CMPS115 Sprintplan 1
LoLStats

(here we have to divide the stories into tasks and then assign working hours)

- Task 1 - Deploy master branch on digitalocean, with pull hooks to fetch and recompile source
    - 4 hours
- Task 2 - Get development environments up and running, with MySQL server and Apache running
    - 3 hours
- Task 3 - Incorporate RiotAPI wrapper into our project
    - 3 hours
- Task 4 - Define basic API routes for accessing player information via AJAX
    - 2 hours
- Task 5 - Design a player schema to represent all sorts of match type specific information
    - 6 hours
- Task 6 - Create menu bar with urls to different pages (ie. player page, community, etc) with a search bar to search by username
    - 4 hours
- Task 7 - Create a responsive layout for a player page with multiple sub sections (recent stats, most played champions, etc)
    - 3 hours
- Task 8 - Create a reusable ui component for percentage and non percentage stats
    - 3 hours
- Task 9 - Integrate Google Analytics with laravel to be able to see how many API calls are being made, how many live sessions are active, and other real time stats
    - 1 hour
- Task 10 - Add authentication to internal API so that outside applications can't use our database
    - 2 hours
- Task 11 - Integrate Bootstrap CSS library to provide a responsive web-design
    - 1 hour
- Task 12 - Have the api calls happen asynchronously, so the webserver can handle a render request but wait on API calls (which must query database + riot API when info is not in database)
    - 3 hours
- Task 13 - Make sure JS written in ES6 compatible code, so it works on majority of browsers
    - 3 hours

Player Analysis

- As a developer I want to be able to work on the project.
  Tasks:
    1. Get development environments up and running, with MySQL server and Apache running.

- As a developer, I want to be able to track analytics about users accessing my website in real time
  Tasks:
  1. Integrate Google Analytics with laravel to be able to see how many API calls are being made, how many live sessions are active, and other realtime stats
  2. Add authentication to internal API so that outside applications can't use our database

- As a player I want to access the website online.
  Task:
  1. Deploy master branch on digitalocean, with pull hooks to fetch and recompile source 4 hours

- As a player, I want to be able to see stats about myself to be able to analyze and improve my gameplay.
  Tasks:
  1. Integrate D3 into wireframe JS/HTML mockups
  2. Research graphing frameworks (Flot, Highcharts, etc) to determine which will work best
  3. Pick a unified typography and frontend design to organize data in a visual hierarchy
  4. Create a reusable ui component displaying percentage stats (ie. K/D/A)
  5. Create a reusable ui component displaying non percentage stats (ie. turrets killed)
  6. Store fields in a database to provide analysis over time

- As a player loading into a game, I want to quickly view stats so I can outplay my opponent.
  - The website should be fast and respond asynchronously

  Tasks:
  1. Integrate Bootstrap CSS library to provide a responsive web-design
  2. Have the api calls happen asynchronously, so the webserver can handle a render request but wait on API calls (which must query database + riot API when info is not in database).

- (As a mobile app user, I want to be able to use the site quickly on my phone so that I don't have to tab out to see stats. I want to be able to view the website on my mobile phone, so the interface needs to be responsive)
  - Start a mobile-compatible website
  - Fix the mobile site to fit better on many multiple devices

  Tasks:
  1. Integrate Bootstrap CSS library to provide a responsive web-design
  2. Make sure JS written is ES6 compatible code, so it works on majority of browsers

- As a teammate, I want to be able to see summoner statistics filtered for a specific role so I can recommend them what to improve on.
  - Enable filtering tools

Tasks:

1. Write API endpoint to query match database table for specific role and summonerId, returning win percentages