Approach to Solve Problem

Problem statement is **Project Validator.**
Create a validator to evaluate code submissions for a learning platform's Web Development assignment. Your solution should provide clear feedback to users, indicating any areas where their code does not meet the **specified criteria**.

I wanted to created a project validator that can be used for any project. So, I have created a admin panel where the host/tutor can create an assignment and specify all the required criteria. Then when a user submits a solution to the assignment either by entering the code or uploading the file, the project is validated. I have designed the validator with an intuitive UI so that the host/tutor can specify any custom criteria needed within seconds without considering any technical details or writing the validator code.

The website mainly contains three pages. First is the create new assignment page where the host/tutor can create a new assignment and specify all the required criteria. Before the new assignment is stored, all the specified criteria are checked and validated. If there is any error, it is notified to the user.
Second is a page where all the assignments are displayed. Once the user chooses an assignment, the user will be directed to another page where the user can submit the code for validation. The backend will verify the code and provide the feedback to the user.

Assignment is modelled as :

{

Title: String,

Question: String,

Criteria:[CriteriaObj]

Difficulty: String,

CreatedAt: Date

}

CriteriaObj structure:

{

Selector:

Message:

Contains:[this]

}

The 'selector' key refers to the element (div, h1,..), the tutor is expecting there to be in the code. The 'message' key refers to the objective the tutor is expecting in words (Eg: Your portfolio should have a welcome section with an id of welcome-section). The 'contain' key is used to represent the further elements inside the particular element.

Core Logic:
Validation Function - ValidateHtml(html_code, Criteria).

The **validateHtml** function uses the **jsdom** library to create a DOM from the submitted HTML, enabling server-side manipulation of web pages. It iterates over a criteria array, applying each criterion to the DOM. For each criterion, it checks if elements matching a specified selector exist. If elements are found, it may perform additional checks, such as verifying attributes or ensuring content is not empty. Successful checks are marked as passed, while failures are marked as failed. If a criterion includes a **contains** property, the function performs nested checks within the context of the initially selected element, effectively creating a recursive validation pattern for more detailed scrutiny.

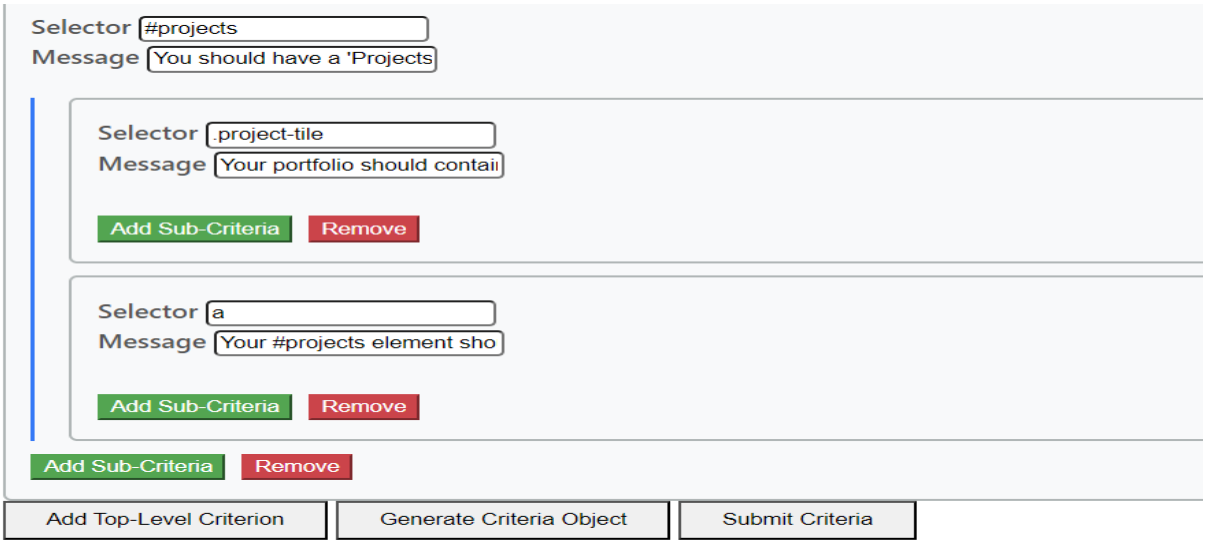
Defining Criteria :

The unique advantage of this solution lies in its ability to empower administrators, here assumed to be tutors with web development expertise, to swiftly create new assignments. This is achieved without the need for manually coding the validation logic. Instead, tutors can specify the desired criteria by listing required elements, using 'selectors' combined with certain symbols, and accompanying messages to articulate their expectations. For more detailed requirements within these elements, tutors can utilize the 'contains' feature to specify additional conditions. This streamlined approach facilitates a highly efficient and flexible assignment creation process, enabling tutors to focus on educational content rather than technical implementation.

For example, The Tutor has to create a assignment based on this criteria
Your portfolio should have a projects section with an id of projects

The projects section should contain at least one element with a class of project-tile to hold a project

The projects section should contain at least one link to a project



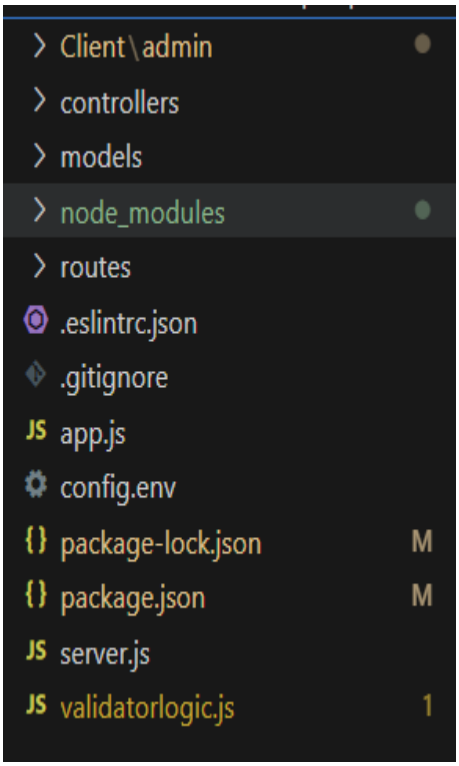

```
[
  {
    "selector": "#projects",
    "message": "You should have a 'Projects' section with an id of projects.",
    "contains": [
      {
        "selector": ".project-tile",
        "message": "Your portfolio should contain at least one element with a class of pro…"
      },
      {
        "selector": "a",
        "message": "Your #projects element should contain at least one a element."
      }
    ]
  }
]
```

The criteria system embedded within this solution offers a robust and flexible framework for validation, capable of addressing complex requirements. This power, however, brings with it the potential for complexity in the user interface and experience. To navigate this, tutors, leveraging their foundational understanding of web development principles, can efficiently construct intricate validation rules without getting entangled in the UI/UX complexity.

- **.name** targets elements by class,
- **#name** identifies elements by their ID,
- **[attribute='value']** is used for attribute presence and value verification,
- **[attribute1='value1'][attribute2='value2']** enables the checking of multiple attributes simultaneously,
- **Parentname>childname** ensures that the specified elements are directly related as parent and child.
- We also able to achieve regular expression matching by using ^=

If there is a wrong in criteria sends a failure response to tutors providing the reason the critera may fail

Code Structure and Setup Instructions:



To get started with the application, please follow these setup instructions:

1. **Starting the Server:**
   - **Navigate to the root folder of the project using a terminal or command prompt.**
   - **Execute the command node server.js to start the server. This command initiates the server and establishes a connection with MongoDB, ensuring the backend is up and running.**
2. **Accessing the Website:**
   - **With the server running, open a web browser of your choice.**
   - **In the browser's address bar, enter the file path to client/index.html located within your project's directory. This action will render the application's user interface, making it accessible for use.**