Task 2 Reflection

1. Overview

The analysis of the pilot data (N=4) in week 9 reveals a significant difference between the user experience of HTMX (enable JavaScript) and No-JS (diabled JavaScript). Although the server-side performance indicators show high stability in both modes (server response time median < 3.0 ms, absolute median MAD < 1.0 ms), the user-side indicators expose serious security and functional defects. Based on quantitative data analysis, this redesign adopts the "inclusiveness-severity priority matrix" to centrally solve two key problems that affect users disproportionately in a restricted operating environment. The goal is to ensure that the "Dual-Path Architecture" meets the WCAG 2.2 Level AA standard and achieve real progressive enhancement.

2. Priority Determination Criteria

This redesign scores the backlog problem according to two dimensions to determine the repair order:

- **Severity (Severity, 1–3)**: 3 points represents key function blocking or safety hazards.
- **Inclusion Risk (Inclusion Risk, 1–3)**: 3 points represents systemic obstacles (such as completely unavailable).
- **Priority score**: severity + inclusive risk (up to 6 points).

The analysis identified two key issues with a score of 6:

Priority 1: Security of No-JS deletion operation (Score: 6)

- **Evidence reference**: The confidence score of pilot participant P2 (No-JS group) in task 4 (deleted) is only 2/5, which is a gap of -3.0 compared with the HTMX group. The server log confirms that the medin processing time of the operation is 1.0 ms (MAD 0.0), indicating that the system has completely skipped the confirmation step.
- **Influence Analysis**: This behavior violates WCAG 3.3.4 (error prevention), which requires a confirmation mechanism for irreversible data deletion operations.
- **Decision-making**: Must fix.

Priority 2: Availability of No-JS editing function (Score: 6)

- **Evidence reference**: Task 3 (editing) presents 100% functional failure rate under No-JS conditions. Because the cancel button is completely dependent on the client script (`hx-get`), the user is trapped in the editing state and cannot exit.
- **Influence Analysis**: This defect violates the principle of WCAG 2.1.1 (keyboard accessibility) and robustness, resulting in the control being inoperable in a scriptless environment.
- **Decision-making**: Must fix.

3. Implementation and Repair Strategy

Repair I: Interceptor Route

Statement of the problem:

In the 9th week version, the deletion function is realized as a direct POST form submission. In No-JS mode, this implementation skips the JavaScript-based hx-confirm dialog box, causing the data to be deleted immediately. Qualitative feedback (P2: "Wait, did I delete it? It didn't ask me...") confirmed the user anxiety caused by the design.

Solution:

Interceptor Pattern has been implemented. Restructure the "Delete" button into a progressive enhancement link:

1. **HTMX path**: Keep the `hx-delete` attribute to block clicks and display the browser confirmation box.
2. **No-JS path**: The `href` attribute points to a dedicated confirmation page routing (`GET /tasks/{id}/delete/confirm`) and renders the server-side confirmation template.

Code Comparison Evidence:

Before Fix (Insecure Direct Commit):

```
<form action="/tasks/{{ task.id }}/delete" method="post">
    <button>Delete</button>
</form>
```

After Fix (Interceptor Routing Implementation):

```
<a href="/tasks/{{ task.id }}/delete/confirm"
   hx-delete="/tasks/{{ task.id }}"
   hx-confirm="Are you sure?">
   Delete
</a>
```

Results Verification:

Regression testing confirms that clicking “Delete” in a JavaScript-disabled environment now correctly redirects to the confirmation page. The system enforces explicit interactive confirmation from users before executing database deletions, thereby meeting WCAG 3.3.4 compliance requirements.

Fix 2: Native Form Override Strategy

Statement of the problem:

The data of task 3 (edit) shows that No-JS participants have encountered 100% functional blocking. Since the "Cancel" button is implemented as a `<button>` element with only the `hx-get` attribute, it can neither submit forms nor perform navigation in a script-free environment. This causes keyboard users and No-JS users to be trapped in editing mode, violating the operability principle of WCAG 2.1.1.

Solution:

The **`formaction`** attribute strategy of HTML5 is adopted. Although changing the button to the `<a>` tag is a common repair method, it destroys the visual consistency with the "Save" button. The final scheme retains the `<button>` tag, but uses the `formaction` and `formmethod="get"` attributes to overwrite the default form submission behavior, so that it acts as a navigation link in No-JS mode.

Before Fix (Semantically Incomplete Pseudo-Button):

```
<button hx-get="/tasks/{{ task.id }}/view" class="outline">
    Cancel
</button>
```

After Fix (Dual-Path Robust Implementation):

```
    <button type="submit"
            formaction="/tasks"
            formmethod="get"
            hx-get="/tasks/{{ task.id }}/view"
            class="outline">
        Cancel
    </button>
```

Results Verification: Manual regression testing confirms that after disabling JavaScript, clicking the “Cancel” button now correctly initiates a GET request to `/tasks`, successfully exiting the user from edit mode. Although this approach appends redundant query parameters to the URL (e.g., `?title=...`), the server ignores these parameters, achieving progressive enhancement without compromising functionality.

4. Validation and Regression Test Results

- In order to ensure that no new defects are introduced in the redesign, a comprehensive regression test was performed (see `regression-checklist.pdf` for details).
    - **Functional equivalence verification**: The pilot task has been re-simulated. In No-JS mode, task 4 (deletion) no longer occurs instantaneously, but correctly jumps to the confirmation page. The cancellation function of task 3 (editing) has returned to normal.
    - **HTMX regression check**: Verify that the experience of the standard user has not deteriorated. AJAX features such as Inline Edit, Modal Confirm and Dynamic Search still run smoothly in a JavaScript-enabled environment.
    - **Compliance Audit**: After Axe tool scanning and manual code review, it is confirmed that the repaired components meet the WCAG 2.2 Level AA standard. In particular, Status Messages and Focus Management perform normally under both paths.

5. Deep reflection and careful consideration

Inclusive advantages of server-first architecture

This project confirms the natural advantages of the Server-First architecture in terms of accessibility. Since the core logic (such as verification, routing, state management) is processed on the server side, the front-end is only responsible for hypermedia rendering, which makes generating semantic HTML a default behavior rather than additional work. Compared with single-page applications (SPA) that require manual management of focus and screen reader announcements, server-side rendering (SSR) with progressive enhancement technology achieves higher compliance with lower code complexity.

Reflection on the rigor of statistical indicators

Choosing **Median** and **MAD (Absolute Median Difference)** instead of the mean in the evaluation stage is the key to reaching the correct conclusion. The data shows that the MAD value of the HTMX group is only **1.0 ms**. This low dispersion indicator excludes the assumption of unstable back-end performance and accurately locks the root cause of the problem on the front-end interaction mode. If you only rely on the mean value, the "fast" performance (1.0 ms) of the No-JS group on deleting tasks is very easy to be misread as a performance advantage, thus covering up the security risks behind it. This shows that in human-computer interaction evaluation, quantitative data must be comprehensively interpreted in combination with qualitative contexts (such as confidence scores).

Design trade-off

In order to support the No-JS path, we have accepted specific trade-offs. For example, using `formaction` to implement the cancellation function will cause the URL to become "dirty" (carrying unused form data). However, according to the **Robustness**, this slight aesthetic sacrifice is exchanged for the usability of the system in extreme environments, which is a decision-making in line with inclusive design ethics.

6. Conclusion

Through the evidence-based iterative process - from the abnormal detection of pilot data (low confidence score and fast operation time), to the root cause analysis based on Median/MAD, to targeted code reconstruction - the project has successfully built a task management system with high fault tolerance. The final deliverable not only fixes the serious violations of WCAG 3.3.4 and 2.1.1, but more importantly, it proves that accessibility is not a "functional feature", but a constraint that must run through the architecture design. The system now has complete service capabilities in diverse environments such as script-free, pure keyboard or screen reader.