API AND VUE FRONTEND ARCHITECTURE AND SCORING LOGIC

This document outlines the architecture and scoring logic of the Accessibility API and the Vue frontend, which work together to analyze and score the accessibility compliance of HTML files.

---

1. API Architecture (Backend - Laravel)

The core of the accessibility checking is handled by the API, which provides an endpoint to accept an HTML file, analyze its accessibility issues, and return a score and list of issues. Here is a breakdown of the API's architecture:

- **File Upload Handling**: The analyze method processes an uploaded HTML file through a form input in the frontend. It validates the file's MIME type (html or htm) and size (maximum of 2MB).
- **HTML Parsing**: The file is read and loaded into a DOMDocument object, which parses the HTML structure. The Crawler component from Symfony is then used to query and extract specific elements from the HTML.
- **Accessibility Checks**: The controller performs a series of checks for common accessibility issues:
    - **Missing alt attributes in images**.
    - **Skipped heading levels** (e.g., going from <h1> to <h3>).
    - **Missing lang attribute in the <html> tag**.
    - **Missing ARIA roles on landmarks (e.g., header, footer, nav)**.
    - **Missing ARIA attributes like aria-labelledby or aria-describedby**.
    - **Low text-background contrast**.
    - **Missing labels for form elements**.
    - **Empty links or buttons**.
- **XPath Calculation**: To allow the frontend to highlight issues in the HTML, the API calculates the XPath of problematic elements.
- **Score Calculation**: The API calculates a **compliance score**, which is inversely proportional to the number of issues found. Each issue deducts 5 points from a baseline of 100. The formula is:

compliance_score = 100 - (number_of_issues * 5)

- **Response**: The API returns the compliance score and a list of identified issues. Each issue includes:
    - **Type** (e.g., Missing Alt Attribute, Low Contrast)
    - **Element** (the HTML element with the issue)

    - **Suggestion** (a recommendation to resolve the issue)
    - **XPath** (the location of the element in the HTML structure)

---

2. Frontend Architecture (Vue.js)

The Vue.js frontend interacts with the Laravel backend to upload HTML files, display the accessibility analysis results, and allow users to explore issues interactively.

Components:

- **File Upload Section**:
    - The user uploads an HTML file through an input field (<input type="file">).
    - The file is validated, and upon submission, it is sent to the backend via the uploadFile method.
    - The loading spinner is displayed while the analysis is running.
- **Displaying Results**: Once the backend returns the analysis results, the Vue component displays:
    - The **compliance score** at the top of the results.
    - A list of **accessibility issues** found, including a description of the issue and a suggested fix.
    - **Highlight** buttons are provided for each issue. When clicked, the corresponding element's XPath is used to highlight the problematic section in the modal.
    - If no issues are found, a success message is displayed.
- **Error Handling**: If the API encounters an error (e.g., invalid file or internal server error), an error message is shown to the user.
- **Modal**:
    - When a user clicks the "Highlight" button next to an issue, a modal opens with detailed information about the problem, including:
        - The **issue type** (e.g., Missing Alt Attribute).
        - The **element** HTML code that caused the issue.
        - The **suggestion** for fixing the problem.

---

3. Scoring Logic

The scoring logic is designed to encourage users to create accessible web content by penalizing the presence of accessibility issues.

- **Compliance Score Calculation**:
  - The score starts at **100** and deducts **5 points per issue** found. This deduction continues as more issues are detected.
  - If no issues are found, the score remains at 100%.
  - Example:
    - **0 issues** → Score = 100%
    - **1 issue** → Score = 95%
    - **5 issues** → Score = 75%
- **Types of Issues**: The scoring is based on the following types of issues:
  1. **Missing Alt Attribute**: Important for image accessibility.
  2. **Skipped Heading Levels**: Affects screen reader navigation.
  3. **Missing Language Attribute**: Important for setting the document's language.
  4. **Missing ARIA Roles/Attributes**: ARIA improves accessibility for users with disabilities.
  5. **Low Contrast**: Poor contrast affects readability, especially for visually impaired users.
  6. **Missing Labels**: Forms need clear labeling for accessibility.
  7. **Empty Links/Buttons**: Ensures links and buttons are usable and meaningful.
- **Penalty for Issues**: Every identified issue, regardless of severity, reduces the score by **5 points**. While this approach is simple, it encourages users to prioritize fixing issues to maintain high accessibility standards.

---

4. Communication Between Frontend and Backend

1. **File Upload**:
   - **Frontend**: The file is uploaded using a form submission. The file is captured with the @change="onFileChange" event and processed when the form is submitted with @submit.prevent="uploadFile".
   - **Backend**: The file is validated (size, type), and its content is parsed to identify accessibility issues.
2. **Response Handling**:
   - **Frontend**: Upon successful analysis, the frontend receives a JSON response containing the compliance_score and issues array.

    - **Backend**: The backend returns a JSON response containing the score and a detailed list of issues.
3. **Displaying Issues**:
    - **Frontend**: Each issue is displayed in a list, and users can click "Highlight" to open a modal with additional details.
    - **Backend**: For each issue, an XPath is returned to allow the frontend to highlight the element.

5. **Conclusion**

This API and Vue.js frontend provide a robust solution for analyzing and improving web accessibility. The architecture is designed to:

- Allow easy file uploads and analysis.
- Display accessibility issues in a user-friendly format.
- Provide actionable feedback with a compliance score to help users improve their websites' accessibility.

The compliance score and detailed suggestions help guide users in making improvements and creating more accessible content.