Jaylon Kiper, Aaron Bone, Elizabeth Fultz

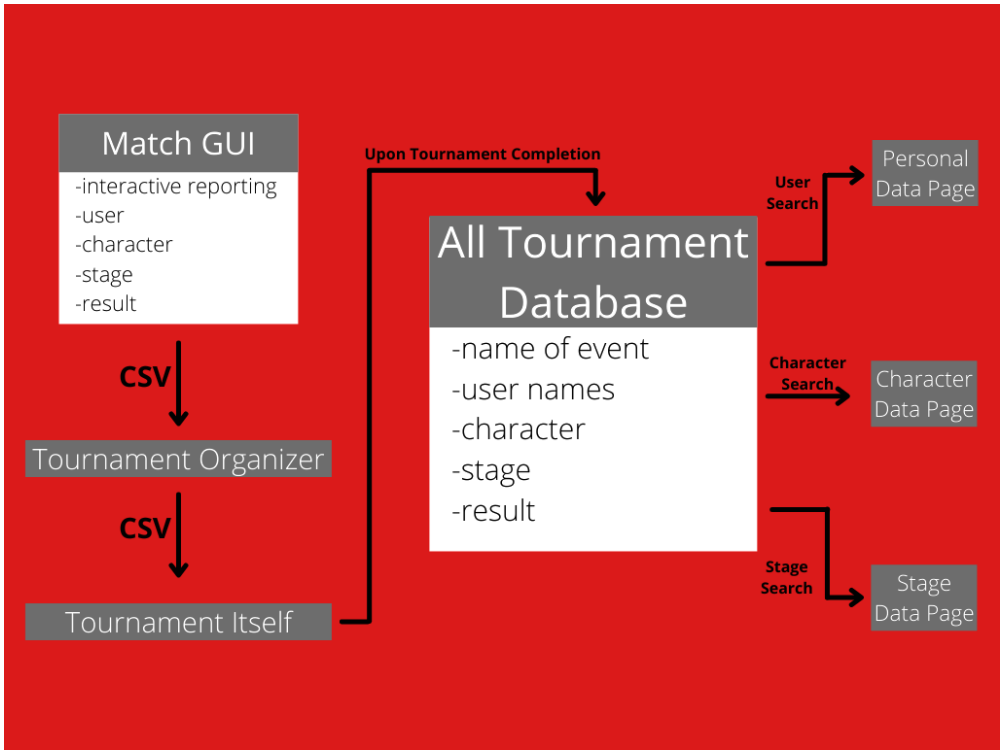
Dr. Kelley

CS-400-01

25 October 2021

Developing a Software Architecture

The architecture of our product, SmashApp affects its efficiency, utility, security, stability and maintainability all from underneath the surface. Deciding how one should partition its components, prepare for server requirements, and build its essential technologies are of the highest necessity. What this actually means though is communication—communication through team members on the most essential functions of our program, communication between component functionality to minimize overlap, and communication on how these components will be presented to the user. The first priority it seems then, before mapping out how components talk to one another, is to decide on what a component actually is. A component is a coherent function of a program. It's something that can be broken down into individual tasks, a gathering of smaller functions that will work in tandem to achieve a larger functionality with other groupings. So our app, SmashApp, is going to organize itself under a couple of headings.



First, we require user input. SmashApp requires a user interface first and foremost to record set data at the tournament itself. Internet connections can be spotty at events, so sending constant updates is going to be too stressful of an option, but collecting all relevant data on a single message first, then sending out a match summary at the end of the set would allow users to then find a better connection, as well as be forced to send full match details. While a GUI of character icons, and stage selection would be present to the user, a CSV file is really all that's needed to properly record all pertinent information about the set for later analysis. Upon selection of a character, stage, result, of the GUI, it simultaneously can be recorded into a text file for later manipulation. So while something such as this may be recorded on the GUI interface for a best of five set,

Match Up	Stage	Result
Wolf vs. Mario	Pokemon Stadium	1-0
Wolf vs Mario	Yoshi's Story	1-1
Wolf vs. Mario	Final Destination	2-1
Young Link vs Mario	Kalos Pokemon League	3-1

the CSV file would record:

```
Mitch, Sam, Wolf, Mario, Pokemon Stadium, 1-0
Mitch, Sam, Wolf, Mario, Yoshi's Story, 1-1
Mitch, Sam, Final Destination, 2-1
Mitch, Sam, Kalos Pokemon League, 3-1
```

The CSV file would then be sent to the tournament organizer, automatically inputted and confirmed, but as a bare minimum could be sent as a text message or email and manually inputted. This structure would lead to a reliable, available nature as well as a quick responsive time due to the client-side nature of its storage, as well as proper security. The main concern would be sending the information. How to communicate with a tournament's preferred website host, dealing with internet availability in spotty places like convention centers, and recording lost data in the event of a crash.

But this is going to result in an absolutely incredible amount of data—there's hundreds of weekly tournaments in the United States alone. So the database we create has to be optimally formatted for storage, a mediocre or bad structure for how data is inputted and retained will slow the entire service down tremendously, but to our knowledge at the moment, retaining information using a SQL database is the best route of action. Though this database is where the main appeal of the app is going to come from, the manipulation of this dataset is going to create personal and theoretical proof of gameplay tenants. A player is going to be able to see just how good a character is on a specific stage, a player is going to be able to see other players' biggest flaws in terms of stage, character, or person to person match-up. The main appeal to hook players in and use the entirety of the platform is this reward of, "If I knew this information, I could make a better decision, and then I'd finally win."

That concludes the minimum product we'd like to produce, so let's step into the side of things more social media based. The capstone of the social side is the group pages. Nurturing a healthy local group page is essential to growth on the platform, this is the hub for players to find one another, to find events, and discuss with one another on a forum style posting. It's where all the drama is going to be, all of the ridiculous opinions people deserved to be ridiculed for will be, and it's the place competitors are going to be able to bond with new and old faces. Having roles like moderator for users and self-policing is going to be the only way to effectively maintain group order and postings.

A group can be made for many reasons. The simplest grouping is going to be ones based on location. This could mean a group for Indianapolis smash, and then another group for the entirety of Indiana smash, as well as ones for smaller cities like Terre Haute or Bloomington. You could have groups based on character, you could have groups of players who just like each other, there's many reasons a group could exist, so the functionality of them has to be adaptable. The main concepts are going to be the event pages, text posting, images, as well as commenting. The main tenant of this side of the app is responsiveness. People need to be able to communicate with one another quickly, they need to have a place to build anticipation for the next event, they need a place to share jokes and memes, it doesn't have to be anything too fancy, it just needs to be easily digestible content. Now this portion of content is going to a cloud based service. This app is seeking to have a base in the United States, so having a cloud based service that scales based on what regions the app is popular in, time zone changes, and hotspots such as when a series of weekly tournaments all happen at the same time, or when national level events happen.

Though groups imply something else, individual user accounts. These user pages are collections of varying information: a player's biography, individual postings, character choice, and a record of all their personal gameplay data. These pages can be intense, but they're still manipulations of the main tournament database. A person's tournament history would gather all tournaments tied to their personal account, which would then produce different returns of data: their tournament placement, their win rate, their varying win rates across stage, player, and character. These accounts would also talk to one another, messages, group chats, these parts of the program shouldn't be centralized because our ability to out-compete other platforms is limited, but these functions are still essential to have as bare necessities of communication. It's a tricky balance, because this app is a combination of an entire social media site that already has a foundation of communities elsewhere, and to a degree, our app's growth is bound by smash's presence on other social medias, Twitch streams, and word of mouth. We simply aren't going to be able to make a communication service better than Discord, but at the same time, we have to prioritize having a private messaging service, which is an interesting divide. We have to create a serviceable product, like a Mexican restaurant having a kid's pizza. Hopefully, a lot of the technology here will be reusable, especially the text based communication forms. Commenting and posting should derive from the same methods, but be implemented in different places. The data analysis tool which sorts differing categories, personal, stage, character, while existing on different pages should all route from the same SQL prepared statements and Python code.

This is a lot of technology to build, and it's a combination of everything we've learned so far at Bellarmine and more. There's an initial client-side program that could easily be done in Java to report match scores, a data analysis project using a SQL database, which is then going to be manipulated through Python data analysis. Then after that, all of the data recorded and manipulated is going to have to be shown through most like Javascript, all of which's GUI needs to be designed, which is going to send our art department (Elizabeth) into a panic attack. This project is large, but everyone I've spoken to about this in Smash sounded very excited by the idea. The data analysis really hooks people in, from top players to newbies. This project is something very large, but ultimately could be such a great addition to the scene.