Matthew Flaig

Project Title: Roll for Encounter



(Bustrillos, 2019)

# Executive Summary

Dungeons & Dragons has been one of the best known and best-selling fantasy tabletop role-playing games since the late 1970’s (Hite, 2006). Its fame and market reach continue to stretch as it has grown into over a billion-dollar industry (Waters, 2004). As Dungeons & Dragons is traditionally played in person, the quarantine caused by COVID-19 would have predictably slowed interest in the product. The opposite however happened, and revenue was up 35 % in 2020, closing out their seventh consecutive year of growth (Pravini, 2021).

The expansion seen during the COVID-19 quarantine is contributed to virtual play. Numerous websites and applications hit the market to fill this need. These sources attempt to replace the aspects tied to the vital elements associated with the in person play style.

Dungeons & Dragons now being played virtually is no longer tied to the hardcover printed rulebooks it has used for decades. Now available in digital form, players can access all materials and play without leaving their computer desk.

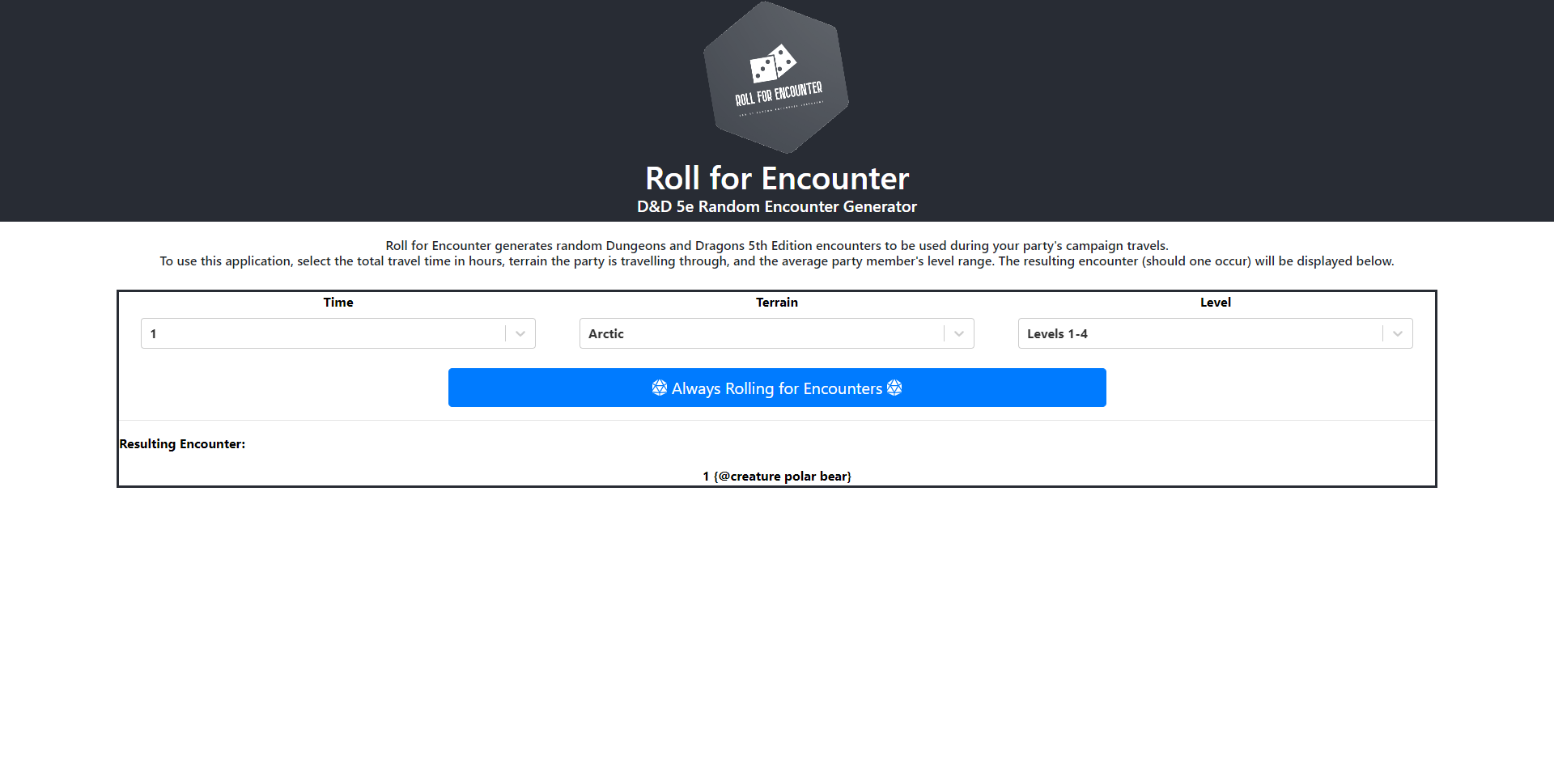
This is where Roll for Encounter enters the scene. Within the Dungeons & Dragons rulebooks exists numerous spreadsheets for creating random encounters. These encounters are based on probabilities, length of travel time, location environment, and player character levels. Determining and building the encounters are extensive and time consuming as they entail consulting several rulebooks, rolling numerous dice, and defining the monsters’ statistics blocks. Roll for Encounter can complete these calculations instantly.

Roll for Encounter is an application that works within the constructs the Dungeons & Dragons rulebooks outline. The application will allow the user to input or select a few data elements to establish what type of encounter needs to be generated and then the application with randomly select the appropriate encounter. The user will then be returned the data regarding the encounter and the monsters involved.

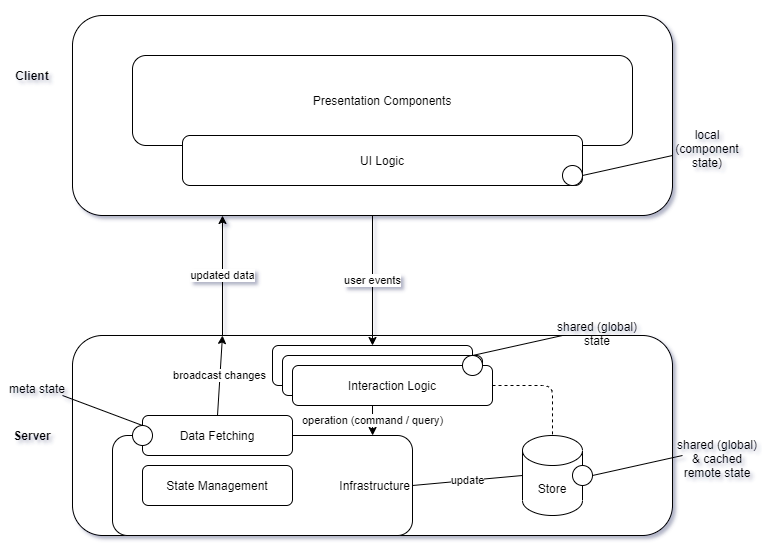
Roll for Encounter is designed to be used for both virtual play and tabletop play. The goal is to allow Dungeon Masters, the game organizer, the ability to seamlessly generate random encounters without the need for breaking gameplay to manually calculate the specifications. The application is expected to save Dungeon Masters an average of five to ten minutes per encounter creation.

# Scenario | Generate Random Encounter

While playing the tabletop role-playing game Dungeons & Dragons, characters often need to travel long distances over extended time frames. During these travels there is a chance that a random encounter can ensue. By entering the length of travel time, the terrain the characters are traveling through, and their average level range, a Dungeon Master can instantaneously determine if a random encounter happens and if so, what that encounter involves.



# System Architecture



## Source Code Structure

Source code structure introduction. The following is a summary of the source code directories and their contents:

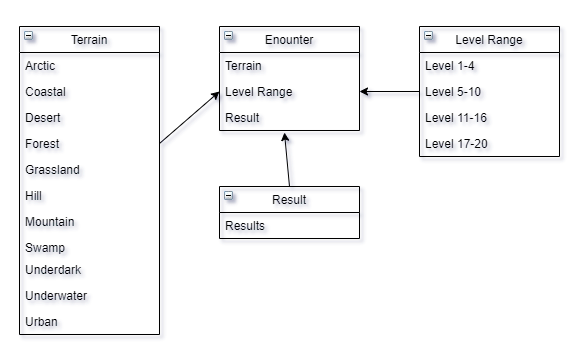
|  |  |
| --- | --- |
| **Code Directory** | |
| **Directory** | **Usage** |
| /client | Frontend of application |
| /client/node\_modules | Dependencies |
| /client/public | Public facing HTML and images |
| /client/src | JavaScript components of user interface and CSS |
| /client/src/graphql | GraphQL query to backend |
| /server | Backend of application |
| /server/node\_modules | Dependencies |
| /server/src | GraphQL components of data store |
| /server/src/datasources | Data Store |
| *Highlighted rows indicate directories containing source code.* | |

# Code Architecture

The application works by allowing the user to select three variables. These variables are then sent to the server where it queries the data store for entries that match its criteria. The server then calculates the percent likely hood of success and if successful selects a random entry of the previously selected data. This data is then returned to the user.

## Database or Data Store

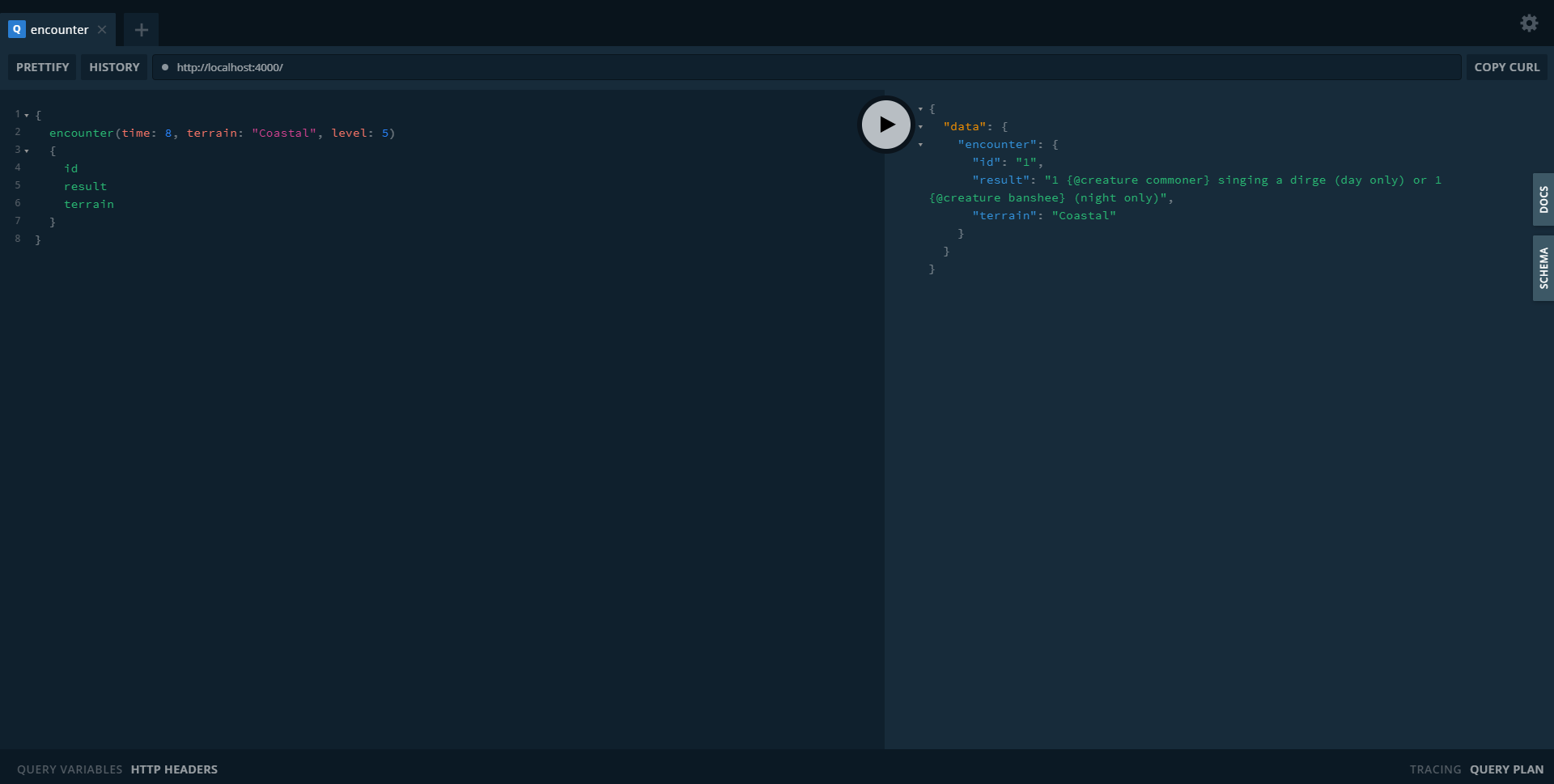
Introduction to the database or data store.



The data store is stored in a flat file database format of JSON. The frontend queries this RESTful API backend to then deliver the returned data.

## Views, Stored Procedures and User Defined Functions

After starting the server, you can interact with the data store on localhost:4000. The primary query for Roll for Encounter is encounter. It takes three arguments: time (Int), terrain (String), and level (Int).



Programming Language | JavaScript & GraphQL

The application is written prominently in JavaScript. On the client side there are additionally small portions of HTML and CSS. The server side is additionally JavaScript with GraphQL mixed in.

Project Classes

The project utilizes these classes:

### Primary Class | App.js

App.js is the formatting of the Roll for Encounter site. As the application expands with new features App.js is used to house them and maintain the branding image as it currently does with EncounterGenerator.js.

### Interactive Class | EncounterGenerator.js

EncounterGenerator.js is the interactive layer of the application. When version 2.0 is released, it will be used as the housing for the Monster statistics block.

Project Modules

The project utilizes these modules:

### Component | Select

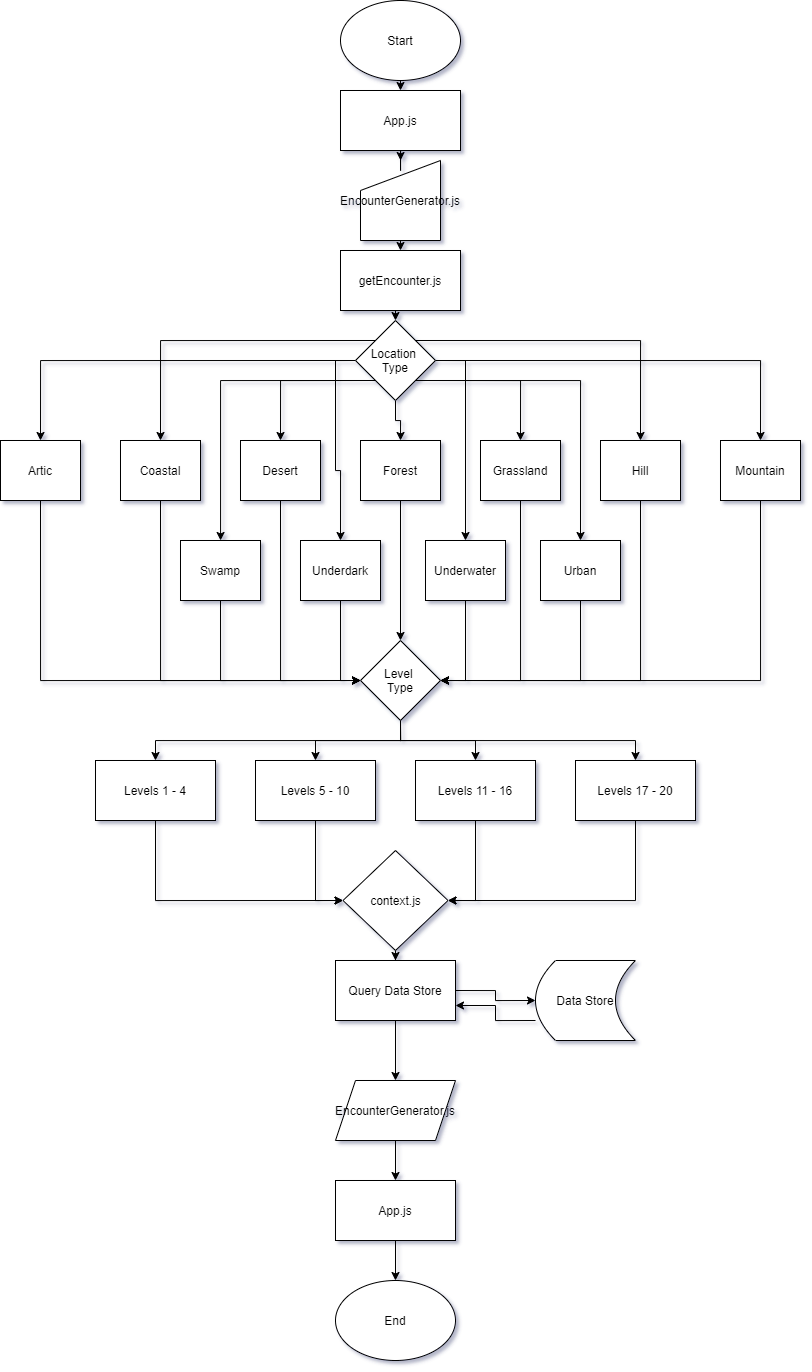
Select allows for the reusability of input validation from the user

### Component | Button

Not actually used as a button as the application performs live queries this is used as a design element

Program Start and End Flow

Program flow of the web based application.



Summary

The web application Roll for Encounter is a tool for Dungeon Masters to create random encounters that logically apply to their game’s needs. Breaking the encounters down by location terrain and level requirements allow for specific elements to be established before the randomness of the encounter details to be programmatically produced.

Imagine a fantasy scenario where a group of adventures are climbing an ice-covered mountain only to be attacked by giant frogs. Does this make sense? Why would amphibians exist in such cold high-altitude location? The correct answer is that they would not.

Roll for Encounter eliminates the result of completely bizarre and random encounters and instead focuses the randomness into location funnels. The application takes this a step further and refines by level requirement.

Imagine the same fantasy scenario on the mountain, but now you are attacked by a literal god of the mountain. This would be a difficult encounter for even the most hardened of adventures but imagine if you were still at a low or mid-level. Your characters would not stand even the slightest of chances of winning or for that matter running away.

Correctly using leveled monsters and accurate locations, Roll for Encounter can save a Dungeon Master from a headache and five to ten minutes per encounter creation. The application functions by using JavaScript and GraphQL to query data based on the appropriate information the user enters. It then produces a random number to allow for the randomization of the encounter details. This specific yet still random encounter is then returned to the user’s GUI for the Dungeon Master to execute in game.

# APPENDIX B (BUILD AND RELEASE PROCESS)

Implementing updates to this application is an easy process. The repository is hosted in GitHub and available for anyone to fork and contribute. The best method to update this application is to clone the repository and checkout the development branch. Here you can make the necessary changes and test before committing the work. Once you are confident the update is bug free, submit a merge request to have the development branch merged into the master production branch.

# APPENDIX C (CLIENT INSTALLATION INSTRUCTIONS)

This application is not installed on any client. It is a web-based application and therefore only exists on the provider’s servers.

# APPENDIX D (DEVELOPER SETUP INSTRUCTIONS)

To work on Roll for Encounter, follow the instructions in the README.md file located in the repository. To begin, first intall Git, npm, and Node.js. At this point you are ready to clone the repository. Once all files have been downloaded locally to your device, navigate to the client and server directories, and run the command npm start from the terminal on each. This will launch both the frontend and backend servers on <http://localhost:3000/> and <http://localhost:4000/> respectively.