

## CMPS115 Sprint 2 Plan

### LoLStats

**Goal:** Provide a website where a player can view stats specific to matches he is interested in or played himself.

#### User stories divided into tasks:

##### Match Analysis

- As a player, I want to see champion picks, bans, runes and masteries selected for a match  
Tasks:
  1. Implement basic live game lookup endpoint
  2. Implement masteries lookup endpoint
  3. Implement full live match lookup endpoint that combines all required information
- As a strategist, I want to see what people do at a certain time frame so that I can strategize the best way to win.  
Tasks:
  4. Pull the time stamped match information from Riot's API
  5. Create a UI that displays the data
- I want to see farm, damage, and other relevant match data at a specific time in the match  
Tasks:
  6. Pull in the time stamped match farm/damage data from Riot's API
  7. Create a UI that displays the data
- As a player, I want to analyze ward placement and it's benefit by viewing ward placement over time and locations on d3 heatmap  
Tasks:
  8. Pull the ward data from Riot's API
  9. Implement D3 on our website
  10. Display the ward data in a useful way as D3 chart
- As a player, I want to see every kill logged and displayed on an image of the in game map  
Tasks:
  11. Pull the location of kills information from Riot's API
  12. Load a image of the in game map
  13. Display the kills on the map

- As a League of Legends player, I want to be able to pick great meta-champions so I can win solo queue. Show stat breakdowns for specific champions (win %, K/D/A, item builds)

Tasks:

14. Design a player schema to represent different types of match specific information

- As a LoL mentor, I want to show my students stats so that they can see how wins relate with game stats.

Tasks:

15. Build frontend view UI to display these stats

16. Specify on a piece of data whether or not that game was a win/loss for the player

17. Filter database queries based on attached win/loss info to display whether an item build/talent build/champion matchup resulted in a win

## List of Tasks:

### *Last sprint*

- Task 10 - Add authentication to internal API so that outside applications can't use our database
  - 2 hours
- Task 13 - Make sure JS written in ES6 compatible code, so it works on majority of browsers
  - 3 hours
- Task 16 - Store fields in a database to provide analysis over time
  - 1 hour

### *This sprint*

1. Implement basic live game lookup endpoint
  - 1 hour
2. Implement masteries lookup endpoint
  - 1 hour
3. Implement full live match lookup endpoint that combines all required information
  - 3 hours
4. Pull the time stamped match information from Riot's API
  - 1 hour
5. Create a UI that displays the data
  - 2 hours
6. Pull in the time stamped match farm/damage data from Riot's API
  - 1 hour
7. Create a UI that displays the data
  - 2 hours
8. Pull the ward data from Riot's API
  - 1 hour
9. Implement D3 on our website
  - 1 hour
10. Display the ward data in a useful way as D3 chart
  - 4 hours

11. Pull the location of kills information from Riot's API
  - 1 hour
12. Load a image of the in game map
  - 1 hour
13. Display the kills on the map
  - 2 hours
14. Design a player schema to represent different types of match specific information
  - 3 hours
15. Build frontend view UI to display these stats
  - 4 hours
16. Specify on a piece of data whether or not that game was a win/loss for the player
  - 1 hour
17. Filter database queries based on attached win/loss info to display whether an item build/talent build/champion matchup resulted in a win
  - 4 hours

### Team roles:

Logan Collingwood: Product Owner, Backend Lead  
Griffin Meyer: Backend {full stack as needed}  
Brandon Chai: Frontend Lead  
Johannes Pitz: Frontend {full stack as needed}  
Michael Le: ScrumMaster (Sprint 2), Frontend {full stack as needed}

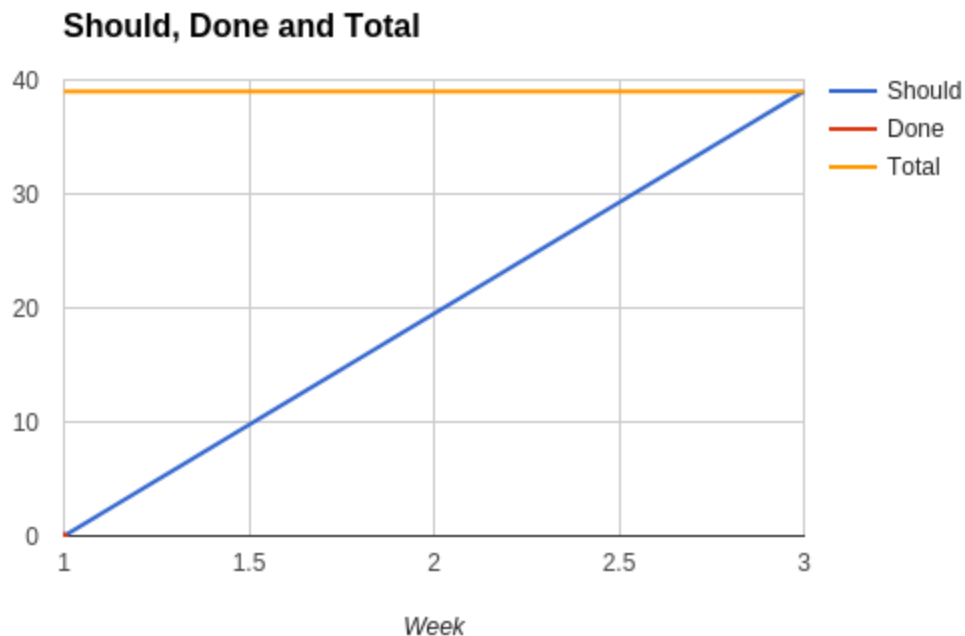
### Initial task assignment:

Logan Collingwood: Task 1: Implement basic live game lookup endpoint  
Griffin Meyer: Task 2: Implement masteries lookup endpoint  
Brandon Chai: Task 9: Implement D3 on our website  
Johannes Pitz: Task 5: Create a UI that displays the data  
Michael Le: Task 7: Create a UI that displays the data

### SCRUM Meeting times:

MoWeFr 12:15-12:25  
Shobhit will attend our Wednesday Meetings.

### Initial Burn-up Chart



# Initial SCRUM Board

Boards

Trello

CMPS115-lolstats

Private

To Do

Implement basic live game lookup endpoint 1 hour

Implement masteries lookup endpoint 1 hour

Implement full live match lookup endpoint that combines all required information 3 hours

Pull the time stamped match information from Riot's API 1 hour

5 Create a UI that displays the data 2 hours

6 Pull in the time stamped match farm/damage data from Riot's API 1 hour

7 Create a UI that displays the data 2 hours

8 Pull the ward data from Riot's API 1 hour

9 Implement D3 on our website 1 hour

10 Display the ward data in a useful way as D3 chart 4 hours

11 Pull the location of kills information from Riot's API 1 hour

12 Load a image of the in game map 1 hour

13 Display the kills on the map 2 hours

14 Design a player schema to represent different types of match

Add a card...

Doing

Task 10 - Add authentication to internal API so that outside applications can't use our database

Task 16 - Store fields in a database to provide analysis over time

Task 13 - Make sure JS written in ES6 compatible code, so it works on majority of browsers

Add a card...

Done Sprint 2

Add a card...

Done Sprint1

Release Plan

Sprint Plan

Task 1 - Deploy master branch on digitalocean, with pull hooks to fetch and recompile source

Task 2 - Get development environments up and running, with MySQL server and Apache running.

Task 3 - Incorporate RiotAPI wrapper into our project

Task 4 - Define basic API routes for accessing player information via AJAX

GM

Task 5 - Design a player schema to represent all sorts of match type specific information

Task 6 - Create menu bar with urls to different pages (ie. player page, community, etc) with a search bar to search by username

BC

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

Task 11 - Integrate Bootstrap CSS library to provide a responsive web-design

Task 12 - Have the api calls happen asvnrhronnislv so the webserver

Add a card...

Add a list...