CODE INSPECTION – RESULT AND SUMMARY

While reading the code, it is evident what the intention of every file and function's purpose happens to be. However, the implementation of our functions is messy. This makes it hard for someone to maintain the code after the original developer has moved on to a new task. There is a distinct lack of comments everywhere. It is appropriate to describe our documentation as completely missing. The saving grace when trying to read a team member's code is our distinct naming convention. Every variable name represents a piece of data we agreed should exist. The data is stored/represented in a collection of database tables. Therefore, everyone can trace issues back to the source function/file. This lets us prompt the correct person to fix an issue. Ideally, we would all be able to fix the issues within anyone's code. Our project contains lots of similar code, but nothing is copy and pasted. We usually start with some data, run a php file, get its output and either display the results or redirect the user to the correct page. Due to the way we organized our javascript functions, each page has a controller. This prohibits a page from accessing the functionality of another page. So far, we have not needed to perform the same tasks in an identical fashion between both user types, but it may be an issue in the future. We do not have constants because there is no initial data or a piece of data that remains static upon launching the program for the first time. Therefore, we have made our code modular in a sense that we can delete and add features very easily. However, we are not able to easily edit each other's work. Which implies a third-party entity would have trouble figuring out how to maintain our software. However, most should be able to add features using our existing code very easily.

Our code does not conform to our original design. However, it does conform to a design choice we made very early on. The first iteration of our project was extremely disorganized there were tons of files with the same name. It was difficult to find the correct place to start working on a new feature. This lead us to using a central javascript file that keeps track of the page we are on and has the appropriate functions for that page. More precisely, each page has a controller and a set of functions associated its events or initial load. While our javascript and php is highly modular; our html is not. We utilized a navigation bar that is the same for every professor on every page yet rewrote it every time we coded a new page. This meant we needed to update every page when we added a new page. This could be improved by using an html header file. We could simply abstract the navigation bar to a sperate file that is linked in each html file that needs it. Thereby reducing the amount of code, we need to update upon creating and deleting pages. Our student pages had a similar issue and we can apply the same method to further modularize our code.

The functionality of our code is rather simple. I can tell what blocks of code do without necessarily understanding the implementation. We chose simple and descriptive variable/file/function names which alleviated the burden of tracing through to figure out which data I am working with within a given function. Everything works as intended. If the javascript controller is told to handle some data, it calls the correct php file to handle the interaction with our database. There exists some dead code within our project. We have some lines commented out in various parts of the project that should be removed. This is more prominent in our php files. They are not debug statements. Rather, they are relics of old

implementation methods used early in the project. We do not handle errors or exceptions at a user level. We prevent bad data from being entered into the database. If someone does not fill in the html fields correctly, we do not prompt them to fill them in correctly. This is something we are looking to fix as we near our final release.