

League of Legends Team Architect

Blake Skaja and Chidike Ubani

Table of Contents

Cover.....	1
Table of Contents.....	2
Introduction.....	3
Expansion of Materials.....	7
SimpleServer.php.....	7
Exploration of More Complex Issues.....	9
Dart.....	6
Asynchronous Event Handling.....	8
Other.....	11
Blooms Taxonomy.....	12

Introduction






League of Legends is one of the biggest video games in the world right now. More than 27 million people **per day** play a game of league. League of legends has a free API, which allows developers to access data about individual players, games, champions and much more. For our portfolio, we decided to make a website that uses the API to do some cool things.

The first thing that our website can do is a player search. This allows a user to type in the Summoner name of another player and get a lot of cool data about the players ranked stats.

The screenshot shows a web browser window with the URL `lolteamarchitect.elasticbeanstalk.com/SummonerLookup.html`. The browser's address bar includes a search icon and the text "Search". The website's navigation bar contains the following items: "League of Legends Application", "Home", "Summoner Lookup", and "Create Lineup". The main content area has a dark background with the title "League of Legends Player Search" in large white font, followed by the subtitle "Know which of your teammates is going to feed before the game even starts". Below this is a search input field labeled "Summoner Name" and a green button with a magnifying glass icon and the text "Get Summoner Data". At the bottom, there is a horizontal menu with the following categories: "Champion", "Total Games", "Win Percentage", "Wins", "Losses", "Average Kills", "Average Deaths", "Average Assists", "Average Farm", "Mid Score", "Top Score", "Adc Score", "Support Score", and "Jungle Score".

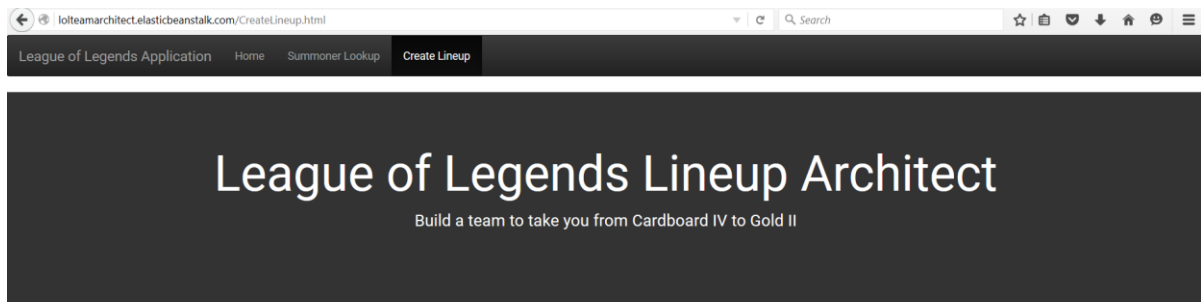
Any Summoner that has played a ranked game will be a valid entry in the Summoner Name field. The resulting data will look like this.

skyman12 : SILVER III

Champion	Total Games	Win Percentage	Wins	Losses	Average Kills	Average Deaths	Average Assists	Average Farm	Mid Score	Top Score	Adc Score	Support Score	Jungle Score
 Aatrox	2	0.00%	0	2	10.00	9.50	13.00	90.50	0.00	2.90	0.00	0.00	5.11
 Ahri	4	50.00%	2	2	8.75	13.00	19.00	180.75	7.82	0.00	0.00	0.00	0.00
 Amumu	3	33.33%	1	2	5.67	7.00	13.67	58.00	0.00	0.00	0.00	0.00	6.49
 Annie	10	60.00%	6	4	10.60	9.40	12.50	135.60	12.84	0.00	0.00	11.00	0.00
 Ashe	2	0.00%	0	2	8.50	9.50	13.00	190.00	0.00	0.00	3.30	0.00	0.00
 Bard	1	0.00%	0	1	1.00	8.00	4.00	8.00	0.00	0.00	0.00	-1.60	0.00
 Blitzcrank	3	33.33%	1	2	4.33	10.33	11.67	19.33	0.00	0.00	0.00	-0.30	0.00

We show how many games the Summoner has played with each champion, their wins and losses and other stats associated with that champion. We also have a sort by drop down at the top, allowing a user to order their data to see specific things.

The second thing that our website can do is help a user create a lineup from the Summoners they are paired with. This page asks for the uses to enter in the 5 players that they are going to be playing with.



Must fill in all 5 summoner slots. You can choose a champion and position for a summoner if you wish. If blank, they will be assigned to a role that will create the best team

Teams will be built using our secret "Best Lineup" algorithm. Teams will include at least 1 tank and 1 mage.

<input type="text" value="Your Summoner Name"/>	<input type="text" value="Which Champion"/>	Role: <input type="text" value="Top"/>
<input type="text" value="Summoner 2 Name"/>	<input type="text" value="Which Champion"/>	Role: <input type="text" value="Top"/>
<input type="text" value="Summoner 3 Name"/>	<input type="text" value="Which Champion"/>	Role: <input type="text" value="Top"/>
<input type="text" value="Summoner 4 Name"/>	<input type="text" value="Which Champion"/>	Role: <input type="text" value="Top"/>
<input type="text" value="Summoner 5 Name"/>	<input type="text" value="Which Champion"/>	Role: <input type="text" value="Top"/>





You can also predetermine where which champion and role a player should play. In this case, I am saying to build a team with the player's skyman12, cotton85, huntres, skyman31 and polishthunder21. I am also saying that skyman12 will play Ziggs mid and huntres will play Caitlyn Adc.

Teams will be built using our secret "Best Lineup" algorithm. Teams will include at least 1 tank and 1 mage.

skyman12	Ziggs	Role: Mid
cotton85	Which Champion	Role: Top
huntres	Caitlyn	Role: Adc
skyman31	Which Champion	Role: Top
polishthunder21	Which Champion	Role: Top

[Generate Best Team](#)

The application is then able to assign cotton85, skyman31 and polishthunder21 the remaining roles of support, top and jungle. Using our advanced stats, the program is able to generate a team, putting players in the correct positions - based on stats, team composition and winning percentage – that gives you the best chance to win. This is the result. As you can see, skyman12 is playing Ziggs mid and huntres is playing Caitlyn Adc. It then fills in the remaining players to their best roles. The application also makes sure to build the team in such a way that there is at least one tank on the team (so you are not all squishy) and so there is at least one mage on the team (so the other team cannot armor stack).

Champion	Total Games	Win Percentage	Wins	Losses	Average Kills	Average Deaths	Average Assists	Average Farm	Mid Score	Top Score	Adc Score	Support Score	Jungle Score
polishthunder21 - Top													
 Wukong	59	57.63%	34	25	6.88	6.42	7.49	169.12	0.00	17.43	0.00	0.00	0.00
skyman12 - Mid													
 Ziggs	3	66.67%	2	1	9.00	9.00	11.33	193.67	6.91	0.00	0.00	0.00	0.00
huntres - Adc													
 Caitlyn	26	50.00%	13	13	7.88	6.15	7.54	181.38	0.00	0.00	10.03	0.00	0.00
skyman31 - Jungle													
 Vi	6	100.00%	6	0	10.17	8.50	12.50	64.00	0.00	0.00	0.00	0.00	20.24
cotton85 - Support													
 Sona	17	47.06%	8	9	4.29	7.12	14.29	16.88	0.00	0.00	0.00	6.91	0.00

I encourage the reader of this document to play around and test out this application. Enter in some Summoners in the Search Summoner page (try skyman12, polishthunder21, huntres, skyman31, icastilloa or cotton85). They are all Summoners that have ranked data to show. You can also use other Summoner names if you happen to know any.

Throughout this document, we are going to cover the different technologies and concepts that we used to create this application. Anyone who has played League of Legends will immediately realize how useful this website can be when used properly. In the event that the reader of this document has not played League of Legends, they should still be able to appreciate the complexity of the technology used and amount of work that this project took.

Here are the different technologies and concepts that we used:

- Dart as our client side language

- PHP as our server side language
- REST API calls
- cURLs for getting JSON data
- Bootstrap for creating a professional looking webpage
- Asynchronous events for handling our server calls
- Object-Oriented programming in the structure of our code
- Using Amazon Web services to put our website on the web
 - <http://lolteamarchitect.elasticbeanstalk.com/>

I would estimate that more than 30 hours were put into the making of this project. It was nice to get a chance to work on something that is both interesting and allowed for the learning of new concepts. The project was designed using object-oriented programming techniques that will allow for me to continue working and adding new features to this project as time goes on, which is something that I plan to do.

I also took the time to use Amazon's free web services to host our project so others can see it. Along with this document, the source files will be attached with instructions of how to run our project locally. Since there is a decent amount of setup that needs to happen in order to get a project of this size to run, it may be easier to test and verify our project on the web at <http://lolteamarchitect.elasticbeanstalk.com/>. I recommend that you test this application with Firefox, as we are getting the best performance on that.

In doing this project, there are several resources that were used.

1. Dart Lang - <https://www.dartlang.org/>
 - The official dart webpage
2. Making REST calls in PHP - <http://rest.elkstein.org/2008/02/using-rest-in-php.html>
3. League of Legends API - <https://developer.riotgames.com/>
 - The LoL API for development
4. W3Schools - <http://www.w3schools.com/>
 - For learning Bootstrap and some basic css/html stuff
5. StackOverflow - <http://stackoverflow.com/>
 - Always

This project demonstrates the highest level of learning in blooms taxonomy of synthesis by taking the League of Legends API's and being able to build something new, useful and functional – using new and cutting edge technologies.

Expansion of Materials

As per assignment requirements, one of the key things that this portfolio had to demonstrate was our interaction with the materials that we had discussed and covered in class and how we took this knowledge and expanded upon it.

In order to make REST calls to the League of Legends API, we had to create our own server in order to make these calls (We will talk about why due to cross-site scripting issues later in this document). Throughout the semester, we have done a lot of work with PHP, so this seemed like a great opportunity to implement a custom PHP server for making these API calls.

SimpleServer.php

```
<?php
header("Access-Control-Allow-Origin: *");

$apiKey = '941d558c-76a3-4e0a-b90a-1c59c34623cd';

if (isset($_REQUEST['action'])) {
    session_start();
    switch ($_REQUEST['action']) {
        case 'getSummonerData':
            getSummonerData($_REQUEST['summonerName']);
            break;
        case 'getRankedStatsData':
            getRankedStatsData($_REQUEST['summonerId']);
            break;
        case 'getChampionData':
            getChampionData();
            break;
        case 'getSummonerRank':
            getSummonerRank($_REQUEST['summonerId']);
            break;
        case 'getChampionRoles':
            getChampionRoles();
            break;
    }
}
```

In the header of our php file, we need to put Access-Control-Allow-Origin: *. This header is used for security purposes in dealing with cross-site scripting. Since we cannot directly call the LOL API server (it does not implement the access-control-allow-organ header) from our client, we first make a call to our server, which implements this header. This server then goes out and makes the call to the LOL API. This is allowed since our server is not a client, so we don't have to worry about cross-site scripting issues. We can then make a call from our client to our server, which will be able to return the data since our server implements the access-control-allow-organ header.

We are able to pass in requests that we call actions. These actions can accept some different values along with them, which allows our server to make the correct calls out to the LOL API. For example, we can get a Summoners rank by passing using action=getSummonerRank&summonerId=123.

```

function getSummonerData($summonerName) {
    // get the basic summoner info
    $result = 'https://na.api.pvp.net/api/lol/na/v1.4/summoner/by-name/' . $summonerName . '?api_key=' . $GLOBALS['apiKey'];
    $summoner = curl_get_contents($result);
    echo $summoner;
}

function getSummonerRank($summonerId) {
    // get the basic summoner info
    $result = 'https://na.api.pvp.net/api/lol/na/v2.5/league/by-summoner/' . $summonerId . '?api_key=' . $GLOBALS['apiKey'];
    $summoner = curl_get_contents($result);
    echo $summoner;
}

function getRankedStatsData($summonerId) {
    $result = 'https://na.api.pvp.net/api/lol/na/v1.3/stats/by-summoner/' . $summonerId . '/ranked?api_key=' . $GLOBALS['apiKey'];
    $rankedStatsData = curl_get_contents($result);
    echo $rankedStatsData;
}

function getChampionData() {
    $result = 'https://na.api.pvp.net/api/lol/static-data/na/v1.2/champion/?champData=all&api_key=' . $GLOBALS['apiKey'];
    $championData = curl_get_contents($result);
    echo $championData;
}

function getChampionRoles() {
    // get the basic summoner info
    // $result = 'http://127.0.0.1:8081/LeagueOfLegendsServer/ChampionList.json';
    $result = 'http://lolteamarchitect.elasticbeanstalk.com/LeagueOfLegendsServer/ChampionList.json';
    $summoner = curl_get_contents($result);
    echo $summoner;
}

```

Seen here are the different functions that are called when our php server receives a request. We build the correct url based on the request and then use `curl_get_contents` to return a json string with the data from the API. This little script uses curl to get the json data.

```

function curl_get_contents($url)
{
    $ch = curl_init($url);
    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
    $data = curl_exec($ch);
    curl_close($ch);
    return $data;
}

```

There was a lot of effort and reading done on this idea of cross-site REST calls. Initially, we thought that it would be easier to make our API calls from our dart code (which is the client). In theory, this would be the easier solution, but for obvious security reasons, it would not be smart to allow clients to transfer data back and forth without more verification.

Exploration of More Complex Issues

Through this course, we have use JavaScript as the main language for our client side programming. For this final portfolio, we thought that it would be interesting to try out Googles new client side programming language called Dart.

Dart

Dart has many advantages over JavaScript in my opinion. First, Dart looks and feels a ton like Java, which is the programming language that most students here at Iowa State are the most familiar with. You can create classes, objects and all the things that make up a great object oriented programming language. They also include a lot of feature from newer languages that Java does not have, such as optional and named parameters for functions and removing the private and public keywords (they are instead indicated by `_variable` for private and `variable` for protected).

Starting a Dart project took a decent amount of reading, as there are many different components that do not exist in other frameworks. For example, a Dart project contains a `pubspec.yaml` file that helps you manage the dependencies of the project. Here is the `pubspec.yaml` file for our project.

```
name: 'LeagueOfLegendsWebApplication'
version: 0.0.1
description: An absolute bare-bones web app.
#author: Your Name <email@example.com>
#homepage: https://www.example.com

environment:
  sdk: '>=1.0.0 <2.0.0'

dependencies:
  bootstrap: "^3.3.4"
  browser: '>=0.10.0 <0.11.0'
  dart_to_js_script_rewriter: '^0.1.0'
  http: any

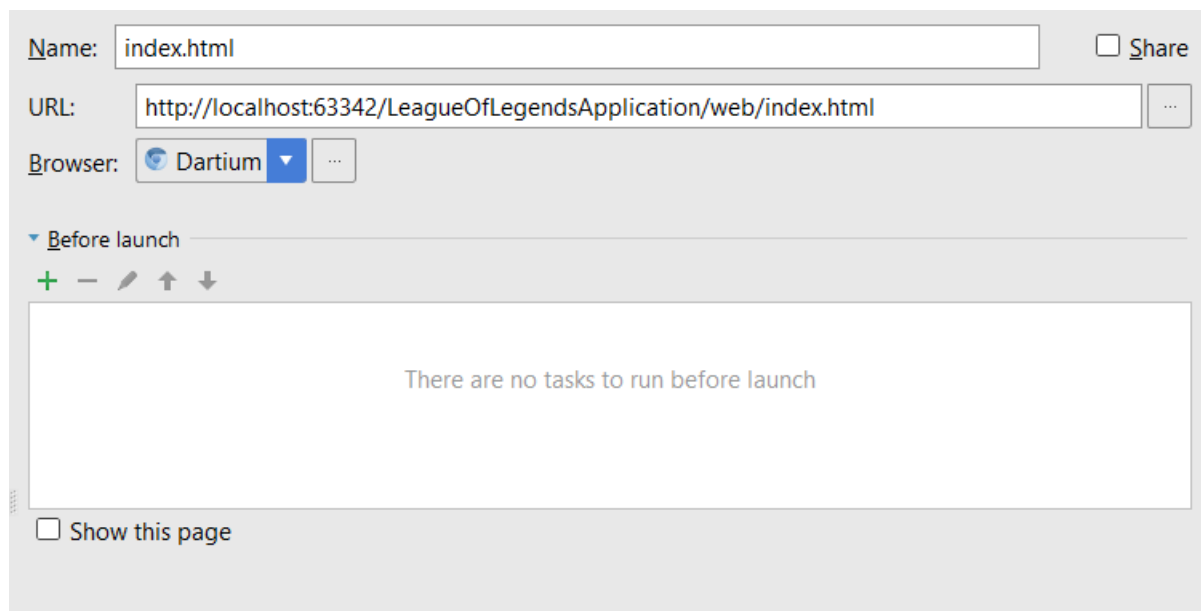
transformers:|
- dart_to_js_script_rewriter
- bootstrap:
  url: //maxcdn.bootstrapcdn.com/bootstrap/{{version}}
  version: 3.3.4
```

We have a few things here. First is environment, which says that our application is only compatible with a Dart version between 1.0.0 and 2.0.0. Next, a neat way to manage the dependencies of imports is included. As you can see, we have included bootstrap version 3.3.4 as part of our project. Using the pub functions, we can build our project and include bootstrap in the release build.

There are two ways to use the different pub functions, either command line by navigating to the project directory with the `pubspec.yaml` file or through Webstorm.

Pub actions: [Get Dependencies](#) [Upgrade Dependencies](#) [Build...](#)

Since I like to use UI for doing things, I choose to use the built in Pub manager that WebStorm provides. There are a few things that we can do. We get do a pub get (Get Dependencies) which will look at our pubspec.yaml file and get all the required dependencies into our project. This is nice, as our pub can go out and get all the bootstrap files that we need. We can also call pub upgrade, which will upgrade our dependencies. Whenever we click the run button in Webstorm, we are calling pub build and then pub server, which builds our project and then serves it on localhost. Pub build has a few options, we can either build in release mode or debug mode. Calling pub build transforms the dart code into .js, which can then be run on all browsers. However, we want to have a way to be able to debug our code in Dart to find issues, which is why Dartium is a neat little browser for Dart development. Dartium is a Chrome-esk browser that runs Dart natively instead of compiling it to JavaScript. This can be setup from within WebStorm.



Using Dart turned out to be an interesting experience. Getting a project up and running was a lot of work, but the reward was very much worth it. The object oriented aspect of the language allowed for a much more controlled programming experience and made it easier to separate our code into individual components. It is a little tough to use Dart, however, as there is not a ton of resources available for troubleshooting and debugging when issues arise.

Asynchronous Event Handling

One of the new things that we had to work with during this project was the idea of using asynchronous events to make calls to our servers. Most programming that you first learn is done synchronously, meaning that one function is called, once that function is done, the next function runs and so forth. Asynchronous programming can be used in web development, especially when making server calls that you are not sure how long it will take to process. In Dart, we are able to return Future objects, which basically means that we are going to make some call and at some time, we will return an object. In the meantime, we return an object called Future. This allows us to do several calls at the same time, and

when they are finished, return the Future objects. We can use the keywords `await` in Dart to tell our code to wait until a Future is returned before moving on to the next line of code. This is useful when we are trying to load some data before loading some other data. For example, in our code, we build a Summoner object to hold all the Summoner's data.

```
Future<Summoner> buildSummoner(String summonerName) async {
  if (championMap == null) {
    await _getStaticChampionInformation();
    await _addToChampionRoles();
  }

  // URL to get the summoner data associated with the summonerName
  var summonerUrl = "http://$host/LeagueOfLegendsServer/simpleserver.php?action=getSummonerData&summonerName=$summonerName";

  // call the web server
  await HttpRequest.getString(summonerUrl).then(onSummonerDataLoaded).then(_assignSummonerData);

  //Build a new Summoner once the data is retrieved
  Summoner s = new Summoner(summonerName, summonerData);
  var summonerId = s.getId();

  // Get the Summoners rank
  var summonerRankUrl = "http://$host/LeagueOfLegendsServer/simpleserver.php?action=getSummonerRank&summonerId=$summonerId";

  // call the web server
  await HttpRequest.getString(summonerRankUrl).then(onSummonerRankLoaded).then(_assignSummonerRank);

  await s.addRank(summonerRank);

  // URL to get the ranked stats data associated with the summoner id
  var rankedStatsUrl = "http://$host/LeagueOfLegendsServer/simpleserver.php?action=getRankedStatsData&summonerId=$summonerId";

  // call the web server
  await HttpRequest.getString(rankedStatsUrl).then(onSummonerDataLoaded).then(_assignRankedStatsData);

  await s.addRankedStatsData(rankedStatsData, championMap, championRoles);

  return await s;
}
```

As you can see, there is a specific order in which this object needs to be built, so we use `await` on some of the calls to guarantee that that object is finished before moving on to the next call.

Some of the issues that we had to deal with in working with the asynchronous calls were issues with trying to call methods on Futures before the objects were created. This was causing issues, which forced us to use `await` to make sure that certain methods were done before moving on to the next call. We also faced some issues in regards to using the LOL API. Since we only have a developer and not a commercial API key, we are only allowed to make 10 calls to the LOL server every 10 seconds. This was causing some issues when trying to load the data for multiple Summoners, as we need to make several calls to the API in order to get all the data that we need. We were able to use `Future.wait()` in order to allow our code to pause between Summoner creation calls, therefore we don't overload the API. I have applied for a commercial API, which is free, but the process can take up to 2 weeks. Once this key is obtained, the speed of the application will be increased, as we will no longer need to delay to make sure we are not overloading the API's call limit.

Other

As this is a rather large project, it would take me far too many pages to talk about in great detail about all the complex things that we did for this application. So I am going to combine a bunch of the topics into this section called other, in which I will briefly cover what we did. The best way to get a good

understanding of this project, is to, in fact, play around with the application at the provided URL and to examine the code provided.

I was able to take this project and use it as an opportunity to learn more about Amazon's free web services. I have always been intrigued with how to actually get a website up and running (not just on localhost), especially because the webpage that we created for this portfolio is not just a static website. Amazon's web service allows you to host a page on their .elasticbeanstalk domain using their tools. After creating a PHP server through Amazon, I was able to use the dart pub manager to build my dart code for release. This, along with my php server code as well as some .json files that I use, were put into a zip file. Amazon allows you to upload a zip file, which is then served. All in all, the process is not super hard, but there is definitely some reading and trial and error in order to get the project up and running.

We also use some advanced Bootstrap in this project, which I think is one of the coolest technologies out there. Bootstrap allows a user to get a professional and nice looking webpage up and running rather quickly. Dart also is very nice for editing the DOM directly, as it allows you to create classes such as DivElement or RowElement. These elements can then be added directly to the DOM. As seen in our code, we create several static classes that implement build functions.

```
class ChampionDisplayComponent {  
  static TableRowElement buildComponent(ChampionStats c, String rowColor) {
```

For example, we are able to call ChampionDisplayComponent.buildComponent() and pass in the ChampionStats and the row color that we want. This is nice, as we are able to use this class to build a row to display our champion data from wherever we want. We can reuse this component in multiple places as well, which is the great part about our object oriented approach to programming.

For more about the OO side of this project, we were able to break our components up into several different classes. We have our Summoner class, which has lots of data, including a list of ChampionStats. ChampionStats contains a Champion and a RankedStats. Champion has the data for each champion while RankedStats contains all the advanced data for ranked games. By storing the data in classes and not primitive data structures, we are able to define methods and functions for our classes. We are also able to use encapsulation, as each class knows about the data it needs for itself and are not linked to other classes. All in all, the project was designed in a way that follows standard agile design pattern practices, as it will be easy to continue work on this project in the future due to the layout of the code.

Blooms Taxonomy

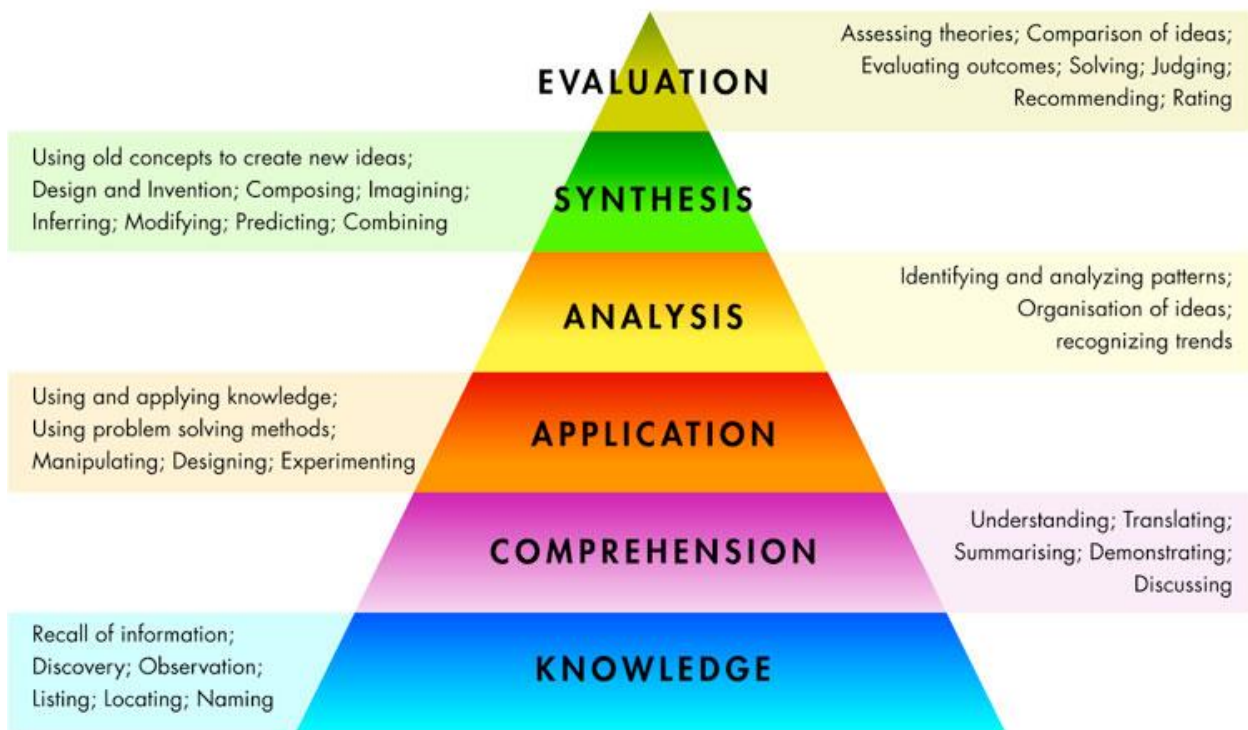
As always, a huge part of these portfolios are to make sure that we are looking at Blooms taxonomy and doing work that really focuses on the highest elements of learning. I can without doubt say that our project is in the highest level of learning, mainly Evaluation and Synthesis.

Evaluation is normally the hardest level to reach, as it requires a person take time to compare people's ideas and make a decision based on their findings. In starting this project, we had to evaluate all of the different technologies that were available to use and make a decision based on what technologies fit the needs of our project. We knew that we wanted to create a League of Legends web application, so we had to choose a language in which to write out client. We decided on Dart after evaluating other options, such as JavaScript. The pros of Dart, such as OO principles and an opportunity to learn a new technology

outweighed the risks of Dart, like not a lot of online support or resources. We also had to evaluate different backend frameworks and eventually settled on using PHP to write our server, because of how lightweight and easy the language. We considered using GO, which is Googles server language, alongside with Dart, but the simplicity and lightweight components of PHP were too good to pass up.

Synthesis the level of blooms taxonomy that really encompasses the entire idea of making a portfolio. You want to take the things that you have learned and be able to create something new and innovative from them. For this project, I would have to say that we succeeded in doing this. We were able to take the ideas of pulling data from API's, writing server code, writing client code, working with user interfaces and create an application that does all of this. Not only did we succeed in doing that, we were able to create a fully functional website that allows users functionally that otherwise would not be provided. We were able to take the API and create an algorithm for building the ultimate league of legends team.

B L O O M S T A X O N O M Y



We were able to meet the requirements of making sure that our project focuses on the components of blooms taxonomy. Using this pyramid as a guideline forced us to consider and evaluate all options when choosing a framework to work with. We also had to be critical of online resources that we found when looking up suggestions to issues that we were having. Blooms Taxonomy also really allowed us to focus on creating a new and innovate project, something that is not only useful for this class but a project that can be used by the millions of people worldwide who play League of Legends.

