INTEGRATIVE PROGRAMMING AND TECHNOLOGIES LAB

Performance Task 1 *(This is a group activity)*

Title: Integrating Interactive stories feature on Web application by utilizing DOM.

Description: This activity focuses on creating an **Interactive Stories Feature** like those found on social media platforms like Facebook and Instagram on a full Web application (Identify the niche of you web app and it should be explicitly obvious in your online application). The implementation of this activity leverages modern **HTML**, **CSS**, and **JavaScript** to provide a dynamic and responsive user experience by integrating DOM techniques.

Objectives:

- **Implement Dynamic DOM Manipulation** - dynamically create, modify, and manage HTML elements using JavaScript to build interactive features like story uploads and display.
- **Utilize Event Listeners for User Interaction** - attach and manage event listeners to handle user actions, such as clicking on a story, uploading files, and transitioning between media elements.
- **Design Responsive and Adaptive UI Components** - create responsive web designs with CSS, including aspect ratio constraints (9:16) and scrollable containers for effective media presentation.
- **Incorporate Media Playback and Timing Control** - master handling media files (images and videos) programmatically, including setting playback durations and using timers for automated story transitions.
- **Implement Progress Indicators with Visual Feedback** - create and animate progress bars in sync with content duration, enhancing the user experience by providing visual feedback.
- **Enhance User Experience for Niche Audiences** - implement auto-fit content to avoid cropping or distortion, ensuring stories display consistently across devices.
- **Facilitate Seamless Story Interaction and Engagement** - enable manual navigation between stories with intuitive "Next" and "Previous" buttons to improve usability.
- **Optimize Performance for High Scalability** - Use lazy loading for thumbnails and preloading for the next story to improve application performance.
- **Empower Users with Creative and Interactive Features** - enhance user engagement by introducing audio integration, reactions, and comments, promoting a more vibrant user interaction.

Key DOM Features in the Code:

1. **Dynamic Story Creation**:
    - When a user uploads files, new div elements representing stories are dynamically created using document.createElement().
    - Images or videos are appended to the story elements with DOM methods like appendChild().
2. **Event Handling**:
    - Event listeners are added to various elements:
        - Clicking a story thumbnail (addEventListener('click', ...)) opens the fullscreen viewer.
        - Video metadata events like onloadedmetadata are used to set the progress bar's duration.
3. **Content Modification**:
    - The innerHTML property is used to populate or clear the content of elements like the story viewer.
    - The textContent property updates the story title in the viewer dynamically.
4. **Style Manipulation**:
    - CSS properties like width, transition, and display are modified at runtime using the style property.
5. **Responsive Interactions**:
    - The progress bar dynamically updates its width using DOM manipulation.
    - The setTimeout function is used to control when the progress bar completes and the story transitions.
6. **Dynamic Queue Management**:
    - Stories are stored in an array (storyQueue), which maps elements in the DOM to logical objects (image/video sources and titles).
    - DOM elements are filtered using the Array.from() method combined with queries like querySelector().
    -

Examples of DOM Methods Used:

- **getElementById()**: To access elements like the story container or viewer.
- **createElement()**: To create new HTML elements for stories.
- **querySelector()**: To find specific child elements within a story.
- **appendChild()**: To add newly created elements to the DOM.
- **addEventListener()**: To attach event listeners for user actions.

Features Already Implemented:

1. **Dynamic Story Upload and Management**:
   - Users can upload an unlimited number of media files (images and videos) with optional titles.
   - Uploaded stories are dynamically added to a horizontally scrollable container.
2. **9:16 Aspect Ratio Enforcement**:
   - Both story thumbnails and the fullscreen viewer are constrained to a 9:16 aspect ratio using CSS techniques like object-fit and calculated dimensions.
3. **Fullscreen Story Viewer**:
   - Clicking a story thumbnail opens it in a fullscreen viewer.
   - The viewer dynamically displays the story title and the corresponding media (image or video).
4. **Media Playback Timing**:
   - Images are displayed for **5 seconds**, and videos for **15 seconds** or their actual duration if shorter.
   - Automatic transitions occur between consecutive stories.
5. **Progress Bar Implementation**:
   - Each story includes a progress bar that visually indicates the playback duration.
   - The progress bar is updated dynamically using JavaScript and CSS transitions.
6. **Responsive and Scalable Design**:
   - The story container uses a horizontally scrollable design to handle an unlimited number of stories without affecting usability.
   - CSS ensures the UI remains visually appealing and functional across various screen sizes.

Additional Requirements/Enhancements – PT1:

Ensure that your web application has a niche.

- i.e. social media site, sport site, education site, entertainment site, blog site etc. with the story feature in the background. Add a banner image in your web app as part of your header.

Story Viewer to 9:16 Aspect ratio:

- Clicking a story thumbnail opens it in a 9:16 viewer. Ensure the story viewer strictly adheres to the 9:16 aspect ratio. This can be achieved with CSS constraints and dynamic scaling for different screen sizes. The *story-viewer-content* container can be modified to calculate and maintain this ratio dynamically.
- **Auto-Fit Content in Viewer** - use the object-fit: contain property for images and videos in the story viewer to ensure proper scaling and no cropping.

Enable Manual Navigation Between Stories

- Include "Next" and "Previous" buttons to allow users to navigate between stories manually.

Include Story Indicators

- Add small dots or bars at the bottom of the story viewer to indicate the total number of stories and the current story being displayed. This improves user awareness and navigation.

Optimize Performance for Large Numbers of Stories

- Implement lazy loading for story thumbnails and preload the next story to improve performance when handling large numbers of stories.

Enhance Story Upload Options

- Add features like cropping, rotating, or resizing images and trimming videos before posting to improve user control over their uploaded content.

Accessibility Improvements

- Use ARIA roles and attributes to make the activity more accessible to users with disabilities. For example:
  - Provide keyboard navigation for story thumbnails and the viewer.
  - Include descriptive alt text for images and captions for videos.

Add a Confirmation Prompt Before Uploading

- Display a preview of the selected stories with a confirmation prompt to ensure users are satisfied before uploading.

Audio Integration

- Allow users to add background music or voiceover to their stories.

Interactive Reactions and Comments

- Allow users to react (like, love, laugh, etc.) or send quick emojis while viewing stories.

Use the structure and framework that you have created in your OE3 activity and leverage the given code on that activity. However, you can also start from scratch if it seems more practical and technically viable.