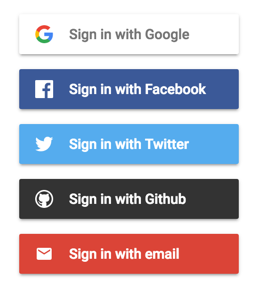
Carson Stevens

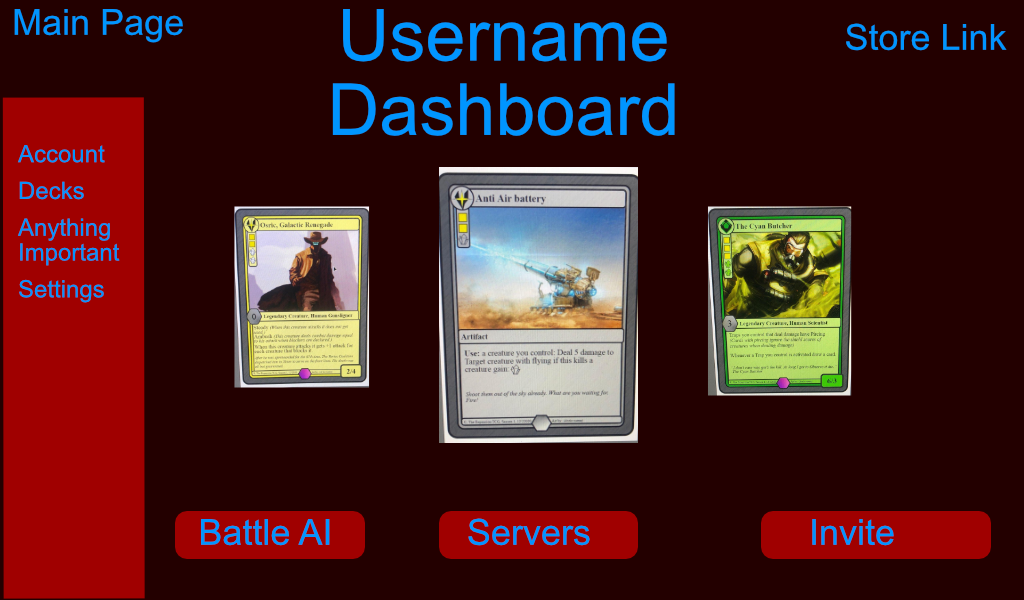
1. The website I will be prototyping is based on the idea a friend wants me to help him implement after graduation. It is a mix between a tabletop card game and an online card game. The incentive behind this idea is that there is a large market for tabletop card games such as Magic the Gathering, but there are some limitations like having to have a physical opponent there to play against. This fact has never been more relevant due to social distancing and quarantines. This is where the online platform game to the rescue. Players could now play games like Magic the Gathering using platforms like HearthStone and Magic Arena. Although this gave players a chance to play against a wide variety of players, the main group of players still gravitated towards physical tabletop cards. *There is just something about the feel of a physical card*. The other thing from a business side was that physical cards have an increasing value that brings customers into the hobby and keeps the game alive; online platforms lack this ability.

So the question is: How do we mix physical cards with an online platform? The technology lies in NFC chips which can be tiny, paper-thin, flexible, printer-friendly, stickers that store small amounts of data that can easily be scanned with a smartphone (approximately 144 Bytes; ~144 characters). When playing online, the card can be scanned by placing it on top of the user’s phone. A simple application interface would be included for scanning on the phone, but the main game is aimed to be played online which is why I will be focusing this prototype on the website and backend primarily. When a signed-in user scans a card, a request is sent to the server to initialize the card for the same user on the computer. Users can then play against each other with their physical cards while in separate locations. The NFC chips store a Primary Key that will link them to a database storing all the cards. There is a lot of potential within this, but possibilities include updating card attributes after release, letting both users clearly see and read the card that has just been scanned into play, validating moves and gameplay logic, and any other meta-information needed for play. The potential is limitless and in the end, will be up to the game designer (my friend).

The result is a tabletop card game that can be played in person, but when that is no longer a choice, the game provides a way for players to continue playing online. As briefly mentioned earlier, this project will be prototyping the website and backend and not the mobile app or game logic.



1. The first page a user will encounter is a login page. Although this could be implemented solely with the database, I’d opt for the Google Firebase option for login as a user experience benefit. Nobody likes having a thousand different logins. This also logins them in on all their devices which works perfectly for ensuring user pairing between mobile and desktop later. If the user declines/exits the login popup, they will simply enter the Game’s Home Page. On the homepage, a simple introduction to the game will be displayed. This may include things like the different card sets, the most recent game information, videos instructing users how to set up the cards they’ve bought, and a link to the marketplace where users can buy more cards (I will not be focusing on the e-commerce side since we already covered that). If a user logs in, instead of being shown the home page, they will be redirected to their dashboard page which will contain information such as all the cards scanned, user-designed decks, and other general user settings.

From the dashboard, the user can choose to play from 3 different modes. The first mode will be against an AI which will probably end up being a NN model that runs on a Python API backend we can then query with the current game information and return the CPU’s game choice. The next mode will be an open server that will match players. The matching process will probably be more involved than just finding to available players (such as matching some sort of ranking system for more balanced gameplay). Once two players are matched, a game instance will be instantiated on the server for those specific users. Similar to the mode above, any time a player scans a card or makes a move, a request will be sent to the database which will then send responses to the client-side to update the UI with the relevant information. The last mode is simply a more personal version where players can invite another player by the username to a game. When creating that request, a response invitation is sent to the corresponding user. The player will then be entered into a ‘waiting room’ page. If the invited user is logged in, the next time their page refreshes information from the server (not sure what a good update rate will be yet. I think that will have to be done through testing so we don’t eat up to much overhead in sending, receiving, and updating). When the user clicks to accept the request, they will join the opponent in the waiting room while any game variables and needed information are loaded. Once the game instance is loaded, a response will be sent to the clients with the gameplay page that will start their game. As before, the UI is updated any time a request is sent from a move or card scan, or server update.

1. The database will be composed of a few main tables. The first table will contain every instance of a card that has been built for the game. Updating values here will propagate to individual instances of cards. The card name can be used as a primary key and information that might be stored are the moves, stats, card modes the card is allowed to be used in play with, and other information that should stay consistent from card to card.

The next table will be the main user table. This will contain basic user information like name, email, preferences, and a Primary Key username. The other important thing that this will contain is the list of Primary Keys to the cards the player owns. When a user requests their cards, their updated information can then be pull from the card table. Any unique information that is different from card to card will be stored on the NFC side which eliminates the need for user card-specific database entries.

As mentioned above, there will also be a merchandise side to the website where users can purchase more cards. The database information would be setup similar to what we have done for the e-commerce site.

Basic commands will allow for updating a user’s information such as if a player scans a new card to be inputted into their deck. Other important things like updating user stats and contact information may be an important feature as the final pipeline continues to fruition. Things like deleting cards from a user’s deck may also be needed if a form of trading is implemented. The main card database will also need to have commands to add new cards and their relevant information as well as updating any card that already exists.

1. For technologies used in the class, we have the basic HTML and CSS framework for structure and style. jQuery would be used for basic animations, styling changes, basic navigation queuing for showing and displaying things, and applying event listeners to any DOM elements. The backend however would not be implemented in PHP. Because the project uses NFC and NN models, the obvious language of choice would be Python as it has the vastest ecosystem for processing sensor inputs and easily creating/deploying NN models.

So if not PHP, the language would probably be Python. The framework of choice will probably end up being Django as it is the most horizontally scaleable and versatile of the python web frameworks. The process is very similar to the MVC model though. The server has routes that accept different request methods that apply whatever logic is needed and then return the new template or data that the request was wanting. Another possibility that Python lends itself to is easy integration with the Electron Frontend Framework that applications like Github’s Atom and Discord are built on. Using a few commands, the website becomes compiled into an application that is a web server and servers the same HTML and CSS templates from the website. This would make this into more of an application, but the web framework lends itself perfectly to the Electron frontend.

I think the hardest part to implement will be keeping the UI updated with information without having the user do something that sends for a request. Setting an interval for the continual updating process seems inefficient and like it would apply a ton of overhead to the application that would slow it down. Another challenge will be the easy and efficient communication of scanning NFC information and updating the desktop application which falls into a similar category as the above. In contrast, the easiest part for me will most of the frontend and backend things. I’m not sure anything is really easier or harder than the rest for me. Having implemented NN APIs and websites in the past, the process is relatively familiar. The part that excites me the most is working on connecting NFC to IoT and making something that combines the two biggest sects of gaming. I’m least excited about working with someone not familiar with the computer science side of things; explaining what is needed and why will be a lengthy process. And on the flip side, fully understanding the game concept will be important for making sure that I am able to offer a bridge between idea and design. Things are much easier to design when I have my own vision in my mind.