# EldenBot: The Best Elden Ring Companion

- Overview
- About the Game
- Running the Program
  - Logic Structure
  - How to Run
    - Packages to Install
    - Starting the Bot
  - Sample Runs
    - New User
    - Returning User
    - Storing User Models
- Diagram
- NLP Techniques
- Knowledge Base
  - Web Scraping
  - Level Recommendations
  - Building REST API
    - Area and Bosses Recommendations
    - Bosses
    - Builds
    - Classes
    - Comparing Classes
    - Items
    - Stats
    - Weapons
- Evaluations & Analysis of its Strengths and Weaknesses
  - Strengths
  - Weaknesses
    - Duplicate Names
    - Recognizing Names

Understanding Intents

# Overview

EldenBot is a chatbot designed to assist players on their journey through Elden Ring, a vast game filled with numerous hidden secrets. With so much to discover, players often consult the game's Wiki during each playthrough. Fortunately, EldenBot offers a convenient and efficient alternative, allowing players to quickly access valuable information such as boss details, build guides, and location data, thus streamlining the path to becoming Elden Lord.

# About the Game

Elden Ring is an action role-playing game set in a vast open world. Players take on the role of a character who must explore the world, fight challenging enemies, and collect powerful items in order to regain their strength and challenge the Elden Ring. The game features challenging combat, customizable stats and equipment, and a rich story told through the game's environment, characters, and items.

# Running the Program

## Logic Structure

EldenBot was built using a combination of Dialogflow and Python. We used Dialogflow to allow our bot to capture the intent of a user and have it answer the user’s inputs about the game. First, we set up Google Cloud Platform (GCP) authentication and project identification by defining variables for the API key path, setting an environment variable, and creating variables for the project and session IDs. This is so we can interact with a Dialogflow agent - EldenBot.

To get the bot’s responses, there is a function called "get_response" that sends a given text input to the Dialogflow agent and returns a response as a dictionary. The response object is parsed into a dictionary with fields such as the query text, detected intent, fulfillment text, and sentiment analysis results. Although all this information is provided the only information we used is the detected intent and the fulfillment response. The intent helped us decide which backend function to call, and we used the fulfillment response to obtain any parameters we needed.

As the user typed their queries into the terminal, we opted to save the conversation in a temporary file. This information was utilized to compute and exhibit the user's highest TF-IDF scores.

# How to Run

## Packages to Install

You must install the following program via the following

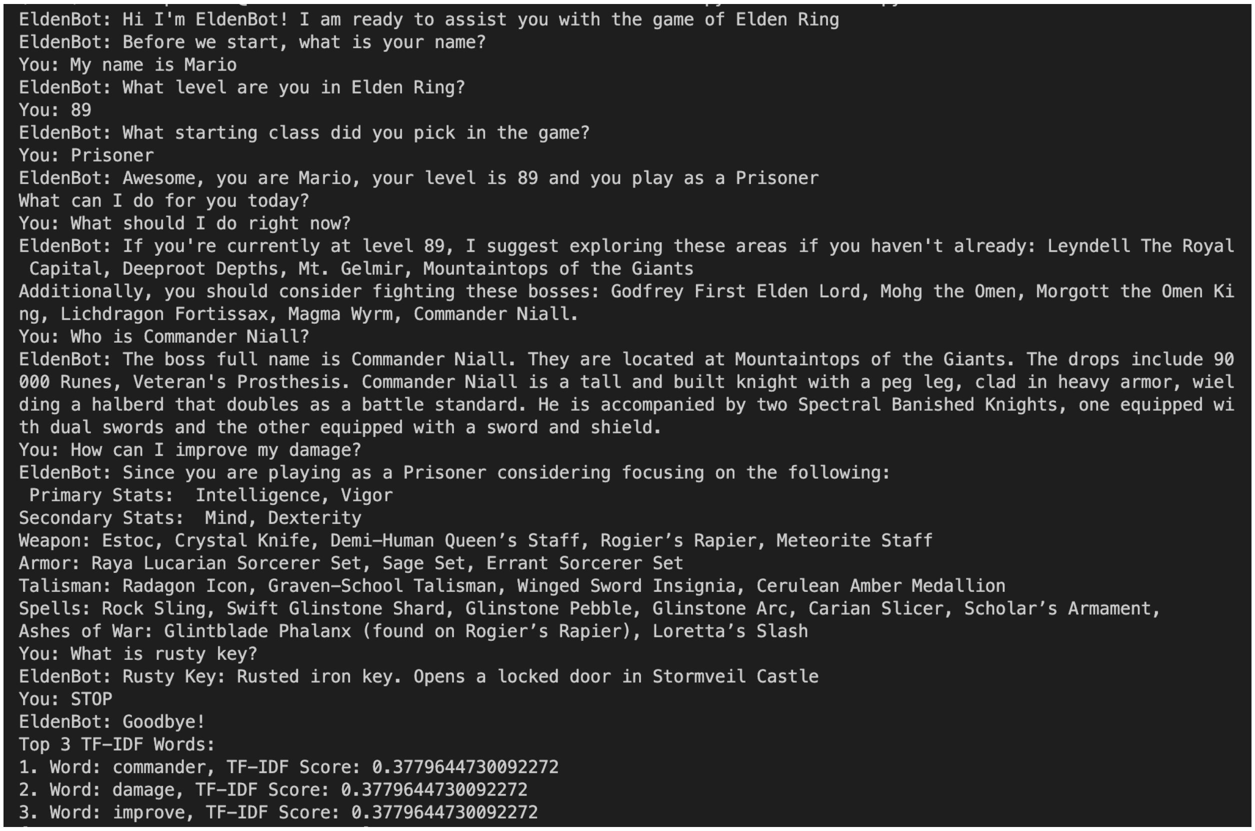
```
pip3 install beautifulsoup4, nltk, numpy, dialogflow, scikit-learn, flask
```

## Starting the Bot

There are two servers running. To run them, type `python3 controller.py` in one terminal and `python3 eldenbot.py` in the other.

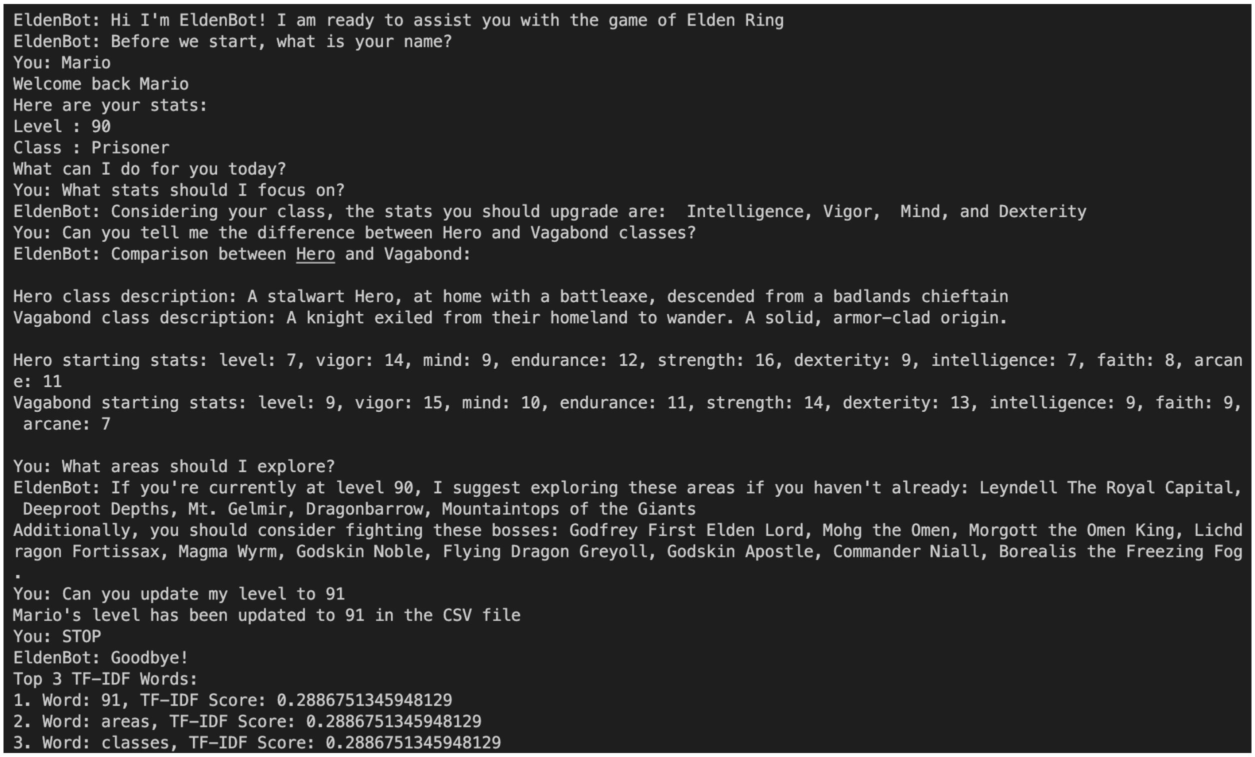
# Sample Runs

## New User



```
EldenBot: Hi I'm EldenBot! I am ready to assist you with the game of Elden Ring
EldenBot: Before we start, what is your name?
You: My name is Mario
EldenBot: What level are you in Elden Ring?
You: 89
EldenBot: What starting class did you pick in the game?
You: Prisoner
EldenBot: Awesome, you are Mario, your level is 89 and you play as a Prisoner
What can I do for you today?
You: What should I do right now?
EldenBot: If you're currently at level 89, I suggest exploring these areas if you haven't already: Leyndell The Royal
 Capital, Deeproot Depths, Mt. Gelmir, Mountaintops of the Giants
Additionally, you should consider fighting these bosses: Godfrey First Elden Lord, Mohg the Omen, Morgott the Omen Ki
ng, Lichdragon Fortissax, Magma Wyrm, Commander Niall.
You: Who is Commander Niall?
EldenBot: The boss full name is Commander Niall. They are located at Mountaintops of the Giants. The drops include 90
000 Runes, Veteran's Prosthesis. Commander Niall is a tall and built knight with a peg leg, clad in heavy armor, wiel
ding a halberd that doubles as a battle standard. He is accompanied by two Spectral Banished Knights, one equipped wi
th dual swords and the other equipped with a sword and shield.
You: How can I improve my damage?
EldenBot: Since you are playing as a Prisoner considering focusing on the following:
 Primary Stats:  Intelligence, Vigor
Secondary Stats:  Mind, Dexterity
Weapon: Estoc, Crystal Knife, Demi-Human Queen’s Staff, Rogier’s Rapier, Meteorite Staff
Armor: Raya Lucarian Sorcerer Set, Sage Set, Errant Sorcerer Set
Talisman: Radagon Icon, Graven-School Talisman, Winged Sword Insignia, Cerulean Amber Medallion
Spells: Rock Sling, Swift Glinstone Shard, Glinstone Pebble, Glinstone Arc, Carian Slicer, Scholar’s Armament,
Ashes of War: Glintblade Phalanx (found on Rogier’s Rapier), Loretta’s Slash
You: What is rusty key?
EldenBot: Rusty Key: Rusted iron key. Opens a locked door in Stormveil Castle
You: STOP
EldenBot: Goodbye!
Top 3 TF-IDF Words:
1. Word: commander, TF-IDF Score: 0.3779644730092272
2. Word: damage, TF-IDF Score: 0.3779644730092272
3. Word: improve, TF-IDF Score: 0.3779644730092272
```

## Returning User



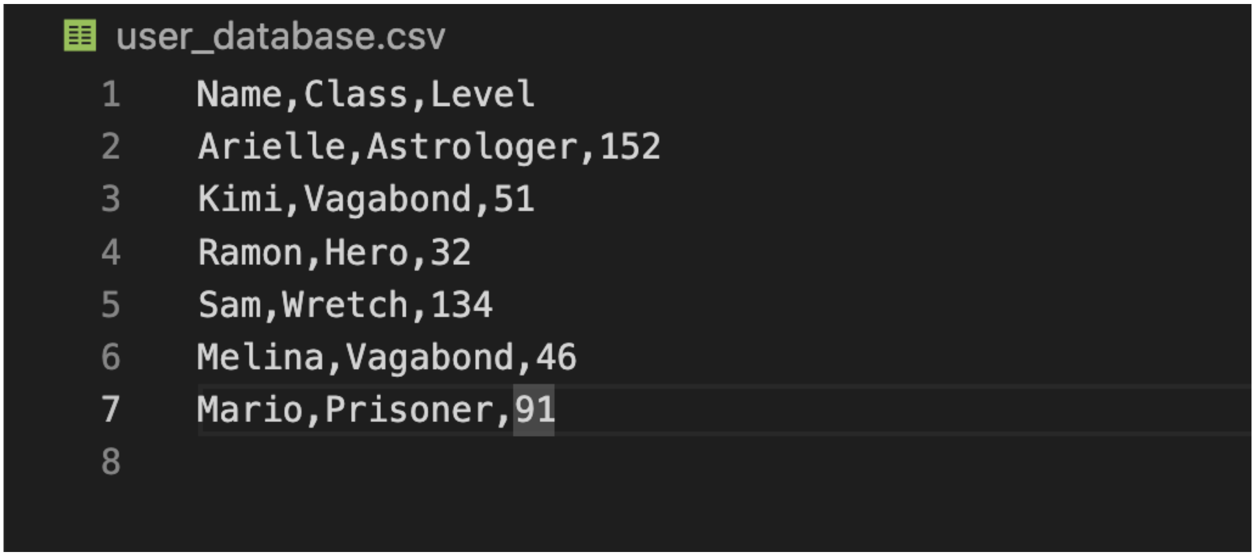
```
EldenBot: Hi I'm EldenBot! I am ready to assist you with the game of Elden Ring
EldenBot: Before we start, what is your name?
You: Mario
Welcome back Mario
Here are your stats:
Level : 90
Class : Prisoner
What can I do for you today?
You: What stats should I focus on?
EldenBot: Considering your class, the stats you should upgrade are:  Intelligence, Vigor,  Mind, and Dexterity
You: Can you tell me the difference between Hero and Vagabond classes?
EldenBot: Comparison between Hero and Vagabond:

Hero class description: A stalwart Hero, at home with a battleaxe, descended from a badlands chieftain
Vagabond class description: A knight exiled from their homeland to wander. A solid, armor-clad origin.

Hero starting stats: level: 7, vigor: 14, mind: 9, endurance: 12, strength: 16, dexterity: 9, intelligence: 7, faith: 8, arcan
e: 11
Vagabond starting stats: level: 9, vigor: 15, mind: 10, endurance: 11, strength: 14, dexterity: 13, intelligence: 9, faith: 9,
 arcane: 7

You: What areas should I explore?
EldenBot: If you're currently at level 90, I suggest exploring these areas if you haven't already: Leyndell The Royal Capital,
 Deeproot Depths, Mt. Gelmir, Dragonbarrow, Mountaintops of the Giants
Additionally, you should consider fighting these bosses: Godfrey First Elden Lord, Mohg the Omen, Morgott the Omen King, Lichd
ragon Fortissax, Magma Wyrm, Godskin Noble, Flying Dragon Greyoll, Godskin Apostle, Commander Niall, Borealis the Freezing Fog
.
You: Can you update my level to 91
Mario's level has been updated to 91 in the CSV file
You: STOP
EldenBot: Goodbye!
Top 3 TF-IDF Words:
1. Word: 91, TF-IDF Score: 0.2886751345948129
2. Word: areas, TF-IDF Score: 0.2886751345948129
3. Word: classes, TF-IDF Score: 0.2886751345948129
```

### Storing User Models



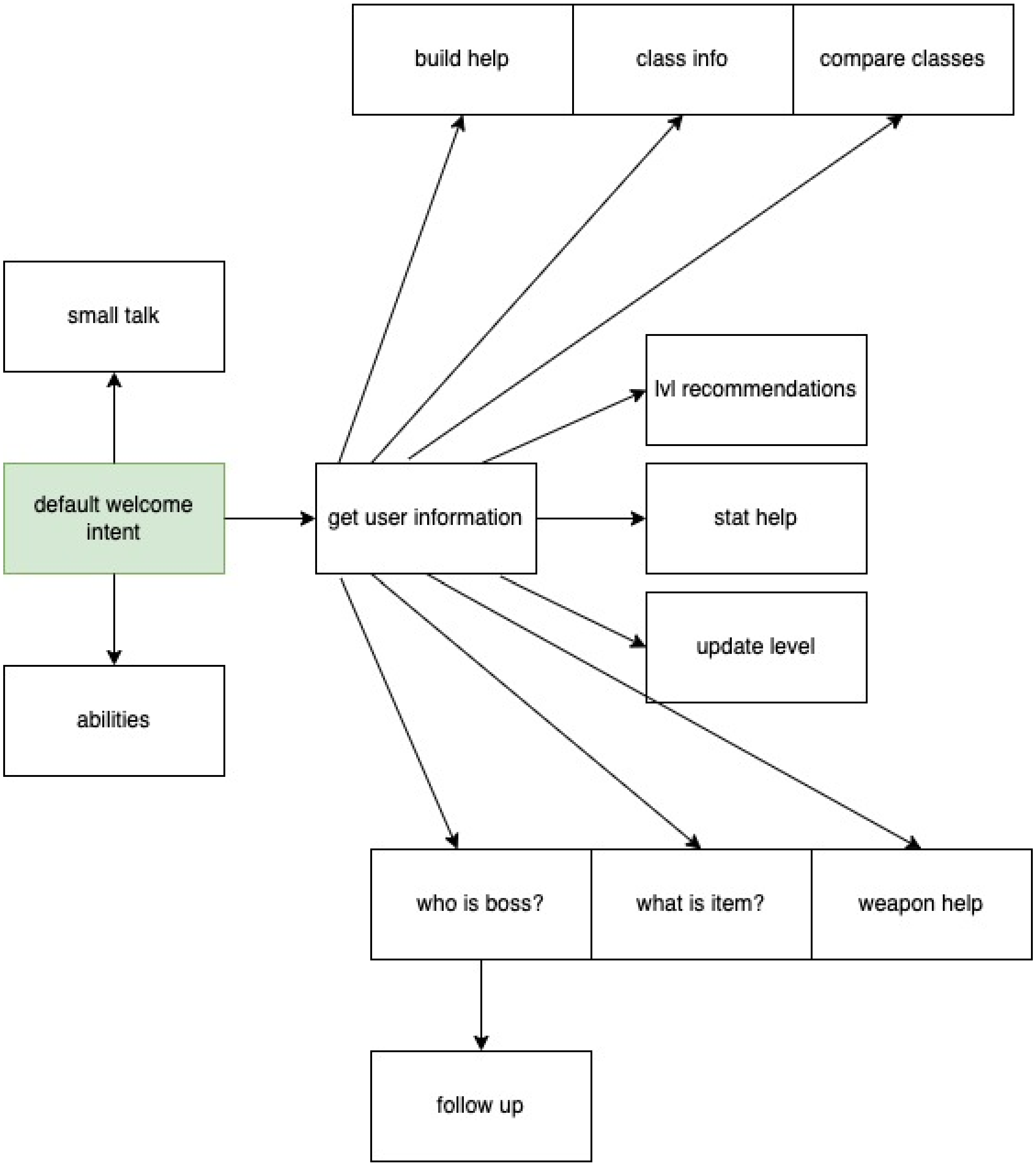
user_database.csv

```
Name,Class,Level
Arielle,Astrologer,152
Kimi,Vagabond,51
Ramon,Hero,32
Sam,Wretch,134
Melina,Vagabond,46
Mario,Prisoner,91

```

We store each user’s data through a csv file. This is how the program checks if the user already exists for the bot. If they don’t exist, the program writes them into this file.

# Diagram



EldenBot greets the users with its default welcome intent. Once connected, users are presented with three options to choose from: engage in casual conversation with the bot, inquire about its capabilities, or provide information such as their name, starting class, and current level.

Following this, the bot offers users nine unique routes to explore. They can seek advice on how to optimize their character build, gain more insights about their starting class, or compare their class with two others. Additionally, the bot's capabilities encompass guiding users on what to explore or which battles to engage in based on their level, recommending which stats to upgrade based on their class, and even updating the user's current level.

Users can also pose questions about bosses or items within the game, as well as receive recommendations on the best weapons to wield. Notably, the bot even includes a follow-up feature, where if a user inquires about a boss, the bot will prompt them for further details by asking, "What boss?".

After giving the bot your information, consider asking these types of questions:

1. How can I improve my build?
2. What stats should I upgrade?
3. What weapons should I get?
4. Who is the boss Margit?
5. What locations do you recommend I explore?

Overall, EldenBot offers a range of features and options for users to explore and optimize their gaming experience.

# NLP Techniques

As the user talks to EldenBot, we record its lines to a log file. When the user decides to stop talking to the bot, we calculate the top TF-IDF values and find the most frequent words the user used. There is a `top_tfidf_words` function reads the contents of a file specified by the `filename` argument and performs TF-IDF (term frequency-inverse document frequency) on the user’s text. It then returns the top `n_words` words with the highest TF-IDF score. The function uses the `TfidfVectorizer` from the scikit-learn library to compute the TF-IDF scores. The top `n_words` words are identified using the `heapq.nlargest` function to obtain the indices of the top `n_words` scores, and then the corresponding words and scores are printed. The function returns the list of top `n_words` words. This function can be used to identify the most important words in a given text document, which could be useful for summarization or topic modeling.

There’s also another function called `extract_name` that uses the Natural Language Toolkit (NLTK) library to extract a name from the user’s input sentence. The input sentence is tokenized into individual words using the `nltk.word_tokenize` function, and then each word is tagged with its part of speech using `nltk.pos_tag`. The function then extracts proper nouns from the tagged words by checking for parts of speech 'NNP' or 'NN'. It assumes that the name is the first proper noun in the sentence and returns it. This is how the bot get the user’s name.

# Knowledge Base

To ensure that our EldenBot is equipped with the necessary data, we have sourced information from these websites:

- Elden Ring API (https://docs.eldenring.fanapis.com/docs/)
- GamesRadar (https://www.gamesradar.com/elden-ring-ancient-dragon-smithing-stones-somber/)
- DenOfGeek (https://www.denofgeek.com/games/elden-ring-best-builds-every-class-stats-pvp-pve/)
- GetDroidTips (https://www.getdroidtips.com/recommended-level-boss-elden-ring/)

## Web Scraping

We created a function called `get_build` that takes a string argument called `name` that represents the user’s class and returns a string. The function scrapes a website at the URL "**https://www.denofgeek.com/games/elden-ring-best-builds-every-class-stats-pvp-pve/**" using the Python requests and BeautifulSoup modules to obtain information about the builds.

The function searches for `h2` tags on the webpage that start with "Elden Ring: Best", and removes any that don't meet this criterion. It then searches the remaining `h2` tags for one that matches the `name` argument, after converting `name` to lowercase and removing any instances of "astrologer" and replacing it with "astrolger" using regular expressions (since the author misspelled “astrologer”).

If a match is found, the function extracts the text from the first `p` tag after the matching `h2` tag and all subsequent `p` tags until it reaches one with a `strong` tag. The resulting text is then cleaned up and returned as a string. The string contains each build’s primary stats, secondary stats, weapons, armors, spells, and ashes of war.

## Level Recommendations

There is a function called `get_areas_and_bosses` that takes an integer level as input that represents the user's current level and returns a string that tells them which areas to explore and which bosses to fight.

The function first creates two empty lists, `matching_bosses` and `matching_locations`.

It then checks if the level is between 1 and 20, and if so, it returns a string indicating that the location is Limgrave and no bosses match the level.

Next, the function loops through a dictionary called `data`, a self-made dictionary that maps level ranges to areas and bosses, and checks each key to see if the level falls within the range specified by that key. If it does, the corresponding location is added to the `matching_locations` list, and the associated bosses are checked to see if they match the level. If a boss matches, its name is added to the `matching_bosses` list.

After looping through all the keys in the `data` dictionary, the function checks if there are any matches. If so, it creates a string with the matching locations and bosses, and returns it. If not, it returns a string indicating that no matches were found.

*Note: The `data` dictionary was hard-coded in, but the information was gathered from here: https://www.getdroidtips.com/recommended-level-boss-elden-ring/.*

# Building REST API

We used Flask to create a REST API that does GET requests, retrieve the information we need, and assemble responses for the chatbot to use. The controller has 8 endpoints described below.

## Area and Bosses Recommendations

The Flask route at `/areas-and-bosses` handles HTTP GET requests and accepts a query parameter called `level` to recommend areas and bosses suitable for a given game level.

The function first checks if the `level` parameter is present, and if not, an error message is returned. If present, the parameter is converted to an integer using `int()` and checked to be within the range of 1-200 inclusive.

If within the range, a recommendation string for the areas and bosses at that level is retrieved via `get_areas_and_bosses()` and returned as the response. If `level` is not a valid integer or outside the range, a custom error message is returned.

Example: http://127.0.0.1:5000/areas-and-bosses?level=85

Output:

```
At your current level (85), you may want to explore these areas if you haven't already: Leyndell The Royal Capital,
Deeproot Depths, Lake of Rot, Mt. Gelmir, Mountaintops of the Giants
You may also want to fight these bosses: Godfrey First Elden Lord, Mohg the Omen, Morgott the Omen King, Lichdragon
Fortissax, Astel Naturalborn of the Void, Magma Wyrm, Commander Niall
```

## Bosses

There is a function that creates a route for a Flask web application. The route is `/bosses`, and it only accepts GET requests.

When a GET request is made to the route, the function retrieves the value of the "name" query parameter from the request. If the name is "margit" (case-insensitive), the function returns a string that describes Margit, the Fell Omen boss in a game called Elden Ring. If the name is "morgott" (case-insensitive), the function returns a string that describes Morgott, the Omen King boss in the same game.

If the name is not "margit" or "morgott", the function makes a GET request to an external API that provides information about other bosses in Elden Ring. The function retrieves the information for the boss whose name matches the value of the "name" query parameter, and then constructs a response that includes the boss's name, location, drops, and description. The function randomly chooses one of three different sentence formats for the response.

If an error occurs during the process of retrieving boss information from the external API, the function returns an error message that indicates that the boss was not found.

Example: http://127.0.0.1:5000/bosses?name=godrick

Output:

```
The boss's full name is Godrick The Grafted. They can be found in Stormveil Castle. They drop Godrick's Great Rune,
Remembrance of the Grafted. Here's a brief description: Godrick the Grafted is the ruler of Stormveil Castle, wielding a
large golden axe. He has grafted many arms on his body, a symbol of power to all those who might try to depose him.
```

## Builds

The route `/builds` listens to GET requests. The route expects a query parameter `name` to be provided, representing the user’s class.

When a GET request is received with the `name` parameter, the function `get_build_advice` is called, which calls the `get_build` function with the provided name parameter to retrieve a build description for the character class. The `get_build` function is not provided in this code snippet and is assumed to be defined elsewhere.

The `get_build_advice` function returns the build as a string, which is sent back as the response to the GET request.

Example: http://127.0.0.1:5000/builds?name=samurai

Output:

```
Primary Stats: Dexterity, Vigor
Secondary Stats: Endurance, Mind
Weapon: Uchigatana, Rivers of Blood, Longbow, Lion Greatbow
Armor: Briar Set, Raging Wolf Set
Talisman: Lord of Blood’s Exultation, Arrow’s Reach Talisman, Erdtree’s Favor
Ashes of War: Keen Ash of War, Unsheathe, Mighty Shot, Enchanted Shot
```

## Classes

There is a Flask route at `/classes` that handles HTTP GET requests with a query parameter called `name` to retrieve information about a starting class, including Hero, Bandit, Astrologer, Warrior, Prisoner, Confessor, Wretch, Vagabond, Prophet, and Samurai.

The function retrieves the `name` parameter from the request using `request.args.get()` and sends an HTTP GET request to a remote API via `requests.get()` method with `name` as a query string parameter in the URL. The API response is converted to a JSON object using `response.json()` method.

The relevant data, such as class name, description, and starting stats, are extracted from the JSON object. The description is modified to ensure the first letter is lowercase. A string is constructed with this data and returned as the response.

If an error occurs, such as the requested class name not being found in the API, a custom error message is returned.

Example: http://127.0.0.1:5000/classes?name=confessor

Output:

```
The Confessor class is a church spy adept at covert operations. Equally adept with a sword as they are with
incantations. The starting stats are: level: 10, vigor: 10, mind: 13, endurance: 10, strength: 12, dexterity: 12,
intelligence: 9, faith: 14, arcane: 9
```

## Comparing Classes

This code defines a function named compare_classes that is executed when a GET request is made to the `/compare-classes` endpoint. The function retrieves information about two Elden Ring character classes specified by the `class1` and `class2` parameters passed as query strings in the request URL. It does so by sending HTTP requests to an API that returns JSON data, which is then parsed to extract the class name, description, and starting stats. The function then returns a string that displays a comparison between the two classes, including their names, descriptions, and starting stats. If either of the specified classes is not found, an error message is returned instead.

Example: http://127.0.0.1:5000/compare-classes?class1=samurai&class2=hero

Output:

```
Comparison between Samurai and Hero:

Samurai class description: A capable fighter from the distant land of reeds. Handy with Katana and Longbows
Hero class description: A stalwart Hero, at home with a battleaxe, descended from a badlands chieftain

Samurai starting stats: level: 9, vigor: 12, mind: 11, endurance: 13, strength: 12, dexterity: 15, intelligence: 9,
faith: 8, arcane: 8
Hero starting stats: level: 7, vigor: 14, mind: 9, endurance: 12, strength: 16, dexterity: 9, intelligence: 7, faith: 8,
arcane: 11
```

## Items

One function defines a Flask route at `/items` that handles HTTP GET requests. It accepts a query parameter named `name`, which is used to search for an item in the Elden Ring API.

If the API returns a successful response (status code 200), the function extracts the description, effect, and name of the first item returned by the API and creates a sentence with this information. The sentence is returned as the response to the HTTP request.

If an error occurs while querying the API, an exception of type `Exception` will be raised, which is then caught by the `except` block. In this case, the function returns a custom error message indicating that the item with the given name was not found.

Example: http://127.0.0.1:5000/items?name=perfume+bottle

Output:

```
Perfume Bottle: Glass bottles used by perfumers. Used to seal various scent compounds. Container required for crafting
perfume items
```

## Stats

The route at `/builds/stats` accepts a query parameter `name` to generate a recommendation for stat upgrades based on the player's class. The function extracts the stats from a build string using string manipulation, concatenates them into a sentence, and constructs three different sentences to provide a recommendation. A randomly selected sentence is returned as the HTTP response.

Example: http://127.0.0.1:5000/builds/stats?name=samurai

Output:

```
Considering your class, the stats you should upgrade are: Dexterity, Vigor, Endurance, and Mind
```

## Weapons

There's an endpoint for the `/builds/weapons` route that takes a GET request. It expects a query parameter `name` that is used to retrieve a build with the specified name using the `get_build()` function. The weapons for the specified build are extracted from the build data and formatted into a sentence.

There are three different sentences that can be returned as a response, each providing a slightly different message. The sentence to return is selected randomly using the `random` module.

The response format is a plain text string that includes a message with the recommended weapons for the build based on the class.

Example: http://127.0.0.1:5000/builds/weapons?name=samurai

Output:

```
To optimize your build for your class, it is recommended that you upgrade the following weapons: Uchigatana, Rivers of
Blood, Longbow, and Lion Greatbow
```

The Elden Ring API is an open-sourced API that contains information we needed such as describing each item, having all bosses information, and getting descriptions for each starting class in the game.

# Evaluations & Analysis of its Strengths and Weaknesses

## Strengths

Our chatbot has several strengths that make it effective and efficient. Firstly, it is able to read and write CSV files, which enables it to save important user data such as name, class, and level. Additionally, the chatbot is built on Dialogflow, Google's chatbot framework, which provides a robust and reliable platform for its operations. Another strength is that the chatbot has the ability to comprehend questions even if words are misspelled, making it more user-friendly. Furthermore, our code is designed to handle errors, ensuring that the program won't break, and the chatbot understands the intent of user queries.

## Weaknesses

While Dialogflow provides a convenient framework for building chatbots, there are some potential weaknesses to consider. Here are the main weaknesses we found during our testing.

### Duplicate Names

The chatbot does not allow duplicate names to be saved to the CSV file, which could potentially create issues for users with similar names. If you have the same name as someone else you have the ability to overwrite the stats.

### Recognizing Names

When asking for the user’s name, sometimes the bot doesn’t recognize the input as the name. Additionally, the knowledge of the chatbot is limited to information gathered from our REST API, which may not provide a comprehensive understanding of all user queries.

### Understanding Intents

The chatbot requires additional training phrases to be added to understand certain intents fully. These weaknesses, though, are being actively worked on to improve the functionality of our chatbot.