eSportGuru

Persia Ghaffari

Thien Vo

Rachel Beale

Eric Liu

Kevin Zhang

Introduction

Over the last decade, the market for competitive gaming has exploded, producing an almost insatiable demand for high quality competitive multiplayer video games and highly skilled players. The competitive gaming scene, coined as “eSports,” has raked over five hundred million dollars in revenue in 2017 alone. Financial news institutions, such as the Business Insider and Forbes, have projected that eSports will become the next multibillion dollar industry within the next two years. The exponential demand for highly competitive gaming has been fueled by new emerging technologies, such as online streaming through Twitch or YouTube.

While the video game industry has always supported competitive local multiplayer options, developers didn’t have a reason to produce highly competitive multiplayer video games with an emphasis on broadcast quality presentation or production, until now. Not only have developers and publishers created these types of video games to meet consumer demands, they’ve also promoted and advertised them through sponsorships. Publishers sponsor big tournaments by adding enormous amounts of prize money as a means to maintain interest of their games, as well as to increase their popularity. Developer sponsorship draws in an enormous amount of people to play and view tournaments, simply because eSports is one of the few activities that anyone can become good in, regardless of background.

Without a doubt, this decade in video games has been be defined by rise and dominance of eSports and eSports culture. However, eSports has also been slowly creeping its way into mainstream media. For instance, ESPN has started broadcasting huge League of Legends, Dota 2, and Hearthstone tournaments, and Disney has been experimenting with broadcasting Super Smash Bros. tournaments. The fact that these companies have been testing eSports viewership and seeing favorable results is proof that eSports will find its place in mainstream entertainment, alongside traditional sports such as football or baseball.

Motivation

There are several databases on the internet devoted to eSports, however they either cover one game or a few major games in depth. For instance, gosugamers.net is a major website dedicated to providing ranking information for the top eight competitive PC games, such as Dota 2, League of Legends, and Hearthstone. While this information is highly useful to those already initiated to these games, it would be very hard for casual viewers to absorb of all their information. These websites are overloaded with statistics and rankings that would mostly make sense to people already familiar with the ranking system.

The goal of eSportGuru is to create a general but universal database for all competitive games. The general information included would be basic information over players, tournaments, teams/sponsors, and the games themselves. While other websites dive into great detail with in-depth statistics over specific matches, teams or players, our site is aimed at presenting data in an aesthetic fashion that, by nature, is wide in scope. What our site would lack in depth would make up for in scope, and this would allow viewers new to eSports to quickly find general data that could serve as a stepping stone in their research over the game and its players. While the developers of eSportGuru would love to add rankings and useful statistics for each player and team, we do not have the resources to do so with the given scope; it would cost too much money to maintain.

Use Cases

Users can query our database through four main models: players, teams, games, and tournaments. If a viewer tunes into a live stream of a prestigious event for the first time they may find themselves asking questions such as: What game are they playing? How popular is that game? Is this tournament a big deal and who’s the favorite to win? What’s the prize pool and who’s on what team? The answer to these questions are not always immediately apparent from the stream itself, so the viewer can make a query on our database to find out.

If the viewer wants to find out more about a tournament, they can simply search for the tournament and will receive information about the winner of the tournament or the current standings, date(s) of the tournament, the game(s) played, as well as what teams and players were involved. For a particular game, the viewer can see when the game was released, who published it, what genre(s) it falls under, and relevant links to the game’s website. For a given team, a viewer will receive the acronym for the team, the current roster of the team, and the video game(s) the team is involved in. Lastly, the viewer can query a player and see their full name, hometown, their role (if any), current team (if any), and current video game. Furthermore, all of our models are interconnected. For example, if a viewer queries a player they’re interested in, and see that the player has a team, they can immediately click the team to find out more information. This type of interconnectivity can be found throughout the website and increases the ease of access for the user, allowing them to bypass the act of researching. This interconnectivity also ties into the philosophy that esportsGuru should be a very user friendly site to visitors who have little to no eSports experience, allowing the user to access a slew of data with minimum interruptions.

RESTful API

The majority of our preliminary data was scraped from [pandascore.co](http://pandascore.co/). Their API allowed us to request information from any game, player, team, or tournament, either requesting all of a type of model, or allowing us to request a model based on their ID in the PandaScore database. All of our models can be filtered or sorted by their respective attributes. For example, we can send a request for all players with the hometown “Los Angeles”, or we can request all players but sort them alphabetically by first or last name. This same functionality applies to all of the data models. Below is an example of a player entry returned by the PandaScore API:

{

"id": 7659,

"slug": "q1",

"name": "q1",

"first\_name": null,

"last\_name": null,

"role": null,

"bio": null,

"hometown": null,

"image\_url": null,

"current\_team": {

"id": 583,

"name": "DAN Gaming",

"acronym": "DAN",

"image\_url": null

},

"current\_videogame": {

"id": 1,

"name": "LoL",

"slug": "league-of-legends"

}

}

For the games League of Legends and Dota2 in particular, PandaScore API can also return details of particular matches from a player or team, such as gold earned, heroes or champions selected, and the level and kills of each player in the match. However, the Pandascore API did not provide enough information specifically about the game itself, rather than the professionals playing the game.

Another useful resource was [igdb.com](http://igdb.com/). This website provided the data to populate our game models. This game database allowed us to search for game statistics based on a given ID, or allowed us to filter games by genre or by rating. The igdb database also provided sorting functionality such as ordering by popularity. It also provided much more in-depth statistics than Pandascore. For example, for League of Legends we can see the MetaCritic rating, the genre and the description, as well as links to the official website and social media outlets. It also provides links to members of the community currently live streamers, reviews of the game, as well as recommendations to games that are similar.

The API we are currently developing will be similar to the PandaScore API. We will be able to provide all of our players, teams, tournaments, or games, as well as the capability to search for specific models by their id. If the user wants to filter by tag or name to find a particular player in a given game, they must use the Get All Players functionality and filter it there, since there can be multiple people across the game with the same name/gamer tag.

Models

Our database is composed of four main models : players, teams, tournaments, and games. The player has attributes specific to itself, such as name, and hometown, as well as attributes connecting it to other models. For example, the player “Doublelift” has the first name “Yiliang” and last name “Peng”, but also is a part of the team “Team SoloMid”, so his model will contain his attributes unique to himself, and a foreign key to the TSM team model in our database.

Our player model has a private ID as a unique identifier, a name (equivalent to their gamer tag or in game name), a first and last name, a role, a hometown, an image, a current team, and a current video game. The current team and current video game attributes are foreign keys to other models in our database. Our team model has a private ID as a unique identifier, a name, an acronym, an image, a list of players, and a current video game. The players attribute and the current video game attribute are foreign keys to other models in our database. Our tournament model has a private ID as a unique identifier, a name, a begin and end date, a video game played, and a list of teams involved. The video game played attribute and the list of teams attribute are foreign keys to other models in our database. Our games model has a private ID as a unique identifier, a name, a developer, a genre, a release date, a website, and a logo. The games model has no foreign keys associating it with other models.

Tools

To set up our website we used a couple of tools. We used React, Bootstrap, Apiary, Github, Flask, Google Cloud Platform, Slack, Trello, and PlanItPoker. React and Bootstrap were key tools used in allowing us to develop the aesthetic and visual features of the website. Some of the key aspects that were implemented using React and Bootstrap were the navigation bar, that allowed users to traverse the website to pages or topics that they are interested in, as well as the carousal on the front page, a circular queue of images that changed on user input. Apiary was a tool that allowed us to document the APIs in a presentable and clean fashion. This tool was useful when creating the webscraper, as well as when storing the information from the web scraper into the database. We could see the exact JSON format from what we would receive from the API’s we scraped, as well as the exact JSON format that our SQL database required when storing our information. Github was used for version control and on-going development of different features. We made different parts of the website on different branches then proceeded to merge the final product after going through review by at least one teammate. Since everyone had different roles, we would all be working on our own separate development branches, so Github allowed us to manage these separate features and functionalities and eventually combine them into a functioning website. Flask is a micro framework for Python, and handled our routing for back-end. Google Cloud Platform was used to host our website and used to store our SQL database.

Slack was used for communication between team members. It also notified us when new branches, pushes, or pull requests were made on Github and also show when something was being done on Trello. Trello was also an organizational tool that allowed us to keep track of projects, whether they were currently on going, waiting for reviewed, or yet to be started. It allowed us to keep our priorities focused and see what features were close to being finished or what features needed more attention. This helped keep us organized and showed who was doing what and what needed to be done. PlanItPoker was a tool used to decide the difficulty in implementing user stories and features for the website. We used this to create stories or consumer experiences that we thought should be a functionality in our website. We then each anonymously voted for the difficulty, ranging anywhere from 0 to 5 in determining the estimated cost of implementing said functionality. This tool allowed us to determine what could be easily added on to the website and what would require more time, which, in turn, logically paved the development pathway and dictated what features could be finished in a timely manner. Slack, Trello, and PlanItPoker were mainly used to communicate between team members while BootStrap and React were used for frontend and Apiary and Github were used for back-end and Flask and Google Cloud Platform was used for platform stuff.

Hosting

We used Google Cloud Platform’s App Engine to host our web server and their CloudSQL for database storage. Flask