Project proposal

# 1.Project name

Sellswords & Spellcrafts

# 2. Team member names

Mason Evans

Luke Schnetlage

Nicolas Towery

# 3. Abstract

We intend Sellswords & Spellcrafts to be a web-based 1v1 deck-building game. Users will connect to a central web server in which they can be matched with other users. We want to allow users to be able to take their turn, while the other user takes their turn once the first user has completed their turn. We hope to achieve this so as to allow each user to see their opponent’s moves or actions in real time. The central gameplay loop will follow 2 players drawing cards from their own randomized decks until they have a number of cards equal to the starting hand size. A random player will be given the ability to play first, after the first turn, players will alternate turns until one player’s life total has been reduced to zero or have zero cards in their library. With the gameplay mechanics in mind, we will implement a system which allows users to better understand the rules of the game and provide helpful tips as needed. Our hope is to provide users with a unique gaming experience and to support multiple diverse play-styles including aggressive, controlling, “midrange”, and other niche options.

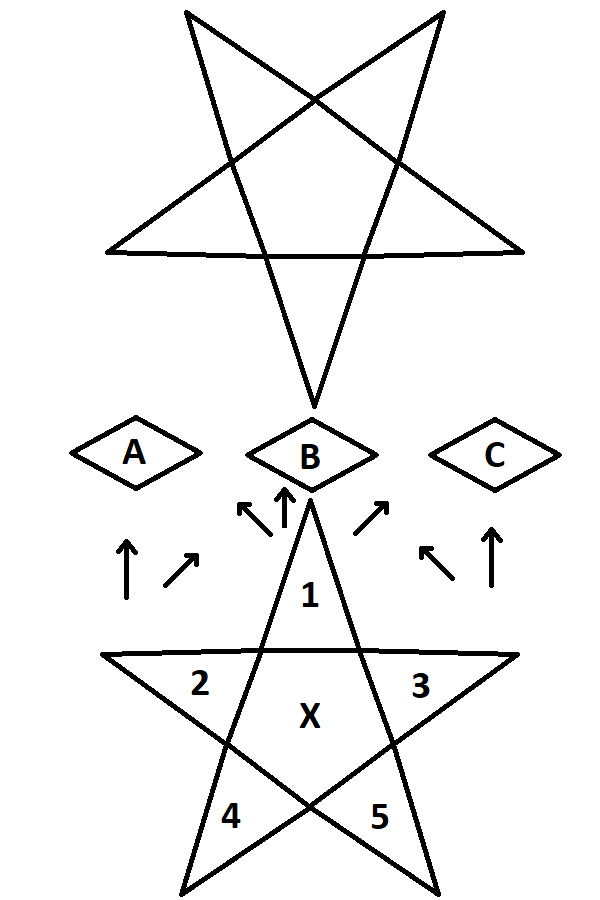
# 4. Description

Our website will serve as a client to connect with friends and play Sellswords & Spellcrafts. We intend players to send direct links to their friends to challenge them to a game. Once on the site, players will have the option to play the match offered or view a detailed explanation of the rules and an example of the basic gameplay loop.

Sellswords & Spellcrafts’ primary gameplay loop will center around players paying life to summon minions, casting incantations for their indicated effects, playing terrain to provide energy for minions and incantations, and players attacking contested ground. On each but the very first turn of the game, the turn player will draw a card to begin their turn. Then the turn player has the option of playing terrain, minions, or incantations before or after attacking.

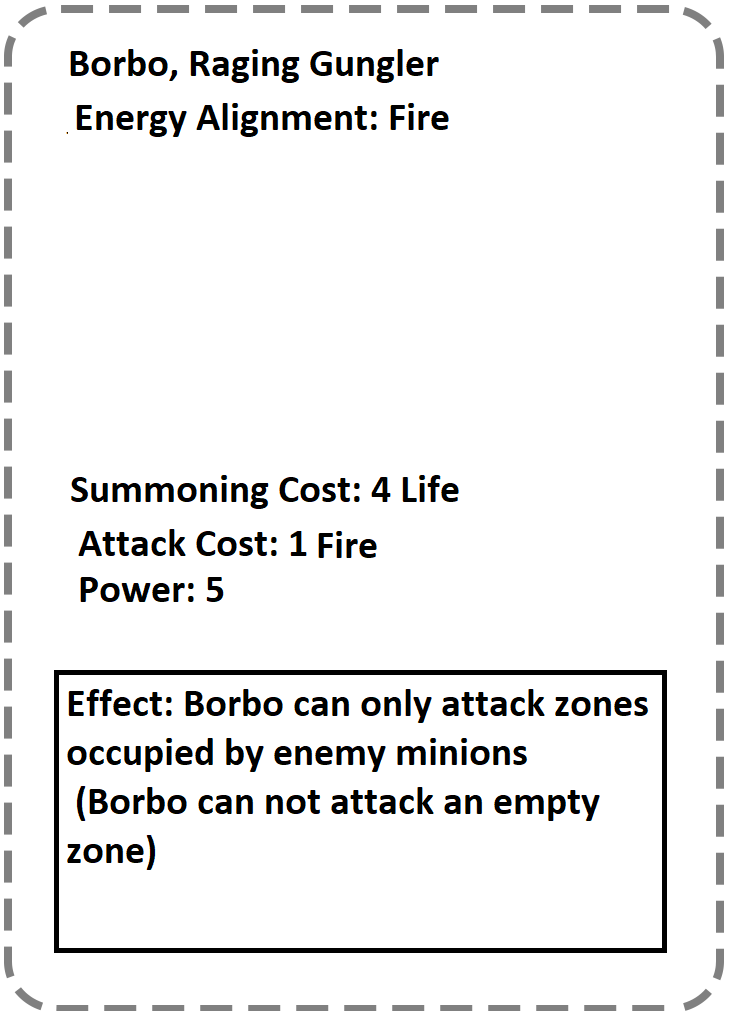
The “board” will be divided into 3 broad zones: Your summoning circle, three contested zones, and the enemy summoning circle. Your summoning circle consists of three minion zones (Figure 1: 1,2,3), two invocation zones (Figure 1: 4,5), and a terrain zone (Figure 1: X). As the names imply, minions may only be summoned to minion zones, invocations may only be placed in invocation zones, and terrain may only be placed in the terrain zone. A minion may only attack contested zones or the creature occupying contested zones adjacent to itself (illustrated by arrows in Figure 1). Once a minion attacks an unoccupied contested zone or destroys a minion occupying a contested zone, that minion will leave your summoning circle and occupy that zone. Each contested zone will contain a contested zone card randomly selected from a pool of possible zones. Once occupied, a contested zone card will confer its specific benefit or drawback to the creature occupying its zone. A minion may only attack your opponent directly once it occupies a contested zone.

**Figure 1**.A labeled diagram of the game board



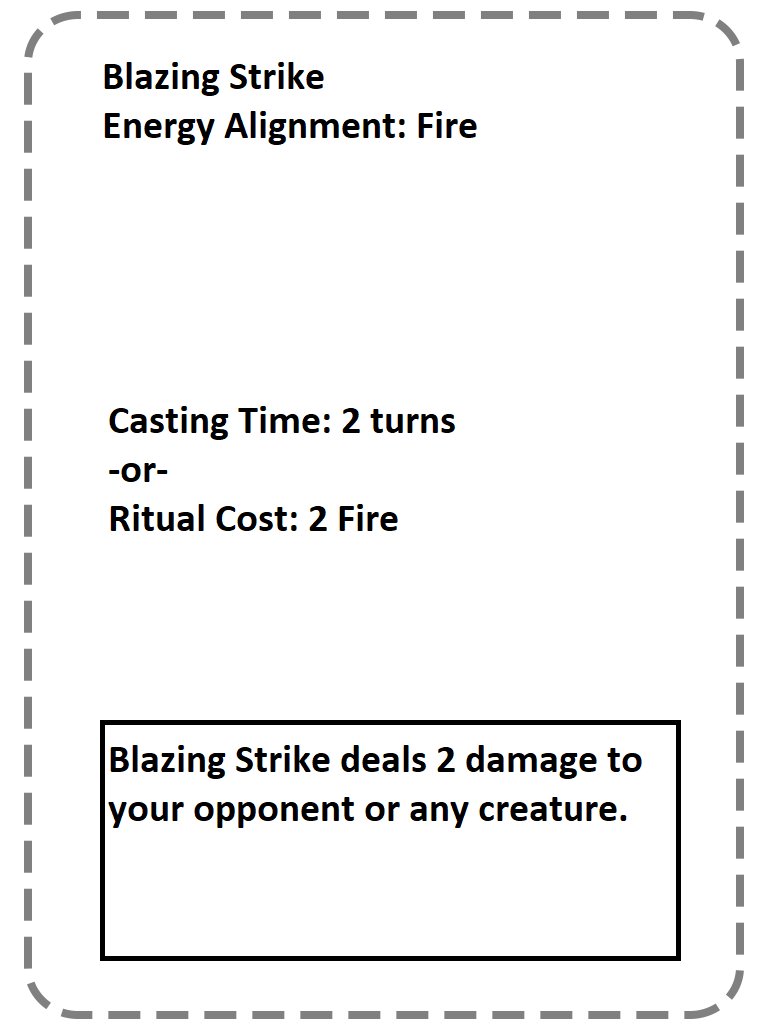
Your deck will consist of 3 primary card types: minions, invocation, and terrain. All minions have 4 fundamental attributes:a summoning cost, a power stat, an attack cost, and an energy alignment. A summoning cost indicates the amount of life to be paid in order to summon a minion to any valid position in your summoning circle. A power stat indicates how much damage a minion would deal to minions and players they attack as well as how much damage they can take before being destroyed. An attack cost indicates the amount of energy that must be paid to attack a player, minion, or contested ground. Alignment indicates what type of energy must be paid for attack costs as well as provides subdivisions for effects that affect specific alignments. There are 4 possible alignments that correlate to desired play styles: fire, wind, water, and void. Fire typifies high aggression and trading resources like your own life total and cards in exchange for immediate damage. Wind typifies the “midrange” playstyle, focusing on moderately sized creatures of all costs with an emphasis on utility such as drawing cards and gaining life. Water emphasizes a slower playstyle, focusing on using cheap, weaker creatures that prevent damage in the early game to buy time for casting expensive haymakers to control the end game. Void serves as a utility color, slotting into all decks and requires no specific type of energy to activate its effects.

**Figure 2**. A minion card prototype



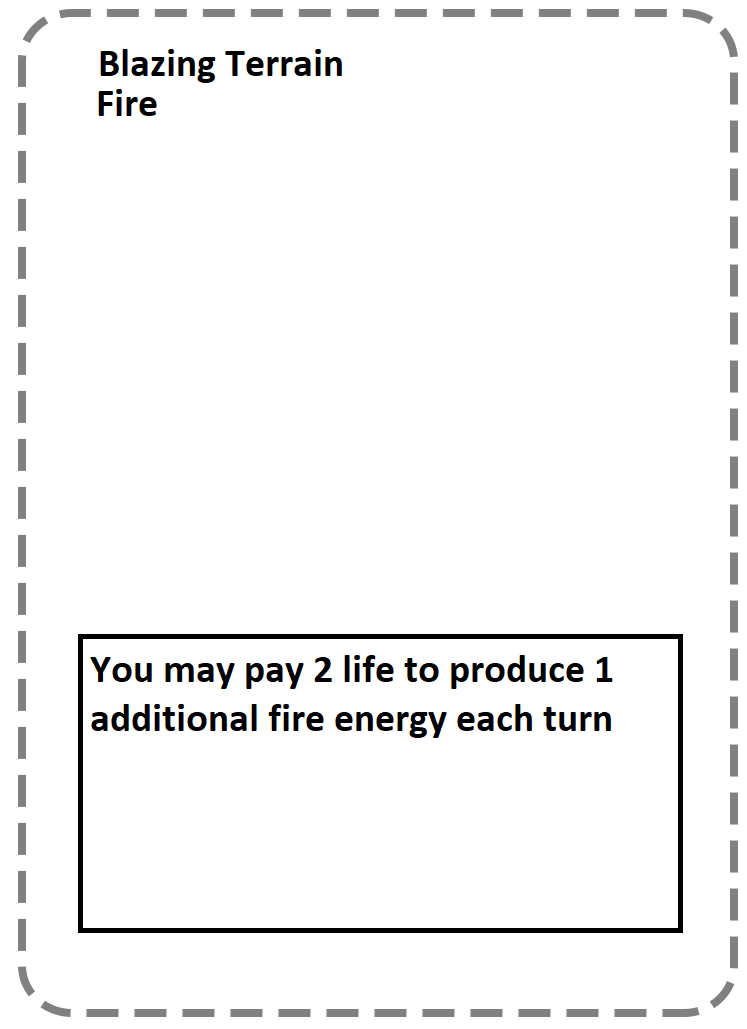
An invocation has 4 important pieces of information: an energy alignment, a casting time, a ritual cost, and its effect. An invocation may be placed in your invocation zone during your turn for no cost, however its effect will not resolve until the number of turns indicated by its casting time have passed. Alternatively, a player may pay an invocation’s ritual cost to immediately gain its effect.

**Figure 3**. An invocation card prototype



Finally, terrain is the simplest card type in your deck. All terrain produces one energy of its stated type per turn, although some terrain may have additional effects. A player may only play one terrain per turn, thus forming the primary “resource curve” of the game, keeping players from using all cards in their hand on turn one.

**Figure 4**. A terrain card protype



# 5. Feature list

## Features that will be completed by the end-of-semester

* Functional website hosted on a web server with a “game” page and “rules” page
* Networking functionality to connect two end users together via a central web server
* Functional, playable game for the two users to play against each other, complete with animation and original artwork to some capacity
* Pre-packaged, complete decks for users to pick up and play

## Features that will be completed if there is time

* Ability to save pre-made card decks
* Ability to make custom decks
* Video tutorial page
* Aesthetic polish

## Features we would like to implement but cannot be completed this semester

* Replace instance userID with username password combinations
* Ranking / leveling system
* Interactive tutorial

# 6. Technology

* Desktop / Windows 10 - For simplicity sake, All team members own a desktop computer running Windows 10. Given the time and scope of the project, developing for one OS on one platform will be easier.
* Firefox - All team members have experience developing and optimizing web based applications for Firefox.
* Visual Studio 2019 - Provides the ideal development environment given our use of tools in the .Net ecosystem(C#, Blazor,...) .
* C# - C# is a prerequisite for Blazor development as well as a language all team members are familiar with.
* Blazor Server - All code will run server side. This will streamline the process, as all code will be written in one language, C#. This will also aid in simplifying stylization.
* Discord - Discord provides persistent records of team discussion as well as the ability to designate specific discussions to their own channels.
* Google Cloud - Google Cloud’s services are free and links directly to GitHub, streamlining the process.

# 7. Server information

The server will be run on Google’s Cloud services. The service we are using is the free version. It links directly to GitHub, so as we update our repositories, it will automatically link to the server, which will make running the server simplistic. If we want, we can implement an authentication service when players initially join the game, as the Cloud services make providing this feature as easy as clicking a button.

# 8. Data sources

N/A

# 9. Team members’ backgrounds.

Mason Evans has experience working with HTML, CSS, JavaScript, C, C++, SQL, Java, and has general experience working with languages such as C# and .NET. Nicolas Towery has experience working with HTML, JavaScript, CSS, C#, and .NET with Blazor. Luke Schnetlage has experience working with HTML, CSS, C#, TypeScript, MSSQL, XAMPP, and general game development.

# 10. Dependencies, limitations, and risks

Dependencies, risks, and limitations that we are already aware of and that we may encounter are that we have very different work schedules, so we have to account for that when we are planning to meet outside of class. This also holds true for our class schedule, as we all have several different classes, and will require us to plan when we will do work for these classes and we will do work for this project. The other issue that has come to our attention is that we are generally inexperienced when it comes to working with real-time player versus player web applications, which will require us all to learn as we go along.

# 11. Timeline

Tasks are to be completed by 11:59pm on specified dates, unless stated otherwise.

* 1/22: Proposal Due
* 1/29:
  + Luke - Fine tune game mechanics, researched potential needed libraries
  + Mason - Set up web server
  + Nicolas - Draw out a diagram of entire website link functionality
* 2/5:
  + Luke - Begin creation of class files and main file, develop ERD
  + Mason - Begin drawing cards and research related network functionality
  + Nicolas - Draw 3 sketches of potential web designs and select one
* 2/12:
  + Luke - Develop the “read, calculate, render” cycles for core gameplay
  + Mason - Finish drawing cards, begin development of instanceID
  + Nicolas - Make all web pages and begin on CSS
* 2/19:
  + Luke - Develop game board-card interaction systems
  + Mason - Scan in drawings
  + Nicolas - Web page layouts are in first draft
* 2/26:
  + Luke - Develop specific card interactions and events
  + Mason - Format drawings to .png files
  + Nicolas - Web page layouts are in second draft, begin frontend scripting
* 3/5:
  + Luke - Successfully link frontend to backend
  + Mason - Update server as needed
  + Nicolas - Successfully link frontend to backend
* 3/12:
  + Luke - Successfully implement helper files
  + Mason - Successfully implement instanceID
  + Nicolas - Completed final draft of ”rules” page
* 3/19:
  + Luke - Successfully implement “pre / post game menus” (backend)
  + Mason - Successfully implement “pre / post game menus” (networking)
  + Nicolas - Successfully implement “pre / post game menus” (frontend)
* 3/26:
  + Luke - Successfully implement beta version of gameplay (backend)
  + Mason - Successfully implement beta version of gameplay (networking)
  + Nicolas - Successfully implement beta version of gameplay (frontend)
* 4/2:
  + Luke - Develop custom deck creation system
  + Mason - Manage upload and storage of all content available at this. point to the server
  + Nicolas - Create SFX files / game animations
* 4/9:
  + Luke - Validate and refine custom deck creation system
  + Mason - Manage static resources going to and from server
  + Nicolas - Successfully implement SFX and image files
* 4/16:
  + Luke - Refactor existing codebase where applicable
  + Mason - Test server by running different jobs to determine efficiency
  + Nicolas - Successfully implement animation and remaining SFX files
* 4/21: Poster presentation - Demo most recent stable build
* 4/23:
  + Luke - Quality testing for obscure interactions and complex boardstates
  + Mason - Ensure responses within server are efficient
  + Nicolas - Complete final draft of “game” page, polished UI
* 4/30:
  + Luke - Iron out remaining bugs (backend)
  + Mason - Iron out remaining bugs (networking)
  + Nicolas - Iron out remaining bugs (frontend)
* 5/3 at 10:30am: Class presentation - Demo final build
* 5/4 at 11:59am: Final report and support documentation