tr16879 & bm16015

WEB TECH ASSIGNMENT – REPORT

TOM RUSSELL & BEN MURRAY

INTRODUCTION

League of Legends is a popular free to play competitive multiplayer game. The developers (RIOT Games) provide an API for retrieving statistical data about the game. For our assignment, we decided to make a website that accesses the RIOT API and presents data about a players recent matches in a visually pleasing way. Our website is a desktop first project, but also functions reasonably well for smaller resolutions such as tablets (but not mobile).

PROBLEMS UNIQUE TO THIS PROJECT

Dealing with the API created a number of problems. Primarily, for an unpublished project there are very strict rate limits on API calls. Additionally, a large number of data anomalies exist: many times while testing throughout the project we encountered new exceptions, such as missing data. We needed to deal with requests to an external server, which presents a number of extra problems outside of our control and dealing with these in a graceful manner while communicating the error to the user required some effort (balancing rate-limiting, external server and user input errors).

With the basic API key, you can only make 10 requests in 10 seconds or 500 requests in 10 minutes. Because of this, if you try to visit many profile pages for the first time in a short space of time, the pages will load very slowly but will eventually load. This is thanks to our rate limiting system – if the rate-limit is reached, each request will attempt a retry after a request can be made again. This issue is further reduced by our database system that stores already retrieved data locally – this is explained in depth in the databases section.

HOW TO USE OUR WEBSITE

To view any data on our site, you will require a valid ‘summoner name’ (player account must exist and have more than 5 games played). Here are some valid names to try:

- Gurglesprain
- Firoozan
- Lvl5Dialga
- Justinius888
- Gangnam samurai
- Yeezys
- Baozi1993
- Bluemat
- showmesomeLove
- gummihaj

You should be able to guess some invalid options!

On the profile page the radio buttons below the graph will switch the type of data showing, and the points will move with an animation. By default the match data at the bottom shows the data for the most recent game, and if you click on any of the graph points the match data will update for the chosen match.

OUR GRADE ESTIMATES

SERVER

Grade Estimate = A

We used Django as our server, and have become familiar with the setup. While Django does take care of some the work for you, we were still required to set up all the views and different url patterns for form handling and setting up our own requests. Additionally, our backend deals with all of the API access, including rate-limiting, JSON decoding and data extraction, and more complex error handling due to the added limiting factor of relying on an external server.

DATABASE

Grade Estimate = A

While Django does simplify database use somewhat (providing an ORM, having SQLite 3 installed by default etc.), we made the use of the features by creating a complex system of data storage.

Due to the issues mentioned before with rate limits, we chose to use our database to store data retrieved from the API. This means that the same user returning to our site can load their data much quicker, since many of the API won't need to be run.

Additionally the database 'cleans' itself by removing obsolete data: we only show the data for the last 5 matches, so if any new matches are added then extra ones are deleted.

We have flushed our database in the uploaded copy of our server to illustrate how our database system speeds up data loading for repeat users.

DYNAMIC PAGES

Grade Estimate = B

We use dynamic pages in two main ways: firstly, when loading a new page we insert simple data into placeholders. Examples of this are error messages on the homepage and the summoner name on the profile page. However, due to the sheer amount of data required for 5 full match views, this method wouldn't work for the entire profile view. To solve this, we used JQuery on the clientside to request data from our database in JSON form, so we can more easily work with the data.

HTML

Grade Estimate = A

We used XHTML5, which we configured on the server side so pages are delivered to the client in XHTML format. We created some complex page structures and tried to make it as friendly as possible for multiple resolutions.

CSS

Grade Estimate = A

We worked a lot with CSS to achieve a visually appealing style. We spent a significant amount of time learning how to lay out all the elements correctly.

JS

Grade Estimate = A

For our project we wrote some complex JavaScript functions to achieve animations. Most notably are the animations on our SVG graph and the background transitions on the homepage and profile page (when switching between matches). Additionally, when a node on the graph gets clicked all of the match data gets updated.

PNG

Grade Estimate = B

We did some basic work with PNGs – cropping and transparency of our background texture, and our logo which was exported as a PNG.

SVG

Grade Estimate = A

We worked with SVG in multiple ways: our graph is set up as an SVG canvas in the HTML, then the SVG is animated and modified in the JavaScript. We additionally created our SVG logo in Inkscape (but converted it to PNG due to unexpected positional errors in the SVG image). The logo has been saved as an Inkscape SVG file and is stored in our server folders.

FINDING OUR FILES

Because Django uses a specific file structure, we have explained in our readme file where all the relevant files for the above sections are located.