My target audience is NBA fans who want to search NBA player stats. Although current NBA website allows the fan to search player stats, the website does not work intuitively. For example, the website cannot return multiple players, need to resolve name perfectly, and the user interface is hard to see. Therefore, I decided to implement a new NBA website.

My original method of setup AWS was to start an EC2 micro-instance with SUSE Linux AMI. Unfortunately, the SUSE Linux AMI downloaded the “index.php” instead of rendering the “index.php”. In the process of meeting with Chris Oh, it was brought to my attention that Amazon Linux AMI also has default images including PHP, MySQL, and other packages.

I set up security rules for my EC2 instance with the Amazon Linux AMI and used FileZilla Client to upload “index.php”. It successfully showed “index.php” on the web. Then, I terminal-ed into AWS EC2 instance to start Apache and installed a LAMP web server with the Amazon Linux AMI. The “index.php” was successfully shown on the web.

Subsequently, I launched an RDS instance and connected the RDS to my MySQL database through myPhpAdmin. In the myPhpAdmin, I uploaded the CSV data to create a database and the tables associated with the database. Thus, setups for the web development were complete.

Before I built the website, I abstracted the architectural style and pattern to make scalable architectural design decisions. I decided to use N-tier to separate the user interface, the logical decision, and the database information into presentation, logic, and data tier. The N-tier architecture made the website modifiable, portable, reusable, and most of all, scalable.

I first implemented my Presentation Tier with HTML consisted of form, input text, and submit input in “searchPage.php”. In order to query the list of all player’s name that contains the user input from RDS database (Database Tier) when submit button is clicked, I implemented Logic Tier, “database.php”. I also built “Player.php”, that also stays in the logic tier, in Object Oriented Architecture using encapsulation to produce organized player information retrieved from the “database.php”. Communication between “search.php” and “database.php” produced div and p on the user interface, “searchPage.php”.

After the implementation of the N-tier architecture in PHP and HTML, I decided to catch misspelled names, make AJAX search button, and create CSS to make a neat user interface. In order to catch misspelled names, I used Levenshtein distance to determine if the misspelling is within 5 characters in the “database.php”. For AJAX search button, I added onkeyup in the text input, implemented javascript function, and coded “search.php” that outputs the maximum of five relevant names. Lastly, I used CSS to produce beautiful presentation.

Navigating between modules works in XAMPP, but not in EC2 somehow. It shows my PHP code when I click on submit in EC2.