Mini Project 2: Course Explorer

This mini project involves importing course data from a JSON file, implementing filtering and sorting functionality on a webpage to help users explore different courses. The project aims to build a comprehensive user interface that allows for interactive data exploration, including filtering by course attributes and sorting course listings. This will give you practical experience in working with JSON data, JavaScript array methods, and DOM manipulation.

Please check the rubric before you start to see if there are points for implementing a feature a given way. For example, it is required to use the filter method for the filtering.

Basic Requirements

1. **HTML Layout**: Create a well−structured HTML page that provides an intuitive user interface for users to upload data, filter courses, and apply sorting. The layout should include an input element for file uploads, drop−down forms to filter based on different criteria, and a display area to show the details of a single focused course. Ensure the page is styled appropriately for easy navigation. You are not required to exactly replicate the style shown in the layout, but use CSS to visually define the different areas of the page (e.g., with colours or borders).

HTML Layout

2. **Data Loading**: The webpage should allow users to load course data from a JSON file. The data must be properly parsed, and any errors during loading should be handled gracefully, ensuring the program doesn't crash. Users will first browse and upload the JSON file containing course listings, after which the course data will be displayed in a list format. **Note: Due to browser security restrictions, JavaScript cannot directly access local files without user interaction. Therefore, users must manually select and upload the JSON file; attempting to directly load local files via JavaScript will not work.**
3. **Course Class Definition**: Define a `Course` class to encapsulate the details of each course entry, such as title, posted time, type, level, skill, and detail. Use JavaScript to create methods for retrieving and formatting course details.
4. **Filtering Functionality**: Implement filtering functionality that allows users to filter courses based on criteria such as course `Level`, `Credits`, and `Instructor` (see image for complete list). The filtering options should be generated based on the available data, and the project may be tested on slightly different objects than those provided. (That is, **don't** hard−code a list of categories!) As users select different filters, the course listing should be updated to reflect only those course that match the selected criteria. The user should be able to select multiple filters at once and have them all applied.

Filters

5. **Sorting Functionality**: Provide sorting options to allow users to sort courses by title, id, or date. You will need to write a function to sort ! You **may** assume that any dates you recieve are in the exact format "Winter XXXX", "Spring XXXX", "Summer XXXX", or "Fall XXXX" where XXXX is a 4−digit year (like 2025 or 2026).

Sorting

6. **Error Handling**: Ensure the application handles incomplete or incorrect data gracefully, providing informative error messages without crashing.

Error

7. **Interactive Course Details**: Users should be able to click on each course listing to view more detailed information about the course, including attributes such as id, title, and credits. (See the first image for all attributes.)

Details

Grading Criteria

Functional Requirements and Points Allocation

1. HTML Layout and User Interface (1 Point)

- **Knowledge Points**: HTML structure, CSS styling, basic UI design.
  - **Proper HTML Layout (1 Point)**: Ensure a well−structured HTML layout to display course data and interactive elements such as filtering and sorting dropdowns. The webpage should provide an intuitive user interface that is easy to navigate. The page should include an input element to load data, filter forms, and a section to display course listings.

2. Load Course Data from JSON File (1 Point)

- **Knowledge Points**: File handling, JSON parsing.
  - **File Input Handling and Data Loading (1 Point)**: The project should allow the user to select a JSON file via an `<input type="file">` element. The file should be read using a `FileReader` to parse the JSON data and create Course objects. Implement error handling to ensure that if the file is missing or the format is incorrect, an appropriate error message is displayed.

3. Define and Use a `Course` Class (1 Point)

- **Knowledge Points**: Object−oriented programming, constructors.
  - **course Class Creation (1 Point)**: The `Course` class should encapsulate all relevant course details, including title, posted time, type, level, skill, and detail. The class should have a constructor and methods to provide detailed information about each course. Ensure that the `Course` class provides methods for retrieving formatted details and creating course instances from data.

4. Filtering Functionality (2 Points)

- **Knowledge Points**: Array methods (`filter`), DOM manipulation.
  - **Filter by Criteria (1 Point)**: Implement filtering options that allow users to filter courses based on the criteria shown in the image. Use the `filter` method to achieve this.

- **Dropdown Generation (1 Point)**: Generate dropdown options based on the available course data, ensuring that users only see relevant filtering options. Use `Set` (look this up!) to collect unique filter values and update the dropdown menus.

5. Sorting Functionality (2 Points)

- **Knowledge Points**: Array methods (`sort`), data normalization.
  - **Sort by Title (1 Point)**: Implement a sorting option that allows users to sort courses alphabetically by title (A–Z and Z–A) and ID. Use the `sort` method to achieve this.
  - **Sort by Posted Time with Normalization (1 Point)**: Implement a sorting option that allows users to sort courses by Semester offered. (For example, Fall 2025 should come after Summer 2025 and before Winter 2026.)

6. Error Handling and Edge Cases (1 Point)

- **Knowledge Points**: Error handling, robustness.
  - **Error Handling (1 Point)**: Add error handling logic to manage data anomalies (e.g., missing fields) during the data loading and processing phases. Ensure that the program handles incomplete or incorrect data gracefully, displaying relevant error messages without crashing the application.

Total Points: 8

The grading will be based on the successful implementation of each feature as described above, with an emphasis on the underlying skills and concepts required for each functionality. Proper usage of JavaScript, understanding of DOM manipulation, object–oriented programming, and good coding practices are essential for achieving a high score on this project.

Submission Requirements

1. Submit index.html as well as any required scripts or stylesheets on Gradescope. **Make sure the repository you submit only contains files needed for this assignment.**
2. I recommend separating your JavaScript code, HTML code, and CSS for better clarity and maintainability.