WarpMind #4 Assignment Report: The Paths of Refinement Option 1: CSS and JavaScript frameworks

Adrián Fernández-Paniagua Belinchón
Grazia Cossu
DA4 5

October 3, 2025

Contents

1 Introduction

For this assignment I decided to rework my WarpMind interface by introducing one CSS framework and one JavaScript client-side framework. I chose **Bootstrap** and **jQuery** because they are well-established, easy to integrate into a small project, and flexible enough to reshape the interface without forcing a full rewrite. My goal was not only to modernise the appearance of the tool, but also to make the code easier to maintain and to improve the experience across different devices.

2 Refactoring Process

2.1 Choosing and Integrating the Frameworks

The first step was simply getting the frameworks into the project. Once I added the Bootstrap and jQuery CDNs to the `index.html`, the entire interface immediately changed character. Bootstrap brought a much more uniform style than my handwritten CSS, and I quickly realised that many of my custom rules were no longer necessary. jQuery, instead, felt like adding a layer of convenience over the functions I had already implemented. Even small changes, like switching to `$(#id)` selectors, made the code feel lighter and more coherent.

2.2 Rebuilding the Layout

What took the most time was reorganising the layout to use Bootstrap's grid system. The original interface worked, but it was rigid and definitely not friendly on smaller screens. With Bootstrap I structured the page into a simple three-column layout: the concept list on the left, the PDF viewer in the centre, and the workspace on the right.

The surprising part was how easy it became to make everything responsive. Once I replaced my fixed widths with Bootstrap's `col-` classes, the interface naturally collapsed into a vertical layout on a phone without further tweaking. Even simple components like buttons and sliders instantly looked cleaner once wrapped in Bootstrap classes.

2.3 Refactoring JavaScript with jQuery

Moving the JavaScript logic to jQuery required more thought. A lot of my code interacted with elements that were created dynamically from a template, and these elements did not respond to event listeners in the same way as static ones. I had to rethink how interactions were triggered and eventually adopted jQuery's event delegation pattern.

Although this initially felt counterintuitive, it ended up simplifying the entire script. Instead of scattering listeners everywhere, I could attach a single listener to a parent element and let jQuery handle the rest. The result was cleaner, more compact, and easier to read.

3 Reflections

3.1 Positive Aspects

What I appreciated most in this refactoring is the immediate sense of order that frameworks bring. Bootstrap removed the visual roughness of the prototype and gave me a layout that behaves well even in edge cases. jQuery, on the other hand, encouraged me to restructure parts of my script in a more thoughtful way, especially regarding dynamic elements.

Another unexpected benefit was how quickly the interface started to feel like a more mature application. Even though the underlying logic did not change dramatically, the combination of better styling and clearer interaction patterns made the tool feel more professional.

3.2 Challenges and Less Positive Experiences

The main difficulty emerged from the combination of old and new code. As soon as Bootstrap's styles became active, several of my previous CSS rules started clashing with it, creating some awkward layouts. It took patience to gradually remove or adapt the conflicting parts without breaking the entire interface.

The switch to jQuery also required learning some habits from scratch. Event delegation in particular took a while to understand, because the behaviour is less explicit than directly attaching event listeners. Debugging dynamic behaviour in the workspace section was the most time-consuming part of the refactor.

4 User Interface Before and After the Refactoring

To better illustrate how Bootstrap and jQuery changed the interface, I compared the previous layout with the refactored version. The screenshots below show both the original state of the tool and the redesigned version.

4.1 Original Interface

The initial interface relied mainly on custom CSS with a fixed layout, which made it less adaptable and visually inconsistent. Figure 1 shows a typical view of the older version.

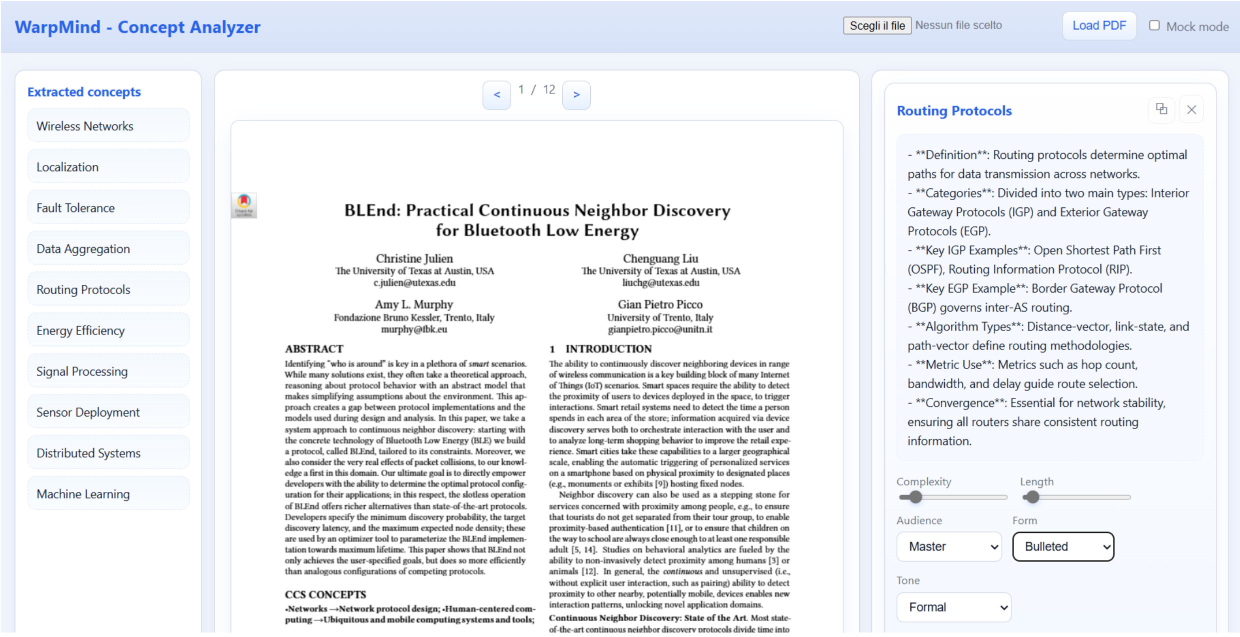

Figure 1: Original WarpMind interface before refactoring, based entirely on handwritten CSS.

4.2 New Layout with Bootstrap

Figure 2 displays the updated interface after integrating Bootstrap. The three-column layout is now handled by the grid system, making the interface more balanced and immediately more readable.

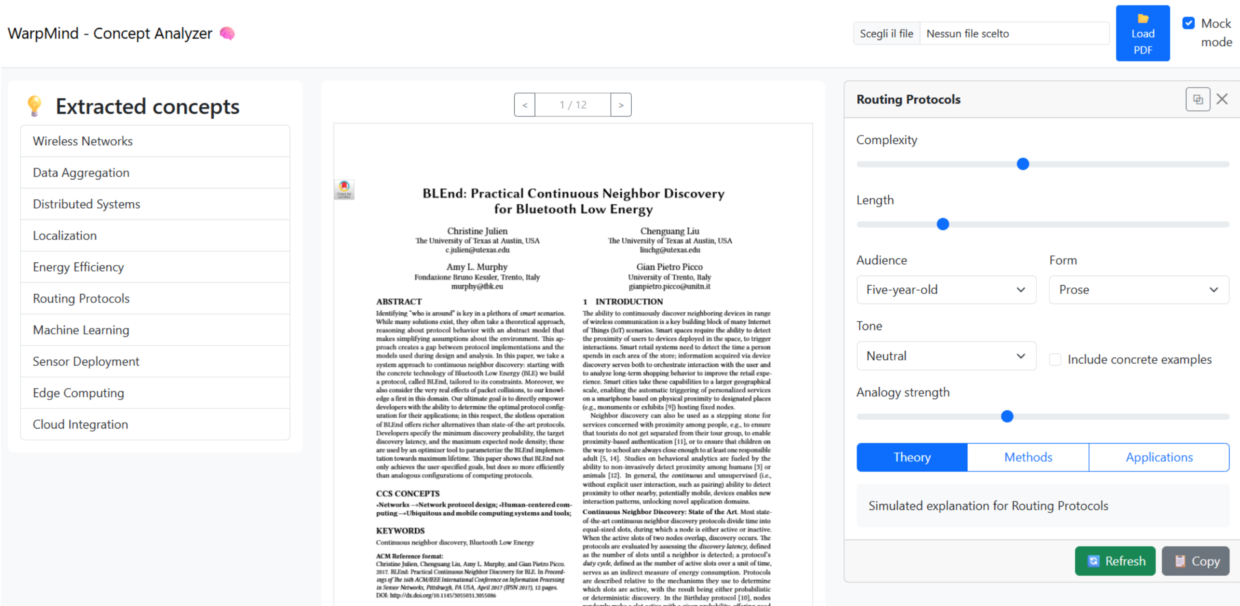

Figure 2: Refactored WarpMind interface using Bootstrap's grid system and improved spacing.

4.3 Concept Extraction and Card Rendering

Bootstrap list-groups and cards allowed for a cleaner and more structured presentation of dynamically generated content. As shown in Figure 3, the extracted concepts appear neatly in the sidebar, and clicking on them produces an interactive card.

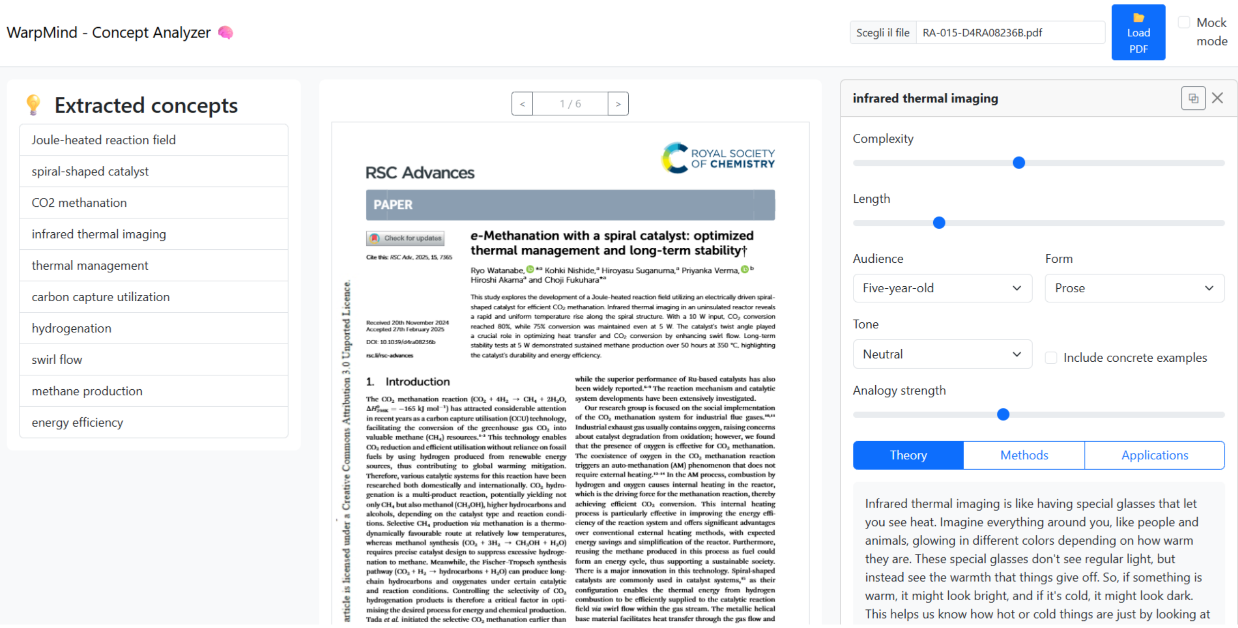

Figure 3: Extracted concepts (left) and example of a dynamically generated concept card (right).

4.4 Responsive Behaviour

Bootstrap's responsiveness significantly improved usability on smaller screens. Figures 4 and 5 show how the interface reorganises itself without requiring additional CSS.

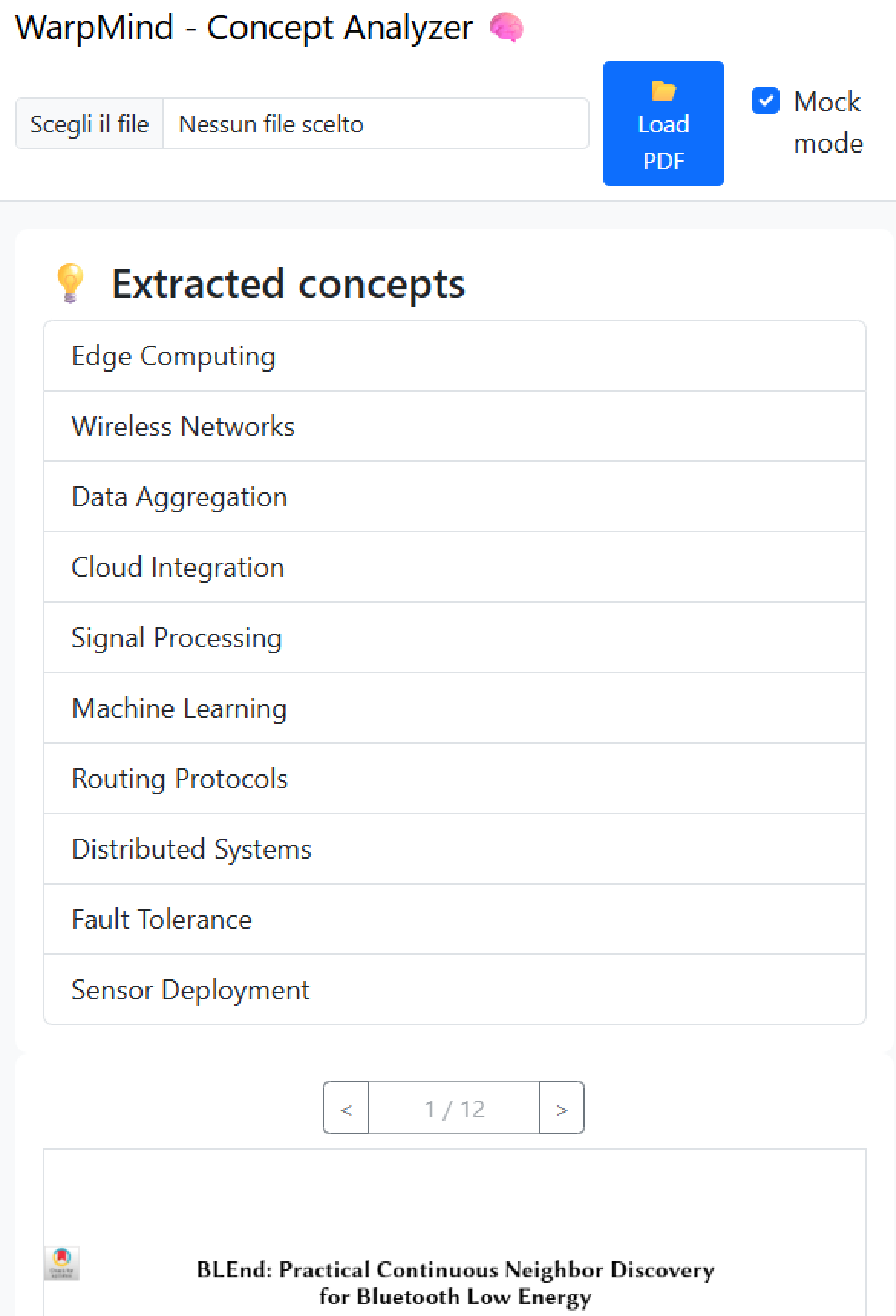

Figure 4: Medium-sized view: components reorganise while maintaining clarity.

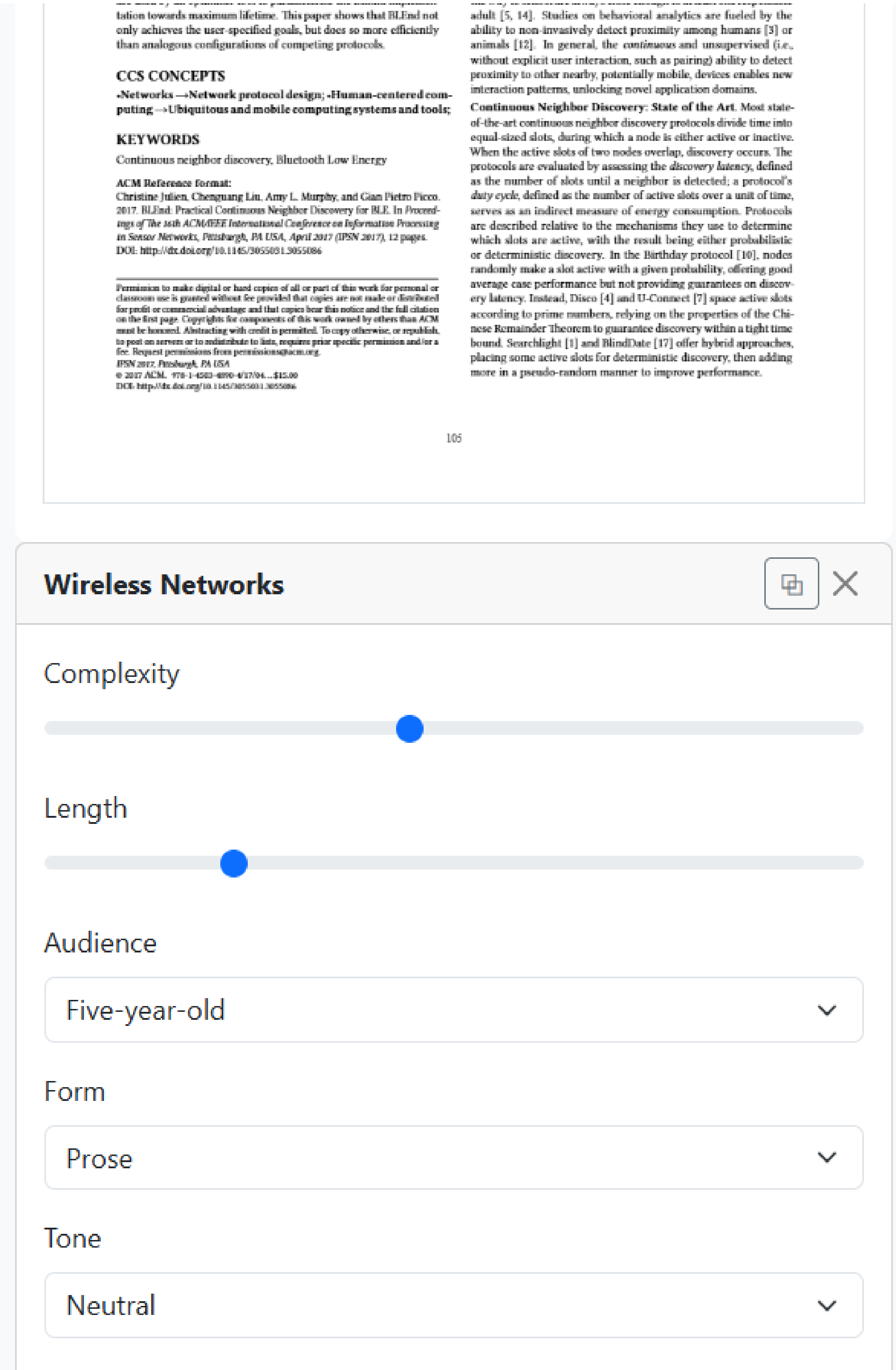

Figure 5: Mobile-friendly layout where columns stack vertically and buttons adapt.

5 Conclusion

Overall, integrating Bootstrap and jQuery proved to be a very positive experience and gave me a clearer sense of how frontend frameworks can support, rather than complicate, a project. The WarpMind interface is now more polished, more flexible, and easier to extend.

While there were moments where it felt like I was wrestling with the frameworks rather than working with them, the final outcome confirmed that the refactoring effort was worthwhile. The application now behaves more consistently and is better prepared for future improvements, including the possibility of evolving into a full Electron-based desktop tool.