessly across mobile, tablet, and desktop devices.

**Step-by-Step Approach**

**Step 1: Set Up the Project Structure**

**1.Create a Project Directory:**

**-** Create a directory for your project to keep all files organized.

- Example: mkdir ielts\_dashboard && cd ielts\_dashboard

**2.Create Necessary Files:**

- Create an HTML file named dashboard\_layout.html.

- Create a CSS file named styles.css.

**Step 2: HTML Structure**

**1.Basic HTML Template:**

- Create a basic HTML structure with <!DOCTYPE html>, <html>, <head>, and <body> tags.

- Link the CSS file using the <link> tag in the <head> section.

**2.Test Overview Section:**

- Use semantic elements for better readability and SEO.

- Create a section with information about the test (Test title, Current test section, Time remaining).

**3.Results Section:**

- Create a grid layout to display user performance.

- Use columns for Section, Score, and Status, and rows for data entries.

**Step 3: Styling with CSS**

**1.Use Flexbox for the Test Overview Section:**

- Align items horizontally on desktop.

- Stack items vertically on smaller screens using media queries.

**2.Use CSS Grid for the Results Section:**

- Create a table-like structure with columns and rows.

- Ensure the grid is responsive and adapts to different screen sizes.

**3.Add Media Queries for Responsiveness:**

- Mobile: Stack items vertically, reduce padding and margins.

- Tablet: Implement a two-column layout where applicable.

- Desktop: Use a full-width layout with horizontal alignment.

**4.Styling and Hover Effects:**

- Apply a consistent color scheme.

- Add hover effects for interactivity.

**Detailed Steps**

**Step 1: Set Up the Project Structure**

**1.Create files:**

- dashboard\_layout.html

- styles.css

**Step 2: HTML Structure**

**1.Basic HTML Template:**

- Create the basic structure and link the CSS file.

**2.Test Overview Section:**

- Use <header>, <section>, and <div> elements to structure the content.

- Add placeholders for test title, current section, and time remaining.

**3.Results Section:**

- Use a <section> element for the results.

- Create a grid using <div> or <table> elements for data.

**Step 3: Styling with CSS**

**1.Flexbox for Test Overview Section:**

- Use Flexbox to align items horizontally on larger screens.

- Stack items vertically on smaller screens using media queries.

**2.CSS Grid for Results Section:**

- Define a CSS grid to handle the layout of columns and rows.

**3.Responsive Design:**

- Use media queries to adjust the layout based on screen sizes.

- Ensure content is accessible on mobile, tablet, and desktop devices.

Example Layout Guide (in prose, not code)

**Test Overview Section (Flexbox)**

**1.Desktop Layout:**

- Items aligned horizontally.

- Ensure equal spacing and alignment using Flexbox properties.

**2.Mobile Layout:**

- Items stacked vertically.

- Adjust font sizes and spacing for readability.

**Results Section (CSS Grid)**

**1.Grid Layout:**

- Create columns for Section, Score, and Status.

- Use rows for each data entry.

**2.Responsive Design:**

- Adjust column width and padding for different screen sizes.

**Steps for Creating the Layout**

**Step 1: HTML Structure**

**1.In dashboard\_layout.html:**

- Create the basic HTML structure.

- Add semantic elements (<header>, <section>, <footer>).

**2.Content for Test Overview Section:**

- Use header and div tags to structure the title, current section, and time remaining.

**3.Content for Results Section:**

- Use a section element for the results container.

- Structure the data in a tabular format using div or table elements.

**Step 2: Styling with CSS**

**1.Initialize CSS:**

- In styles.css, start with basic styles and reset any default browser styling.

**2.Apply Flexbox to Test Overview Section:**

- Use Flexbox properties (display: flex, justify-content: space-between) for horizontal alignment on desktop.

- Use media queries to adjust the layout for smaller screens (e.g., flex-direction: column).

**3.Apply CSS Grid to Results Section:**

- Define grid properties (display: grid, grid-template-columns) for consistent column layout.

- Use media queries to ensure responsiveness.

**4.Use Media Queries for Responsiveness:**

- Define breakpoints for mobile, tablet, and desktop views.

- Adjust layout, padding, and margin properties to ensure visual consistency across devices.

**Step 3: Final Adjustments and Testing**

**1.Ensure Cross-Browser Compatibility:**

- Test the layout on various browsers and devices.

- Use developer tools to simulate different screen sizes.

**2.Optimize and Clean Code:**

- Review and refine the HTML and CSS for readability and performance.

- Ensure accessibility features are in place (e.g., ARIA labels, proper HTML semantics).

**Submission Guidelines**

**1.Include All Required Files:**

- Ensure dashboard\_layout.html and styles.css files are included.

**2.Include Screenshots or GIFs:**

- Provide visual evidence of the layout on different devices (mobile, tablet, desktop).