CP476 - Final Project Documentation - Mackenzie Dang

Implementation Documentation:
Web-based YuGiOh was designed to implement an interactive deck builder and play-testing area for players of all varieties to have access to online testing resources that Konami does not provide themselves. Especially in the COVID-19 times, the functionalities provided by Web-based YuGiOh are extremely useful. These functionalities include:

- Interactive Deck Builder
- Local Deck Import/Export to txt file
- User Logins For Personalized Deck Saving
- Online Deck Saving in Database
- Card Information Poppers
- Interactive Drag-n-Drop Play Testing Board
- Automatic Random Hand Generation
- PHP/jQuery based Chat Room
- Cached decks that persist through sessions

The interactive deck builder allows for users to search for any card within the card database and have results appear in the Card List section of the application. The card database is retrieved from an external API which an overview can be accessed from here: https://yugiohprices.docs.apiary.io/#. This functionality is the basis for the entire web application as everything relies on the user making their deck.

Local deck import/export allows the user to save their current deck to a text file at any time, or load a previously saved text file into the deck builder. This functionality is available without requiring a login, and is made specifically to accommodate those who do not wish to save their decks online.

User logins and signup are fundamentals for allowing personalized data to be shown. Using a PHP based login, we work with mySQL to create a login system that allows the user to save and load decks from the database. Security involved with the login system is fairly standard, using parameter binding to prevent SQL injection attacks from occurring, as well as using the PHP password_hash method to hash passwords into the database. The PHP password_hash method follows the following steps:

1. Select a new random salt each time it is called
2. Applies a costly hash function that takes the input of the random salt + password
3. Combines the algorithm parameters, random salt, and hash into an output string

This prevents passwords from being leaking in an event of a database leak. On the following page, one may see the schema generated from the database used for the web application:

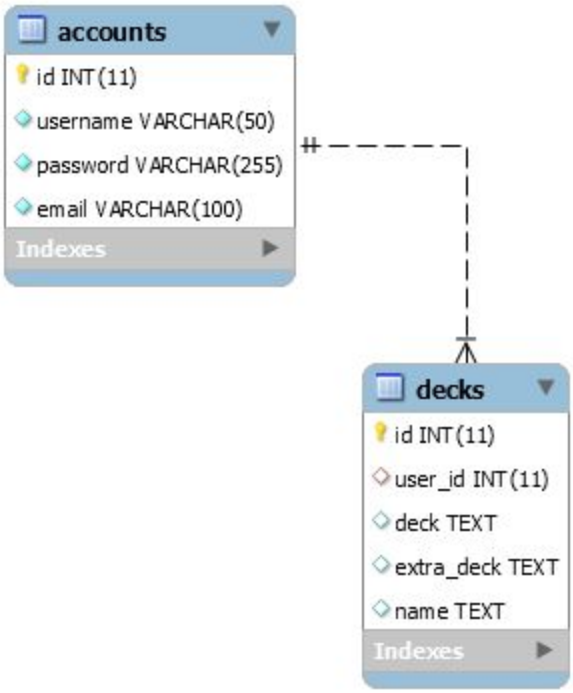

Figure 1.0: Database Schema for Web-Based YuGiOh

The schema is fairly simple since most of the data comes from an external API. The only two tables needed are accounts and decks because that is the only data that is required to be saved.

Card information poppers makes use of an external JS library called poppers, which implements the functionality to provide information popups anchored to any HTML element. This is extremely useful because it allows for each card to have individual full length descriptions just by clicking on the card. This allows an efficient way to present a large amount of information that the user may need access to.

The testing board is the main area where users can use their currently loaded deck and simulate playing using the predefined play area. This makes use of drag and drop events to allow for full interactivity between HTML objects meaning the user can drop from one square to another, simulating actually playing with the cards.

Automatic hand generation is a simple functionality that takes random cards from the users loaded deck and displays them as a hand. Although simple, this is extremely useful for YuGiOh players because this allows one to test probabilities using the deck one has created which is one of the biggest factors for determining whether or not one has made a good deck.

The chat functionality is a PHP/jQuery based chat room that pulls on a server-sided text file to retrieve messages on a set interval. This is useful for allowing users to interact with one another and provide deck feedback.

Technologies/Frameworks/APIs used in the web application include:

- BootStrap 4.0
- AJAX
- jQuery
- JavaScript
- PHP
- Popper
- YuGiOh Prices API

Bootstrapping the project was a design decision in favour of standardization and modernization. Using bootstrap simplifies the whole process for designing a web-page and gives users a standard fluid design that is widely used. It allows for a responsive design to be made adjusting to any screen size along with browser compatibility for all modern browsers. Bootstrap provides extensive documentation for any implementation issues. On top of all that, bootstrap implements design templates for different typographies, fonts, forms, buttons, tables, popup modals, carousels, etc. all of which take a significant time to implement otherwise. My familiarity with BootStrap is high as I have used it in many projects beforehand. This is the exact reason that led me to choose this specific framework for CSS design, because I have had first-hand experience with how simplified BootStrap makes a project's design.

AJAX is a natural choice when working with any XMLHttpRequest objects, which is required to work with the YuGiOh Prices API, as well as working with PHP asynchronously. This means that it is possible to use AJAX to send PHP data requests without requiring the user to refresh the page. This is very useful when working with PHP because often sending data with PHP requires navigating to a new page and sending a header back. I was previously familiar with AJAX because it is commonplace for working with external APIs in JavaScript, and this influenced the decision to make use of AJAX because of my experience with its usefulness and irreplaceability.

jQuery should be an essential library in any complicated web application that requires JavaScript. jQuery takes many tasks that may take many lines in traditional JavaScript and wraps them into methods that can be called in a single line of code. jQuery is the foundation of many other libraries as well so often it is impossible to work without jQuery if one decides to use another JavaScript library. jQuery selectors and the sorts allow for simpler and more efficient access to elements, and allow for the full interactivity by handling the multitude of different user events such as drag, drop, click, key presses, etc. The reason for jQuery over other frameworks that are rising in popularity like React, is because jQuery is fast, small, and feature-rich. jQuery allows DOM interaction and manipulation extremely easily because it directly accesses DOM, which is much more convenient in a single-page web application than virtual DOM that many newer frameworks use.. React and others are better at handling states or updating information. I was previously familiar with jQuery, and chose jQuery because of the reasons mentioned previously.

JavaScript should require no explanation. It is the basis of all user interaction, page states, element interactivity, etc. It is the de facto language of the web and almost no page on the internet would function

properly without it. I was familiar with JavaScript, and because of this know that it is irreplaceable in terms of creating a web application.

PHP was a difficult choice to settle upon, but the justification for using PHP is that the MVC architecture in PHP allows for easy code maintenance, the stability provided by the language which has been in existence for a long time, as well as the easy implementation with mySQL databases that streamlines database management. I was not previously familiar with PHP, and used this as an opportunity to create a web application using one of the older server-side languages that is still widely used. The main reason the implementation was not done with Node is again, due to myself being previously familiar with Node and wanting to learn PHP since it is still so popular amongst the swath of server languages these days.

Popper is a no compromise JS library that allows for lightweight DOM-less access to popcorn displays with no delay. There are very few established libraries that provide the same functionality and customization options like popper. Using popper, it allows for the application to provide popcorn displays for card information which would otherwise be too much to fit in just the card element itself. This way the user is not limited to information they can access, and the design remains sleek. I was not previously familiar with popper, but after reading the documentation and seeing the uses provided to the application it became an obvious choice to implement.

The YuGiOh Prices API was used due to there being no publicly available official API provided by Konami for card information. With YuGiOh Prices, the database is kept constantly updated with new set releases and because the website is used by many players to calculate prices. Due to the constantly updating dataset, and vast information that provides for extensibility options for the future, using this API became an obvious choice. I am previously familiar with YuGiOh prices in general but have never worked with their API extensively, but I came to choose their API because I had knowledge of the information they provide on their website and how useful it would be towards this web application.

Usage Documentation:

Upon loading the web application, the main screen shown can be seen below:

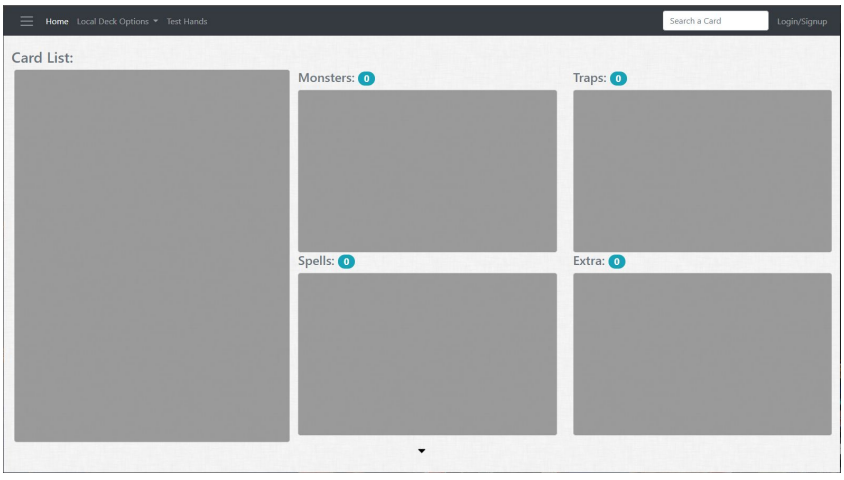

Figure 2.0: Main landing page for Web-based YuGiOh

On this page, one is able to create a deck using the search bar in the upper right corner. Search results appear in the “Card List:” Section on the left. Cards appear in a list of names corresponding to the lexicographical distance to the search characters. For example typing “Zo” into the search bar will display results such as:

Figure 2.1: Search results on search query “Zo”

Where as searching for the query “Zood” will result in:

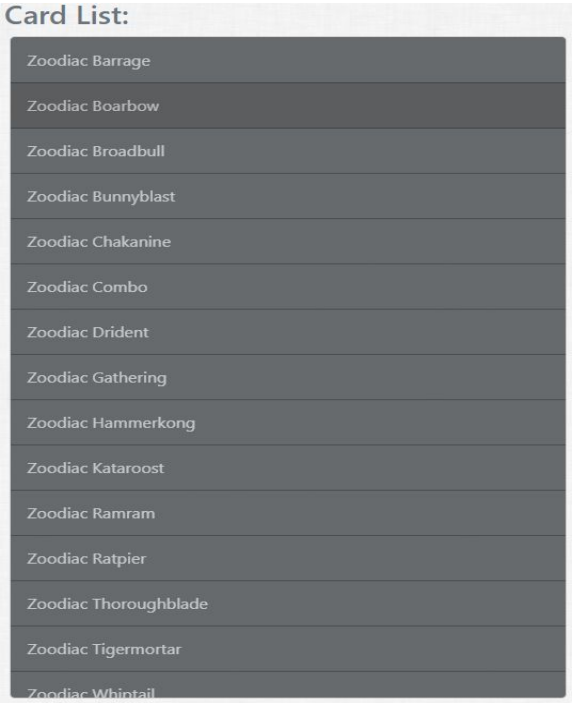

Figure 2.2: Search results on search query “Zood”

Left or right clicking on any card shown will open a popper with the card information:

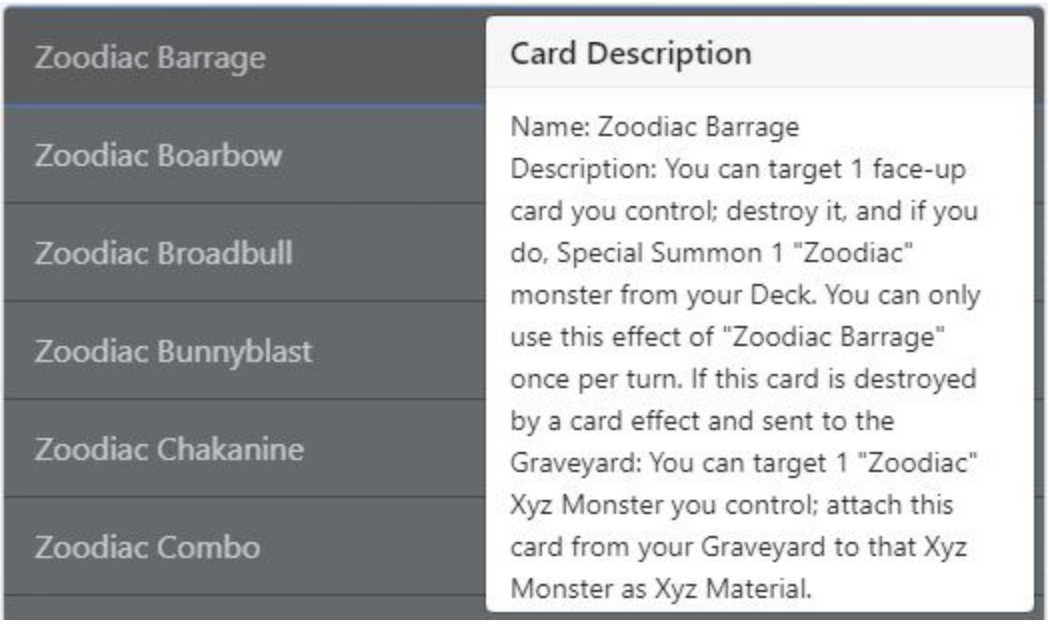

Figure 2.3: Card Information Popper

The popcorn display is scrollable and provides in-depth information about the card selected.

Specifically right clicking on a card in the list will place the card into one's deck. The card will then appear on the right depending on the type of card it is. For example, right clicking on "Zoodiac Barrage" which is a spell card will appear in the spell section on the right:

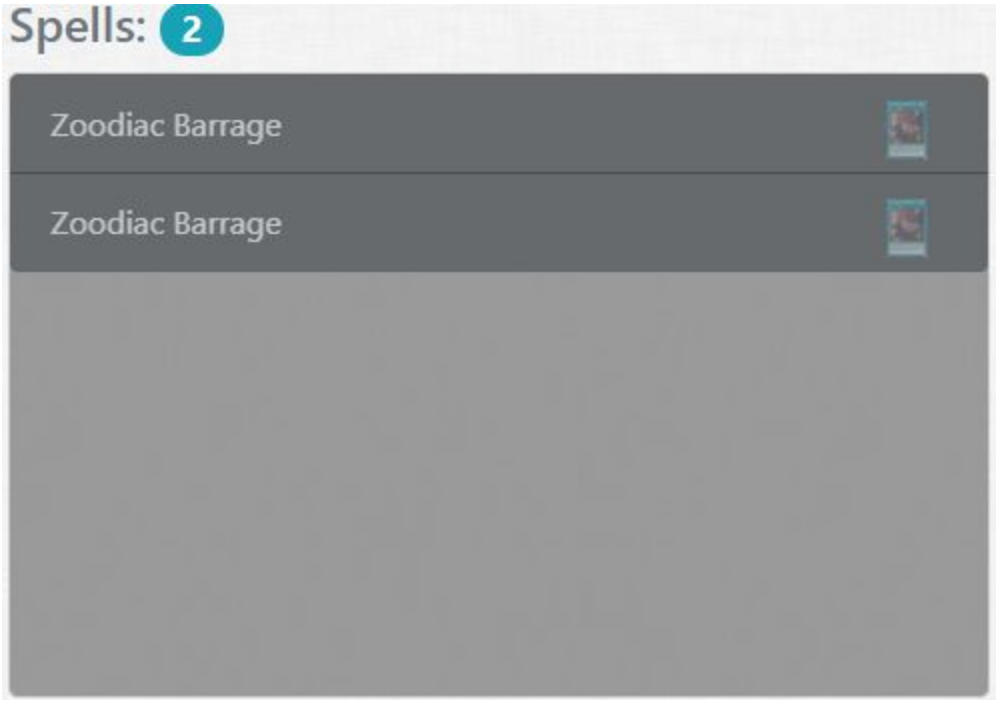

Figure 2.4: Deck Builder

The user may double click on any card in their deck to remove it from the deck.

Once a user has finished creating their deck there are several options that can be done. The user can decide to save their deck to a local text file using the local deck options link in the navigation bar:

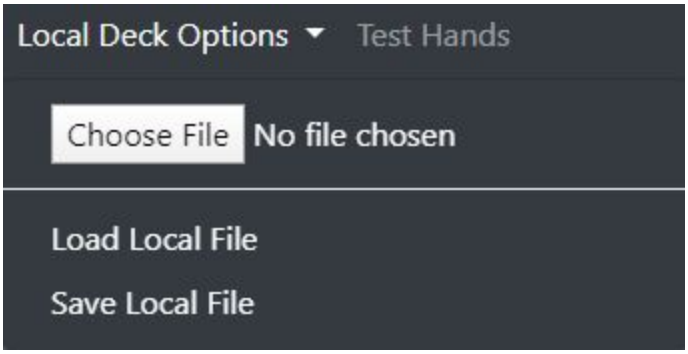

Figure 2.5: Local saving option

Which provides users who are not signed in the ability of saving and loading decks locally. The user may also click on the Test Hands link in the navigation bar which can be seen in figure 2.5. This will open a modal with a generate button in the bottom right corner that can be pressed to generate a random hand to test hypergeometric probabilities with one's created deck.

Figure 2.6: Test hand generation modal

The user also has the option of signing in to unlock the online save and load options for their deck. By pressing the login/signup link in the far right of the navigation bar which can be seen in figure 2.0. A login modal is displayed. This is a tabbed modal which means that both registering and logging in are displayed in the same modal.

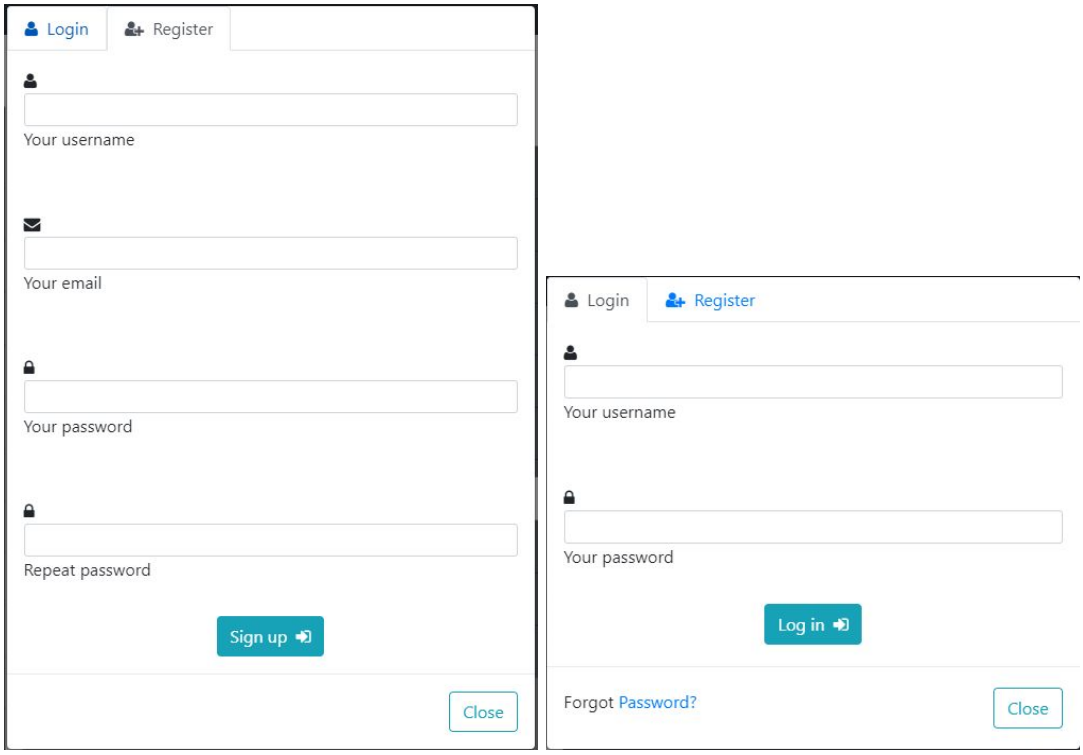

Figure 2.7: Login/Register Modal

The user can sign in using a test login with the following credentials:

Username: test

Password: test

Otherwise, one may create a new account using the register tab. All information is validated on the server side using PHP, while it makes use of javascript to generate what feels like seamless error displays even if the page is required to be redirected.

Once a user is signed in the page will look like the following:

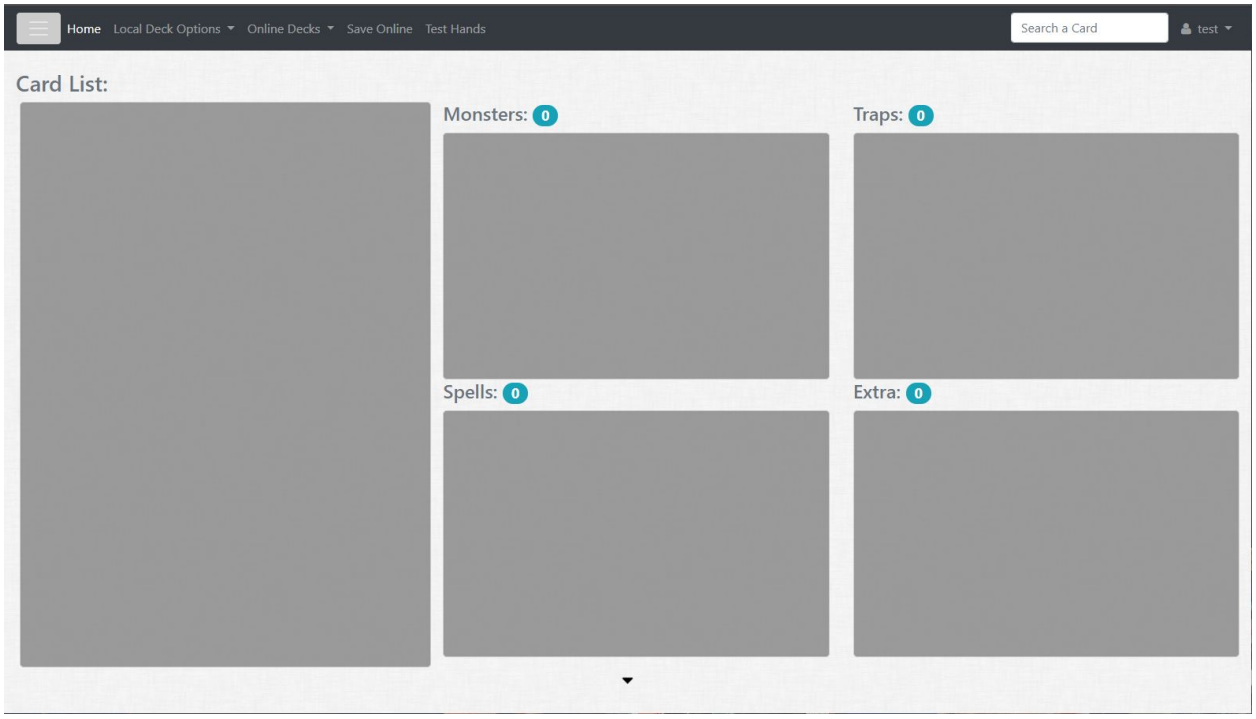

Figure 2.8: Signed in landing page

The user may notice that there are several new navigation options such as online decks and save online. Online decks will present the user with their list of saved decks, left clicking on any of these will import the deck into the deck builder, while right clicking will remove the deck from the database. The save online button will present a modal display with an input field to present a name for the deck in the deck builder and a save button. The save button will save the deck to the database under the users decks.

The main functionality that can be used upon all the deck creation and saving is the play testing board. The user can either scroll down manually to the board or press the animated caret down button that can be seen in figures 2.0 and 2.8 to be automatically scrolled to the play board.

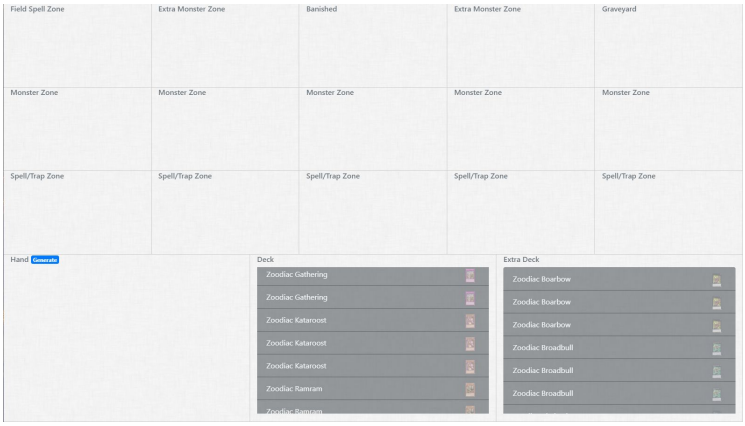

Figure 2.9: Play testing board

The testing board will automatically fill with the users created deck and extra deck. The user may press on the generate button in the hand section to take random cards from their deck and place it in the hand section. The user may also interact directly with any card element by selecting and dragging the element to their desired space of their choice.

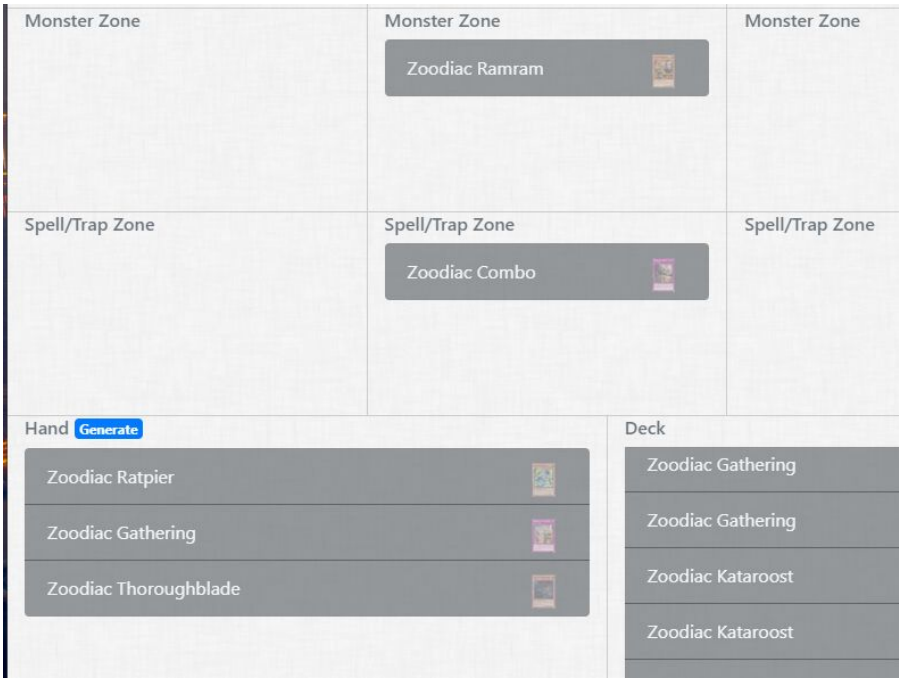

Figure 2.10: Moving card elements

The user may restart by emptying their deck in the deck builder.

To use the built-in chatroom one can press on the chat button in the navigation bar. The chat button will display a modal where if you are logged in you will have your username as your chatting ID, and if you are not then you will be listed as “Anon”.

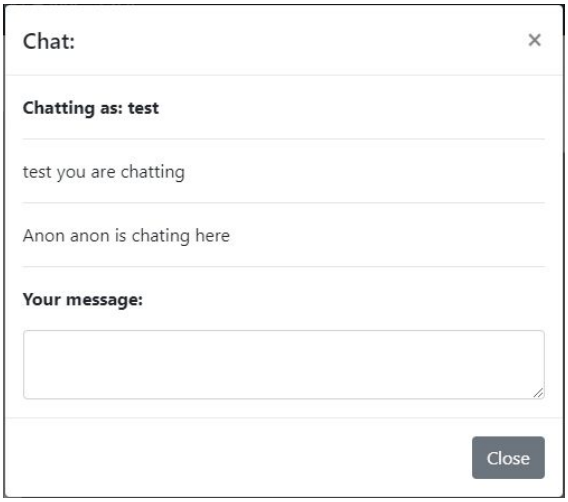

Figure 2.11: Chat Modal

The chat modal has a message box where one may enter any message they wish. The max length is 100 and upon completion of one's message, just press enter to send the message. The message will be pushed to any browser that currently has the web application open.

Experiences, Future Prospects & Inspirations:
During the development of this project I gained valuable experience in terms of integrating front-end with back-end for full stack development, making full use of JS and other libraries to enrich my knowledge in DOM related programming to provide user interactivity and dynamic elements, standardization and modernization of design using BootStrap, and well-founded PHP programming; all of which open new doors for future opportunities. The greatest challenge encountered among these experiences was definitely working with DOM so closely to provide an optimal user experience, as well as integrating full stack. This was also the most time-consuming part to implement, making sure that all the user interaction was error-free and that cards were placed exactly where the user wanted through multiple events was strenuous to achieve and without a doubt the most time consuming part.

In terms of extending the project, one feature that would be really useful for users would be the implementation of a share deck option that would allow users of the website to share their deck in a format that automatically imports into anyone that wishes to import their deck. This could be done by having a public deck page that has a list of all decks, with an import option. I would also like to change implementation frameworks if I were to develop further since the benefits of technologies like Node and React may provide for more functionality.

My main inspiration for creating this web application was an attempt at replicating and modernizing the currently existing application DuelingBook which can be accessed here: https://www.duelingbook.com/. This application is a flash based dueling application which has full online functionality to allow users to test, chat, and play with online cards. However, it is exactly as it seems; it requires other people to play with, whereas often when a player wants to test they only wish to test opening hands or how to practice a combo, etc. Also the implementation of DuelingBook runs on an archaic codebase based in Flash meaning that it is becoming obsolete soon in many modern browsers. Given more time and further implementation, a full modernized version of DuelingBook could be made from this web application and that was my inspiration behind starting this project.