TO: *Dr. Green*

FROM: *Elijah Rose (elirose)*

DATE: *7.8.2019*

SUBJECT: *Project-Logger Web Application Review*

SUMMARY

The project-logger web application seeks to prove basic working knowledge of both HTML form inputs and posting as well as PHP server-side manipulation. Currently, though perhaps not the prettiest of web applications, the attached web project meets these criteria under the tab HW5. Due to the limitation of using no table in the design, as well as being locked into exact variable names, a very simple and rudimentary design was sought, as seen in *Figure 1*. Titles were put on top of the input fields to encourage more thoughtful, slow form-answering[1], and the related percentage fields grouped together with a fieldset; care was also taken to change the margins of the headers to be closer to their respective input fields to better serve as labels.

The only unique implementation, perhaps, is the progress bar: a dynamic div element which changes according to the values of the percentage. Javascript, of course, is used to implement this functionality, relating each value with a color. The difficult part, however, is keeping the values in-line – when one value changes, say the percent of Coding in the project, naturally all of the others should take up less of a percentage, if they are all to equal 100% combined. As such, though other methods were considered (and some tested), the solution devised was such that the four non-changed values maintain their ratios and are changed based on the difference of the new and old value of the changed input. Additionally, as nice round numbers are desired in the input field yet rounded numbers would cause gross errors in repeated calculations, a static variable *progressBar.values* is used to store the floating-point number that the input fields approximate.

The form view page is similarly unremarkable, as well as the all-view page. Each data entry is separated into their own div tags (due to the ambiguity of the rubric regarding if tables were disallowed everywhere or just on form input). When the user prints or "Save as..", care was taken to remove excess elements such as the URL and time using CSS media queries, and each entry is given its own page for clarity and neatness. Additionally, users can search on the project-log-all page, either appending "?BlazerID=" followed by the BlazerID, or entering it in the search bar and clicking "Search"; this also produces a summary element as the last block which gives some aggregated data on the user's projects as a whole. The data is stored in a local CSV file, which is in a directory protect by a .htaccess file with the command "Deny from all", only permitting local access to the file.

The main negatives of the web app right now are its design (it looks excessively plain); printing typically results in one extra blank page; and the composition bar on the form does not have a "number lock" button for each input yet, making inserting exact values a bit more of a chore than it should be. These are features that would not be too difficult to fix and may be implemented in future builds.

Figures and Tables

Figure 1. Form Page, enabling users to input project data as well as see the approximate ratios in the colored ratio-bar to the right.
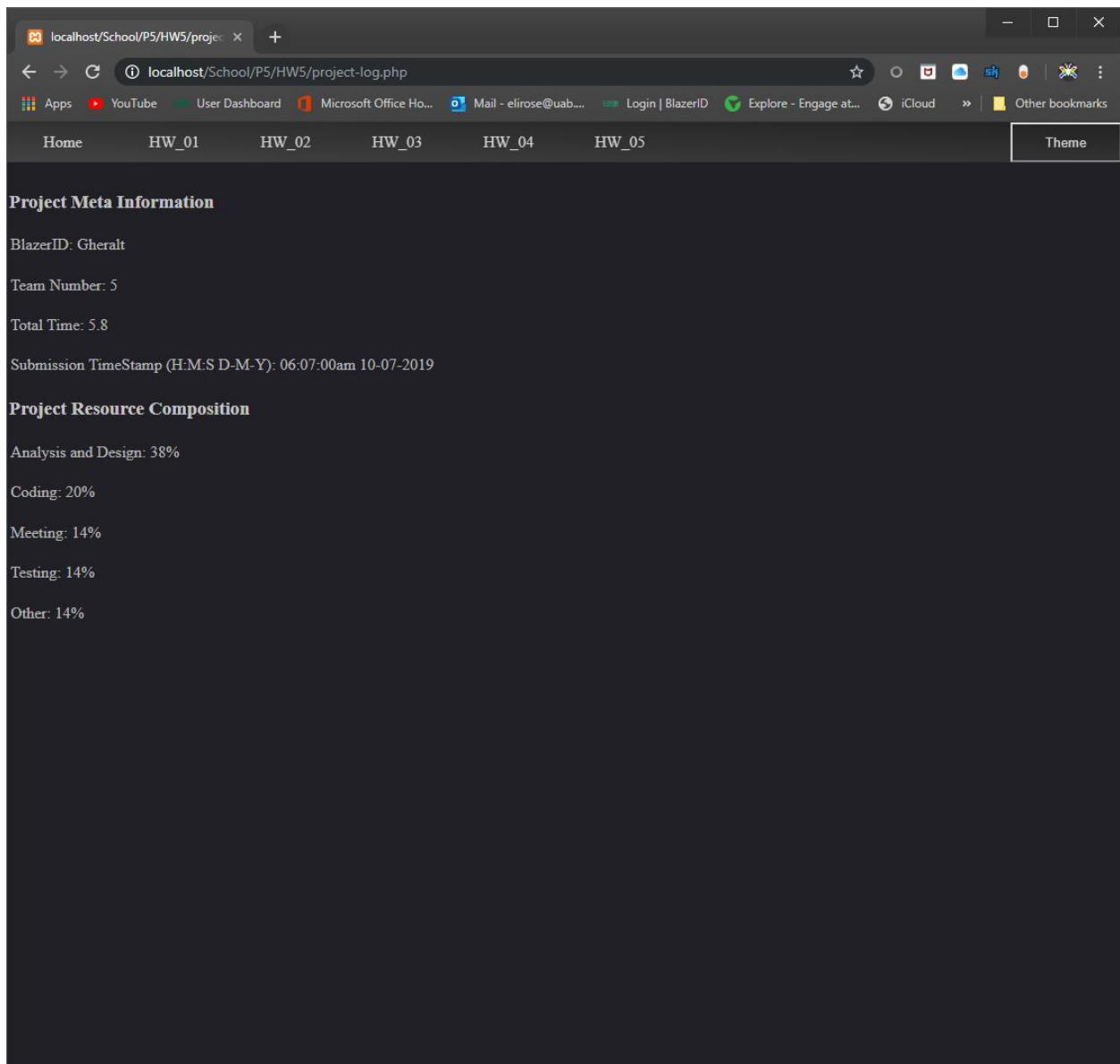
Figure 2. Simple Receipt page, allowing the user to save or print their data and ensure that it was transmitted correctly.
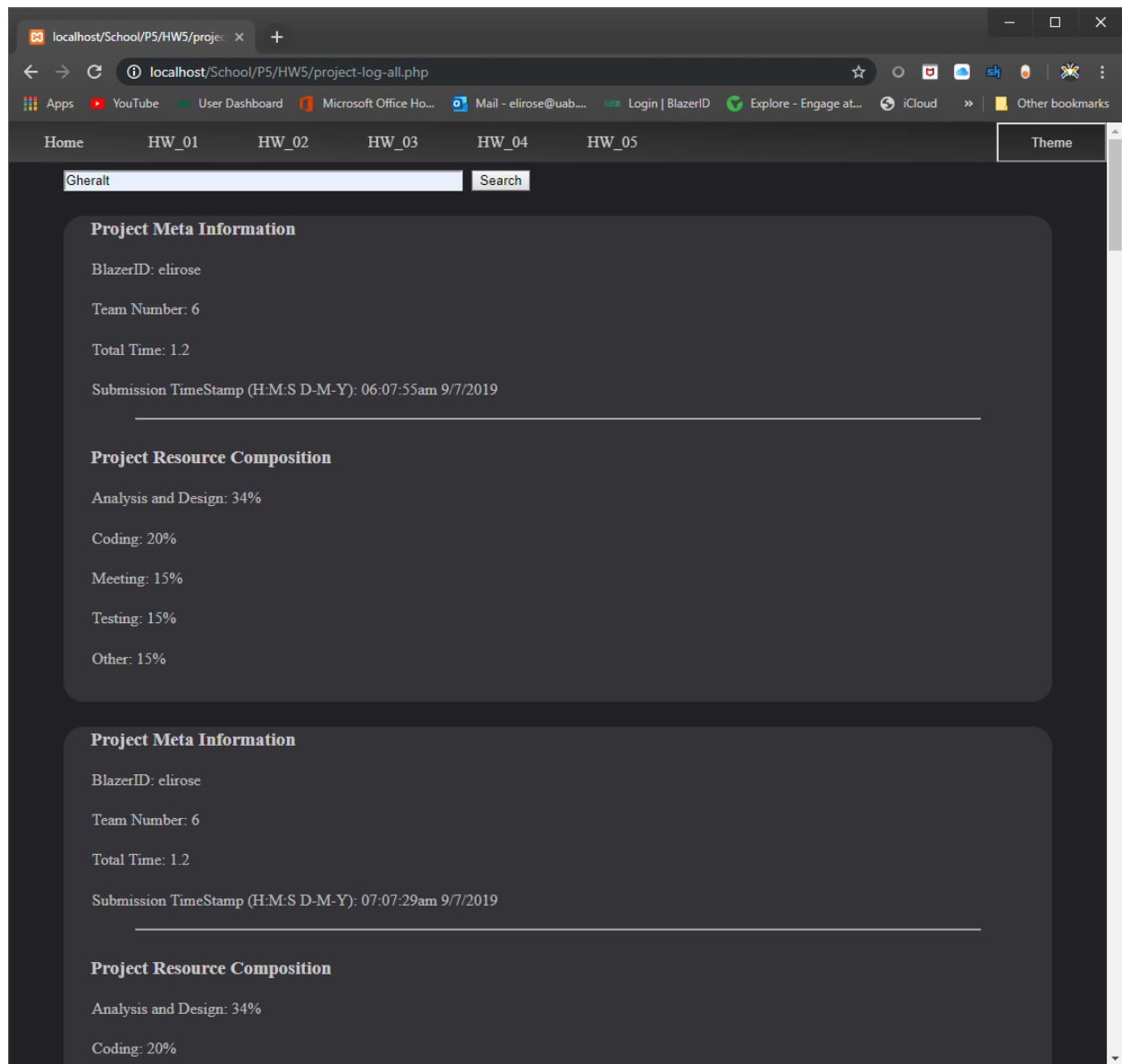
Figure 3. Using the search bar to type a BlazerID for searching; see Figure 4 for the result.
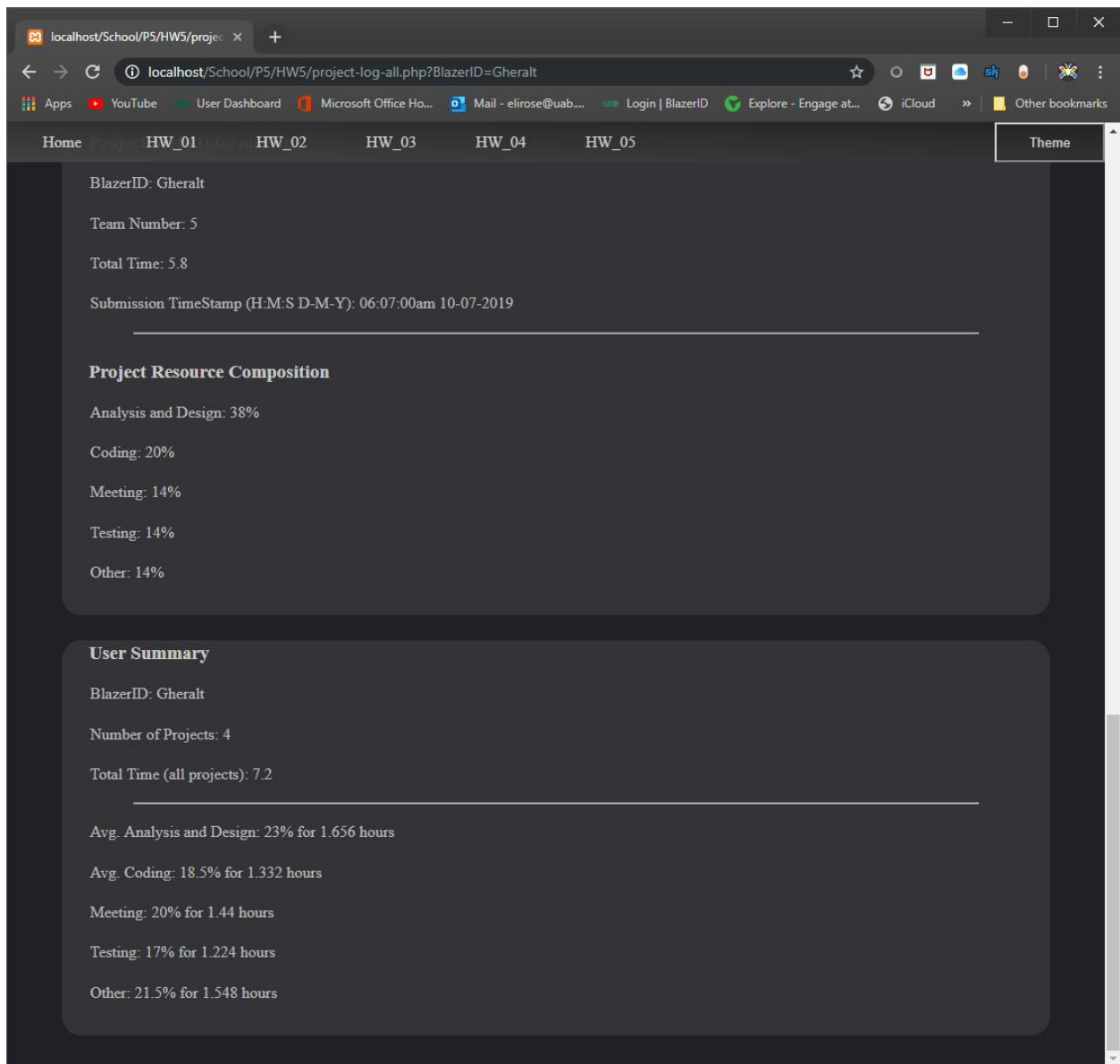
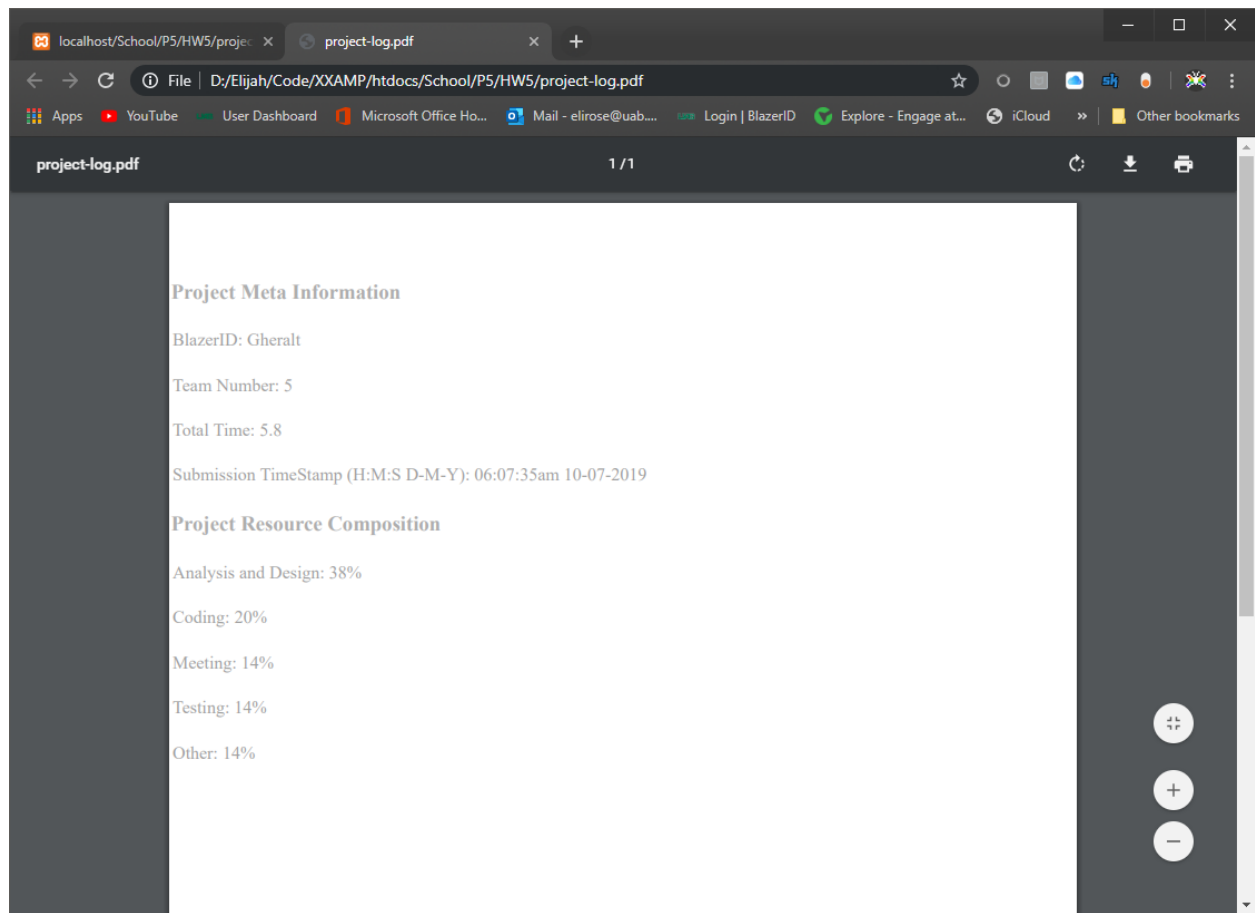Figure 4. Result of clicking the "Search" button from Figure 3. Note the last block is a summary block.

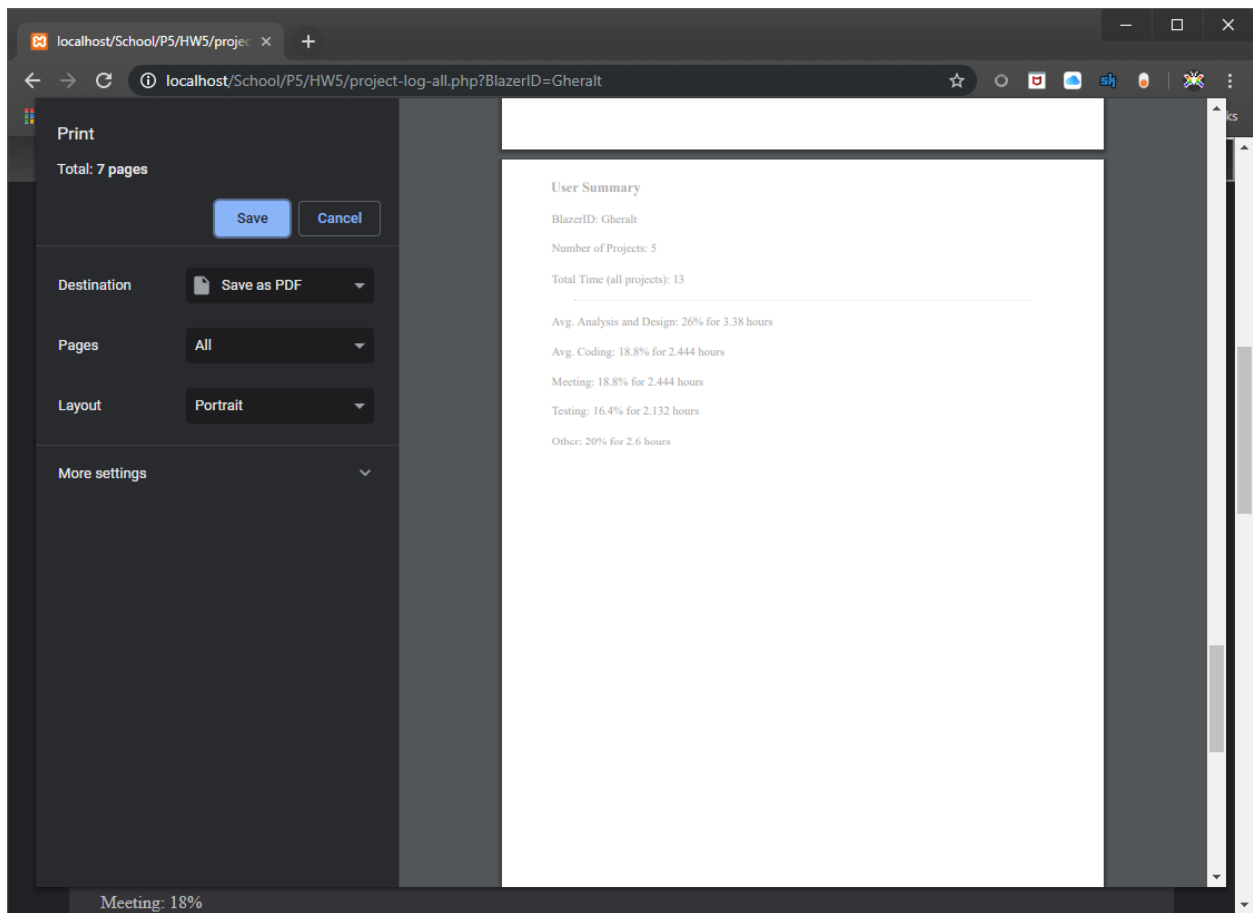Figure 5. PDF Result of Receipt print.

Figure 6. Gheralt's all-log print, note that each data section has its own page.
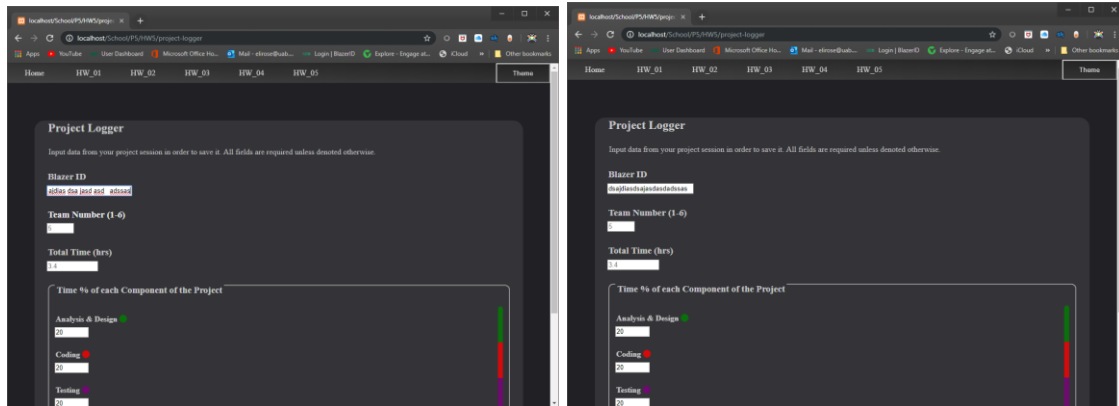
Figure 7. Client-side Validation Type 1, pre-submission editing of text fields (removing whitespace, making sure no number is above min or max).
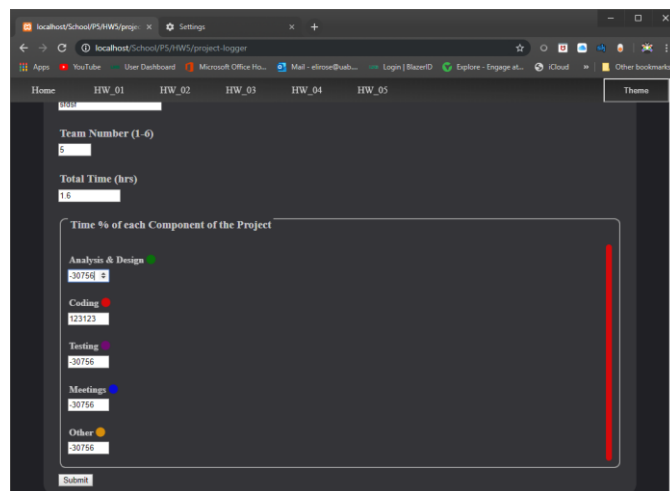

Figure 8. Client-side validation method Type 2 (Type 1 method turned off for this demo), double-checking the values before the data is submitted; *should* be impossible to trigger, but just in case. This is Chrome's HTML form features kicking in, though additionally a post-validation function is run even if that should fail using those same min and max attributes.
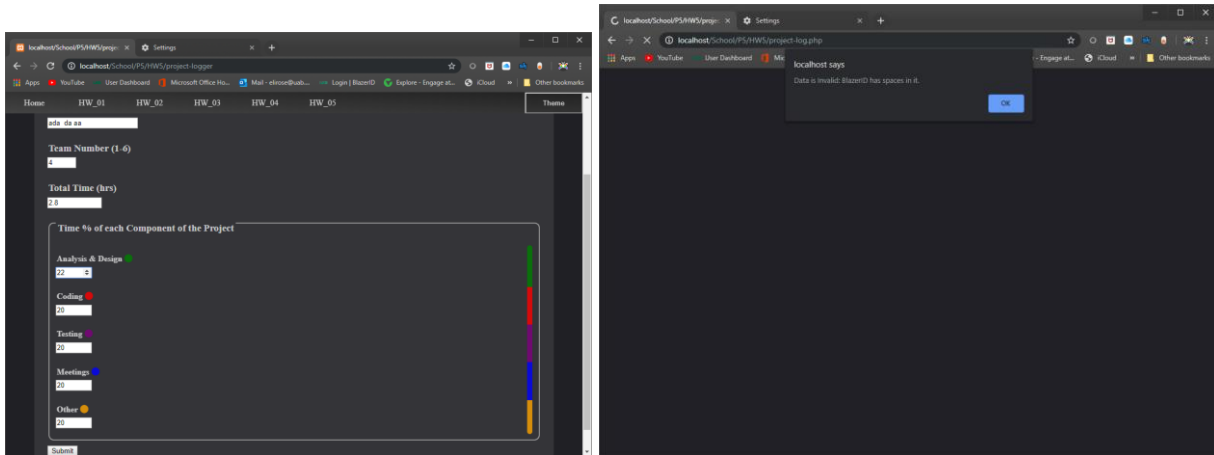
Department of Electrical and Computer Engineering
The University of Alabama at Birmingham

Figure 9. Server-side validation. After the data is submitted, if it is incompatible, the user is told what exactly is wrong via alert before being ~~redirected back to the form page~~. This is not ideal as they lose their form data, hence additional client-side validation.
Returning to form page via Header was removed so users can see the error message on the page, and if Javascript is turned off (as may be the most typical users who have to run into server-side validation errors), the redirect is near immediate. Future iterations may send the error message back to the form page to be displayed to circumvent this shortcoming, alongside perhaps the valid data to partially refill the form.

REFERENCES

1. K. L. Segwoth, C. Hang, and J. T. Barton, "Porosity malformation in modified and unmodified Fe-Si alloy castings," *AFSP Transactions*, vol. 102, pp. 276-361, July, 1974.
2. *Nick Babich. "Designing more Efficient Forms Structure Input Labels and Actions".* https://uxplanet.org/designing-more-efficient-forms-structure-inputs-labels-and-actions-e3a47007114f?gi=1f8f9edb4a80

3. (2006, May, 22) *IEEE Transactions LaTeX and Microsoft Word Style Files* .http://www.ieee.org/portal/pages/pubs/transactions/stylesheets.html