University of North Carolina Asheville
Journal of Undergraduate Research
Asheville, North Carolina
May 2022

D&D Engine: Digitally Enhancing Interactive Storytelling

Keegan Bruer
Department of Computer Science
The University of North Carolina Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisors: Dr Marietta Cameron, Dr. Adam Whitley

Abstract

The essential experience of Dungeons and Dragons (D&D) is to live out a story-like adventure, grounded in specific rules and mechanics, but ultimately guided by a storyteller dubbed the Dungeon Master (DM). This project, DnDEngine.com, is a website for playing D&D campaigns online that captures and enhances this interactive-storytelling experience. Existing platforms, such as Fantasy Grounds and Roll20, provide systems for keeping track of information about the campaign including character components, monster stats, item descriptions and spell details. Roll20 provides simple interfaces for storing this information, but unfortunately it lacks the ability to store custom information. Roll20 systems provide a digital-paper like interface that allows you to store character information, yet it does not take full advantage of its digital format. Fantasy Grounds does allow for more customization, but has a complex interface that is hard for new players to navigate. D&D Engine borrows from the best parts of these platforms, and presents a full-featured application that better utilizes the project's digital format to provide an enhanced experience over traditional paper gameplay. Regarding monetization, Roll20 offers a free platform that lacks advanced features, while Fantasy Ground offers advanced features but requires each person to purchase a copy of their software. D&D Engine offers a free “basic play” component as well as following a tiered subscription model that provides adjustable levels of premium features for advanced players.

1. Introduction

1.1 What is D&D

First published in 1974, Dungeons & Dragons (D&D) is a tabletop role-playing game created by Tactical Studies Rules, Inc. New versions of the game have been published by Wizards of the Coast since 1997. D&D is published in the form of a set of books, each defining a different aspect of the game. The game is played by a group of players led by what the book calls a “Dungeon Master”, generally shortened to simply “the DM”. While Wizards of the Coast provides all the necessary aspects of play in a very detailed format, DMs are encouraged to adapt these rules to best tell the story they want to tell.

The game is traditionally played using pen and paper, not only writing down character details in the format of a Character Sheet, but creating environments in the form of hand drawn Battle Maps. In a digital format, D&D still requires digital implementations of Character Sheets and a Battle Map. A digital version of D&D would also require a digital implementation of the content of the books, mainly focused on the provided information for creating imaginative worlds including magical items, spells, classes, races, and monsters. Most existing digital D&D implementations call this collection of premade content the “Compendium”.

This project focuses on the interactive storytelling that D&D exemplifies and not the specific worlds or rules that are defined in the official D&D content. The goal is for the platform to be able to handle any type of interactive story that a user may want to tell. While developing a user interface for general storytelling aspects, it is easier to focus on a specific example of the type of content that will be added instead of blindly working at an ideal. D&D is the perfect

choice to use as an example because of how expansive and customizable the content is.

1.2 Enhancing The Experience

This project provides a web application as an alternative to existing products and offers the best experience to make creative D&D campaigns. The website serves as a central location for D&D enthusiasts to enjoy a traditional pen and paper feel while also gaining the benefits of using a digital medium. There are a few similar products that attempt to accomplish this same goal.

For example, Fantasy Grounds is an extensively featured product, containing a plethora of customizable features that can act "as a virtual online gaming table primarily intended for pen and paper style narrative role playing games." [2] Fantasy Grounds provides detailed menus and an integrated user interface to speed up some of the aspects of the game most players would consider tedious or meticulous (e.g. keeping track of damage). The downside of Fantasy Grounds is that it requires a large upfront cost to start creating as a DM and also requires each of your players to put down a large upfront cost. This deters not only beginner players but experienced players that want to introduce their friends.

Another example, Roll20 [3] is a free to play website, built by a GoFundMe Campaign in 2012, that contains the basic features needed to play D&D online. Roll20 has a simplistic and outdated map engine that can support basic features needed for operating a D&D campaign. Roll20 is a great place for inexperienced players or for experienced players that want to introduce their friends, but it doesn't allow for growth as the players and DM gain experience.

This project is the middle ground between these two platforms that provides an inexperienced user with a free to use website that can support their creativity, while also providing experienced users with a plethora of features to fully customize their game-play experience. With a focus on intuitive design for the Map Viewer, new players will have a straightforward experience. While the Map Editor will have a feature-set that allows DMs to build creative and extensive stories.

2. Background

2.1 Exploring Existing Solutions

To develop a base understanding for what features are wanted by the D&D community, we need to discuss other D&D related websites that are already in use. All of the projects below fulfill different aspects of what the D&D community wants and requires. The goal of this project is not to replace existing products but to become the hub where these products collide, so it is important to understand the extents of each of these products.

Fantasy Grounds is an extensively featured application that is great for experienced enthusiasts who can afford a large upfront cost. The application provides users a technical application that allows for keeping track of the detailed mechanics of the campaign. Fantasy Grounds provides many different licensed game systems allowing users to play different types of games with their system. This project is different from Fantasy Grounds because the website will not include much integrated content, having most of the content crowd-sourced/user-entered. The project does aim to replicate the integrated feel of the user interface and the automated features that can speed up game-play and emphasize the storytelling aspect of D&D.

Roll20 is a free to use website that provides all the basic tools that are required to create a campaign. These features include Character Sheets, a Battle Map, and a Chat System. The Character Sheets use nearly the same layout as the official Character Sheets released by Wizards of the Coast. The Character sheets utilize almost no additional features from being digital instead of paper (namely expression evaluation). There are of course ways to interact with the digital medium like arrow buttons for changing counters or check boxes to indicate proficiencies, but there are barely any features that take full advantage of the digital medium. For example, being able to rearrange stats or being able to evaluate math expressions (i.e. 50+43). Another notable issue with Roll20 is how long it takes for a user to resume their place after the session has timed out. Without a valid session, Roll20 is unable to determine what game or content a user was using before the session timed out. This reliance on session data to determine a user's location on the website causes the user to have to start over from the site's homepage and spend time navigating back to their game.

D&D Beyond is another website that hosts a plethora of D&D information. It aims to be a digital version of the D&D Books that allows you to search through it by providing details about different character builds, monster stats, magic items, weapons, armor, and spells; providing both official licensed content as well as hosting additional home-brew content.

DungeonMastersGuild provides licensed D&D adventures and supplements. This website provides different content than D&D Beyond in that it provides full pre-made campaigns that include monster stats, maps, and story details.

2.2 Outlining Community Needs

With an understanding about the products that currently service the D&D community, we are able to develop an understanding of what features a free-to-play campaign-details management and storage website requires.

From Roll20, we understand the basic features that are needed by the community to play their campaigns. These features focus on multiple ways to store information including the Battle Map and Character Sheets, as well as being able to communicate with other players via chat messages or virtual dice. Roll20 is not optimized for players that are using it regularly; when leaving pages open for extended periods of time, Roll20 is unable to quickly resume a player‘s place in the campaign. From Roll20’s shortcomings, we understand that the project needs to be able to resume with no more information than what is provided in the website’s URL.

Fantasy Grounds provides an understanding of what advanced players require, and how to implement the previously discussed basic features with a customizable and expansive mindset.

From both D&D Beyond and DungeonMastersGuild, we can understand that there are already extensive content libraries available. This broad and available library allows that this project’s Compendium doesn’t have to be initially extensive but could be focused on crowd sourced and user generated content that gets more expansive as the user becomes more engaged.

2.3 Best Practices For Designing An Interactive Storytelling Interface

The user experience impacts a variety of design decisions and interface paradigms. Daniel Green, a professor at Bournemouth University, states that without a well-designed user interface, “accessibility is reduced and systems become restricted to users with the appropriate technical know-how, which can result in a frustrating user experience and can contribute to a reduced rate of adoption by interactive fiction communities.” [6] With one of the main goals of this project being to make introducing new players to D&D easy and intuitive, it is very important to construct a user interface that does not require “technical know-how” and won’t frustrate and disrupt the game play experience. This sentiment is continued in Green’s explanation of Hick’s Law. “Hick’s Law suggests reducing complexity where possible, as the time it takes to make decisions or take actions is altered by the number and complexity of options available… Keeping complexity low can aid [the user interface] and increase productivity.” [6] Hick’s Law is vital in the design and implementation of the Map Editor. The Map Editor provides a large feature-set, so it is vital for the adoptability of the project to reduce complexity where-ever possible.

Everett McKay, author of UI is Communication, states that “a user interface is essentially a conversation between users and a product to perform tasks that achieve users’ goals.” [14] This concept is the cornerstone of how this project was designed. All conversation between the user and the user interface needs to be easy to understand, responsive, and concise. While designing each different system, the intended audience of the conversational interface has to be considered. For example, the Map Viewer’s intended audience is general D&D players. As opposed to DMs, general D&D players may have less experience with D&D and don't need to be overwhelmed with intricate details of the campaign. The Map Editor’s intended audience is the DMs, generally having more experience, and requires an interface targeted towards users with more “technical know-how.”

The two other best practice concepts that aided in the design of this project are the Law of Similarly and the Doherty Threshold. Green states, “The Law of Similarly advises elements of differing functionality to be visually dissimilar, and those that are similar to be treated as related or as a group regardless of physical separation.” [6] The visual grouping reduces ”potential confusion for users and helps them focus on authoring rather than understanding the system.” [6] Green’s explanation of Doherty latency threshold states that “system feedback should be ≤400ms else user attention and productivity can suffer.” [6] These design practices provide necessary advice on designing intuitive user interfaces that don’t impede the user’s experience.

2.4 Understanding The Essence and Invisible Rules of Interactive Storytelling

Markus Montola defines three categories to divide interactive story rules and goals including endogenous, diegetic, and exogenous. Montola defines endogenous as the “rules and goals defined in the game structure,” [16] diegetic as the “rules and goals existing within the fiction of the role-play,” [16] and exogenous as the “rules and goals brought to the game activity by players to give it meaning.” [16] These categories help to outline the different important game details

that need to be considered when developing systems to store this information. Endogenous rules and goals are stored inside the content of the Compendium, the collection of game data and rules providing pre-made rules and goals that the DM can adapt and utilize in their storytelling. Diegetic aspects of the role-playing experience are created by the DM and are needed to keep track of the story progression; this type of content mostly comes in the form of DM notes. Exogenous goals are constructed by the player as they play and are stored in the Character Sheets of each player.

Montola also suggests a number of invisible rules that all interactive storytellers should follow. Montola's first invisible rule to roll-playing states that "role-playing is an interactive process of defining and re-defining the state, properties and contents of an imaginary game world." [16] To allow the fulfillment of this rule of role-playing, the project has to be designed to not only define stored information but easily adapt and redefine the stored information. Another of Montola's rules states that "typically the decisive power to define the decisions made by a free-willed character construct is given to the player of the character." [16] This invisible rule is an important aspect of interactive storytelling, because it explicitly states the type of interaction that a player engages in. A player interacts with the story through a free-willed character construct, where the only one making decisions for that character is that individual player. When designing the project, how the player's interactions with the story need to be considered in terms of what the character in the story is able to perceive. The player needs to be given a complete model of the world that they inhabit so that they are able to act as if they were the character construct.

3. Project Description

3.1 Project and File Management

The project was managed with a loose version of Scrum. Scrum is a "framework for developing, delivering, and sustaining products in a complex environment, with an initial emphasis on software development." [21] Scrum defines a system for taking complex problems, breaking them down into realistically implementable feature-sets called sprints, and finally into small tasks that can be completed in no more than a day. It is essentially an outline for how to create and manage small achievable goals and complete them in a realistic timeframe. Utilizing a Kanban board to collect, sort, and manage tasks, I organized the tasks needed to implement each requirement, described in the following sections, into groups, described in section 3.2, and then into smaller sprints. The sprints include tasks that implement new parts in different groups, adding up to implementing a project requirement. After the design stage, detailed in section 3.3, sprints were reorganized into the steps to implement the individual systems.

File management was extremely important in designing a containerized full stack website built for scalability. The project contains many complex systems, so keeping the project's file structure well organized aids in maintainability and reusability as well as easing initial development. The file management was also crucial to the development of the Docker [9] Swarm, detailed in following sections, by allowing a Docker compose file to construct multiple containers from subfolders within the project directory.

3.2 Building For Scalability

When designing a web-application built to allow large groups of people to play together, scalability and load management are extremely important. To accomplish this goal of scalability, the project utilizes Docker to create multiple containerized systems that implement different aspects of the overall structure. Docker is "a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers." [9] From the definition of Docker, we understand that we can build small targeted virtual machines that can run on either a single machine or across multiple machines without changing the functionality of each application. Linking multiple machines together is called a Docker Swarm and allows for quickly scaling the amount of containers that are created for each task. This integrated scalability means that with only a few configuration changes, the web-application can increase the amount of traffic that can be handled.

3.3 Requirements/Specifications

Utilizing our basic understanding of what features are needed and wanted by the D&D community, we can create an extensive list of requirements for what systems need to be implemented across both the front and back ends of the website. These systems, detailed in Table 1, include the Battle Map, Chat, Character Sheets, User and Game Management, URL Routing, Compendium, Inventory System and Monetization.

Table 1. Project's List of Requirements by System

Battle Map	The Battle Map consists of a Map Editor and a Map Viewer. The Map Editor will include features that help with the development of a map. While the Map Viewer is where a map is played and consists of more interactive features.
Map Editor	The Map Editor is the main page for the DM to create interactive stories. The Map Editor has tools for creating the map. For example, drawing tools include shapes and a pen, and the ability to add, rearrange, lock, hide, snap to grid, and scale images. The Map Editor also has features that allow for easily managing items and layers, allowing for large sections of the map to be hidden, deleted or rearranged.
Map Viewer	The Map Viewer is the central focus of the user gaming experience. The Map Viewer not only has the responsibility of allowing for normal D&D operations, but for the user's experience as well. For example, moving player and monster tokens in the grid, and measuring distances between game pieces, while also adding features to access character information.
Chat System	The Chat is a simple interface that allows users to send to, and receive messages from, one another. You can specify another person, a group, or the DM to send messages to. The Chat is also the place where you can roll virtual dice.
Character Sheets	The Character Sheets are a place to store a characters' stats, backgrounds, and any other information that a player may want to remember about their character. Some of these character details, like hp, would also benefit from being able to use the same syntax evaluator that the Chat uses, allowing damage to be automatically summed, or the addition of a new dice roll when you level up. An important feature that should not be missed is the ability to rearrange the Character Sheet, allowing a player more freedom to write about their character, instead of trying to fit the information into predefined boxes.
Backend User and Game Management	The Backend Server handles the creating and managing of users. The Backend creates a new database for each game, adding default information into the database to begin with. The game's database saves data relevant to the different maps, characters, items, spells or even custom classes. The Backend also includes a database that contains global components that are accessible by everyone. These components include publicly shared maps, items, spells and classes.
Game Specific URL Routing	The project's Backend facilitates server side route management, so that routes are specific to the user. This routing reduces the chance of a user editing their URL to get another game's information as well as reducing the amount of time required to resume gameplay after a session timeout.
Compendium	The Compendium is the main resource for searching for information on items, monsters, and spells. The Compendium's content is a mix between crowd-sourced D&D content and custom user submitted content.
Inventory System	The Inventory System is an extension of both the Character Sheets and the Compendium and facilitates the transferring and storage of in-game items. Oftentimes players would have to copy information about an item their DM gave them into whatever online resource they are using. The inventory system will allow for a DM to search the item in the Compendium and instantly transfer the item to the player. The player would then be able to view all of the details of an item along with the amount and any other information about the item they would like to add (i.e. context about the item in the current campaign). The DM will not be limited by the items already in the Compendium because a DM will also be able to easily

	create custom game components that they could later submit to the Compendium.
Monetization	Advertisements will be placed in both the Character Sheets and the Map Viewer as well as alongside the Compendium. The goal of the Advertisements is to demonstrate the feasibility of creating a self-supporting platform.

3.4 Design

When discussing the technical design of this full stack website project, we have to break down the design into both Backend and Frontend sections. The Backend includes all the systems that run on the server hardware and supports the functionality of website Frontend. The Frontend consists of the systems and files that are transferred to/run on the user's browser.

3.4.1 backend

The Backend of the platform consists of several containerized systems built for scalability utilizing Docker Swarm. The swarm consists of five different custom built containers: a Load Balancer, a Session Manager, a SMTP Server, a MongoDB [15] database, and multiple Workers.

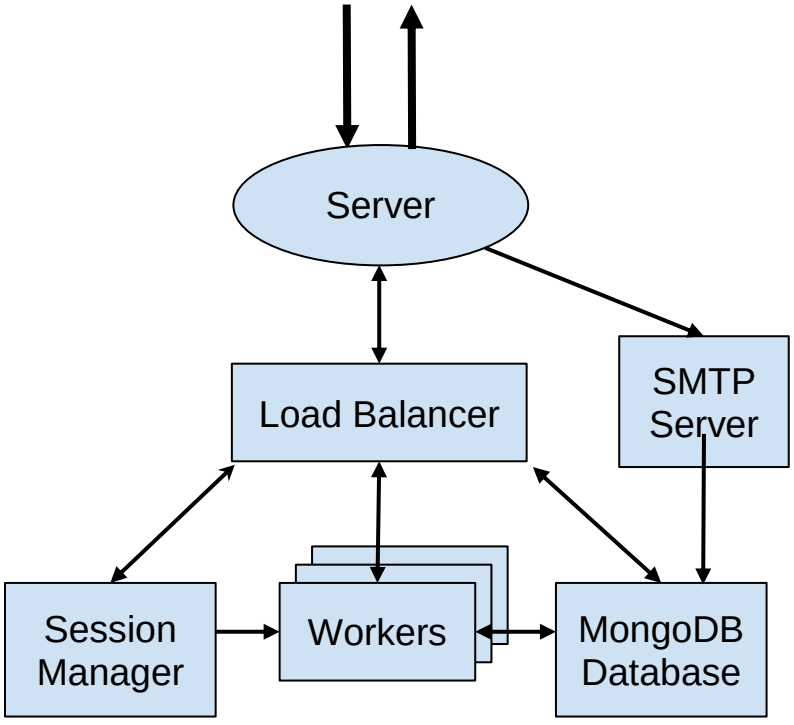


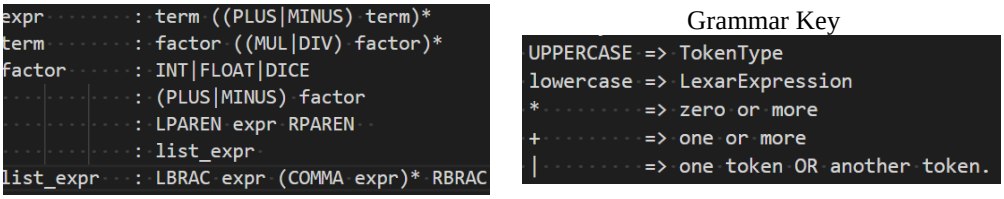
Figure 1. Project Backend Design UML

Figure 1 shows the design of the Backend represented. The UML diagram depicts squares representing the docker containers and arrows representing the direction of data flow. The Server Hardware represents the computer hardware the Docker Swarm is run on and the large arrows represent incoming and outgoing user requests.

Except for the database, all of these containerized systems were built with NodeJs. [1] Utilizing NodeJs to develop the project in a single language allowed algorithms to be consistent across all aspects of development. This algorithmic consistency saved a lot of time and frustration especially with the Battle Map, because the same functions utilized to apply map updates to the map model could be used on multiple different systems. Utilizing an update based system, with a consistent way to apply those updates, reduces the amount of data transferred between the different users and

the Backend. Another important choice when developing the Backend was to use Web Sockets instead of individual HTTP requests. Qigang Liu defines Web Sockets "as a technology that enables web pages to use the Web Socket protocol for full-duplex communication with a remote host." [13] In comparison to HTTP requests, Sockets have very low overhead. After the initial connection request sending all the necessary information on the sender, sockets messages consist of little more than the data that is transferred. The advantage of sockets, as stated by Liu, is that the Web Socket protocol "delivers an enormous reduction in network traffic and latency compared to Ajax polling and… HTTP connections." [13]

Table 2. Backend Design Details by Docker Container

Load Balancer	The purpose of the Load Balancer is to evenly distribute incoming requests to one of the many workers that would process the requests. A major feature of the Load Balancer is to manage the SSL communication, decrypting incoming messages before passing them to the workers and encrypting outgoing messages to the user. Another important feature of the Load Balancer is to manage consistent sessions, allowing different workers to access the same session data. A challenging part of implementing sessions was to allow workers to edit session data and have that change take effect immediately for all new incoming requests. This challenge was overcome by having the worker communicate session changes back to the Load Balancer instead of trying to edit the sessions themselves (see Figure 1).
Worker	The Worker receives socket requests from the Load Balancer, processes the request, and sends a response back to the Load Balancer. With the SSL encryption handled by the Load Balancer, the Worker operates using a low overhead and fast socket connection. The Worker handles any of the requests sent by the Frontend, including routing requests; game, map, character, compendium, and user data requests; Chat evaluation; and admin requests. Routing requests are first processed by the Worker based on the session data provided from the Load Balancer. Utilizing this session data, we check whether the player is authorized to access the content. Data requests are the most common requests made to the website. Data requests utilize the session and request data while interacting with the database to retrieve and filter JSON data. Chat evaluation requests are processed by the Worker by calling the dice-integrated evaluated language (DIEL). DIEL was built to aid this project in rolling dice and evaluating expressions and is an interpreted language built in Javascript. Currently DIEL is limited to basic evaluations (Figure 2), because building a full language is a project of its own. The current functionality of DIEL is built to fulfill the virtual dice requirements of the project. In the future, DIEL could be expanded to interact with the Battle Map allowing the creation of advanced functionality. With limited access to site information, including character and map data, DIEL can provide users a secure way to create custom functionality in their campaigns.  Figure 2. Basic DIEL grammar. Admin requests aid in the development and management of the website. These requests check the session data to make sure a user is logged in and has the correct permissions. Once a user is verified, they are able to access information including the mailing list, reported issues, database and storage usage, and emails from the SMTP server.

Session Manager	The Session Manager exposes a socket connection that processes requests for updated session information. Even though each request through the Load Balancer gets updated socket information, the Session Manager can also be accessed by the Workers and the SMTP Server. Worker access is especially useful when performing long operations where a user's authorization may change.
SMTP Server	The SMTP Server allows for the integration of emailing services into the automation of the website, allowing users to subscribe to the mailing list with subscribe@dndengine.com and unsubscribe using the email unsubscribe@dndengine.com. Administrators can also access emails that are sent to an email based on their username.

3.4.2 frontend

The Frontend of this platform was built using React. React allows you to "build encapsulated components that manage their own state, then compose them to make complex UIs." [22] Component-based architecture for the Frontend was chosen because the design and content of the Compendium, Character Sheets and more importantly the Battle Map all heavily rely on the JSON data received from over socket communications from the Backend. Another reason for using React was that "since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM." [22] This feature was appealing because, even though the data handled on the Frontend was either backend-verified user specific or public information, it meant that the Character Sheets, Compendium, and Battle Map could be quickly regenerated when receiving new data. The Frontend includes three major systems in addition to the static website pages: the Character Sheets, Compendium, and Battle Map.

Table 3. Frontend Systems Design Details

Character Sheets	The Character Sheet page retrieves both the game and character ID from the URL and uses them to make a request to the Backend. The Backend verifies that the user is allowed to access the character information and then returns a dictionary of canvas components. Making the character information a dictionary, instead of an array, allows each component to be referenced by a static key instead of an index. Having the components referenced by a key makes the JSON character information more human readable, but also makes it easy to integrate with DIEL. Eventually, DIEL could be built to programmatically access character details, providing users a way to create custom automated functionality for their characters, whether that be in the form of complex multi-attack actions, or spells that apply area of effect components to the Battle Map. One of the benefits of a digital format over traditional paper is that you could rearrange the components after creation, allowing you to adapt and customize how your character information is displayed as you play. To accomplish component rearrangement, each component contains positional attributes that describe where to place the component on each tab. Another benefit of using a digital format is that each component can be evaluated using the same system that the Chat uses. This evaluation is accomplished by sending a request to the Backend and updating the character information based on the response.
Compendium	The Compendium system is a React component that will be reused on multiple pages, and is built to be versatile to fulfill the requirements of each situation. The Compendium page gave rise to a couple of major design challenges. The Compendium consists of many different content types including items, spells, monster stats, and pre-made map components. To accomplish this multi-type list, all components needed standardized attributes that signified how to render the specific component. React was exceptionally useful in this case, creating separate components for each content type and rendering the component based on its attributes.

Battle Map	The Battle Map consists of two separate pages, the Map Viewer and the Map Editor. Each of these pages was built using a Base Map system that renders components to multiple individually rendered HTML canvases. Separating out each layer onto its own HTML canvas reduces the rendering time during user interactions by only re-rendering layers that had been interacted with. While developing the custom engine, Daniel Kee's article, titled *Comparing interactive web-based visualization rendering techniques*, informed both the decision on which technologies to utilize and the choice to adopt an individually rendered multi-canvas design. [11] The Map Viewer page is built utilizing this Base Map system but overwrites the user interactions. By overwriting the user interactions including mouse interactions and keyboard presses, the Map Viewer can change how the user interacts with the Base Map. The Map Viewer's user interactions are focused towards making a simplistic interface that keeps users engaged in the story. The user is able to select and move unlocked components, move the camera position, measure in-game distances, mark locations to communicate with other users, and send chat messages to log player interactions and roll virtual dice. The Map Editor page also utilizes the Base Map system with updated user interactions. The Map Editor's interactions are focused on allowing the user to build out a map quickly and easily. The creativity of interactive storytelling often comes from the feedback of the players, so the Map Editor needed to be designed to support the DM's need to quickly build creative maps on the fly.

3.5 Implementation

Discussing the implementation of the project's design will be broken down into both Backend and Frontend sections, similar to how the design was discussed.

3.5.1 backend

The containerized Backend was sectioned into a few separate stages of development. Initially, the Load Balancer and Worker were built. Through challenges, the need to maintain user sessions and consistent file access required the creation of the Session Manager. Finally, the SMTP server was built later in development to allow the site to receive emails, adding mailing list automation to the project. The implementation of the containers are discussed in Table 4.
.

Table 4. Backend Implementation Details by Docker Container

Load Balancer	The Load Balancer is built with ExpressJs [7] and Socket.io. [17] ExpressJs is a NodeJs package for handling HTTP and HTTPS requests. Socket.io is also a NodeJs package and allows for fast inter-container communication. In addition to these two major packages, I also utilized Express Sessions [8] and Express Fileupload [5] to assist in the management of user sessions and custom image uploads respectively. The Load Balancer starts by using DNS to search for available workers on the Docker Swarm. Once socket communication has been established with each of the workers, the Load Balancer accepts not only both HTTP and HTTPS requests on their respective ports but also Socket.io connections from users after the website is loaded. When the Load Balancer receives a request, it decrypts the message with the SSL certificate and then parses the request header into a JSON format. Making sure to attach the sending session information and whether the request is sent from a mobile device, this newly generated JSON data is then sent over one of the sockets to a Worker to be processed.
Worker	The Worker utilizes Socket.io to receive requests from the Load Balancer. As discussed in section 3.3.1, the Worker handles any of the requests sent by the Frontend, including routing requests; game, map, character, compendium, and user data requests; Chat evaluation; and admin requests.

	The routing requests' first step is checking if the URL requested is locked, redirecting to the login page if the user is not logged in or to the homepage if they are logged in but not authorized to access the page. The next step checks whether the route is for a React static file and responds with the static build file, careful to restrict file access to only subfolders. The second to last step is to process the request's URL for user specific file requests. Files can be requested from two main locations, DnDEngineAsset and DNDEngineGlobalAsset. DNDEngineGlobalAsset URLs respond with global assets stored on the server based on the path specified in the URL. DnDEngineAsset URLs respond with files stored on the server with a path based on the URL and the session information of the user sending the request. The final step is to return the React build index.html if none of the other routing options are triggered. Data requests are the main requests made to the Backend because they manage how data is accessed and manipulated. All of this project's systems require at least basic access to information stored on the database, but most of them also require the ability to change the stored information as the users play their campaigns. Chat evaluations are a special kind of data request that, in addition to accessing and manipulating stored data, call the dice-integrated evaluated language (DIEL) interpreter to evaluate the received request data. The DIEL interpreter is built on top of NodeJs to provide easy integration with the website and in the future will interact with the website's database to provide users with a specialized language to build custom website functionalities. DIEL, while being built using Javascript, does not utilize Javascript event based architecture and is more similar to the Python interpreter. Admin requests are also a special kind of data requests, requiring users to have a certain authorization level to access. Admin requests include actions like promoting users, accessing database statistics, and editing site defaults. The promoting users action allows current admins with the correct permissions to promote another registered user to any admin level lower than their current one. The accessing database statistic action generates a JSON formatted structure of the database with usage statistics on how much storage space each campaign or user requires. The final action, editing site defaults, allows admins to access and adjust the default databases that are utilized when creating new campaigns and content.
Session Manager	The Session Manager, like the Worker, is a socket based server that accepts connections from the Load Balancer, Workers, and SMTP Server. The Session Manager implements a custom version of the Express Sessions store, with the functionality to get, set, and destroy session data. The storage of session data uses a simple Javascript dictionary with the session ID as the key.
SMTP Server	The SMTP Server is built utilizing a NodeJs package node-mailer, [18] specifically the smtp-server [19] wrapper implementation. Once the SMTP Server is connected to the MongoDB database, the smtp-server is set to listen on multiple ports. When an email is received, it is parsed using the NodeJs package mailparser [20] and then added to the database.

3.5.2 frontend

The React Frontend was sectioned into a few separate major systems, discussed in Table 5. Initially, however, a couple of general features, and the challenges they produced, need to be addressed.

The first of these general Frontend features is Google account binding. Xi Gao's article published in the Journal of Information Security and Applications, titled *A research of security in website account binding,* was useful in assuring that the Google account binding for this project is as secure as possible. [4]

In addition to Google account binding, the project also implements Google Analytics. A challenge that arose while adding Google Analytics was that the page navigation was handled by the React Router, [10] which handles routing changes locally without any new requests to the server. This caused Google Analytics to only be able to detect the initial page load and not any subsequent page navigations. To overcome this issue, the project utilized Google's GTag package that allowed the website to send Google a page view notification whenever a new page is selected.

With GTag sending the correct event notifications to Google Analytics, Google Adsense could then be implemented. Implementing Google Adsense was straightforward once obtaining a client ID from Google. A React component was created that managed the type of ad displayed, as well as the styling of the ad container. These ad components were then placed throughout the Frontend of the application. One challenge that arose was how to integrate advertisements into the dynamic content of the Compendium. To overcome this challenge, the React ad component was integrated into the generation of Compendium components after any filtering has been applied. This placement in the generation processes allows for advertisements to be placed based on the shown components, resulting in ads being placed 10 components apart but won't show any ads when there are less components than the minimum. A minimum number of components to display advertisements improves the clarity of the interface when a user is searching for a specific component.

Another general feature of the website is that it uses css variables to maintain consistent color scheme and styling between pages and allows for page content to be scaled relatively. After general features, the major system of the Frontend - including the Character Sheets, Compendium, and Battle Map - can be discussed.

Table 5. Frontend Systems Implementation Details

Character Sheets	The Character Sheet page is built by generating absolute positioned DIV tags for each component and placing them into one of the relatively positioned wrapper DIV tags based on the component's "tabs" parameter. The wrapper DIV tags are then individually rendered based on which tab is currently selected by the user. The character components generation leaves the character details severely disconnected, making it difficult for information to be updated and shared between the different components. To overcome this challenge, each component was passed a reference to a shared character data model. With functions to edit character information and event listeners called when different types of components are changed, the shared character data model manages which components need to be updated and re-rendered. The shared character data model also manages sending and receiving character information updates with the Backend and, by extension, other users.
Compendium	The implementation of the Compendium had many challenges. The first major challenge was to build one page that could be accessed by different URLs to change the interface design and filter components by their content types. This complex URL navigation was aided by the React Router package that allows for fine control over how URLs affect rendering. The Compendium page attempts to parse a game ID and a subtype from the URL. The Compendium fetches game content if a game ID is found but fetches global content otherwise. If the subtype is found, the Compendium renders the list filtering the content and displaying the correct labels for the subtype. If the subtype is not found, the Compendium doesn't apply any filters and displays basic labels.
Battle Map	As discussed in section 3.3.2, the Battle Map is designed and implemented in three separate parts. These parts include the Base Map, the Map Viewer, and the Map Editor. The Base Map consists of a React component that first requests the full current map, and then facilitates applying and transferring small map updates. Utilizing an update system that consists of a path to the changing data and the new data, we can reduce the amount of redundant data that is transferred resulting in less latency between all of the players. Another major feature of the Base Map is the infinite dimensional capacity of the map allowed by utilizing a custom built two dimensional graphics rendering engine. The custom rendering engine projects the screen into the map's world space based on a virtual camera's three dimensional position. The camera's position being three dimensional allows the virtual camera to zoom in, adjusting the relative screen size of each component. An important component of the rendering process was calculating the world position given a screen position and the world camera. The equation below (Equation 1) requires three parameters including a screen position, often originating from a mouse event, camera position and camera zoom. Equation 1 returns a world position that can be compared to the map component positions that are defined in terms of world coordinates.

	World Pos = Screen Pos / Camera Zoom + Camera Pos (1) Using the calculated world position of each map component, the rendering engine filters out map components that don't appear on screen and then render each layer onto a separate HTML canvas. With this rendering setup, map components can be placed anywhere on the map that can be specified by two Javascript numbers, which can safely represent up-to roughly plus or minus 9 quadrillion. While this value is not technically infinite, it should be larger than any standard campaign map should require. The Map Viewer and the Map Editor are both component wrappers of the Base Map that implements different user interactions. The user interactions are dictated by which tool is selected. Many of the tools function by either applying updates to the map or open menus that change different map or tool configurations. The Edit tool is a great example of both of these functionalities. You are able to click on the Edit tool button and edit the configuration settings for what update is applied when using the tool. With the Edit tool selected, clicking on the map will create a new update based on the tool configurations. As you move the mouse, new updates will be applied to the same path as the newly created update.

3.6 Required Resources

A major goal of this project was to create a platform for D&D that minimized the barriers for new players. By making this project web-based, new players can start playing D&D without buying or installing anything.

3.7 Testing/Validation

To validate the functionality of the project, unit tests were drafted to validate each component as well as performing integration testing in later stages of development to validate inter-component communication. While developing validation tests for the project, it was vital to have a security focused mindset. Ron Lepofsky's article, titled *The Manager's Guide to Web Application Security*, provided insight into how to approach security testing and designing products that have foundational security integration. Lepofsky states that "a straightforward way of reporting both security vulnerabilities and compliance violations is by using a table showing the correlation of security vulnerabilities with compliance violations. Several security regulations and standards can be referenced in one table along with security vulnerabilities." [12] Lepofsky then provides an example of a table containing the most common security vulnerabilities and compliance violations that was useful in developing tests of the project's overall security.

3.7.1 unit testing

The purpose of the unit testing was to assist in the development of the project by allowing the quick confirmation that individual components function as expected. This improvement to the development experience also assists in catching accidental changes to components, so faulty existing components don't contribute to the development time of new components.

Unit tests were created using the MochaJs testing framework in tandem with the assertion library ChaiJs. MochaJs provides the framework to write and execute tasks while ChaiJs aids in the human readability of tests. ChaiJs uses explicit chains of functions and attributes to provide sentence-like structure to the unit test assertions. This structure improves readability by making the test code not only self-documenting but also more intuitive.

3.7.2 integration testing

The purpose of the integration testing is to smooth the user experience, catching problems in inter-component communication and problems with re-rendering components. To assure that integration testing was consistent throughout development, simple user stories were created. These user stories detailed the expected paths that users might take while navigating through the different systems. For example, integration testing was extremely important for developing the campaign invitation system because the invitation system would start from a URL, navigate to sign up or login based on the session authentication, then redirect the new player straight into the campaign. The testing

assured that the intended campaign information, and the authenticated session data, is carried through each of these systems.

3.8 User Verification/Feedback

In the later stage of development, a small group of volunteer testers provided feedback on many occasions. The project was tested by four different D&D enthusiasts, providing feedback on the user interface and needed features for easy map creation. A few specific examples of user feedback that were integrated into development include improvements to the hitbox of the navigation bar items and the ability to rotate map components.

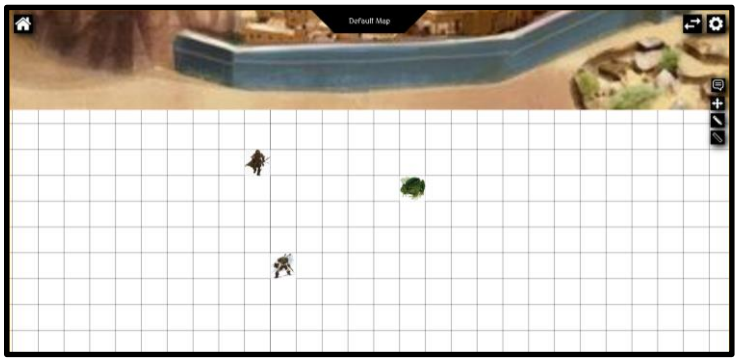
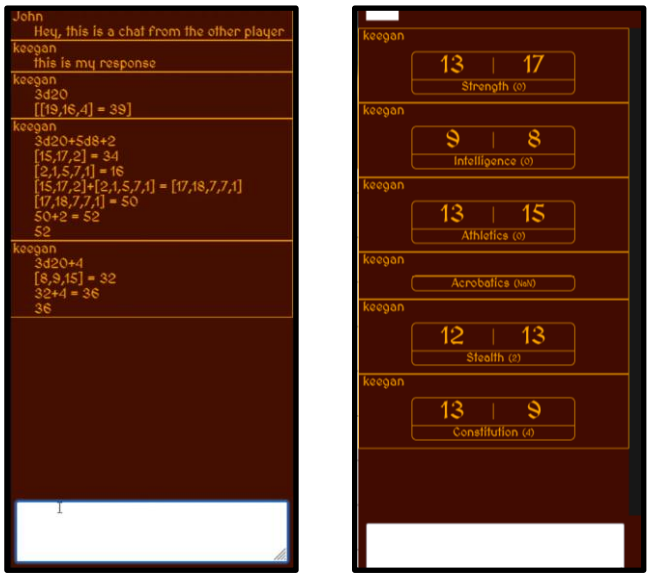
4. Conclusion

4.1 Current Functionality of Application

The project currently implements all of the systems defined in section 2, including the Battle Map, Chat, Character Sheets, User and Game Management, URL Routing, Compendium, Inventory System and Monetization.

Table 6. Accomplished Functionality by Project Requirement

Battle Map	The Battle Map is implemented on both the Frontend and Backend. The Backend consists of data requests that provide the initial full map data as well as map updates. The Frontend consists of two pages, implementing the Map Editor and Map Viewer systems.
Map Editor	The Map Editor is a page on the website accessed by opening a map and enabling editing. The Map Editor has tools for creating the map, including drawing tools, an “add component” menu, a grid menu, and a layer menu. The drawing tools include a menu that allows the user to change the tool’s configurations, producing both a pen and shapes of different colors and line thicknesses. The ability to add, rearrange, lock, hide, snap to grid, and scale components is accessed by right clicking on the intended target component and using the adjustment menu. The Map Editor also has the three menus that allow for easily managing items and layers, the aforementioned “add component,” grid, and layer menus. The layer menu allows for large sections of the map to be hidden, deleted or rearranged as well as changing the render order of the component.  Figure 3. Frontend Map Editor
Map Viewer	The Map Viewer is also a page on the website accessed by selecting a campaign, and opening one of the maps listed. The Map Viewer allows for the moving of player and monster tokens around the map, and measuring distances between game pieces based on the grid.

	 Figure 4. Frontend Map Editor
Chat System	The Chat is a simple interface that allows users to send and receive messages from one another. You can specify another person, a group, or the DM to send messages to. The Chat can also roll virtual dice utilizing the DIEL interpreter.  Figure 5. Frontend Chat System
Character Sheets	The Character Sheets are a place to store a characters' stats, backgrounds, and any other information that a player may want to remember about their character. Each character component can specify whether its value should be evaluated, sending an evaluation request to the Backend whenever the value is changed. The important feature of rearrangeable Character Sheet components, implemented with absolutely positioned DIV tags, allows a player complete customizability of how their character information is stored and visualized.

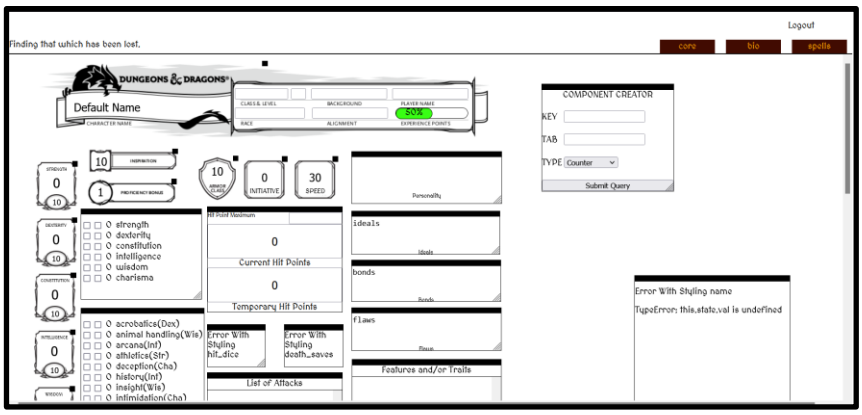
	 Figure 6. Frontend Character Sheets
Backend User and Game Management	The Backend Server handles creating and managing users through the Backend data requests detailed in section 3.3.1 and 3.4.1. The Backend does create a new database for each game, adding default information into the database. The game's database saves data relevant to the different maps, characters, items, spells or even custom classes. The Backend includes a "Compendium" database that contains global components that are accessible by everyone. These components include publicly shared maps, items, monster stats, spells and classes.
Game Specific URL Routing	The website utilization of React Router and the Backend route management allows for routes specific to the user. This routing restricts a user's ability to access another game's information. This Frontend React Router and Backend management places user specific information in the URL during the website redirection process. By injecting user specific information into the URL while the user is logged in, the project reduces the amount of wasted time after a session timeout by only requiring a user to login before the systems have all the necessary information to resume gameplay.
Compendium	The Compendium system is accessible both through a public website page or on through a campaign's details page. If the compendium system is accessed via the public page, The Compendium's content will primarily consist of crowd-sourced D&D content. If the compendium system is accessed via a campaign's details page, it displays the DMs curated content specific to the campaign. Optionally, the campaign specific Compendium can also include global Compendium content.  Figure 7. Frontend Compendium

	Figure 7 shows the current state of the global compendium.  Figure 8. Frontend Campaign Specific Compendium Figure 8 shows the current state of the campaign specific Compendium described above. Note the added options at the top of the Compendium to allow players to add content and toggle the inclusion of global Compendium content.
Inventory System	The Inventory System is an extension of both the Character Sheets and the Compendium and facilitates the transferring and storage of in-game items, allowing the DM and players to drag and drop items from the Compendium into a Character Sheet's inventory component. The players are able to view all of the details of an item along with the amount and any other information about the item they would like to add (i.e. context about the item in the current campaign).
Monetization	Advertisements are placed in both the Character Sheets and the Map Viewer as well as alongside the Compendium content.

4.2 Continued Work

This project has implemented a great flexible foundation for creating an expansive feature-set focused on customizability. From the solid foundation built during this project, this project can continue to expand the feature-set.

One specific feature that can be expanded and built upon is the DIEL interpreter. As stated in section 3.4.1, DIEL can be expanded to provide a secure way to interact with the website's database to provide users with a specialized language to build custom website functionalities.

Another aspect of the website that can be improved with continued work is the Battle Map. A small improvement to the Map Viewer would be to be able to view a character's basic information by right clicking on a map token. A significant feature that would be implemented with more time is interactable map components. For example, a map button component would allow the DM to create interactable buttons with custom functionality. Map button components could be used to navigate players between different maps, spawn tokens, play animations, or open in-game menus.

The Character Sheets can also be improved with additional development. While the current Character Sheet component includes all the necessary types for recreating the original D&D character sheets, the types can be expanded to include advanced components. For example, a Character Sheet party-money component could facilitate the management of funds shared between multiple characters.

4.3 Summary

This project serves as an alternative to a traditional pen and paper feel, with added benefits from the utilization of a

digital medium, providing the best experience for making creative and interactive stories. With a focus on intuitive design for the Map Viewer, new players will have a straightforward experience. While the Map Editor will have a feature-set that allows DMs to build creative and extensive stories.

The current website creates an inclusive experience for inexperienced users with a free to use website that can support their creativity, while also providing experienced users with a plethora of features to fully customize their game-play experience. This dual purpose experience is accomplished by separating these intended audiences and providing only the necessary features for each group, reducing the cluttering and confusion that could be caused by trying to service both audiences with one interface.

The website's current user interface is designed and implemented to allow the DMs to be fully able to follow Montola's invisible rules of role-playing, enhancing the DMs ability to intricately detail an immersive interactive fantasy world. Specifically, the website is able to neatly store Montola's three main types of game rules and goals, endogenous, diegetic, and exogenous.

Overall, this project effectively accomplishes its goal of being an intuitive experience for introducing inexperienced players to the wonders of interactive storytelling, while also empowering and facilitating experienced players to tell creative and immersive stories.

5. References

[1] Dahl, Ryan. NodeJs, OpenJS Foundation, 27 May 2009, Software. https://nodejs.org

[2] Davison, Doug. Fantasy Grounds, SmiteWorks USA LLC, 2004, Software. https://www.fantasygrounds.com/home/home.php

[3] Dutton, Riley et al. Roll20. The Orr Group, 2012, Software. https://roll20.net/welcome

[4] Gao, Xi, et al. "A research of security in website account binding." Journal of Information Security and Applications 51 (2020): 102444. https://doi.org/10.1016/j.jisa.2019.102444

[5] Girges, Richard, Express Fileupload, 2020, Software. https://www.npmjs.com/package/express-fileupload

[6] Green, Daniel, Charlie Hargood, and Fred Charles. "Contemporary Issues in Interactive Storytelling Authoring Systems." *Interactive Storytelling.* Springer International Publishing, Cham, 2018. http://eprints.bournemouth.ac.uk/31466/1/icids-paper-59.pdf

[7] Holowaychuk, TJ et al. ExpressJs, StrongLoop, 16 November 2010, Software. https://expressjs.com/

[8] Holowaychuk, TJ et al. Express Sessions, StrongLoop, 2014, Software. https://www.npmjs.com/package/express-session

[9] Hykes, Solomon. Docker, 18 November 2021, Software. https://www.docker.com/

[10] Jackson, Michael. Remix Software, 2014, Software. https://reactrouter.com/

[11] Kee, Daniel E., Liz Salowitz, and Remco Chang. "Comparing interactive web-based visualization rendering techniques." Poster Proc. IEEE Conf. InfoVis. 2012. https://valt.cs.tufts.edu/pdf/kee2012comparing.pdf

[12] Lepofsky, Ron. The Manager's Guide to Web Application Security: A Concise Guide to the Weaker Side of the Web. Apress L. P, Berkeley, CA, 2014. https://go.exlibris.link/4kgq1Ydz

[13] Liu, Qigang, and Xiangyang Sun. "Research of web real-time communication based on web socket." (2012). https://www.scirp.org/html/1-23101_25428.htm

[14] McKay, Everett N. UI is communication: How to design intuitive, user centered interfaces by focusing on effective communication. Newnes, 2013. https://learning.oreilly.com/library/view/ui-is-communication/9780123969804/?ar=

[15] Merriman, Dwight et al. MongoDB. DoubleClick, 2007, Software. https://www.mongodb.com/

[16] Montola, Markus. 2009. "The Invisible Rules of Role-Playing: The Social Framework of Role-Playing Process." International Journal of Role-Playing 1(1): 22–36. http://www.ijrp.subcultures.nl/wp-content/uploads/2009/01/montola_the_invisible_rules_of_role_playing.pdf

[17] Rauch, Guillermo. Socket.io, Automattic, 10 March 2021, Software. https://socket.io/

[18] Reinman, Andris. Node Mailer, 2014, Software. https://nodemailer.com

[19] Reinman, Andris. SMTP Server, 2015, Software. https://www.npmjs.com/package/smtp-server

[20] Reinman, Andris. Mail Parser, 2018, Software. https://www.npmjs.com/package/mailparser

[21] Schwaber, K. and Sutherland, J. (1991) The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game. www.scrum.org

[22] Walke, Jordan. React, Facebook, 2013, Software. https://reactjs.org/