

Lightning Draft

Benjamin Sweedler

Senior Project
Computer Science Department
California Polytechnic State University
San Luis Obispo
2018

Abstract

Lightning Draft is a web application for drafting *Magic: the Gathering* cards. Users can visit www.lightningdraft.online to build a deck from randomly generated booster packs. This app was inspired by digital card games such as *Hearthstone*. Lightning Draft is a quick, fun, and simple alternative to drafting with physical cards.

Table of Contents

Abstract	1
Table of Contents	2
Introduction	5
Application	5
Choosing a Guild	6
Beginning the Draft	6
Drafting the Deck	7
Features During the Draft	8
Finishing the Draft	9
Background	10
Frontend	10
TypeScript	10
Component Based	10
Scenario Based Testing	10
Node Package Manager	10
Backend	11
Symfony	11
Routing	11
Controllers	11
Doctrine	11
Composer	11
Design	12
User Interface	12
Decklist	12
Flexbox	12
Responsive Webpage	13
Implementation	13
API	13
API Control Flow	14
Standard Use Case	14
Returning to a Draft	15
API Documentation	15
GET guilds	15

URL	15
Description	15
Returns	16
Example Response	16
POST drafts	16
URL	16
Description	16
Body Parameters	16
Example Body	16
Returns	16
Example Response	16
Example Card Object	17
PUT picks	17
URL	17
Description	17
Query Parameters	17
Example Query	18
Returns	18
GET drafts	18
URL	18
Description	18
Example Query	18
Returns	18
Database	19
Card Tables	19
Cards	19
Arts	19
Sets and Formats	20
Draft Tables	20
Draft	20
User	20
Player	21
Pool	21
Picks	21
Booster Pack Generation	21
Color Rules	21
One Extra Land Per Draft	22
Creatures are Preferred	22
Rarity	22

Reprints Are Not Favored	22
Related Works	22
Wizards of the Coast Products	22
Third-Party Magic Companion Apps	23
Cube Tutor	23
5 Color Combo	23
Mana Stack	23
Fantasy Football Apps	23
Digital Card Games	23
Hearthstone	23
Shadowverse	24
Google Sheets	24
Analysis and Verification	24
User Statistics	24
Colors	24
User Retention	25
Users Who Left Early	26
User Survey	27
Player Profiles	28
Card Algorithm Rating	30
Future Work	31
Icon and Logo	31
Sets and Formats	31
Uploading Cubes	31
Authentication System	32
Seperate Servers for Subdomains	32
Code Reuse	32
Dynamic Card Algorithm	32
Triple Picks	32
Conclusion	33

Introduction

Magic: The Gathering is a revolutionary trading card game, released in 1993. As the first trading card game, it bridged the gap between traditional card games like Poker and collectible trading cards like baseball cards. Fans were immediately drawn to this new gaming innovation. Since then, *Magic* has inspired many other games, including the relatively new genre of digital card games.

These digital card games leverage technology in a way that *Magic* has yet to embrace. Gameplay is accessible anytime. The user interfaces are clean and simple. My senior project is a web application that seeks to bring this experience to *Magic* cards. Compared to a traditional *Magic* draft, my app aims to be less *time-consuming*, have a lower *barrier-to-entry*, and have *clearer rules*. Quick, fun, and simple.

Application

Playing *Magic* involves building a deck of spells and mythical creatures. A booster draft is a popular method of building a deck. It requires a getting a group of eight players, opening booster packs and choosing cards. Although there is exist an official desktop application for this format, many players don't consider this option worthwhile. *Magic Online* charges an expensive fee for each draft, it has a confusing user interface, and it crashes often.


Lightning Draft is inspired by the draft format used in the most popular digital card games such as *Hearthstone*, *Shadowverse*, and *Elder Scrolls: Legends*. These games feature "arena" drafts that involve picking one card from a randomly generated pack of three cards. This simulates opening booster packs in a manner than is quick to play. My application allows *Magic* players to experience this arena draft format for the first time.

The following section will explain the user experience of visiting www.lightningdraft.online.


Choosing a Guild

Lightning Draft


Choose a Guild



Orzhov
Draft white and black cards



Azorius
Draft white and blue cards



Dimir
Draft blue and black cards

Decklist

0 / 40
Cards

©

The application begins by presenting a choice of three guilds. This choice decides what color cards the user sees during the draft. Having this choice upfront simplifies the deck building experience for both the user and the booster pack generation algorithm.

Beginning the Draft

Lightning Draft



Decklist

8 Plains

8 Islands

16 / 40
Cards

©

After choosing a guild, the user is presented with the first three cards to choose a card from. They also begin with a deck of 16 basic lands. This is considered a normal land count for a 40 card deck, which means the player has 24 left picks to complete the draft. The current card count is displayed in the lower right area of the screen.

Drafting the Deck

The screenshot shows the 'Lightning Draft' interface. On the left, three cards are displayed: Deftblade Elite (1 mana, Soldier), Advance Scout (1 mana, Summon Soldier), and Jeskai Windscout (2 mana, Bird Scout). On the right, a 'Decklist' sidebar shows the current deck composition: 8 Plains, 8 Islands, Flooded Strand, Erase, Sun's Bounty, Hand of Honor, Coral Barrier, Zephyr Scribe, Coastline Chimera, Myr Quadropod, Glassdust Hulk, and Invincible Hymn. At the bottom right of the sidebar, it indicates '26 / 40 Cards'.

Clicking a card adds it to the decklist in the sidebar. The card names and their mana cost are listed for each card. These cards are sorted by how much mana they cost to cast. fetches a new bucket cards.

Features During the Draft

Lightning Draft

Divebomber Griffin (3/2) - Creature — Griffin

Neurok Commando (2/1) - Creature — Human Rogue

Spirit Weaver (2/1) - Creature — Wizard

Decklist

- Erase
- Deftblade Elite
- Sun's Bounty
- Hand of Honor
- Coral Barrier
- Zephyr Scribe
- Alaborn Trooper
- Coastline Chimera
- Myr Quadropod
- Bident of Thassa
- Glassdust Hulk
- Tobias Andrion
- Invincible Hymn

30 / 40 Cards

You can [view this project on Github](#).
Made possible thanks to [MTGJSON](#), [SVG mana symbols](#), and [Visual Decklist](#).
Magic: the Gathering and all associated card images and names are Copyright © Wizards of the Coast
This website is not affiliated with Wizards of the Coast in any way.

A new bucket of cards is fetched after each choice. All choices are saved as the user drafts them. The user can return to the website's URL at any time to continue the draft where they left off.

Once the decklist grows beyond the screen height, a scrollbar appears on the sidebar column. Users can scroll through their past picks while keeping the current bucket on the screen.

Players can expand the card information in the decklist by hovering over the card names.

Finally, if the user hovers over the copyright symbol, a footer with external links scrolls up. This footer also includes a copyright disclaimer.

Finishing the Draft

Lightning Draft

Thanks for playing!

- You can take a [quick survey](#) to help me improve the app.
- You can print out your deck using [MTG Press](#).
- You can [copy](#) the decklist to your clipboard.

After the user's deck reaches 40 cards, the draft displays a grid of all the cards. The application thanks the user with these three links.



Thanks for playing!

- You can take a [quick survey](#) to help me improve the app.
- You can print out your deck using [MTG Press](#).
- You can [copy](#) the decklist to your clipboard.

The first link is a survey that I discuss later in this report. The second and third link are for helping the user play with the cards they drafted. They can either print the cards out on a printer, or export a text version of the decklist to a variety of third-party Magic applications.

Background

Web development has two sides. Front-end and back-end. This project uses Angular 2 for the front-end technology, and Symfony for the backend. All of the logic for organizing cards and drafts is done on the Symfony side, while the logic for displaying cards and moving elements around on the web page is done in Angular.

Frontend

My project uses Angular as a *frontend* web framework. The main job of Angular is to generate the HTML for a web browser to display. Here are some features Angular provides.

TypeScript

Angular uses TypeScript as its primary language. Angular then generates vanilla JavaScript from this TypeScript. One advantage to this is Angular is *platform agnostic*. Web sites created with Angular theoretically look the same regardless of the browser. Angular handles all of the details to display correctly on different browsers.

Component Based

In many frontend web frameworks, dynamic web pages are defined by templates. Each page has templated HTML that relies on variables or loops. A complex page may include nested templates.

Components take the dynamic abilities of templates a step further. Each component has its own template, its own stylesheet, and its own TypeScript file for behavior. This means the style rules and variable names won't interfere with another component's appearance and behavior.

This is similar to the Model View Controller design pattern that many frameworks enforce.

Scenario Based Testing

Angular Playground is an open-source tool for testing individual components. Lightning Draft has several sandboxed scenarios for testing how components look in certain states.

Node Package Manager

The node package manager (NPM) is crucial to managing projects in Angular. Libraries such as Angular Playground were installed through the NPM.

Backend

Symfony

As mentioned, the backend of my project is not written in Node. My project uses Symfony, a backend framework written in PHP. This backend was written as a generic MtG card database and draft tracker. This means both the Rotisserie and Arena types of drafts use this same backend to save user information and draft progress.

Routing

The frontend communicates with the backend through HTTP requests. These are handled by Symfony. The API routes follow a standard known as REST (Representational State Transfer).

Controllers

When the server receives a request at one of the API endpoints, Symfony runs a PHP function. Each function is part of a class that extends the Symfony Controller class. This design pattern is common in modern frameworks. It's known as the Model View Controller design pattern. Because this app only uses Symfony as backend framework, it only uses Models and Controllers.

Doctrine

The Models in Symfony are called Entities. The base Entity class relies on a library called Doctrine to persist information in the Database. Doctrine is an ORM (Object Relational Model). Lightning Draft only manipulates the Entity objects while Doctrine handles the connection to the underlying MySQL database. One advantage of an ORM is the programmer writes less SQL statements. Another is automatic creation of database tables and other resources, such as the API routes mentioned above.

Composer

Symfony uses a package manager, Composer, that is directly inspired by NPM.

Design

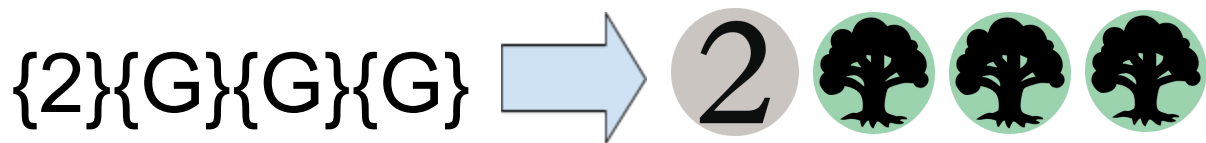
User Interface

Lightning Draft is based on *Hearthstone's* Arena draft.

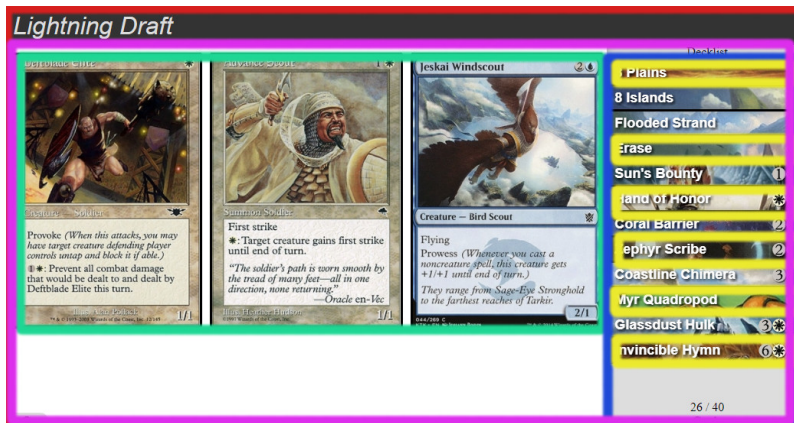


Decklist

The background for each card is a zoomed in version of the artwork. The mana cost symbols are SVG images. The symbols to display is based on a string representation of the mana cost.



Flexbox



All of the highlighted areas are styled with flexbox. Flexbox helps keeps the spacing between elements consistent. It defines which elements should grow and which should shrink when the page size changes.

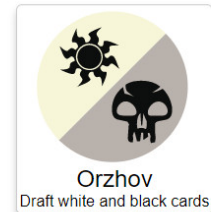
Responsive Webpage

Lightning Draft is functional on mobile web browsers. The screenshots show the page displaying correctly on both horizontal and vertical phone screens.



Lightning Draft

Choose a Guild



Decklist
0 / 40
Cards

Implementation

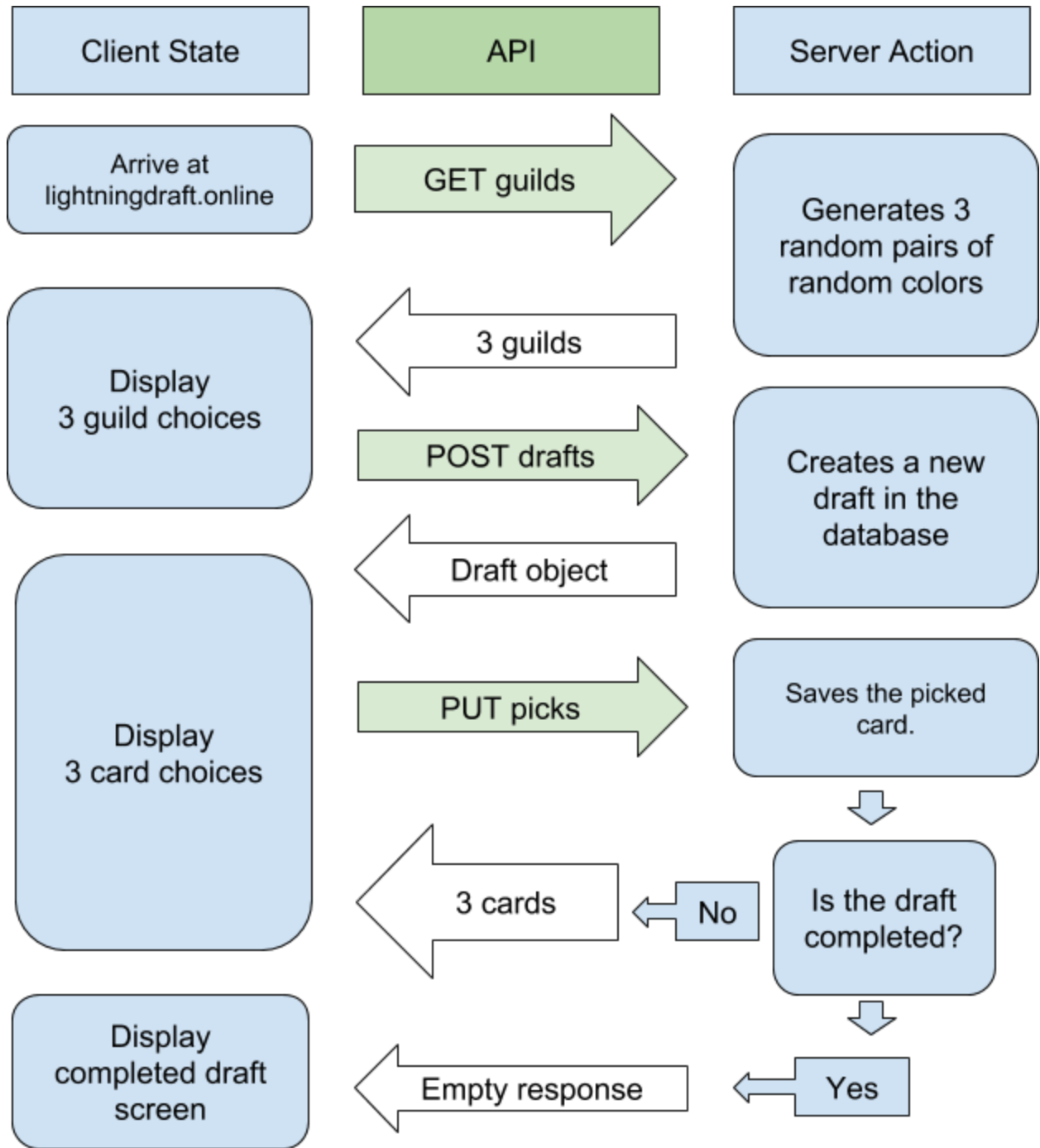
The source code for Lightning Draft is publicly available online at <https://github.com/pharaxe/lightning-draft>

API

Lightning Draft's API facilitates communication between the front-end and the back-end. The next section illustrates this communication. The section following documents the specification for the API.

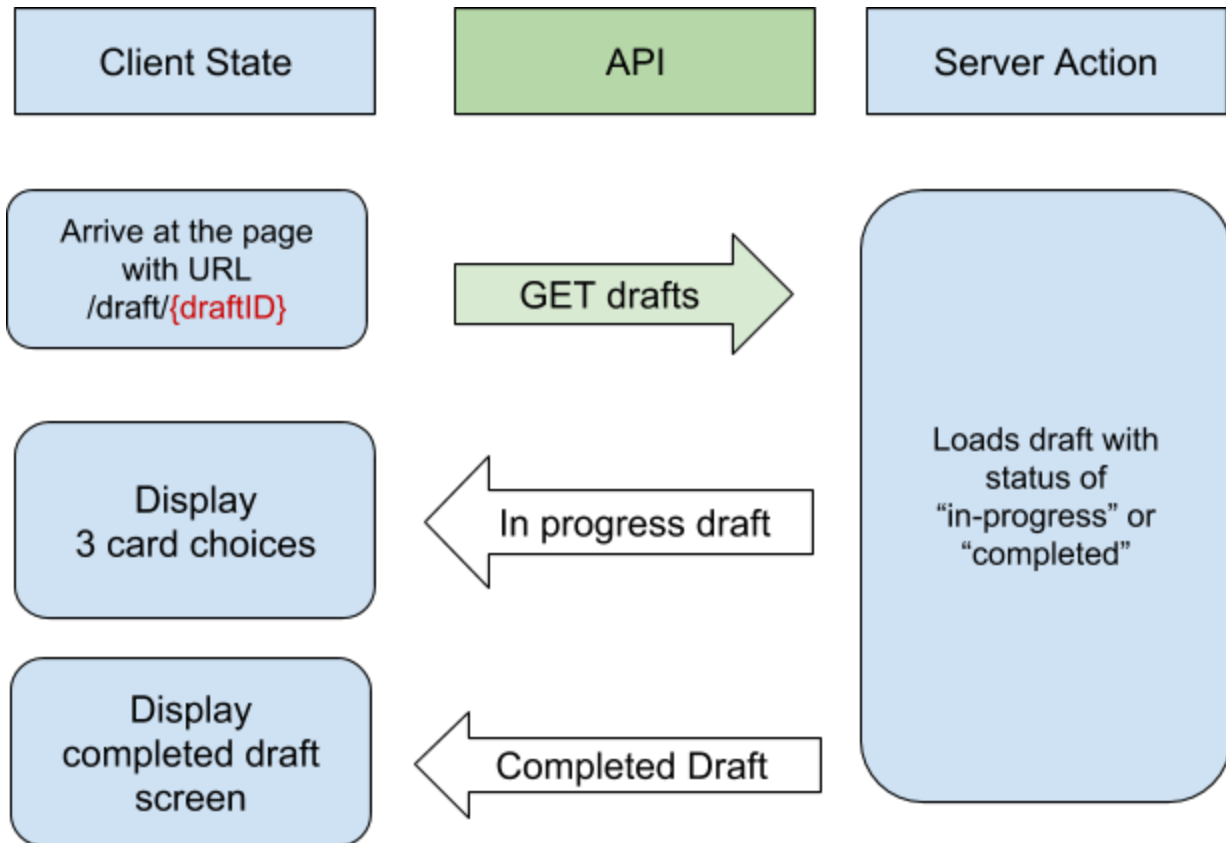
API Control Flow

Standard Use Case



Returning to a Draft

The application pushes a new URL to the browser history when the draft starts. Users can return to the draft by navigating to the url: `www.lightningdraft.com/draft/{draftID}`



API Documentation

The Lightning Draft API can be served over HTTPS.

All the queries have the base URL of `http://api.lightndraft.online`

All responses are serialized JSON objects.

GET guilds

URL

```
api.lightningdraft.online/guilds
```

Description

Gets three random color pairs for the user to choose from.

Returns

A two dimensional array of Color objects. Each pair of Colors represents a possible guild choice.

Example Response

```
{ "guilds": [
  [ { "name": "Blue", "id": 2}, {"name": "Red", "id": 4} ],
  [ { "name": "Green", "id": 5}, {"name": "White", "id": 1} ],
  [ { "name": "Black", "id": 3}, {"name": "Red", "id": 4} ]
] }
```

POST drafts

URL

```
api.lightningdraft.online/drafts
```

Description

Create a new draft, using the colors provided as the guild choice.

Body Parameters

colors: An array of objects with an id field to specify the color.

uuid (optional): A identifier to track a single user across multiple drafts.

Example Body

```
{
  "colors": [ {"id": 3}, {"id": 4} ],
  "uuid": "123e4567-e89b-12d3-a456-426655440000"
}
```

Returns

The newly created Draft object. Nested in this object is the Player object. Inside the Player object is a list of Cards for the first bucket, the Colors for the player, and a list of the Player's deck (which is empty for a new draft).

Example Response

```
{
  "draft": {
    "status": "running",
    "id": 203,
    "player": {
```

```

    "id": 203,
    "pack": [ Card, Card, Card ],
    "colors": [Color, Color]
    "picks": [ ]
  }
}
}

```

Example Card Object

```

{
  "art": {
    "id": 400
    "rarity": "common"
    "multiverseid": 1034
    "card": {
      "id": 533,
      "name": "Giant Growth",
      "cmc": 1,
      "mana_cost": "{G}",
      "url": "http://img.lightnindraft.online/1034"
    }
  }
}

```

PUT picks

URL

```
api.lightningdraft.online/drafts/{DraftID}/players/{PlayerID}/picks/{CardID}
```

Description

Send a card choice to the server, and then fetch the next bucket of cards.

Query Parameters

DraftID	The ID number of the draft.
PlayerID	The ID of the player. This number will be the same number as the draft ID in most cases.

	This is because Lightning Draft only supports single-player drafts at this time.
CardID	The card ID of the user's choice. This number should match one of the inner Card object IDs that was received from the API earlier.

Example Query

```
http://api.lightnindraft.online/drafts/30/players/30/picks/533
```

This would signify a user picking card number 533 for draft number 30.

Returns

On success, returns the next bucket of cards.

```
{
  "cards": [ Card, Card, Card ]
}
```

If the card specified in the query is not one of the player's valid choices, the response returns an error code of 500 and resends the current bucket of valid card choices.

GET drafts

URL

```
api.lightningdraft.online/drafts/{DraftID}
```

Description

This api call is used if the user refreshes the web page. It loads the deck and bucket data from the saved state on the server.

Example Query

```
http://api.lightningdraft.online/drafts/30
```

Returns

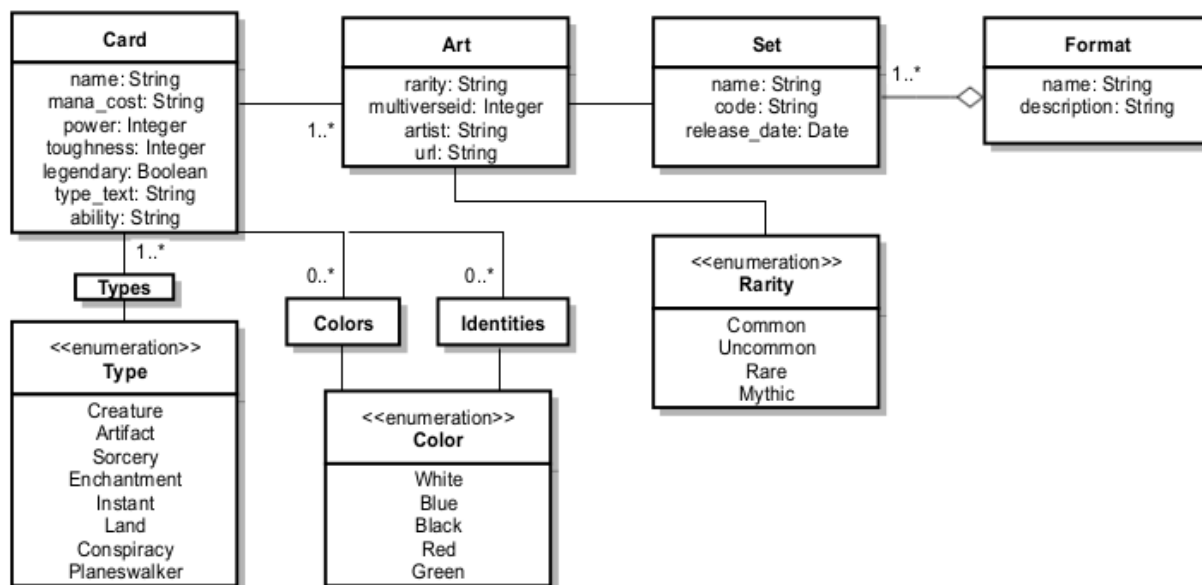
The draft object for the specified draft ID number. The response format for the draft is the same as the POST drafts API call.

Database

The MySQL database has around 16,000 magic cards, imported through a JSON file that can be downloaded from MTGJSON.com for free.

What follows are Entity Relationship graphs for the various tables in the database. Even when not explicitly in the graph, a joining table exists for relationships between entities.

Card Tables



Cards

Mana_cost: String describing the mana cost of the card.

Types: Each card can have many types. Most have a single type. This is used in the card algorithm, which prefers to present creature cards over non-creatures.

Colors: Each card can have many colors. Players are only presented with cards that include a color in their guild.

Identities: Similar field to colors. It includes colorless cards that have colored mana symbols in their abilities. The algorithm includes these as valid cards.

The rest of the fields represent various text information on the card. They are not used in the algorithm currently, but are being stored for their use in future features.

Arts

Each Card has one to many Arts. An Art entity represents a specific printing of a card.

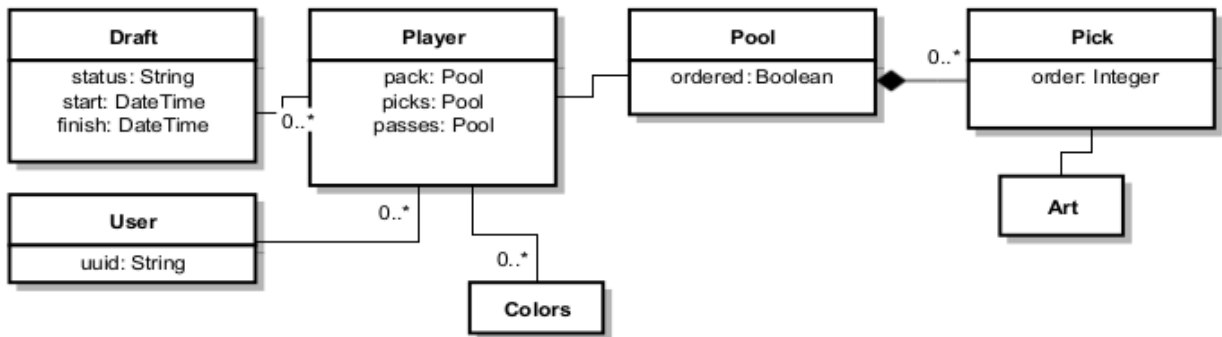
Multiverseid: this field directly identifies the URL for the image of the art.

Rarity: different printings have different rarities. Can be common, uncommon, rare, or mythic rare.

Sets and Formats

A Set is an expansion of Magic cards, which typically includes 200 to 300 cards. A format is a group of Sets. Eventually, Lighting Draft will include a feature for drafting cards from chosen Sets and Formats.

Draft Tables



Draft

In the current version of this app, the Draft Entity is only for keeping track of meta information related to the draft.

Status: Can be in one of three modes: beginning, running, or completed.

Start: The time the draft was created.

Finish: The time the last pick was saved.

User

Eventually, login information would be stored in this table. For now, it only has a single field.

UUID: unique string that the app generates on the front-end. Used to track a user across multiple drafts.

Player

Pack: the bucket of cards the player can choose from.

Picks: the deck of cards the player has picked so far.

Passes: all the cards from previous packs that the player did not pick.

Pool

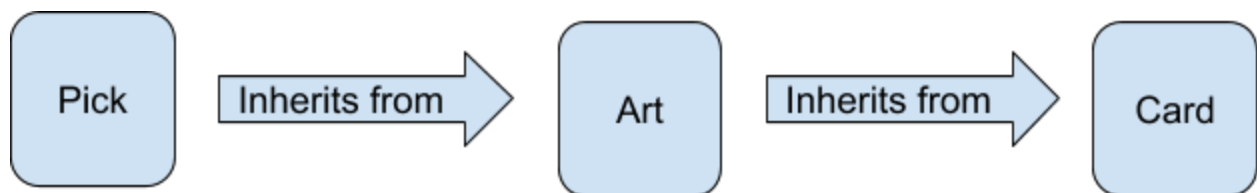
This entity is a joining table for Players and Picks. It represents a list of cards.

Ordered: a flag for if the list should be sorted.

Picks

A joining table between Pools and Arts.

This table creates an inheritance relationship between Picks, Arts, and Cards.



- A Card entity represents a particular Magic card.
- An Art entity includes all the information of a Card entity, plus the rarity and artwork of a specific printing.
- A Pick entity includes all the information of an Art entity, plus a reference to the Player and the Pool it's included in (the deck, the bucket, or the passed picks).

Booster Pack Generation

In most draft formats, booster packs include all sorts of cards that might not work together. This works because a booster pack has 15 cards, there is room for plenty of extras. My application has packs of 3 card choices at a time. This means all 3 cards ought to fit in the user's deck. This is why having the user choose a *guild* is a necessary design decision.

Color Rules

The algorithm excludes all cards with colors different than the chosen guild. There is also a 50% chance per pack to allow colorless artifact cards, which fit in any deck.

One Extra Land Per Draft

Each deck begins with 16 basic lands. They also get to pick 1 extra non-basic land. This non-basic land pack happens in between picks 5 and 20, at a 10% chance per pick until pick 20, when it's guaranteed.

Creatures are Preferred

The booster pack is all creature cards 60% of the time. The other 40% of packs are a combination of creatures and non-creatures cards.

Rarity

The algorithm generates packs so each of the 3 cards have the same rarity.

Common	60%
Uncommon	25%
Rare	10%
Mythic Rare	5%

Reprints Are Not Favored

The random card generator pulls data from the Cards table, not the Arts table. This means even if a card has been reprinted several times, it will not be more likely to show up in a booster pack.

Related Works

This section involves looking at the current market for drafting and card game applications. Some of these applications were inspiration for my project.

Wizards of the Coast Products

Magic: the Gathering Online (known as MTGO) was released in 2002. From a user interface perspective, it seems several years behind the current market of digital card games. It's favored by professional players and avoided by casual players. This application has booster drafts, but nothing similar to an arena draft.

Wizards acknowledges the disconnect between this app and *Magic* fans. They have a new client in beta, called *Magic Arena*. Currently, there is no draft mode in this application either.

WotC also has plans to release phone applications. This shows at there's a market space available for *Magic* software products.

Third-Party Magic Companion Apps

Fans continue to breathe life into Magic by creating companion apps.

Cube Tutor

Cube Tutor is a website for uploading a Magic cube, a personalized collection of around 450 cards that a player maintains for drafting. Cube drafts are popular among all levels of players.

5 Color Combo

5CC is a phone application for deck building and single-player mock drafts. Old sets can be drafted for free, and the paid version includes cards from the newest releases.

Mana Stack

There are many third party deck-building websites. Mana Stack is one such site. These are for maintaining and organizing a player's decks.

Fantasy Football Apps

Fantasy Football takes advantage of web and phone applications to make drafting more approachable. Many of these apps include features for setting up a room, sending out email invites, and then being notified when it's a player's turn to draft. Lightning Draft aims to incorporate some of these features eventually.

Digital Card Games

Magic has inspired many digital cards games.

Hearthstone

Hearthstone focuses making their drafts enjoyable to newcomers. Arena is a single player draft where the game presents only 3 cards at a time. This is much easier than the 15 card booster pack in a *Magic* draft.

Shadowverse

Shadowverse has a twist to drafting where two pairs of cards are presented at a time. The player must consider which pair to pick. This shows there are unique drafting formats yet to be explored.

Google Sheets

One unique draft in *Magic* is a Rotisserie draft. These types of drafts are generally tracked using spreadsheets. Even so, this process can be long and confusing. This is another situation where software could affect the quality-of-life for *Magic* gameplay and organizing.

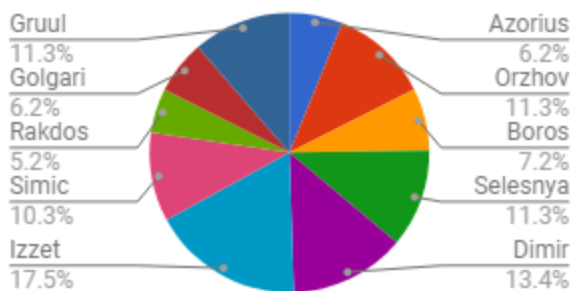
Analysis and Verification

The following graphs are based on data collected from 68 different users who drafted decks on Lightning Draft. This data excludes all drafts run by the creator of the application.

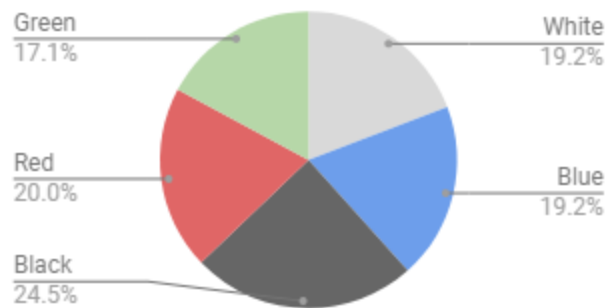
User Statistics

Colors

Guild Choices



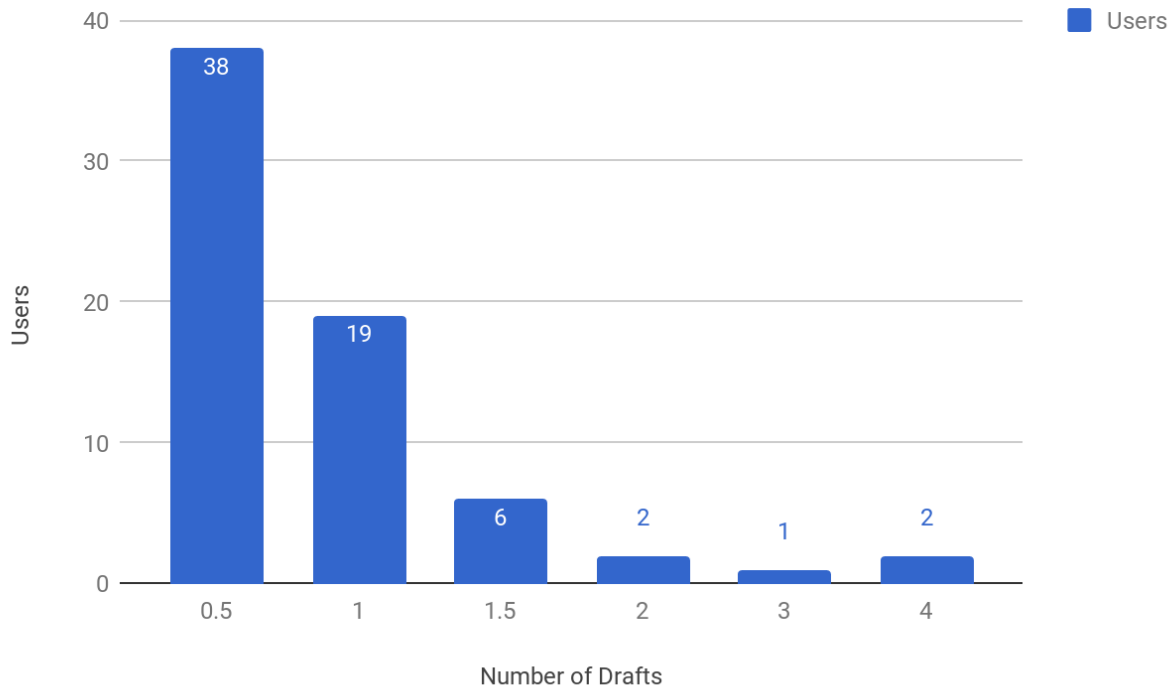
Color Choice Breakdown



The Red/Blue guild, Izzet, was the most commonly drafted. Considering all the colors overall, Black was only slightly favored at 24.5% and Green was slightly unfavored at 17.1%. Color choice in *Magic* often comes down to personal preference, so this breakdown is expected to be about even.

User Retention

Drafts Completed



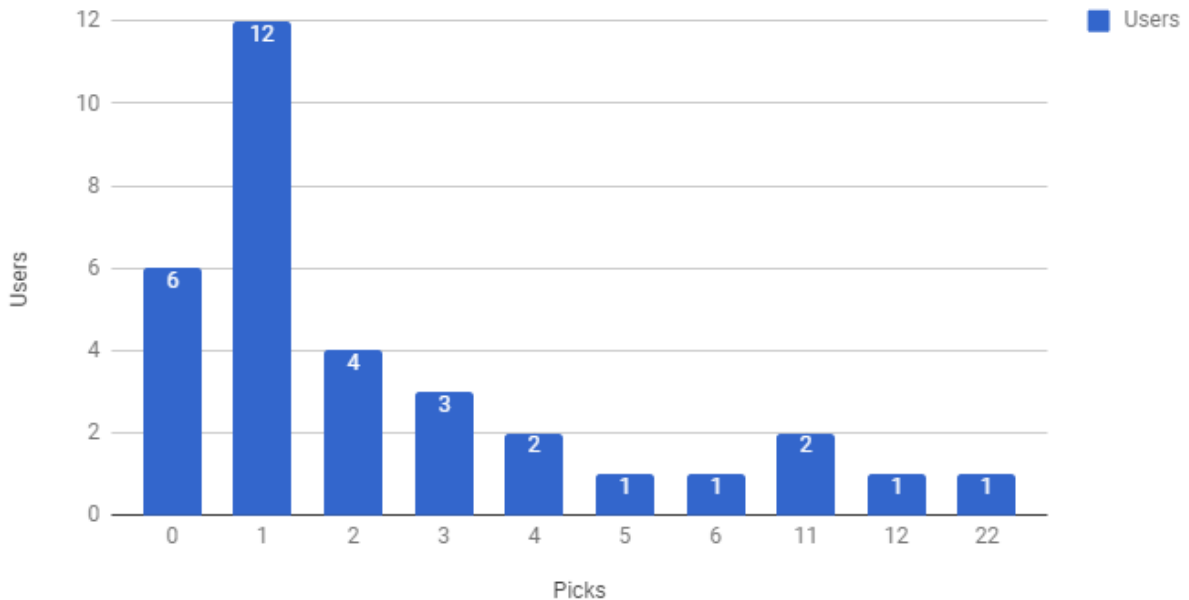
In some of the drafts, users started a draft but did not complete a deck. In the figure above, this is represented by .5 drafts.

This data shows 38 users never finished building their decks. The remaining 30 users completed at least one draft.

Users Who Left Early

Picks Completed Before Leaving

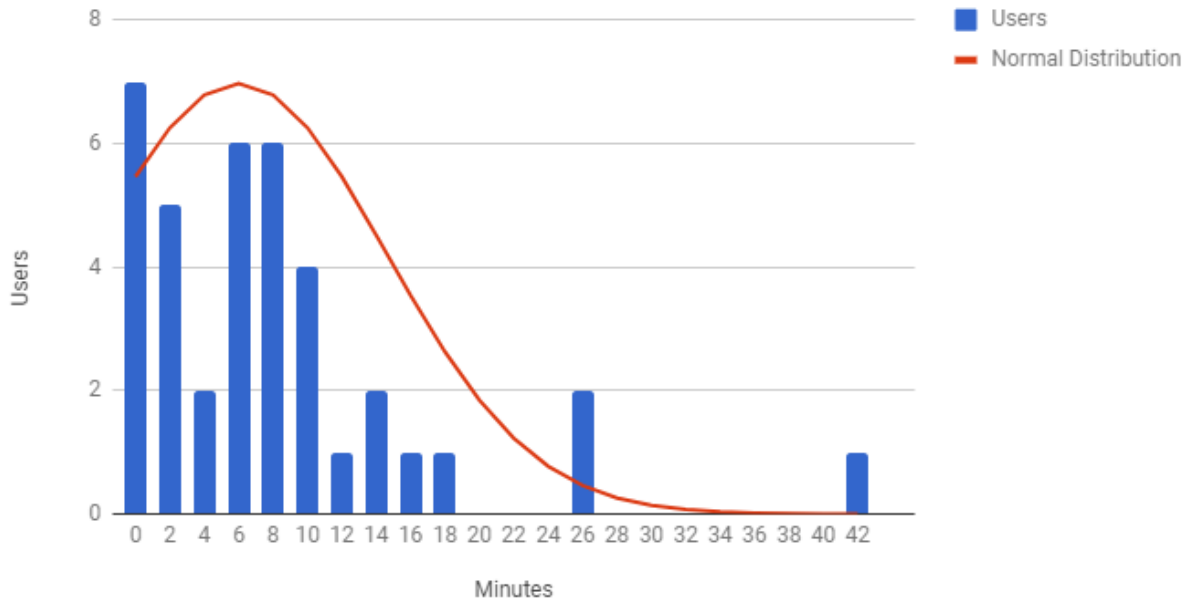
For users who did not finish any drafts



Of the users who did not complete a draft, 6 left after picking a guild. 12 more picked only a single card before leaving the website. More draft data is necessary to determine why this happened. One reason may be a need for clearer instructions during the first two picks. Another possible reason could be users who were mainly motivated by checking out the user interface, and not necessarily visiting the project to build a deck.

Time Spent on Drafts

For users who finished drafts



7 users completed their draft in less than a minute. It's possible they picked more or less randomly. The average time for completing a draft was 7.7 minutes. Three players took over 20 minutes.

A traditional MtG booster draft can take upwards of an hour for drafting and deck building. In all cases it is faster to build a deck using Lightning Draft.

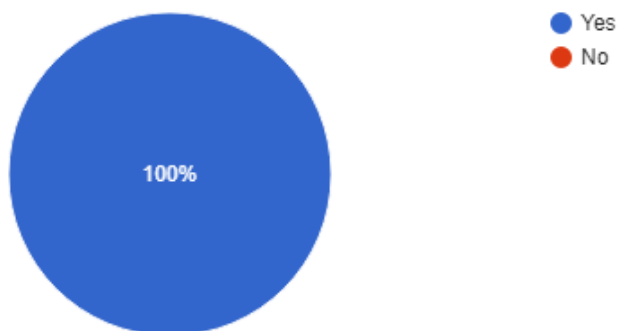
User Survey

On completing a draft, Lightning Draft prompts users to take a survey in order to rate the app. 8 users responded and what follows is an analysis of the results.

Player Profiles

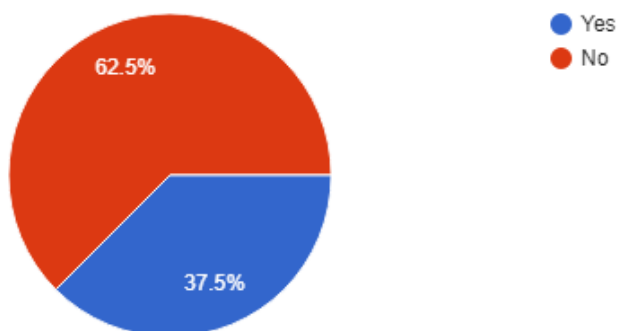
Do you play Magic?

8 responses



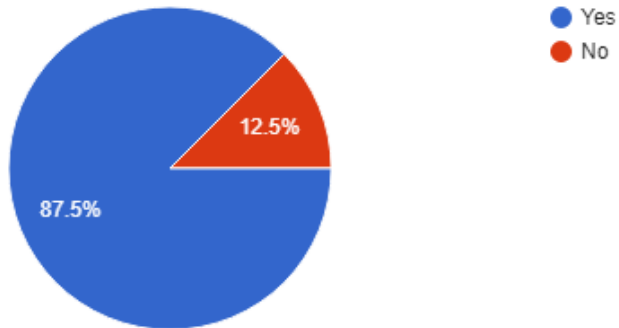
Do you play Magic Online?

8 responses



Do you play any virtual cards games such as Hearthstone, Shadowverse, or Elder Scrolls: Legends?

8 responses

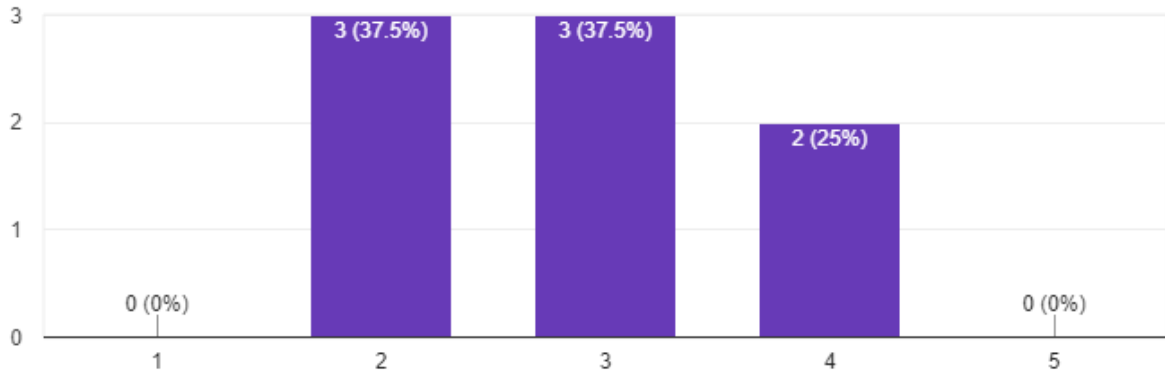


Lightning Draft was created for *Magic* players to experience to draft format that digital card games offer. The users who took the survey match that intended audience. They were *Magic* players, who overwhelmingly played digital card games, but did not play *Magic Online*.

Card Algorithm Rating

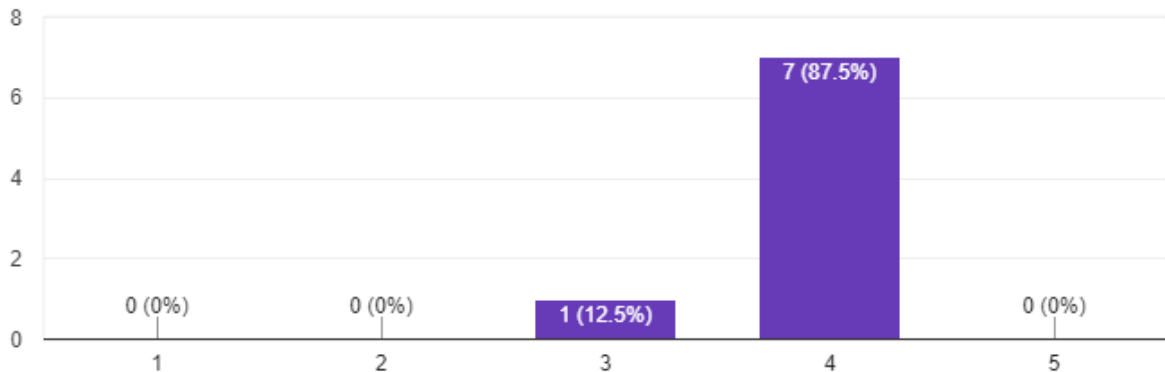
How powerful do you think your deck turned out?

8 responses



How meaningful did the card choices feel?

8 responses



These questions may have been too ambiguous to gain insight from. But even with the random card pool, users felt they got to make meaningful decisions.

App Rating

How would you rate the app in these categories?



Attached was an optional short answer question that gave two insightful responses:

“Mostly just cleaning up the card pool to allow going for themes would be good.”

“Cards were kinda random.”

These responses signal experienced Magic players want more strategic decision making. The users thought the app was undeniably quick, but was only a decently fun experience. There is potential to improve the booster pack generation algorithm.

Future Work

Icon and Logo

Lightning Draft could use a lightning bolt icon.

Sets and Formats

If the card choices were limited to specific sets or formats, then deckbuilding might be more interesting for the players compared to random cards.

Uploading Cubes

This feature could possibly interface with *Cube Tutor*. Players might be more willing to build decks using Lightning Draft if the cards are from their personal collection of owned physical cards.

Authentication System

This would allow them to revisit their old drafts. Currently they can look at their browser history to find old drafts, but this is not ideal. Registered users could also allow for multiplayer drafts.

Seperate Servers for Subdomains

- api - communicating with the backend
- www - serving the Angular app
- img - serving card images.

Currently, all three subdomains are on the same server. A more scalable option would be to have separate services for each subdomain. The images could be hosted by S3, which is Amazon's image hosting service. The frontend could be hosted on Cloudflare, which can protect against DDoS attacks, malicious or otherwise.

Code Reuse

Lightning Draft is a potential a building block for future applications for new or existing card games. The backend Lightning Draft is built upon includes a generalized implementation for keeping track of drafts. There is a possibility of writing future applications for *Magic* using this same system. And because there is an API for Lightning Draft, it's possible that other developers could interface with this backend, and build their own applications.

Dynamic Card Algorithm

The database for the app keeps track of the cards that users pick, as well as do not pick. For instance, a card called *Merfolk Looter* was picked 5 times. This card is one that users liked. Another card, *Crookshank Kobolds*, was shown to users 7 times but never picked. Obviously users did not like this card. This information could be used to improve of the booster pack generation.

Triple Picks

Even though Lightning Draft was less time consuming than a regular draft, many users did not finish their drafts. It's a lot of time to spend on a website, even if it's not a lot a time to draft. 23 seperate card choices is still a lot to ask of a user.

One idea to fix this would be to present mini packs of three cards each, and then present three of those packs at a time. The packs would be a theme. This would help the users create a deck with more synergy. It might also speed up the draft by a factor of three, as instead of 23 choices there would be eight.

Conclusion

Lightning Draft gives the experience a traditional *Magic* draft in a quick, fun, and direct way. The users of Lightning Draft found it less time-consuming than a regular *Magic* draft. Also, the survey responses support the app being decently fun. But, the usage statistics show that 25% of the users left the webpage during the first few picks. One conclusion is the application needs clearer instructions. Another is the app needs to set the user's expectations directly during the beginning of the draft.

Lightning Draft is an example of a Angular web application that communicates through an API. The API provides an abstraction for the complex underlying database. The frontend application only needs to know minimum information about this implementation. This allows the potential for other applications to interface with Lightning Draft in the future.

Lightning Draft is a new addition to the gaming world. There has not been an arena-style draft for *Magic* up until now. With digital CCGs becoming more popular, there's space in the market to create this sort of application for existing or new card games. Hopefully, Lightning Draft serves as an inspiration for new types of games.

Disclaimer

Lightning Draft is unofficial Fan Content permitted under the Fan Content Policy. Not approved/endorsed by Wizards. Portions of the materials used are property of Wizards of the Coast. ©Wizards of the Coast LLC.