CS701 - Mark Outcalt

Prof. Kalathur

Facilitator - Fazil Harroon

**Term Project - Magic: The Gathering Deck Manager**

**Project Code** - <https://github.com/mtoutcalt/CS701>

**How to Run**-

1) Download the Modules with: *npm install*

2) Start the server with ng serve and navigate to http://localhost:4200/ : *ng serve*

**Problem Statement** -

Magic: The Gathering is a strategy card game that involves a player creating a deck of cards, both creature cards and spell cards, and competing against their friend. Magic has been around for decades and there are over 15,000 cards that have been created. I wanted to expose an API that has card information so a user can search for cards, make their own deck of cards, and find resources for how to play the game.

**Application Layout**-

The application uses Angular routing, Bootstrap, and Angular Material Design for the layout and the look and feel. The navigation is at the top and corresponds to each Angular component. The Deck Builder component has a set of children navigation which corresponds to each step the user takes in building a deck of Magic cards.

Components/Pages

1) The **Home Page** has a Search feature so a User can look up cards in a Table. It uses the Card service to search by name.

2) The **Deck Builder** has a user form and then 3 stages of a user choosing a deck color, adding creature cards, and adding spell cards to their deck. Then it has a page that shows their selected deck based on HTML5 localStorage. The color uses SVG. The components use localStorage to persist choices made by the user and then show the cards in the deck at the end. It also checks localStorage to determine what to show since previous steps need to be completed first. The creature and spell components use the Card service and get 100 random cards based on the color chosen by the user.

3) The **Open Booster Pack** page lets a user simulate opening up a new pack of 15 random cards by clicking on the card pack image. It uses the Set service which takes one of the random ID abbreviations saved as a local object and then using a randomizer will call with that abbreviation.

4) The **Tutorial Page** has a list of videos to learn how to play and uses HTML5 video controls. The video element has mouseover event to play and mouseout to pause and will display the video src on the page.

5) The **Phases of Game** page uses HTML5 Canvas to simulate the steps in playing a game of Magic. It consists of 5 canvas drawings that can be either automatically shown with a timer or switched with buttons. It also has a local audio element that has controls to play ambient music.

6) The **Formats of Game** page uses Material Design cards to give info about ways to play. It is mostly just a static page.

7) The **Find Where to Play** page uses HTML5 geolocation and the Google Maps API to find the closest gaming or comic book store that sells Magic cards and gives driving directions to get there. It initially finds the user’s location but also allows any city to be used. The browser HTML5 geolocation coordinates are used on the page initialization and then chained into the Google Maps API call with lat and long to get the closest city. Then it uses the Google Map Directions service to lookup the closest hobby store to that city that sells Magic and displays both the map and the list of directions of how to get there.

**Integration of Project**

I’m using Angular for the project and each Page in the navigation is its own component. The Deck Builder Component then has a subset of children components. I’m using the Angular router and children router to link everything together. The navigation bar at the top is always present and uses Angular Material Design components for its style. Other pages uses either Material Design or Bootstrap styling.

**Code Organization**

The code is organized as a typical Angular app. Each Page described above (1-7) is a stand-alone component. The Deck Builder component has 5 child Components - deck-builder-colors, deck-builder-creatures, deck-builder-spells, deck-form, and deck-builder current. The routing was kept in the app.module.ts file. The model directory contains the Deck.ts data object used for the deck building form and it contains the services. The find-store.service.ts makes a http get call to the google geocode api. The magic-card.ts service.ts uses the mtgsdk-ts library and has 4 methods that takes in filter strings and returns Card Promises. It also has 2 methods that take filters and return Set Promises.

**Services**

Magic API:

-<https://docs.magicthegathering.io/>

-<https://github.com/MagicTheGathering/mtg-sdk-typescript> - Typescript library for the same API above

Data - the imported library abstracts the http/jsonp calls. The objects Card and Set are used to search with different queries. Some example service calls are - findCardsByID, findCardsByName, findCreatureCardsByColor, findSetByName. The service calls return Promises.

Google Maps API:

-<https://maps.googleapis.com/maps/api/geocode/json> - Geocode

-<https://developers.google.com/maps/documentation/directions/> - Directions

Data - I first get local coordinates from HTML5 geolocation and then call Google API giving it the latitude and longitude coordinates. My service call is findClosestGamingStoreWithCoords() and returns an Observable. I also use the DirectionsService with travelMode DRIVING which returns the result into a directionsDisplay.

HTML5 Geolocation - <https://www.w3schools.com/html/html5_geolocation.asp>

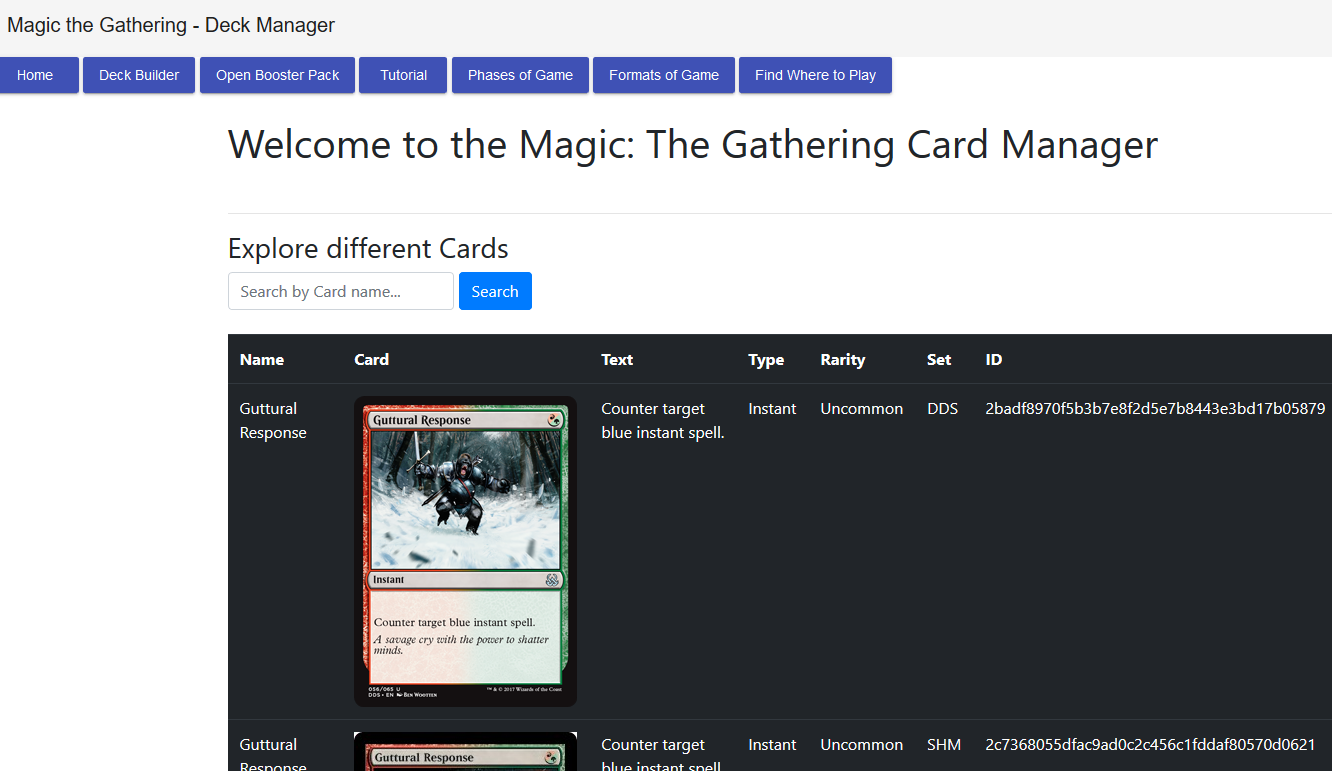
Data - I use the navigator.geolocation.currentPosition with a success callback that resolves my Promise. It also gets options with highAccuracy as true and a timeout as 5 sec. It returns a coordinate object which I then chain into the Google Maps call.

**Course Topics Used**

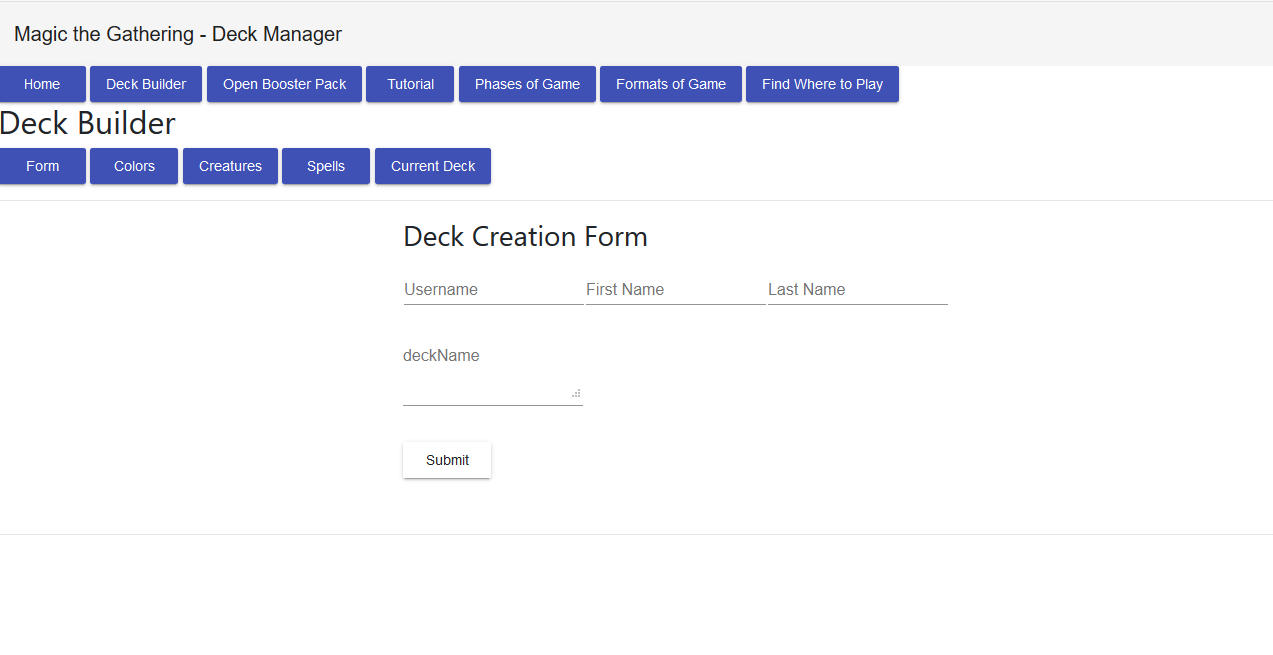
* **SVG** - for DeckBuilding I made an interactive graphic that displays the different colors used in a deck
* **Canvas** - I have a component that consists of a simulation of how to play a game with a background and lots of different shapes and text placed. The code consists of 5 different drawings that can be manipulated with timers and buttons.
* **Geolocation** - I am using the native HTML5 geolocation to get the coordinates of the user in order to find the closest Magic/gaming/comic store nearest to them without them needing to input their address.
* **HTML5 Audio** - For the Phases of The Game component I decided to use the audio element and controls to allow a user to hear an ambient song as they learn to play the game
* **HTML5 Video** - I am using a local video to learn how to play with video controls such as playing and pausing the video when the user hover over it or away from it.
* **HTML5 Form attributes** - The Deck Builder form uses spell check, placeholder, and autofocus features
* **LocalStorage** - the DeckBuilding uses localStorage to keep track of the deck cards that have been selected by the user and the deck name. It also checks storage to determine if previous data items have been set and uses Angular directives to determine what to display based on what data is set.
* **Angular and Typescript -** I chose to use the latest Angular and the entire application consists of Components, Services, and Routing. I also used Angular Material Design Components.
  + Routing - using routing and child routing for the DeckBuilder
  + Services - using angular Services for the Magic API and the Google Maps API

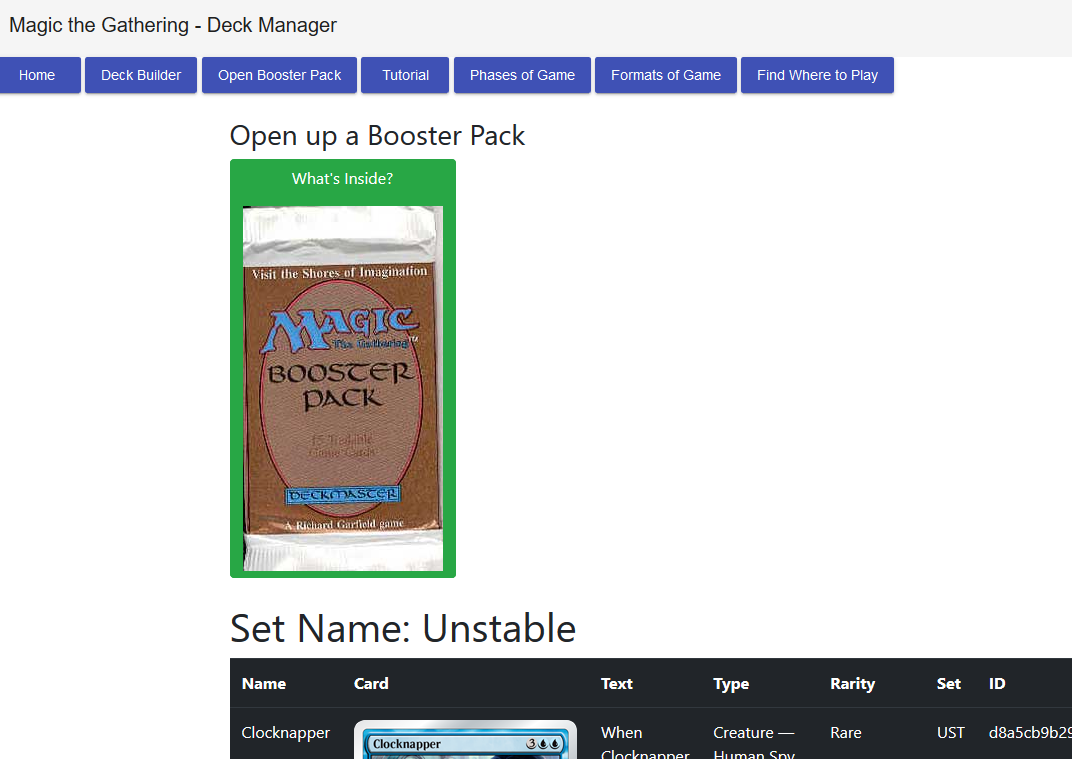
**Screenshots**

Home Page - Search

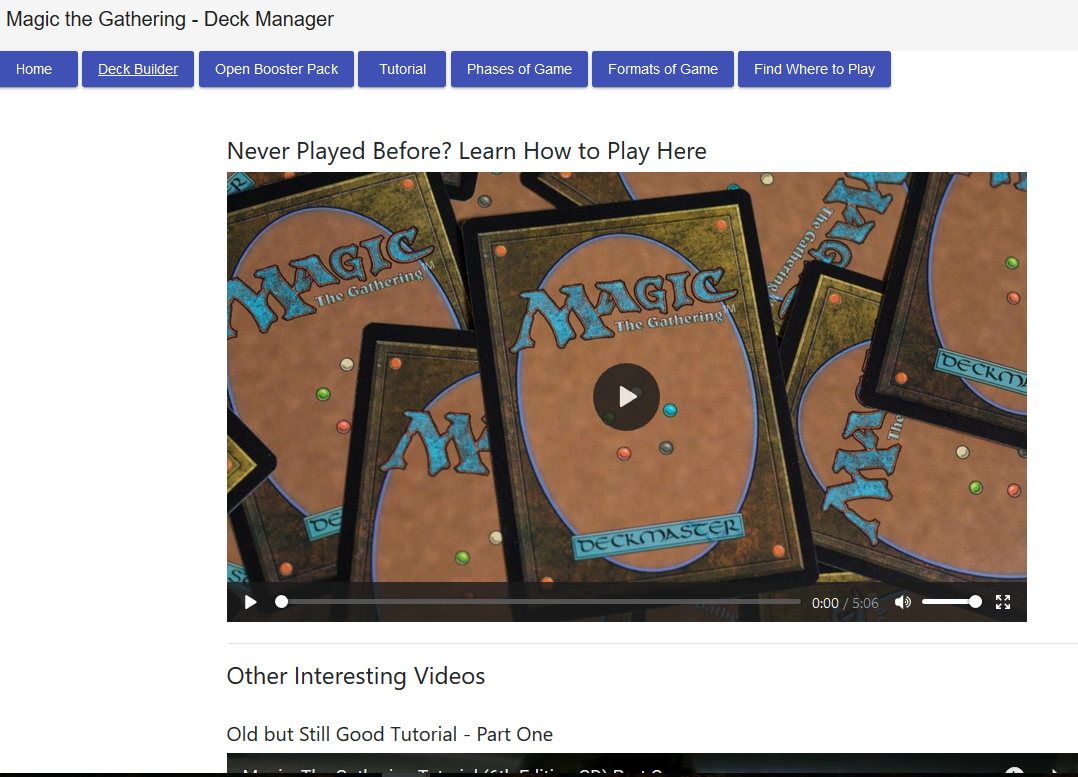


DeckBuilder Page

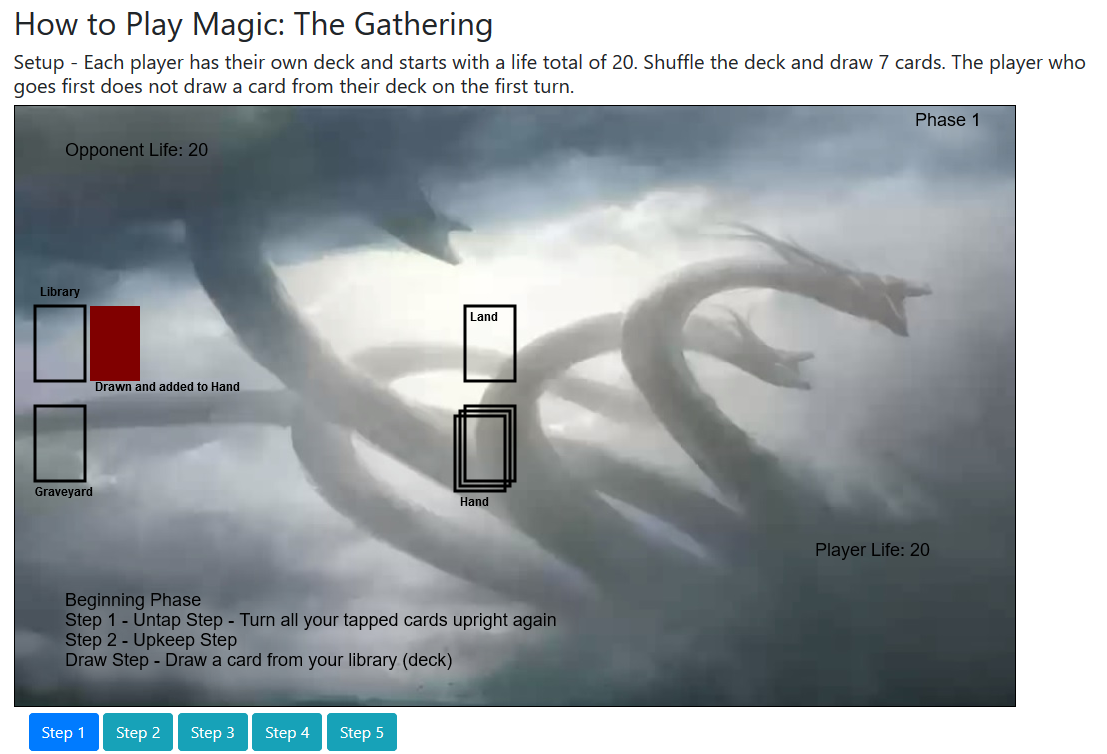
Open Booster Pack Page



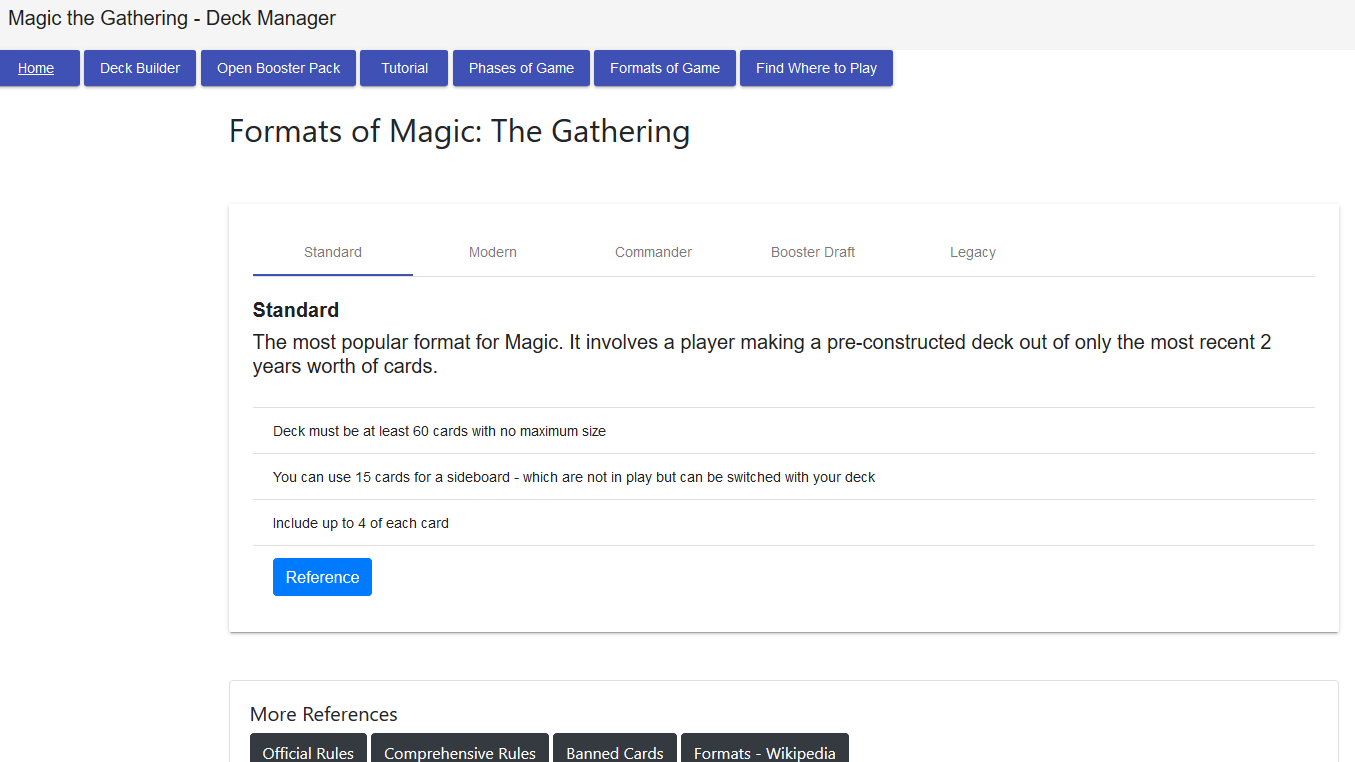
Tutorial Page



Phases of Game Page



Formats of Game Page



Find Where to Play Page

