# Jared Rasmussen – AB Testing Component Review

My task was to develop a component that would allow for AB Testing between two images. The images were to be presented randomly to the user, and Adobe Analytics integration was to be implemented to track which image generated more click-throughs to the target URL associated with both images.

My initial thought process was to modify existing components within WeRetail and reconfigure them to accomplish my goal. I browsed through the source code for both the Image and Hero Image components as an image was the main piece of my component. I copied the Image component and attempted to pare it down to make it simpler to work with, and then attempted to add in the ability to include a second image, as well as a button that would link to an author-defined URL (which is where the Hero Image component was used as a reference). The primary issue I had with this attempt was that while I succeeded in configuring the dialog box to allow for multiple images to be selected, and they would appear in crx/de – the component itself would only ever display the first image selected. This issue would persist even if I attempted to hardcode the second image to be displayed. Looking back, I probably spent more time than I should have attempting to rehash this method in hopes of jury-rigging it to do what I expected it to do (~24 hours).

I spent a fair amount of time throughout the assignment trying to familiarize myself more with Sightly as well as studying the source code provided with WeRetail to understand how all of the pieces fit together and communicate with each other (~6 hours)

I looked briefly into implementing a multifield component as I had come across the concept while browsing the internet looking for solutions to the issues I was experiencing noted above. However, upon closer inspection and attempts at integration, multifield components seemed more complicated/convoluted a solution than what I needed given the requirements I was given (~4 hours).

Ultimately, I ended up going the reverse route from where I started. Rather than attempt to reconfigure an existing component, I built one from scratch, slowly adding and testing functionality as I went to ensure that the frontend and backend were properly linked and communicating as I expected them to. I started by sending a simple hardcoded String value from the Java Class to the AEM frontend via Sightly. From there, I had my Java Class pull a property value defined by an AEM dialog box, which AEM would then display in the component. The next step was to create the code to randomly generate which fileReference property would be sent from the Java code to be displayed in the AEM component (again as a String value). Finally, I moved the Sightly call into the <img> tag as the source value and added in the URL redirection functionality that I was able to source from my initial attempts (~6 hours).

Initially I deleted erroneous code as I discovered issues, but towards the end of the exercise, I decided to just comment it out instead so I could track my attempts better.