File Uploader Component – Javascript Engineer Technical Interview Task © Techzy

Objective:
Create a flexible and reusable `FileUploader` React component that enables users to upload, manage, and preview files. The component should support editing of newly uploaded and preloaded files (which may come from a backend or mock data), allow single or multiple uploads, and include an alt text field for each file.

Functional Requirements

Uploading

- Allow users to upload files by drag-and-drop or file input.
- Support both single and multiple file upload modes (configurable via a prop).
- Allow file type restrictions (e.g., images only) via props.

File Management

- Display file previews (e.g., thumbnails for images).
- Allow sorting files using drag-and-drop.
- Allow users to remove uploaded files.
- Allow users to input and edit alt text for each file.
- Support rendering existing files passed in via a prop (mocked or real).
- Allow editing of preloaded files (e.g., change alt text, replace file).

Data Structure
Each file should follow this structure:

```
{
  file: File | null;
  source: string | null; // URL or local preview
  sortIndex: number;
  altText?: string | null;
}
```

Technical Requirements

- **Use React (TypeScript preferred).**
- You may use any third-party libraries to handle file input, drag-and-drop sorting, UI components, previews, or other features.
- The UI must be fully functional and visually polished, not a barebones prototype.
- The component should be cleanly structured and ready for use in a real-world project.

How to Provide Your Solution

- **Use a public GitHub repository** to share your solution with us.
- The repository should have a **clear and comprehensive README**. This should include:
    - **Project Description:** A brief overview of the component's functionality.
    - **Approaches:** Outline any important design or technical decisions you made.
    - **Steps to Build and Run:** Include clear instructions on how to install dependencies, run the project, and test the component.
    - **Any additional notes** you feel are important for us to understand your work, including challenges faced or solutions implemented.
- **Important Note:**
  While we are not against using LLMs (Large Language Models), we **highly suggest not using LLMs for an end-to-end solution** as it is easily trackable. We have the tools to detect whether the solution was generated by an LLM. We are more interested in

evaluating your **technical skills**, creativity, and thought process, rather than using LLM-based solutions.

Evaluation Criteria

- **Code Quality:** Clean, modular, and maintainable code.
- **UI/UX Design:** User-friendly, polished, and functional UI.
- **Approach:** Logical and efficient approach to solving the problem.
- **Documentation:** Clear and concise documentation for the repository, including how to build and run the project.
- **Testing:** How well the component works under different scenarios (multiple file upload, drag-and-drop sorting, file type restrictions, etc.).