This project explores the design, implementation, and evaluation of a novel interaction technique called the hover button, aimed at improving the user experience when multitasking between windows. The technique was motivated by personal frustrations experienced during coding work, where frequent switching between a code editor and external resources (like requirements documents or documentation pages) disrupts workflow, introduces cognitive friction, and wastes time. The hover button is designed to reduce these interruptions by enabling a temporary overlay of a target window through a simple press-and-hold action, eliminating the need to switch away and back.

The hover button technique introduces a press-and-release interaction model:

- A short press designates the target window to hover.
- A long press or hold overlays the target window semi-transparently over the active workspace.
- Releasing the button dismisses the overlay and returns the user to the active window.

This approach reduces the cognitive load of navigating between windows by providing momentary access to needed information without context-switching. Inspired by the frustrations of frequent Alt+Tab or tab bar switching, the design focuses on minimizing hand movement and mental disruption. The semi-transparent design further allows users to cross-reference information while keeping an eye on their active work, something multiple interview participants praised as enhancing their control and awareness.

The system was implemented as a browser-based web application using HTML, JavaScript, and CSS. Two mock windows were created:

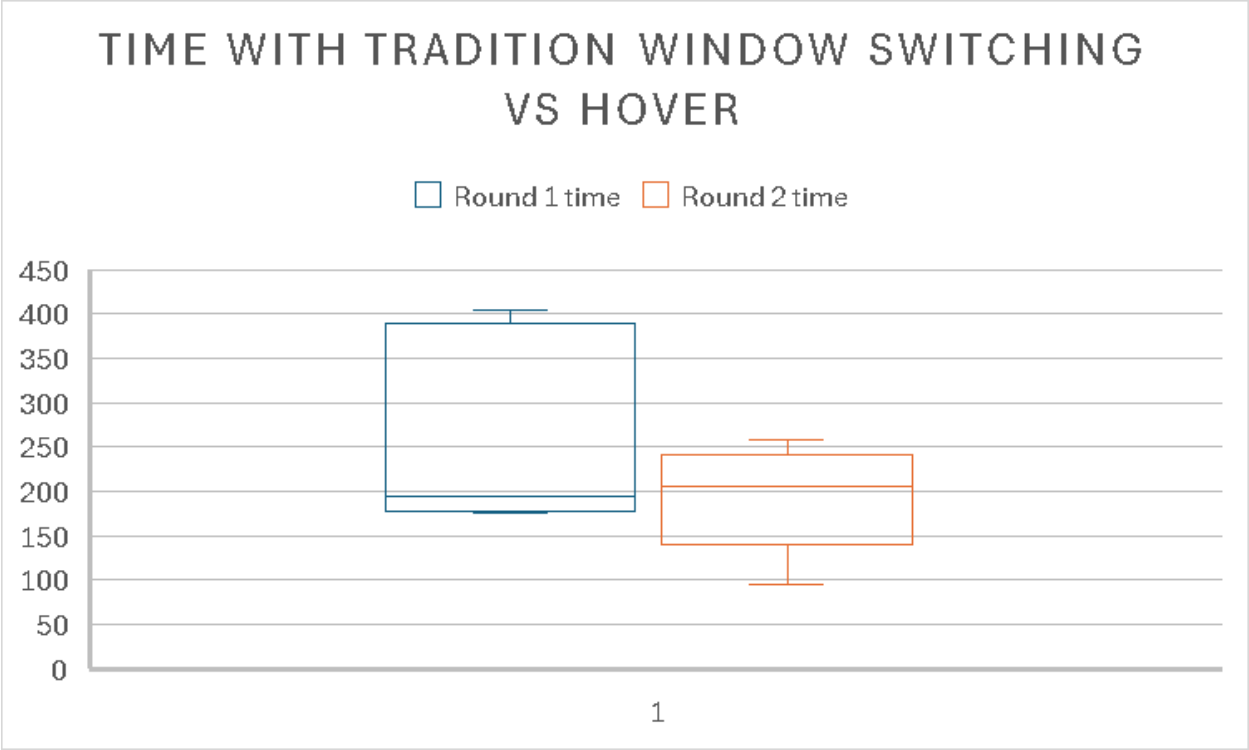
- A Text window containing a paragraph to copy.
- A Typing window where participants reproduced the text.

In Round 1 (traditional switching), users switched between windows using a simulated tab bar. In Round 2 (hover mode), holding the Ctrl key triggered the overlay of the Text window on top of the Typing window. The system recorded:

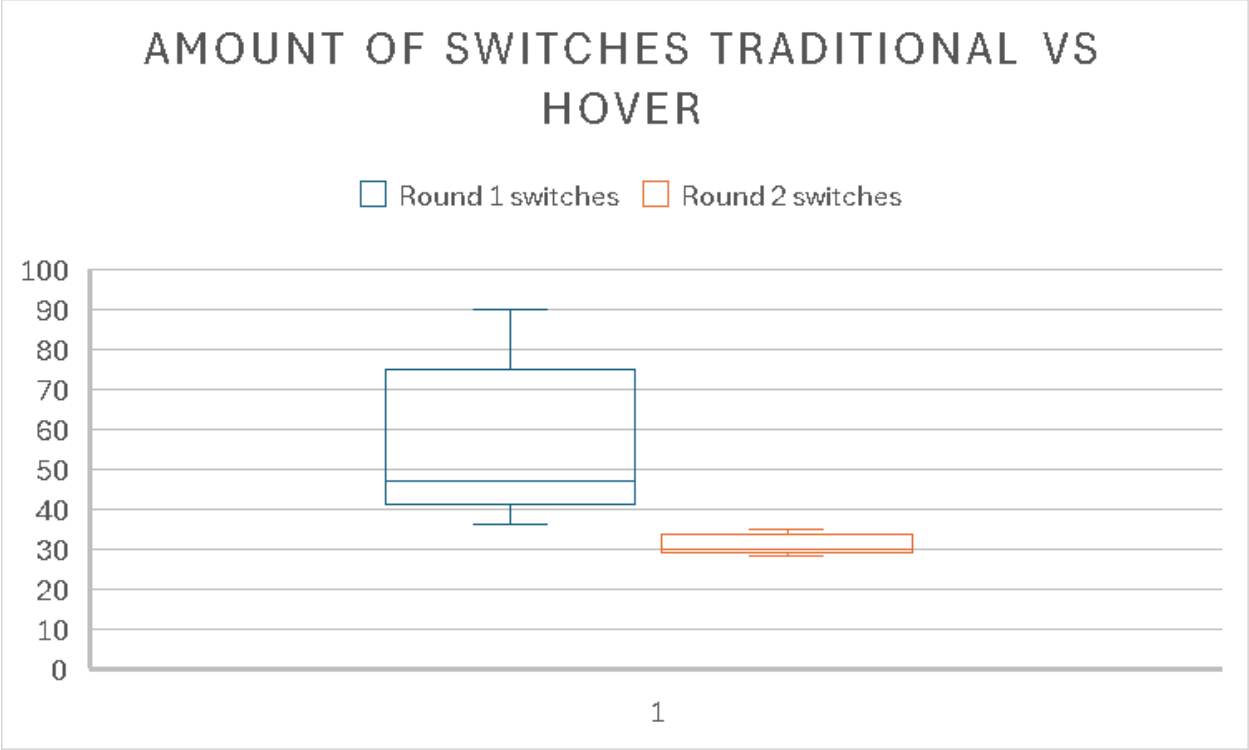
- Time taken to complete the task.
- Number of window switches.
- Number of typing errors.

Participants' consent was collected using an online form, and all results were anonymized and exported as CSV files.

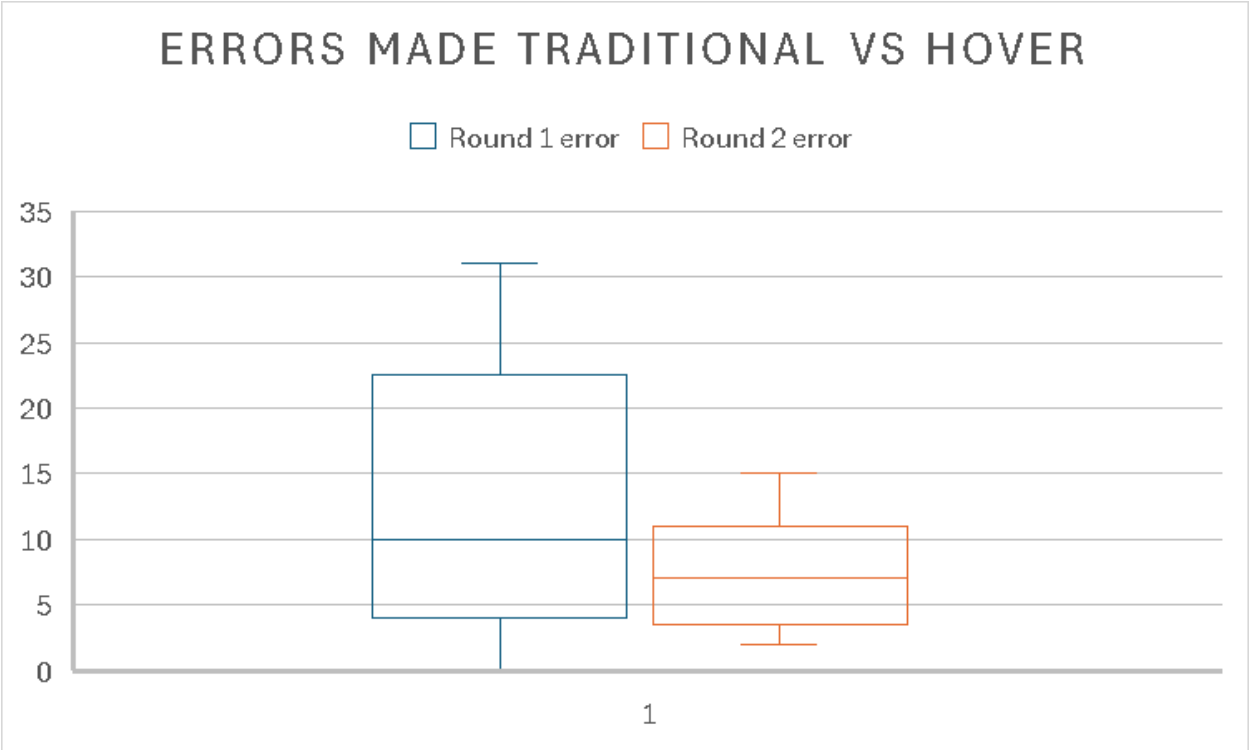
Box plots were generated to compare performance across conditions.



Median completion times were similar between the two conditions. However, the traditional switching condition showed a much larger spread and produced several extreme outliers, suggesting that while some users adapted well, others became significantly slower when forced to switch windows repeatedly.



Hover mode showed a dramatic reduction and stabilization of switch counts. Notably, the upper limit of switches was reduced, and the switch count distribution became very narrow and consistent across users, indicating that the hover technique effectively minimized unnecessary toggling.



Error rates also decreased under the hover condition. While traditional switching produced a wide spread of errors, hover mode reduced both the median and the fluctuation in errors, suggesting that reducing switching demands helps users maintain focus and accuracy.

Participants consistently praised the press-release hover mechanism, describing it as feeling like they were “doing one less thing” compared to tab switching. The semi-transparent overlay was highlighted as a useful feature for cross-examining their work without fully losing sight of the typing window. Suggestions for future improvements included:

- Adjustable transparency.
- Customizable hover keys.
- Visual indicators for hover activation.
- A lockable overlay toggle.

The hover button interaction technique shows promise as a tool for improving window-switching efficiency, particularly in tasks that require frequent reference checking. By reducing cognitive and motor overhead, it improves consistency, reduces errors, and enhances the user experience. Future iterations could further explore customization, integration into real-world window managers, and extended user testing across diverse tasks.