A League Champion Visualization

Paul Kim (pkim62) | Moderator:smarakp2
Johnathan Im (jim20) | Moderator: shivenk2

This is a website that visualizes data scraped from op.gg project for CS242

Abstract

Project Purpose

The project provides a friendly interface that uses data from op.gg (a League of Legends-based website) that helps
players to understand the best champions/characters in the current state of the game.

Project Motivation

We both are avid gamer enthusiasts that have played a multitude of game genres and our latest endeavor is League of Legends.
We are creating this in hopes of helping fellow players to become more informed.

Technical Specification

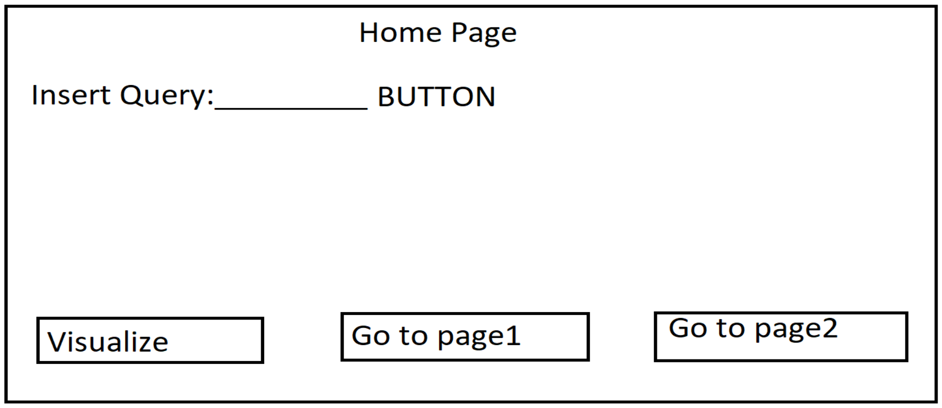
Platform: Browser/Web-Application
Programming Languages: JavaScript for front-end and Python for Flask should backend required
Stylistic Conventions: PEP-8 and Javascript Conventions from Assignment 2
SDK: Python 3.8
IDE: Visual Studio Code, Eclipse
Tools/Interfaces: D3.js
Backend: MongoDB, Flask
Target Audience: League of Legends Players

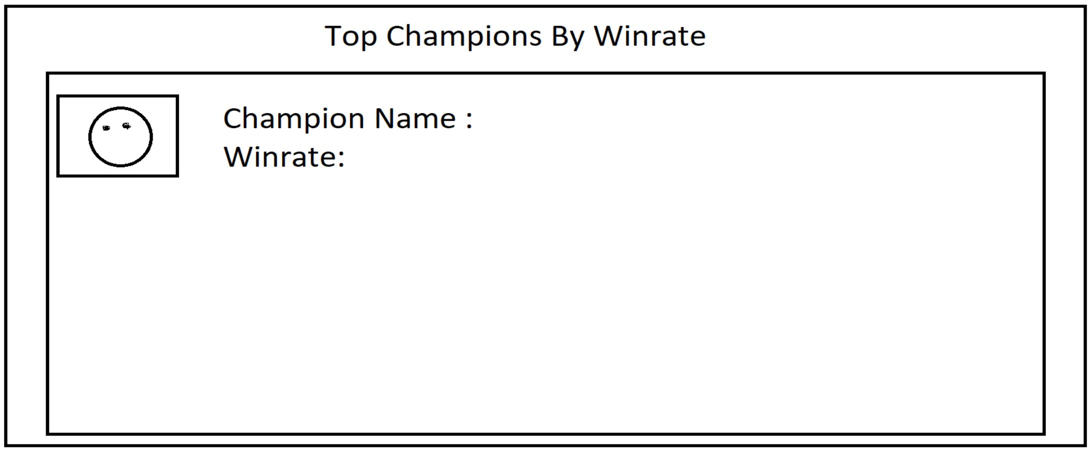
Functional Specification

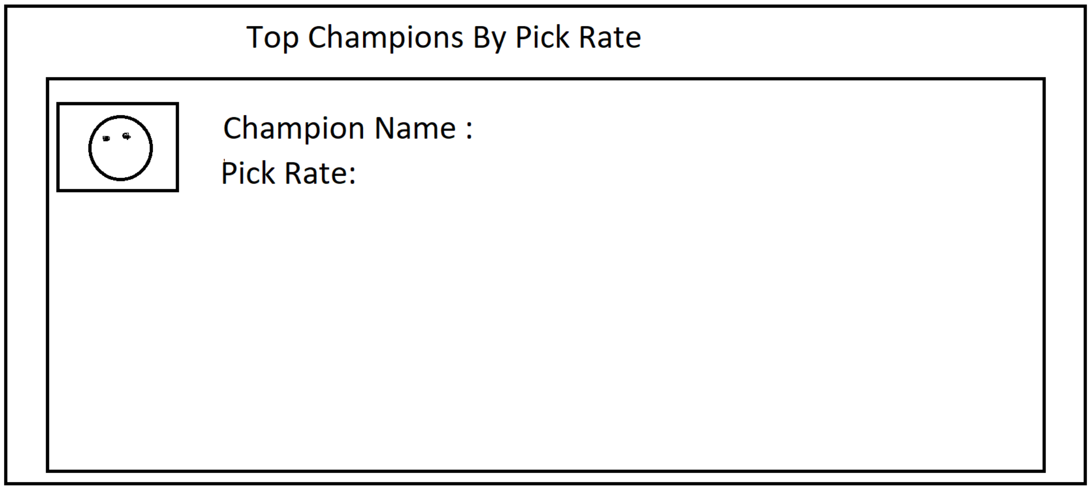
Features

Sort Champions by Win Rates
Sort Champions by Pick Rates
For each champion : Counter champions and Strong Against Champions
Visualization of Data

UI Sketch


Home Page
Insert Query:__________ BUTTON
Visualize
Go to page1
Go to page2


Top Champions By Winrate
Champion Name :
Winrate:


Top Champions By Pick Rate
Champion Name :
Pick Rate:

Scope of the project
Local host

Brief Timeline
Week 1:
i.) design MongoDB Database (Johnathan & Paul)
ii.) set up Flask with API Requests (Johnathan & Paul)
iii.) scrape data off of op.gg (Johnathan & Paul)
iv.) push data to Mongo and test server (Johnathan & Paul)
Week 2:
i.) query parser to pull info from Database (Johnathan & Paul)
ii.) front-end design with static UI to test (Johnathan & Paul)
iii.) implement navigation between pages (Johnathan & Paul)
iv.) Manual test for Web application (Johnathan & Paul)
Week 3:
i.) Design MVC
ii.) Connect backend with frontend (Johnathan & Paul)
iii.) Polish UI design (Johnathan & Paul)

Detailed Division of Labor by Week

Week	Pole	John
1	-design MongoDB Database -set up Flask with API Requests -scrape data off of op.gg -push data to Mongo and test server	-design MongoDB Database -set up Flask with API Requests -scrape data off of op.gg -push data to Mongo and test server
2	-query parser to pull info from Database -front-end design with static UI to test -implement navigation between pages -Manual test for Web application	-query parser to pull info from Database -front-end design with static UI to test -implement navigation between pages -Manual test for Web application
3	-Connect backend with frontend	-Connect backend with frontend

	-Additional functionality -Polish UI design	-Additional functionality -Polish UI design

Rubric
Week 1

Category	**Total Score**	**Details**
design MongoDB Database	5	0: No Database Functionality +5: Database setup
set up Flask with API Requests	5	0: No request handled +1.25: For every API Request(GET, POST, PUT, DELETE)
scrape data off of op.gg	5	0: No scraping +5: Scraping Works
push data to Mongo and test server	5	0: No data in database +5: Data is pushed
Unit Test	10	+1 for each 2 Tests

Week 2

Category	**Total Score**	**Details**
query parser to pull info from Database	5	0: No Query Parser +5: Query Parser Works
front-end design with static UI to test	5	0: No static UI +1.25: Query input textbox +1.25: Submit Query Input Field +1.25: Results Textbox +1.25: Navigation Buttons
implement navigation between pages	5	0: No Navigation +1: Each navigation page
Manual test for Web application	5	0: No Manual Test Plan +1: per test

Unit Test	10	+1 for each 2 Tests

Week 3

Category	**Total Score**	**Details**
Design MVC	5	0: No MVC +1.33: Each component of MVC
Connect backend with frontend	5	0: No connection +5: Connected
Polish UI design	10	0: No polishing +10: Polished
Unit Test	10	+1 for each 2 Tests