Testing/Unit Tests
LoLStats, Release 1
5/29/16

Twitch Status Module: (Michael)

To test this module, I had to ensure the javascript ran together with the website. We had to ensure that the javascript runs when the /streams is loaded. It also means that the page will only be loaded if it is in the AngularJS controllers. The testings I have done to accurately display Twitch Streamers online status would be:
- seeing if they are actually playing on Twitch by visiting Twitch site and checking
- ensure that players that are offline are actually offline by clicking the link provided on the site
- refreshing the page a hundred times to see if the javascript updates properly
- checking on other browsers to see that the javascript did not have any problems there
- ensure that the content on the site is aligned correctly

Content and News Section: (Michael)

To test this module, I had to ensure the websites I linked to provided top-quality content and news to people who accessed the site. It also had to have accurate logos for the website it was on. To test this worked correctly, I simply linked the images to the website using a html image wrapped around a link. To confirm it was working properly, I simply clicked on each site and made sure it linked to where the website the image was designed for. I also had to position the content and news properly on the website so users can click on without problems.

Verify the /api data recieved form the riot-api calls, by looking at my own matches and other matches on op.gg (Logan)

To test this module, I had to ensure the data returned on our /api/player/{region}/{summonername} and our /api/match/{region}/{matchid} endpoint was returning the correct data. If the data returned in our json from our API endpoints is correct, that means our database is also correctly storing the correct values in the mysql backend (because the API response is just a wrapper around our laravel table record classes).

To test the player endpoint, I loaded up /api/player/{region}/{summonername} and viewed the minified JSON in a formatted view. I could then easily compare all of this information with the information provided by riot in game. This will ensure that our data is being sanitized and stored and given a summoner's name and region.
To test the match endpoint, I loaded up /api/match/{region}/{matchid} and viewed the JSON again in a formatted view. I could compare that response object with the recent matches viewer in the League of Legends client.
After I verified these results were correct, I checked our SQL database records by writing a few scripts to dump the contents of our tables. After doing this, I could confirm that our SQL records were correctly being stored as well.
After all this was complete, I simply repeated these tasks on a bunch of different player accounts and a bunch of different matches (because sometimes the response objects from RIOT are not the same, so we need to test edge cases).

dataFactory.js/playerCtrl.js/player.html (Johannes)

To test this module, I had to ensure that the /api/player/{region}/{summonername} and the tables that are displayed by our player view are consistent. I viewed the data that our API returned on /api/player/{region}/{summonername} and compared it to the tables that are displayed in and around the table on /player/{region}/{summonername}. I repeated this for a couple of summonernames on different serves to ensure that there are no special cases that we missed I tried to find a selection of summoners that played a lot different game modes and different champions.

dataFactory.js/matchCtrl.js/match.html (Johannes)

To test this module, I had to ensure that the /api/match/{region}/{summonername} and the tables that are displayed by our match view are consistent. I viewed the data that our API returned on /api/match/{region}/{summonername} and compared it to the tables that are displayed in and around the table on /match/{region}/{summonername}. I repeated this for a couple of summonernames on different serves to ensure that there are no special cases that we missed I tried to find a selection of summoners that played a lot different game modes and different champions.

chart.js/chartCtrl.js/player.html (Johannes)

To test this module I simply looked at the /api/player/{region}/{summonername} page. I computed the percentages by hand and compared those to the charts.

Test responsiveness of page layouts for all screen sizes (Brandon)

To test the responsiveness of page layouts when viewed at different screen sizes, I used Google Chrome's dev tools to manually change the screen size to different common breakpoints for small, medium, large, and widescreen monitors. Components I evaluated at different screen sizes included the search bar and the data tables on the player and match pages.

Test compatibility of page layouts for different devices (Brandon)

To test across different platforms and devices I used Google Chrome's dev tool, which simulates viewing a page on a number of different devices including the Iphone, Galaxy, Nexus, and Ipad.