Exercise 2:

Simplified Movie Editing App

Part 1 (Basic Implementation)
Difficulty Rating: 5/10

Overview: In this initial stage of the exercise, you'll lay the foundation for a simplified app. Your primary goal is to delve into the intricacies of JSON while building a simplified movie editing app. The app is organized into distinct tabs, each serving a specific purpose - from managing folders and files to project-saving functionalities. As part of the task, you'll fetch and parse data from a **root-folder.json** file, housing folders with sets of movie files. Your mission involves rendering folders in the "Folders" tab, allowing users to click and explore the associated files in the "Files" tab. The core functionality lies in loading selected files onto the editing bars (for now only onto T1, and later on T2 aswell), where the user can select a folder and a file, click an editing bar, and calculate the file's duration percentage relative to the 15-minute editing bar. The result is a red-background div, symbolizing the file's position. Additionally, you'll implement project-saving capabilities, prompting users to insert a project name and click the "Save Project" button. The saved project will appear in the "Saved Projects" tab, and editing bars will clear. To complete this part, users should be able to load a saved project, ensuring existing files on the editing bar are cleared before loading.

Instructions:

1. **HTML and CSS:**
    - Examine the provided HTML and CSS files to understand the structure and classes used in the app. Note that the provided HTML includes a hard-coded version of files, folders, files in the editing bar, and saved projects. This serves as a visual reference for how the app should look once dynamic implementation is complete. Remember to remove the hard-coded sections once you have successfully implemented the dynamic functionality.
2. **Tabs Overview:**
    - Familiarize yourself with the app's tabs, which include:
        - Folders
        - Files
        - Project Info
        - Saved Projects
        - Save Progress
        - Preview
        - Editing Section (T1 and T2 editing bars)

- Buttons Section

3. **Fetch and Parse JSON:**
    - Fetch and parse the **root-folder.json** file, which contains an array of folders.
    - Each folder has a folder name, folder id, and an array of files (movies).
    - Each file has a file name, duration, previewURL, and thumbnailURL.
4. **Render Folders and Files:**
    - Render all folders from the root folder in the "Folders" tab.
    - Clicking on a folder should make it active, displaying its files (movies) in the "Files" tab.
    - Clicking on a file should make it active.
5. **Editing Bars:**
    - Select a folder, then a file to make it active.
    - Click on one of the two editing bars (T1 or T2) to load the file onto the bar.
    - Calculate the file's duration percentage relative to the 15-minute editing bar and insert a red-background div with the calculated width.
6. **Project Saving:**
    - Insert a project name in the "Project Info" tab.
    - Click the "Save Project" button in the "Buttons" tab.
    - The saved project is displayed in the "Saved Projects" tab.
    - After saving, clear the editing bars.
7. **Loading Saved Project:**
    - Click on a saved project in the "Saved Projects" tab.
    - The project loads, displaying the files used in the T1 editing bar.
    - Existing files on the editing bar before loading should be cleared.

Example Scenario:

Imagine you have a collection of vacation videos categorized into folders. You want to create a project named "Summer Memories." Start by selecting the "Family Vacation" folder, choose the "beach_day.mp4" file, and load it onto the T1 editing bar. Continue adding more files from various folders to create a dynamic sequence. Save your project, and later, load it to revisit those cherished moments.

Part 2 (Enhancing User-Friendly UI)
Difficulty Rating: 3/10

Overview: In this second part of the exercise, you'll elevate the user experience by implementing additional features to enhance the user interface of the movie editing app. Each file now has two URLs: preview and thumbnail. The thumbnail will serve as the background for each file displayed in the "Files" section, resembling a typical video thumbnail. Additionally, when a file is loaded onto the editing bar, instead of a red background div, the background will now be an image of the **previewURL**, repeating if the file's duration exceeds the image width.

Furthermore, an editing file on the editing bar can now become active. Each time a new file is added to the bar, that editing file becomes active (the last one added). You can change the active editing file by clicking on any other editing file. When an editing file is active, its preview image is displayed in the "Preview" tab of the app.

New Features:

1. **Thumbnail as Background:**
    - Use the thumbnail URL as the background for each file displayed in the "Files" section.
2. **Preview Image on Editing Bar:**
    - Instead of a red background div, use the **previewURL** as the background for an editing file on the editing bar. Repeat the image if the file's duration exceeds the image width.
3. **Active Editing File:**
    - Make the last added file on the editing bar the active editing file.
    - Click on any editing file to make it active.
    - The active editing file's preview image is shown in the "Preview" tab.
4. **Deleting Editing Files:**
    - Delete an editing file by making it active and clicking the "Delete Element" button in the "Buttons" tab.

Example Scenario: Now, when you load a file onto the editing bar, not only does its **previewURL** become the background, but you can also make it active to see its preview image in the "Preview" tab. Moreover, you have the flexibility to delete an editing file directly from the editing bar, streamlining the editing process.

Note: This exercise aims to improve the user interface of the movie editing app. Subsequent parts will introduce even more advanced features.

Part 3 (Improving Capabilities)
Difficulty Rating: 3/10

Overview: In this third part of the exercise, you'll further enhance the capabilities of the movie editing app. Users can now choose between a 15-minute and a 20-minute track by clicking the radio button in the "Project Info" tab. Additionally, users can personalize their projects by choosing a background color. This color will be applied as the background color of the saved project "div" in the "Saved Projects" tab.

In this part, you'll also define the behavior when there is no more space on the editing bar. An alert pop-up should be displayed, informing the user that inserting the file into the editing bar is not possible because it will exceed the specified project duration.

New Features:

1. **Choose Track Duration:**
    - Take in mind the radio buttons in the "Project Info" tab allowing users to choose between a 15-minute and a 20-minute track duration.
2. **Choose Project Color:**
    - Implement the option for users to choose a background color for their projects in the "Project Info" tab.
3. **Background Color for Saved Project:**
    - Apply the chosen background color as the background color of the saved project "div" in the "Saved Projects" tab.
4. **Alert for Full Editing Bar:**
    - Implement an alert pop-up when attempting to insert a file into the editing bar if it would exceed the specified project duration.

Example Scenario: Now, users have the flexibility to choose the duration of their project and personalize it with a background color. When attempting to insert a file into the editing bar, they will be alerted if it would exceed the specified project duration, ensuring a seamless editing experience.

Note: This exercise aims to improve the capabilities of the movie editing app. Subsequent parts will continue to add more advanced functionalities.

Part 4 (Simulating Real Behavior)
Difficulty Rating: 4/10

Overview: In this short yet impactful part of the exercise, you will simulate a waiting period that is required before a project is saved, simulating a render time. The render time is calculated based on the total length in minutes of the editing files on the editing bar, divided by 10. During this waiting period, a progress bar in the "Save Progress" tab (already hard-coded in the HTML and CSS) should start filling up, resembling a typical progress bar. The project is saved only when the progress bar is fully filled.

When attempting to load a project while there are files on the editing bar, a prompt (or a similar popup window) will ask the user whether they want to discard the unsaved project (files on the editing bar). If the user clicks "OK," the editing files are discarded, and the chosen saved project is loaded onto the editing bar. If the user clicks "Cancel," the loading process is canceled, and no changes occur. Additionally, the user's progress will be retained even after refreshing the page through local storage

New Features:

1. **Simulated Render Time:**
    - Calculate the render time based on the total length in minutes of the editing files on the editing bar, divided by 10.
    - Implement a progress bar in the "Save Progress" tab that starts filling up during the render time.
    - Save the project only when the progress bar is fully filled.
2. **Confirm Discarding Unsaved Project:**
    - When attempting to load a project with unsaved files on the editing bar, prompt the user with a confirmation window.
    - If the user clicks "OK," discard the unsaved project (delete editing files) and load the chosen saved project onto the editing bar.
    - If the user clicks "Cancel," cancel the loading process without making any changes.
3. **Local Storage for Saved Projects**:
    - Save each project in local storage when the user clicks the "Save Project" button.
    - Retrieve and load saved projects from local storage when the app is initialized or refreshed.

Instructions:

1. **Simulated Render Time:**
    - Calculate the render time based on the total length in minutes of the editing files on the editing bar, divided by 10.

- Implement a progress bar that fills up over time in the "Save Progress" tab during the render time.
- Save the project only when the progress bar is fully filled.

2. **Confirm Discarding Unsaved Project:**
    - When attempting to load a project with unsaved files on the editing bar, show a confirmation window to the user.
    - If the user clicks "OK," discard the unsaved project (delete editing files) and load the chosen saved project.
    - If the user clicks "Cancel," cancel the loading process without making any changes.
3. **Local Storage Implementation**:
    - When a user clicks the "Save Project" button, save the project details, including project name and configuration, in local storage.
    - When initializing or refreshing the app, check for saved projects in local storage and populate the "Saved Projects" tab accordingly.

Example Scenario: Now, users will experience a simulated render time before their project is saved, giving them a more realistic feel for the editing process. Additionally, when loading a project with unsaved files on the editing bar, they have the option to confirm or cancel the loading process. The local storage implementation ensures that saved projects persist even after refreshing the page, providing a seamless user experience.

Note: This exercise introduces a more realistic aspect to the movie editing app, simulating render times and enhancing user interactions.

Part 5 (Advanced Features Implementation - Second Editing Bar)
Difficulty Rating: 8/10

Overview: In this part, you will leverage the already coded second editing bar (T2) to enhance the capabilities of the movie editing app. The second editing bar runs parallel to the existing editing bar (T1), offering users more flexibility and intricate editing possibilities. T2 enables users to load files independently onto this bar, allowing for overlapping and precise control over the arrangement of videos in the project.

Detailed Instructions:

1. **Utilizing T2:**
    - Utilize the pre-coded HTML and CSS for the second editing bar (T2), ensuring it is displayed parallel to T1.
    - When inserting a file onto T2, follow the same procedure as inserting onto T1. After making a file active, click on the T2 bar to load the file onto it.
2. **Overlapping and Layering:**
    - Enable overlapping of files between T1 and T2, giving users the ability to create layered compositions.
    - Implement a visual representation of the overlapping parts to show users how different videos align in the project.
3. **Saving Projects with Overlapping Parts:**
    - When saving a project, consider only one track for each overlapped part. Identify the top file in the overlapping section and treat it as the primary file for that segment.
    - Provide a visual representation or explanation to guide users on how overlapping parts are managed during the save operation.

Example Scenario: Now, users have the ability to work with two editing bars (T1 and T2), offering a layered approach to video editing. They can load files onto either bar independently, allowing for overlapping and precise control over the arrangement of videos in the project. When saving a project with overlapping parts, only one track is considered for each overlapped segment, enhancing the clarity and organization of the editing process.

Note: This exercise introduces the implementation of the second editing bar, enhancing the movie editing app's capabilities. Subsequent parts can explore additional advanced features if desired.

T1: ****##** T2: **&&**@@ Saved Project and then Loaded: T1 or T2: ****##&&**@@	T1:****##** T2: **&&**@@ Saved Project and then Loaded: T1 or T2: ****##**@@
No OverLap;	With OverLap;