Critical Encounters

Jacob Spigle, Zachary Painter, David Akridge, Hunter Figueroa

November 13, 2017

Contents

Introduction

Role-playing games are games where players act as their characters. Usually in person, around a table, and controlled by the players speaking their actions in turn. This might not always be done by speaking in-character, but rather by following this basic structure:

1. The GM describes the environment.

2. The players describe what they want to do.

3. The GM narrates the results of the characters? actions.

There are countless possible character, monster, and scenario possibilities when constructing a tabletop role-playing campaign, however, it is hard and sometimes impossible to sift through them all and find the perfect encounter. Not only that, but once you commit to a game or role-playing decision you are forced to see it through to completion in order to retain the game?s continuity. Players get one chance to design their character at the very start of a session, and once they do, that character?s path cannot be meaningfully changed. Spending so much time working on a character, and then coming to the first session to find out that they are severely unprepared for the tasks they are presented can be harrowing. On the other side, the GM spends so much time outside of the game to prepare a story and encounters that challenge the Players, so when that GM accidentally wipes out the entire team of Player Characters (PCs) in one unbalanced encounter, or if the PCs simply walk through a fight that was meant to challenge them for the remainder of the session, the GM feels they have failed the PCs in presenting a good game.

Proposed Solution

This project presents a service that can be used by both GMs and players to test out their characters, encounters, team builds, or boss fights in a simulated environment. This will allow users to figure out just how well their creation holds up to the requirements that they have set for themselves. This project seeks to develop a service that follows the rules of the d20 Game System, and by allowing the player or GM to import their character's/monsters' statistics and equipment, they then can play against an automated opponent or opponents using that creation. To further enhance a user's ability to test and refine encounters and PCs, users will be able to upload and share their encounters to their public space where others can try their own creations' skills in that creator's setting.

Broader Impact

Tabletop games bring people together in a world where physical interaction is often lost in lieu of digital communication. This project aims to supplement these interactions by making them more accessible to newer players, as well as help experienced players spend less time on pre-game preparation. This project has the potential to broaden the audience of an already quickly expanding pastime to younger players and DM's. Concepts in the realm of tabletop games are very easily understood by a younger audience, as their imagination and creativity can run free in such games. However, the threshold of understanding for the many rules and balances may prove a bit overzealous for this audience. Having a service that allows improves accessibility to newer players will also bleed into improving accessibility for younger players as well. Additionally, users who do not have a group to play with may find and form groups with other users with whom they have played and shared content with.

Personal Interests

4.1 Jacob Spigle

I discovered role-playing games like Dungeons & Dragons early this year. After playing video games since childhood, D&D was a breath of fresh air; a way to mix the excitement of creating a character, leveling up, deciding your actions, along with an openness that allows nearly infinite stories, infinite worlds, and the tools to create those stories and worlds yourself. Those infinite possibilities can spur creations that might not work as well in-game as you thought. I have written campaigns and characters that on paper seemed great, but when brought to the table were found lacking. When I searched through the many different forums and groups I had joined after I started playing D&D, and scouring the internet for a way to playtest my creations, I was surprised to found that no such tool existed. This service is something I can actually use and wish that I had during the creation of campaigns I have written, and that is why I believe others will find use in it as well.

4.2 Zachary Painter

Some of my primary interests in Computer Science are data structures and algorithms. In my area of research, concurrent programming, these two topics are frequently discussed. Often times in this field, I have had the opportunity to explore many complicated and fascinating algorithms. When tackling these kinds of problems, you often find yourself spending considerably more time thinking and reasoning than implementing/coding. After understanding a problem, you move on to another without getting the chance to apply what you learned in a practical/usable way. This constant cycle of struggle/learn/get-new-problem is something that many students experience throughout college as they rarely have time to stop and demonstrate their rapidly growing skillset in a meaningful way.

My motivation for being part of this project is that I want a chance to slow down and use the skills and algorithms I have picked up in my years at UCF in a useful and creative application instead of an abstract, un-impactful thought exercise. Instead of expecting this project to challenge me with highly complicated algorithms and difficult proofs, I expect this project to challenge me in project management skills, team coding, and the application of numerous techniques that I have learned but never had to implement in a meaningful way. There is an important difference between *knowing* and *doing*; and so I am taking this chance to remind myself what all my learning has amounted to.

4.3 David Akridge

This project caught my interest due to the combination of the creative nature of Dungeons & Dragons mixed with the real world applications that can be gained by accomplishing the project's end goal. In my time at UCF, we have learned tons of information primarily focusing on theories and concepts of computer science. While plenty of classes have had us working hands on, there is still much to learn before being able to stand out in the real world. This project is going to have us utilize the various algorithm implementations we've learned as well as diving into web development. As we haven't covered web development in too far detail in the UCF curriculum, this project easily provides an opportunity for me to grasp the understanding of HTML, Javascript, Angular.js, etc. These are all languages and formats that I have had the desire the learn but lacked the drive to do on my own.

Along with being able to learn practical CS skills, Critical Encounters will allow us to find creative workarounds be it code, web-page design, or even artwork. This is a really important aspect of the project to me, because many of the projects we've had to do prior have been numerically heavy and honestly not all that interesting. An engaging idea could help propel our ability to learn these new concepts, especially if we are interested in them on our own. Now that there is a great project motivating me and a great team to work alongside with, there is no doubt we will learn all the in's and out's of the subject.

4.4 Hunter Figueroa

I have been playing tabletop role-playing games for over 6 years now. They served as a creative outlet where I could experiment with new creative ideas. I prefer acting as a DM, I've only played a single session as a player, I loved the idea of being all powerful creator of a world that I could share with my friends. The tabletop community, namely the Dungeon & Dragons community is like no other that I have seen before. It is composed of highly talented, highly cooperative and interactive people who do their very best to better and expand the role playing community. For close to two years now I have put a considerable about of my free time into constructing a digital service to help this same cause. So when I heard Jacob pitch this idea, I knew it was the one for me. We've got something really special here, and with the team that we have I feel we can create something game changing.

User Interface

5.1 Overall Design and Navigation Bar

The approach we took to user interface (UI) design was focused on a simplistic and familiar look and feel to allow for easy accessibility to new users while the website itself still retaining focus on the encounter tester. The template that website utilizes is reminiscent of the design that many social media sites take, for instance, sites like Twitter, Facebook, and YouTube.

The site also uses the the single-page application structure that allows for easy traversal. The persistent navigation bar (Figure 5.1) has Critical Encounters' name and logo in the top left of each page which redirects users back to the homepage, a searchbar in the top of each page which redirects users to the Encounter Browser page, and the user's username in the top right of each page that links to the user's Arena.

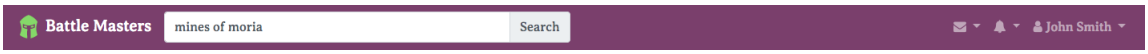

Figure 5.1: NavBar

5.2 Registration Page

The registration page (Figure 5.2) for Critical Encounters requires the user's desired username, their email addresss, and their password.

5.3 Login Page

The login page (Figure 5.3) for Critical Encounters requires the user to enter either their username or email address along with the password tied to that account.

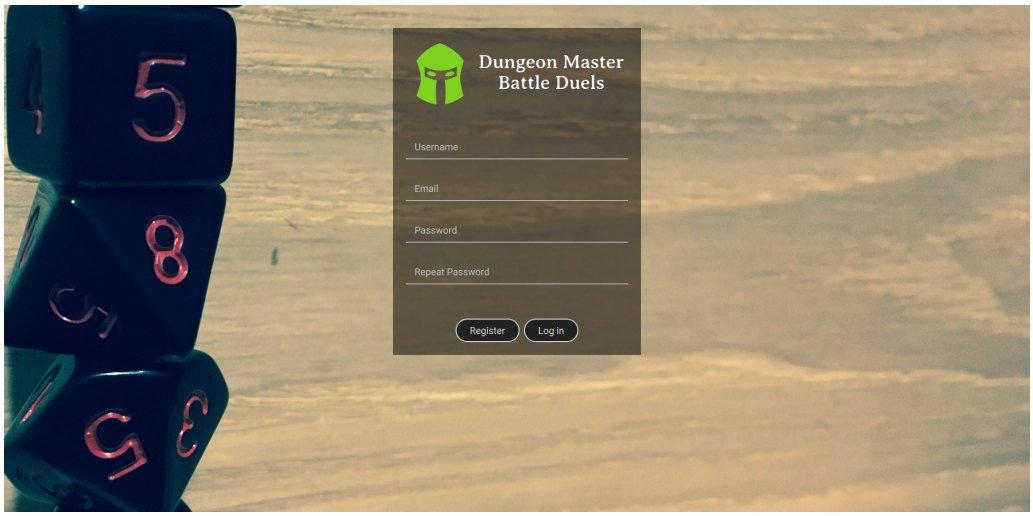

Figure 5.2: Registration Page

Figure 5.3: Login Page

5.4 Dashboard

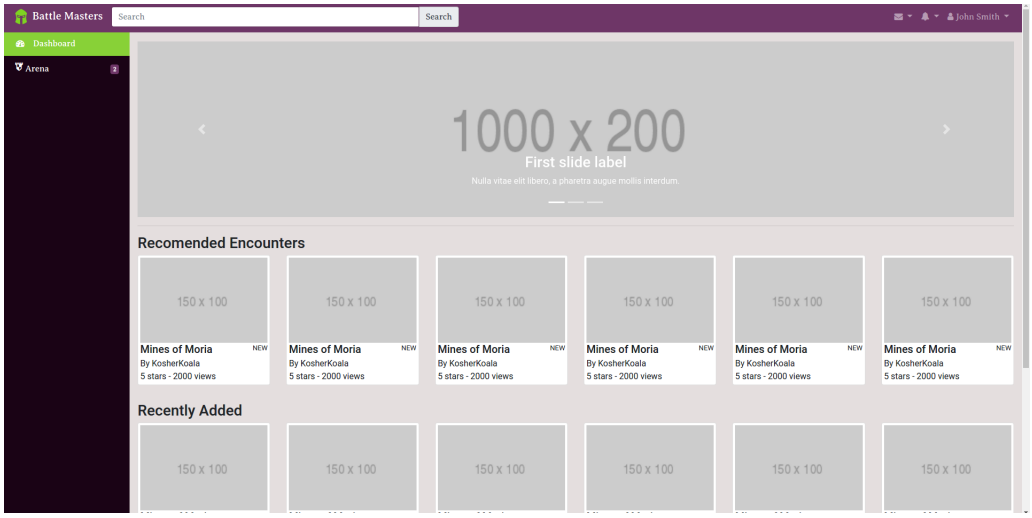

Figure 5.4: Dashboard

The Critical Encounters' Dashboard (Figure 5.4)features notable encounters, updates, and community events in the highlight wheel at the forefront of the page. Below the highlight wheel is a list of recommended encounters that either we the developers suggest to the community, or recommended list that is influenced by the user's recent search history.

5.5 Encounter Browser

The Critical Encounters' Encounter Browser (Figure 5.5)acts as a search results page for queries entered in the searchbar in the navbar. From this page a list of filtered encounters is presented to the user. The user may also filter those results further by selecting settings from the dropdown menu revealed by the 'Filter' button. Through this process we can be sure to provide users with encounters that help with creation or simply encounters that help spark their own imagination. (Figure 5.6).

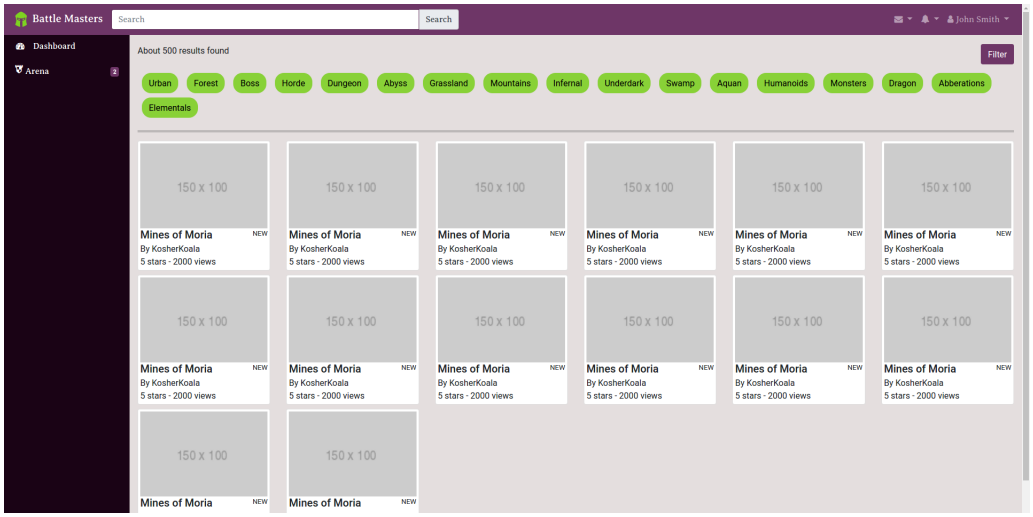

Figure 5.5: Encounter Browser

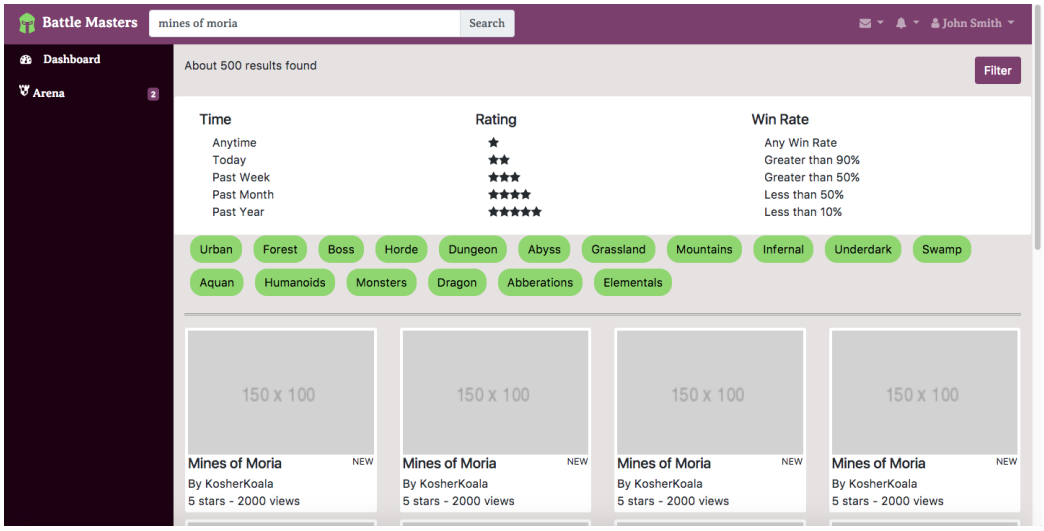

Figure 5.6: Encounter Browser with Filter Options

5.6 Arena

Each Critical Encounters user is provided a page to host their published encounters for the rest of the community to access, their Arena. From the Arena page (Figure 5.7) is where users are redirected to encounter creation within the Battlefield by clicking the "Create Encounter" button on their own Arena. Users also have the ability to edit their Arena, listing their created encounters in what-

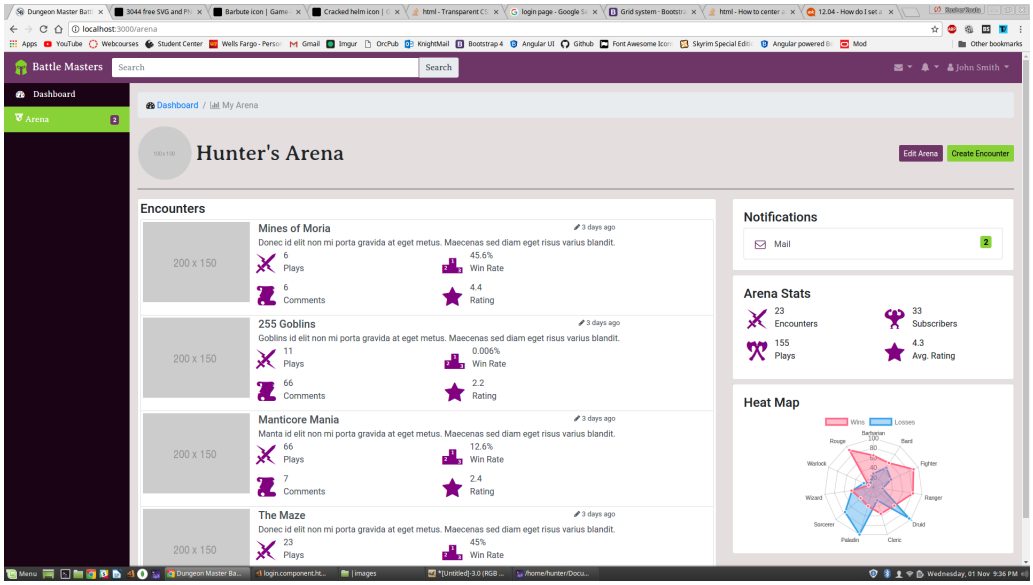

Figure 5.7: Arena

ever order they wish, whether done manually or by analytics (Plays, Win Rate, Rating). At the top of the Arena page is the username and profile picture of the user tied to the Arena. The Arena presents the user's creations in a list format, with pertinent information listed alongside each encounter. Number of plays, PC win rate, overall rating, and the number of comments are shown under the encounter's title and description, and to the right of a screenshot or preset image of the encounter board itself. To the right of the encounter list, additional data on the user's arena is supplied:

- Number of Encounters
- Number of Plays
- Number of Subscribers
- Average Encounter Rating

Below that, a Heat Map figure illustrates how well each class performs (Wins/Losses) in the encounters in the Arena.

5.7 Encounter Page

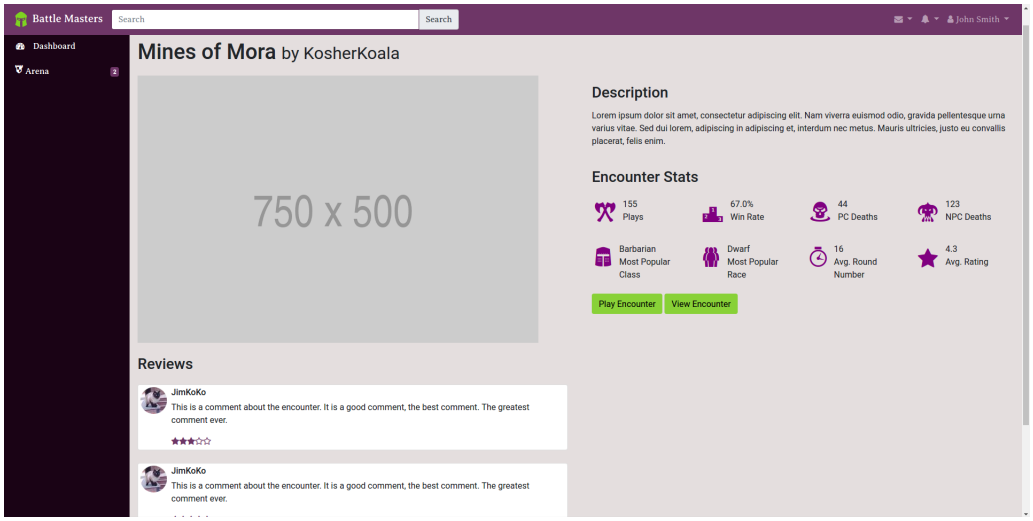

Figure 5.8: Encounter Page

Each encounter created in Critical Encounter is given an encounter page, from which user's navigate to in order to start playing the encounter. At the top of the encounter page is the encounter's name, followed by the username of its' creator. Below is a screenshot or preset image of the encounter board, and the creator's description of the encounter hangs to the left of the image. The encounter page also presents the user with statistics related to the encounter:

- Number of Plays
- Win Rate
- Number of PC Deaths
- Number of NPC Deaths
- Most Popular Class
- Most Popular Race
- Average Number of Rounds
- Average Rating

5.8 Battlefield

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

5.9 Color Scheme

Figure 5.9: Color Scheme

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5.10 Icons

Most of the icons we utilize will be provided by the free icon APIs font-awesome and rpg-awesome. Additional icons will be acquired through Game-Icons.net.

5.10.1 Logo

Figure 5.10: Critical Encounters' Logo

We wanted the logo design to accompany the overall UI of the website with a simplistic design, while also representing the RPG aspect of Critical Encounters. We decided on a helmet because it gave a little more of a human feel, we thought swords or other weapons would be too inanimate. We chose the green color so the logo will "pop" with most backgrounds.This logo can also be easily scaled to fit most container sizes.

Technical Goals

6.1 Overall Goals (Requirements???)

6.2 Project Timelines (Milestones???)

6.3 Communication

6.4 Research

6.5 Budget

Server Design

7.1 User Roles

7.2 Node.js

7.3 Express.js

7.4 Multer

7.5 Mongoose

7.6 Server Sequence Diagrams

Database Design

8.1 Schemas

8.1.1 User Schema

8.1.2 Arena Schema (HAHA THAT RHYMES)

8.1.3 Encounter Schema

8.1.4 Combatant Schema

8.1.5 Obstacle Schema

Application Design

9.1 Design Patterns

9.2 Accessibility

9.3 Types of Views and Controllers

9.4 In-Game Roles

9.5 Map / Environments

9.6 Turn Structure / Order

9.7 Computer Opponent (A.I.)

Since the combat simulator will support only one user, it will be necessary to implement an A.I. capable of controlling either the GM or PC actions. The A.I. must be versatile enough to play either role with an average measure of competency. It is not important that the A.I. makes the optimal tactical decision per action, as this would conflict with the role-playing nature of the D20 system. Instead, the A.I. will have various personality presets, which will be used to guide the A.I. decision making to better match the personality of the person or creature that it is controlling. In some cases, this may lead to a suboptimal action being taken intentionally. Such behavior would be left up to the creator of the encounter and the personality they select for each creature they place in it.

9.7.1 A.I. Responsibilities

If the user is controlling the PCs, the A.I. will control all other allied, neutral and hostile units on the board that are not one of the PCs. If the user is controlling

the GM, the A.I. will control the PCs only. This follows the traditional design of the DnD 5e combat rules. In the case that the A.I. must control two units, one that is hostile to the PCs and one that is in alliance with them, the A.I. will take the actions that are in the best interest of the unit it is controlling for that action, regardless of whether or not that action is in the interest of the side that the A.I. is playing for.

9.7.2 Decision Making

9.7.3 Unit Personalities

9.7.4 Role-Playing

9.8 Encounter Diagnostics (Stress-Tester)

The encounter diagnostics tool, or stress-tester, will be a feature available to users of the website to evaluate their encounter designs and receive highly detailed evaluation based on a variety of metrics. Because combat in DnD5e is quite extensive and highly open ended, it is unlikely that a user seeking to create a well balanced and fluid encounter will be able to do so with some trial and error. Instead of manually playing the encounter repeatedly with multiple variations to check for weaknesses in the design, users will be able to run the built in encounter diagnostic tool to automatically generate useful information regarding their encounter.

9.8.1 Restrictions

The encounter diagnostic tool will require as input a valid encounter that has passed verification. A valid encounter is one that has at least the minimum amount of components necessary for an encounter between the player characters and game master to be simulated. For an encounter to be verified, the creator of the encounter must succeed against a computer controlled game master at least once while controlling a party of player characters that meet the designated restrictions set by the encounter creator.

This requirement is imposed as the diagnostic tool is not meant to be used during the design process to help a creator shape their encounter. It would require

additional functionality to enable the encounter diagnostic tool to work correctly on partially finished encounters and additional functionality to process the resulting data such that it is useful to the encounter creator in designing the remainder of the encounter.

Additionally, the encounter must have passed verification in order to strengthen the notion that all final encounters must have at least one solution found explicitly by a human user. This is to prevent encounter creators from using the diagnostic tool to figure out if their encounter is possible without having to test or verify it themselves.

9.8.2 Interface

The user will be able to access the interface for the encounter diagnostic tool from within the encounter creator. The encounter diagnostics option will only be available on encounters that the user has created. Once the user has selected an encounter to perform encounter diagnostics, they will be prompted to enter details necessary to configure the tests.

9.8.3 Diagnostic Measurement

When activated, the encounter diagnostic tool will generate a set of valid party member compositions that meet both the restrictions of the encounter as well as the additional restrictions provided by the user during configuration. A party composition is a collection of player characters that will be placed on the field of play at the start of the encounter. The diagnostic tool will generate one party composition per iteration of the test. Once all party compositions are generated, the diagnostic tool will simulate the encounter one time for each party composition.

The simulation will consist of each member of the party being placed in the user designated spawn positions for the encounter. Once all player characters are placed, both PC and GM turns will be taken by the A.I. until either side meets their win condition, at which point the board is reset and the next party composition is tested. Once all party compositions have been tested, the diagnostic tool serves the results of the tests to the user.

During each test , statistics relating to the performance of each entity in the encounter will be tracked in order to provide metrics by which the user can evaluate their encounter. Statistics that will be tracked include the following:

- Number of kills/deaths for each non-PC entity in the encounter
- Number of kills/deaths by each "class" of PC
- Overall win rate of PC vs GM
- Damage per ability used
- Location of each kill made on the map
- Turn count

9.8.4 Results and Presentation

One all simulations are complete and the data has been collected, it must be processed into a form that is quick and easy to understand. Each statistics that was tracked on a per-encounter basis will be aggregated into an average value and presented to the user in a separate window. These average values will highlight at a glance measurements such as the average length of each encounter, which enemies are the most dangerous, how difficult the overall encounter is, and which abilities are the most effective in the encounter. Additionally, a heat map will be generated using the location of kills made on the map across all simulations. This heat map will indicated to the user the general flow of combat, which can be used to help fine tune the encounter and improve overall quality.

9.9 Encounter Browser

9.10 Encounter Creator Tools

9.11 Storyboards

9.12 Version Control

9.13 Data Processing from API (WHAT DOES THIS MEAN???)

Research

10.1 MEAN Stack

Figure 10.1: Mean Stack Logo

The creation of developer tools like MongoDB, Express.js, Angular.js, and Node.js (otherwise known as the MEAN stack) forge a collection of technologies that allow for JavaScript utilization when developing a web application. The culmination of these technologies build off each other while staying modular. This can allow for a head start in the initial stages of development with scaffolding for all of the main tools in the MEAN Stack.

10.1.1 Altered MEAN Stack

After scouring the internet for alternatives, we found that Meteor.js is a welcome alteration to the MEAN Stack implementation. Meteor.js is another API that allows for focus on JavaScript coding in development across front-end and back-end applications. While it is a bit more heavyweight than the Express.js framework, on the Node.js platform, Meteor.js is designed to accelerate development of web applications. As an extention of the MEAN Stack, this is achieved just the same, by having a number of components pre-configured out-of-the-box, which allows for an enormous head-start in development.

MongoDB

Figure 10.2: MongoDB

We decided to utilize MongoDB, a NoSQL database that is an essential part of the MEAN Stack. There are quite a few reasons to opt for a more modern approach to constructing databases, for instance, the structure and data type in schemas in a SQL database are fixed. To store information about a new data item, the entire database must be altered, during which time the database must be taken offline. With this project taking on an agile approach in its' development cycle, and a database being so essential to early development with the implementation of the many items, rules, and class/monster types that are available, starting that database and being fixed in the schemas we create could cause a slow in development later in the cycle when changes are made. However, when using a NoSQL database like MongoDB, insertion of data is allowed without any predefined schema. This means development and integration of code is faster and more reliable. A document database model for data storage that we can adopt when using a NoSQL database is another benefit to using MongoDB, because it best fits the way that information is organized while using the d20 Game System.

Meteor.js

Figure 10.3: Meteor.js

Meteor.js is a full-stack JavaScript platform for developing modern web and mobile applications. Like much of the API's we are implementing in the development of this web application, Meteor allows for coding in JavaScript across the front-end and back-end side of development. The biggest advocate for including Meteor in our project is its' focus on development speed. All server-side applications are handled in the development of the client-side. That is to say, when developing client-side applications, any server-side necessities are taken care of by Meteor.js' ability to redirect data in the background. The scaffolding that is included directly out of install with Meteor, such as a fully configured MongoDB database and integration with Angular while still having a fully integrated stack, is a leap forward in the development cycle of this project, so we will have more time for fine-tuning features and performance of the web application later. Meteor also uses web sockets instead of HTTP-style pull requests, so the server pushes updates to the client when new data is available, and the client renders that data. This allows Meteor to implement "hot-pushing", where developers can push updates the client even while it a user is in the middle of a session. Any fixes or improvements can be pushed without the user even knowing.

Node.js

Node.js is a runtime application that starts small but grows more and more useful as additional modules are implemented. There are a few reasons why it is the first choice when building a web application like this project. Node allows for JavaScript coding on the server side of a website or web application. This results in the idea that code can be written once, but ran on both the server and the client browser, optimizing which functions would give users faster response times while also lifting a bit of work for the server. Writing server-side code in JavaScript using Node also keeps the logic and code very similar across the server and client. Also, Node uses an asynchronous even driven mode, so that when a task is performed, other tasks are not prevented from being run.

Figure 10.4: Node.js

Angular 4

Figure 10.5: Angular 4

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Bootstrap 4

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Figure 10.6: Bootstrap4

10.2 RESTful

10.3 Game A.I.

10.3.1 Pathfinding

10.3.2 Basic Rule Following (Non-A.I. Computer Opponent)

10.3.3 Decision Trees

Appendix

Figures

Tables

References