School of Electronic Engineering and Computer Science

Final Report

Programme of study:
Computer Science

Project Title: VIDEO GAME RECOMMENDATION WEB APPLICATION

Supervisor:
Dr. Paulo Oliva

Student Name:
Danyal Hussain

Final Year
Undergraduate Project 2020/21

Date: 04/05/2021

Queen Mary
University of London

Acknowledgements

A large thanks to you, Dr. Paulo Oliva, for the guidance and assistance you have provided over the year. A lot of this would have been considerably more difficult to achieve without your help. Thank you.

And in true gaming fashion:

I would like to thank my dear friend JON_J1 for all the support you have given me. This is not only just over the 3 years of my course, but for all that came before it. You forever have my heartfelt thanks.

AtchaBatcha, I would like to thank you for the conversation we had near the end of my last semester, it helped. Additionally, I would like to take this as an opportunity to cement my heartfelt thanks for all the advice you have given me over the years. I have learnt so much. Thank you.

And of course, a special thanks to omgitsmotrix, Makkyh and Hbb30 – you guys didn't think I would leave you out, did you?

You all know who you are.

Thank you all so much.

Abstract

This report presents the motivations, designs, implementation, and testing of a web application that either recommends, or finds information about, a video game depending on what kind of information a user is looking for.

The aim is to create a custom recommendation algorithm based on video game attributes to provide recommendations for a game entered into the web-based application. This is in addition to providing valuable post recommendation, or search, information for video games using platforms that have either a substantial focus on, or content for, video games. The platforms used are Internet Game Database, YouTube, Twitch and Steam.

The custom recommendation algorithm was successfully developed, with the web application fully integrated with all the aforementioned platforms using their application programming interfaces.

The project has been deployed at the following web address:

https://ugprojectsite.herokuapp.com/

Contents

Chapter 1: Introduction

1.1 Background

Steam is rank one for the most popular gaming client on desktop computers (Prescott, 2020), YouTube has the largest number of visitors for a video streaming service (Top Websites Ranking, 2020), Twitch is the biggest video game live-streaming platform (Stephan, 2020), and Internet Game Database is a gaming focused website that gathers relevant information about games, that has social and explorative features (What is IGDB.com?, 2021).

Yet there is no web application out there that uses all these platforms together to present the user with additional information after searching for, or being recommended, a video game. By creating a web application that presents this information to the user, it would make for a useful tool in assisting the user to select their next video game to play or purchase.

If successful, this web application will save the users time and also present them with information that they may not have otherwise been considered.

1.2 Problem Statement

Whilst there are other web applications available that offer users with video game recommendations. The typical process for these sites is that a video game is suggested, and a link is provided to a platform that sells said video game. There are no links to the most popular, or most relevant, pre-recorded footage of the video games on YouTube, or links to the most popular streams for said video game on Twitch. The user would need to navigate to these sites and perform searches manually, making the process both unnecessary and inconvenient.

As these platforms contain an extreme wealth of content for videos games, it makes for relevant and useful source of information when searching for more information about video games. Using these platforms, alongside Steam and IGDB (Internet Game Database), should prove to be a useful asset when finding a game to be recommended to you or finding more information about a specific game.

With other web applications not integrating these platforms, I believe it is an opportunity that is missed out on.

1.3 Aim

The project aim is to build a web application that provides a recommended game using an algorithm that recommends the most similar game based on a set of attributes, or the option to search for a specific game. This is in addition to providing valuable post-recommendation content that is relevant for the recommended game, or inputted game, from sites IGDB, YouTube, Twitch, and Steam, with a link to a purchasing page on Steam.

1.4 Objectives

1. Build a web application that can do the following:
    a. Use IGDB's API (Application Programming Interface) to get data for the inputted or recommended game.
    b. Use YouTube's API to get a list of the three most "relevant" videos to display for the inputted or recommended video game.
    c. Use Twitch's API to get a list of the current most watch streams for the inputted or recommended video game and display this.
    d. Give the user an option to manually enter a video game in order to attain:
        i. a recommendation
        ii. more information about that specific video game

2. Build a recommendation algorithm using attributes of video games as points for comparisons

Chapter 2: Literature Review

This chapter will focus on the justification for conducting the project, other similar applications, as well as the tools required to conduct it. Additionally, for the remainder of this paper, any mention of the word “game” or “games” should be understood as “video game” or “video games”.

2.1 Justification

Continuing from 1.1, there is no other application, as you will see in 2.1.1, that will use all APIs from major platforms when in search, or after a recommendation has been made, for a game. This is a missed opportunity to bring together these platforms to provide a valuable service that is greater than the sum of its parts.

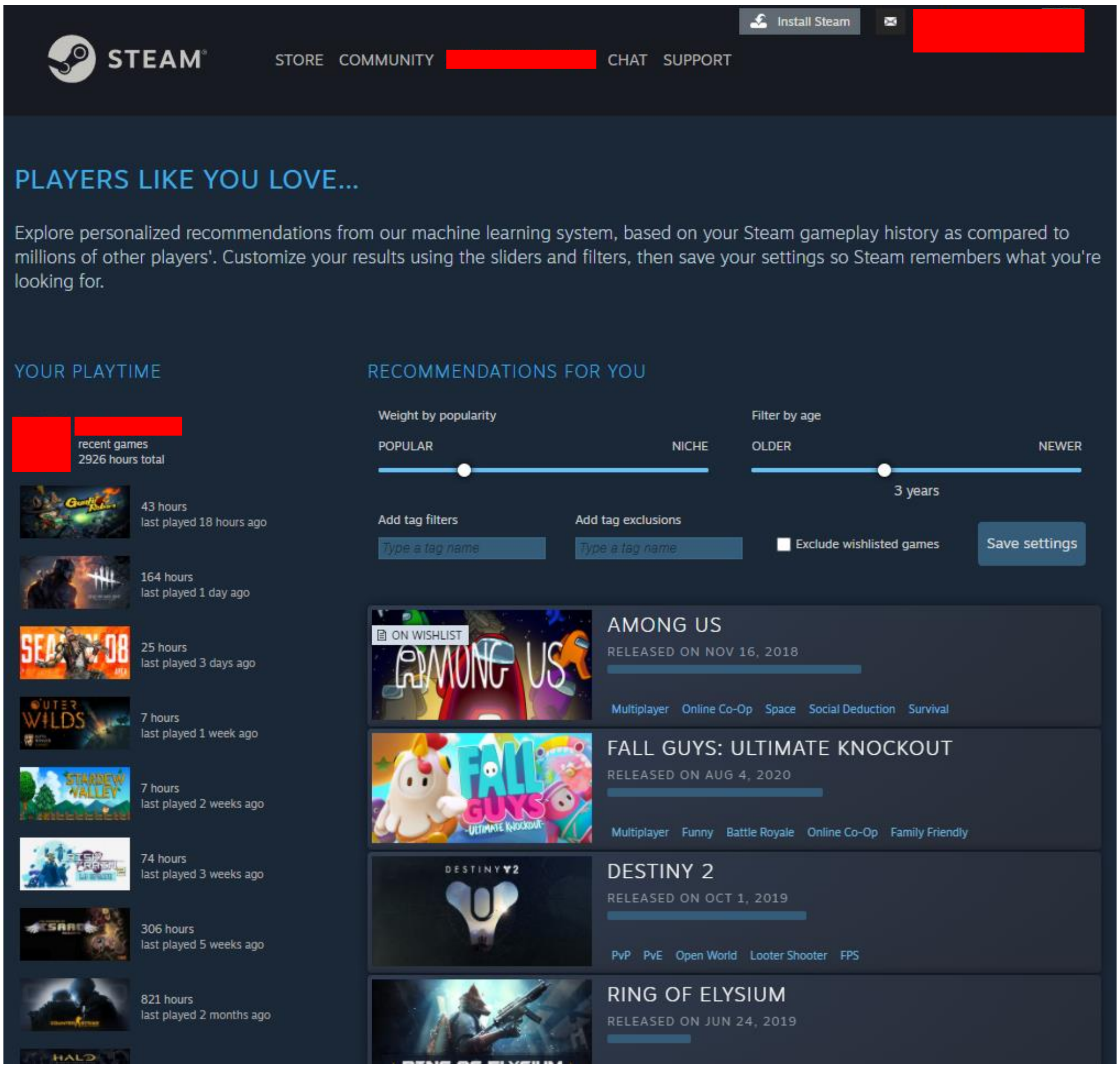
This means that this project will be the first of its kind.

2.2 Existing Applications

In this section, there will be evaluations of the most similar web applications, what they have done well and what makes this project different.

2.2.1 Steam Interactive Recommender (Interactive Recommender, n.d.)

Before we begin, the reference for Steam’s Interactive Recommender (Interactive Recommender, n.d.) requires an account on Steam’s platform in order to gain access. This will be mitigated by providing a screen capture of what the interface looks like using my personal Steam account, with redactions. This can be seen in figure 1.


Install Steam [redacted]

STEAM STORE COMMUNITY [redacted] CHAT SUPPORT

PLAYERS LIKE YOU LOVE...

Explore personalized recommendations from our machine learning system, based on your Steam gameplay history as compared to millions of other players'. Customize your results using the sliders and filters, then save your settings so Steam remembers what you're looking for.

YOUR PLAYTIME

[redacted] [redacted]
recent games
2926 hours total

43 hours
last played 18 hours ago

164 hours
last played 1 day ago

25 hours
last played 3 days ago

7 hours
last played 1 week ago

7 hours
last played 2 weeks ago

74 hours
last played 3 weeks ago

306 hours
last played 5 weeks ago

821 hours
last played 2 months ago

RECOMMENDATIONS FOR YOU

Weight by popularity
POPULAR NICHE

Filter by age
OLDER NEWER
3 years

Add tag filters
Type a tag name

Add tag exclusions
Type a tag name

Exclude wishlisted games

Save settings

ON WISHLIST
AMONG US
RELEASED ON NOV 16, 2018
Multiplayer Online Co-Op Space Social Deduction Survival

FALL GUYS: ULTIMATE KNOCKOUT
RELEASED ON AUG 4, 2020
Multiplayer Funny Battle Royale Online Co-Op Family Friendly

DESTINY 2
RELEASED ON OCT 1, 2019
PvP PvE Open World Looter Shooter FPS

RING OF ELYSIUM
RELEASED ON JUN 24, 2019

Figure 1 – Steam's Interactive Recommender web page (Interactive Recommender, n.d.)

Figure 1 is what would be seen if a user has a Steam account populated with the specific combination of games, as well as "playtimes" (how long a user has played a given game), that is identical to my own. These variables can be seen in figure 1 and will vary from user to user.

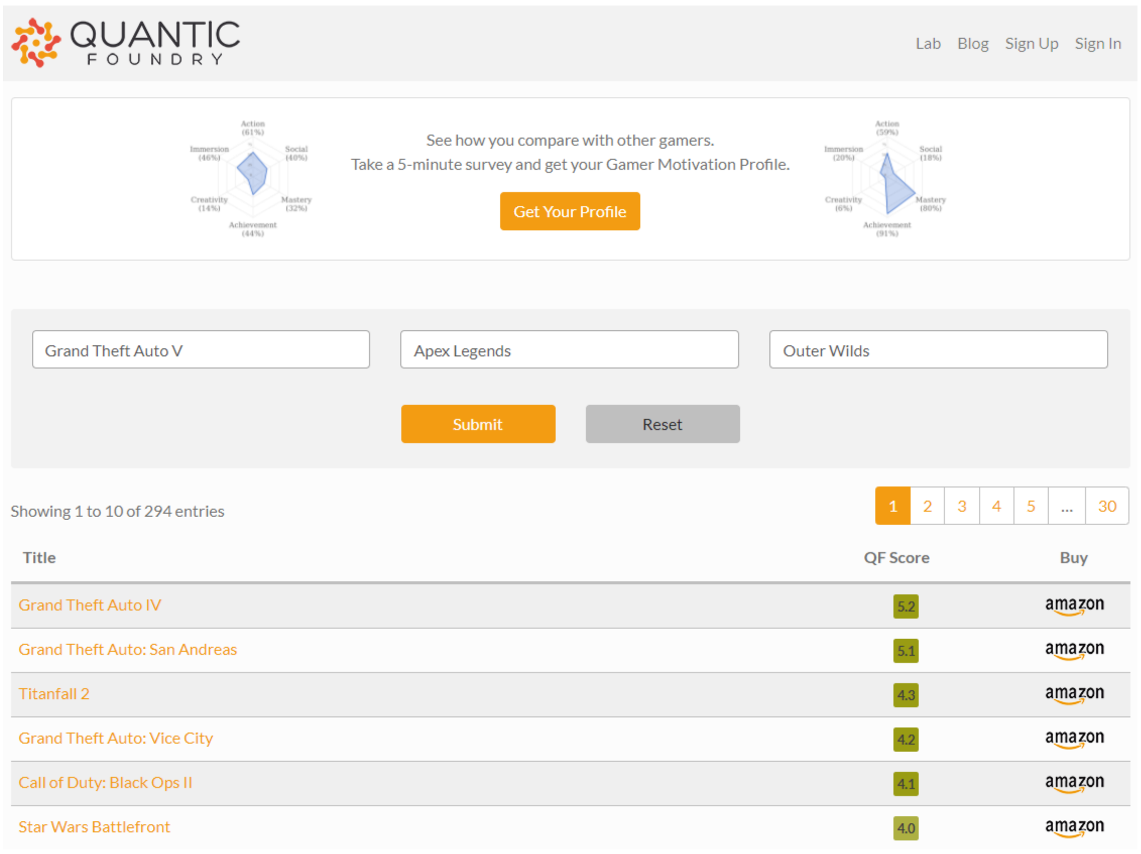
The greatest aspect of Steam's game recommendation tool is the fact it uses ML (Machine Learning) (Introducing The Steam Interactive Recommender, 2020). This ML model is trained on the playtime histories of millions of Steam users (Introducing The Steam Interactive Recommender, 2020). It will recommend games based on other Steam users that have similar playing habits to the user. If these group of Steam users have played and enjoyed a game that the user of the tool has yet to play, it would then be recommended to this user (Introducing The Steam Interactive Recommender, 2020).

In addition to this, the recommendation tool allows for customisations from the user to better adjust the results depending on what the user values more. As seen above in Figure 1, the options available are: Popular, Niche, Older and Newer. This is where moving the sliding bar between any two given categories provides a greater weighting on the category that the slider is closest to.

Overall, this is a strong recommendation tool that is backed by a wealth of user data that Steam has about user playing habits and video games. However, this does not guarantee that the user will like the recommended game/recommendation. This is where it would have been ideal to have post-recommendation content in a variety of forms. Steam does, however, have a store page for each game they have available for sale, where the user can see more information about the game. If the user chooses to view this, they will see details such as screenshots, trailers, and player reviews about the game. Nonetheless, this is still missing videos about the game from YouTube such as recorded reviews of the game, or Twitch's live streams, where the user could interact with a broadcaster that is playing the game and specific questions in real time.

2.2.2 Quantic Foundry (Video Game Recommendation Engine, n.d.)

If you would like to investigate the sources I am providing for this section, what is required are three video game entries from the reader in order to be directed to a recommendation page. This is where the information about the recommended games, alongside the details of how Quantic Foundry recommends games, is found. This is the information that will be referenced in this section (Video Game Recommendation Engine, n.d.). The relevant figures (figure 2 and 3) for this specific page will be provided below for convenience.



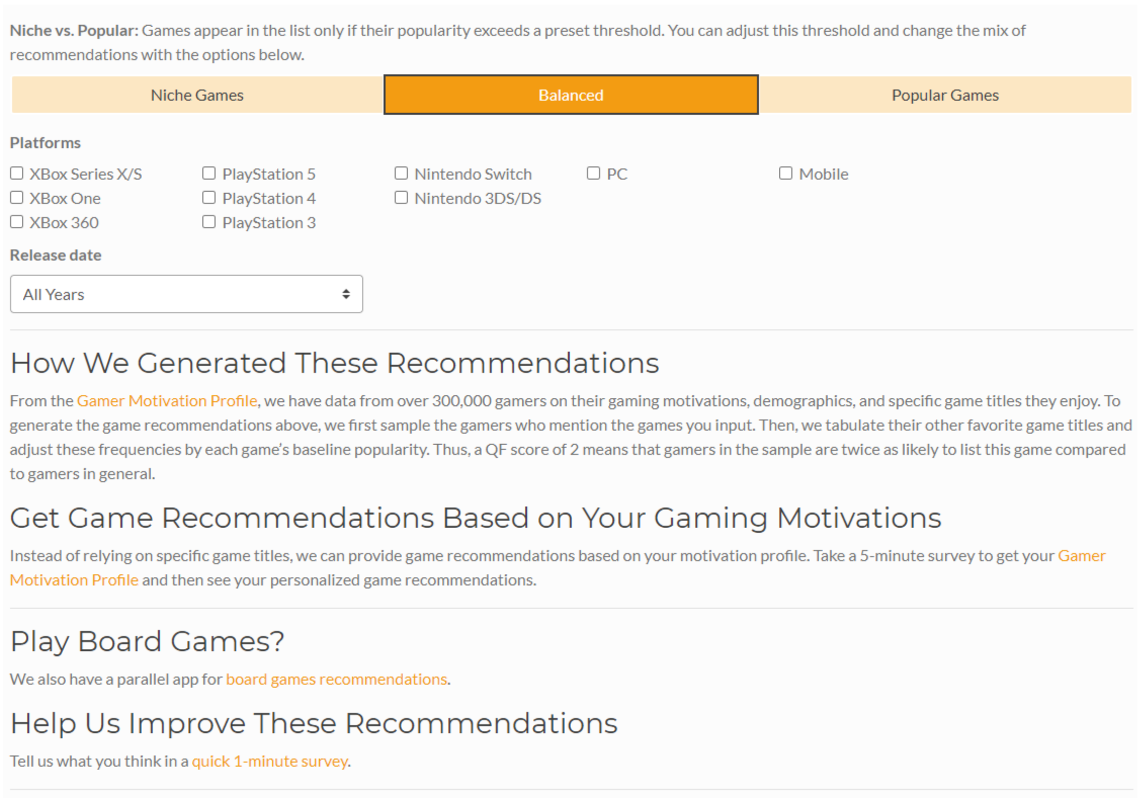
See how you compare with other gamers.
Take a 5-minute survey and get your Gamer Motivation Profile.

Grand Theft Auto V | Apex Legends | Outer Wilds

Showing 1 to 10 of 294 entries

Title	QF Score	Buy
Grand Theft Auto IV	5.2	amazon
Grand Theft Auto: San Andreas	5.1	amazon
Titanfall 2	4.3	amazon
Grand Theft Auto: Vice City	4.2	amazon
Call of Duty: Black Ops II	4.1	amazon
Star Wars Battlefront	4.0	amazon

Figure 2 - Post search result (Video Game Recommendation Engine, n.d.)


Niche vs. Popular: Games appear in the list only if their popularity exceeds a preset threshold. You can adjust this threshold and change the mix of recommendations with the options below.

Niche Games | [✓ Balanced] | Popular Games

Platforms

☐ XBox Series X/S ☐ PlayStation 5 ☐ Nintendo Switch ☐ PC ☐ Mobile
☐ XBox One ☐ PlayStation 4 ☐ Nintendo 3DS/DS
☐ XBox 360 ☐ PlayStation 3

Release date

All Years

How We Generated These Recommendations

From the Gamer Motivation Profile, we have data from over 300,000 gamers on their gaming motivations, demographics, and specific game titles they enjoy. To generate the game recommendations above, we first sample the gamers who mention the games you input. Then, we tabulate their other favorite game titles and adjust these frequencies by each game's baseline popularity. Thus, a QF score of 2 means that gamers in the sample are twice as likely to list this game compared to gamers in general.

Get Game Recommendations Based on Your Gaming Motivations

Instead of relying on specific game titles, we can provide game recommendations based on your motivation profile. Take a 5-minute survey to get your Gamer Motivation Profile and then see your personalized game recommendations.

Play Board Games?

We also have a parallel app for board games recommendations.

Help Us Improve These Recommendations

Tell us what you think in a quick 1-minute survey.

Figure 3 - post search result, remainder of the web page from Figure 3 (Video Game Recommendation Engine, n.d.)

Note that Figures 2 and 3 are part of the same web page. After the submission of games into the engine, in our case "Grand Theft Auto V", "Apex Legends" and "Outer Wilds", the contents below the "submit" and "reset" button in figure 2 are generated dynamically.

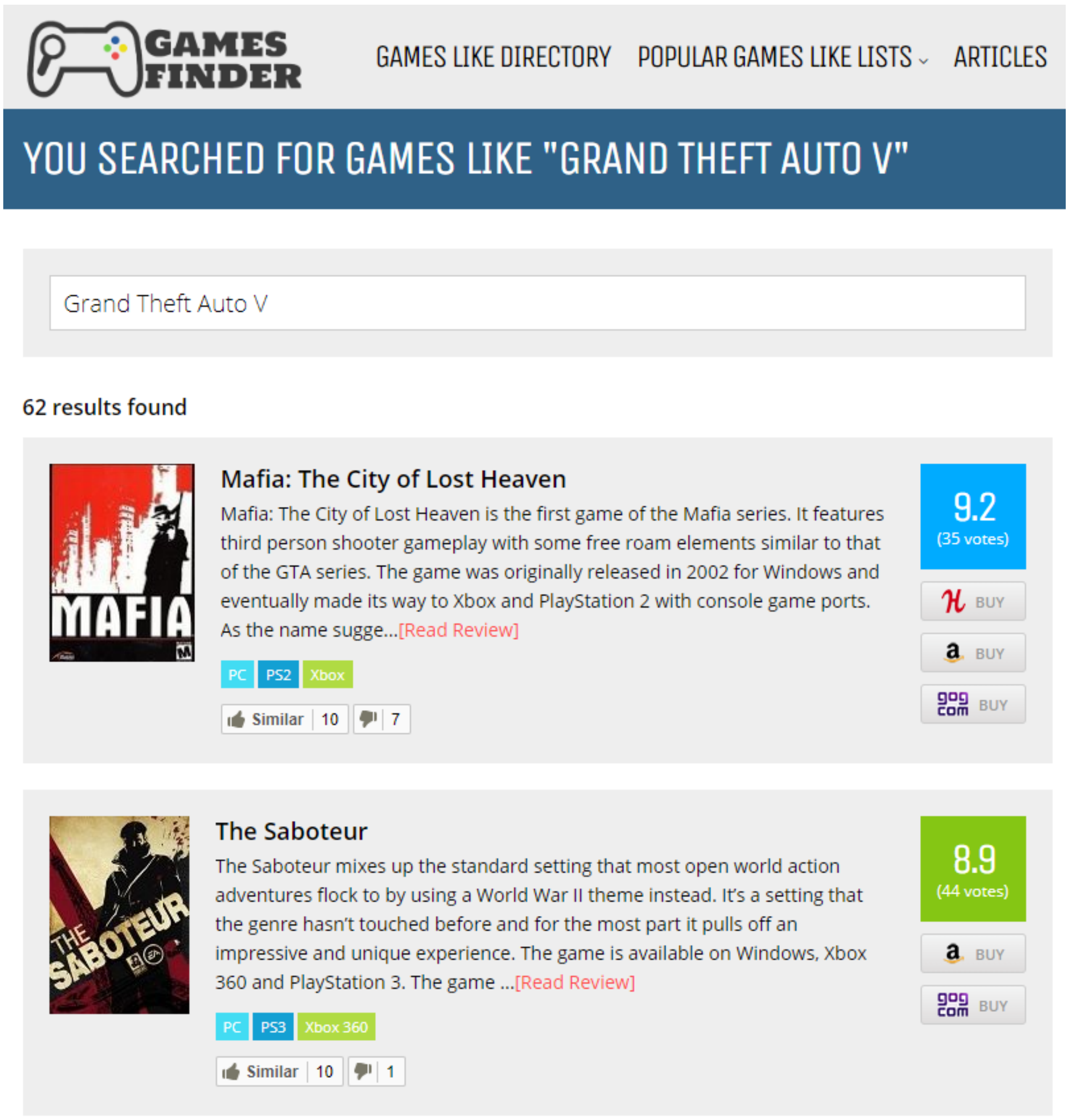
As seen in figure 3 under "How We Generated These Recommendations", Quantic Foundry's recommendation engine essentially uses an algorithm that determines what games to recommend based on other users that have played the same game. It then returns their favourite titles after adjusting the frequency of importance based on popularity (Video Game Recommendation Engine, n.d.).

Much like Steam, this recommendation engine allows for preference on what type of game they value more: niche or popular games. However, unlike Steam's implementation, the user is provided with the options to pick specific platforms. This is useful as almost all gaming platforms have games that are exclusive. A list of these games and their respective platforms can be found at "gematsu.com/exclusives" (Romano, 2021).

As for the post-recommendation experience, there is room for improvement. It only provides links to Amazon to purchase the game, and a link to IGDB's web page for the games that were recommended. This can be seen in the bottom half of figure 2. Similar to Steam's implementation, there are no links to YouTube videos or Twitch streams of any kind.

2.2.3 Games Finder (Games Finder, n.d.)

Games Finder's recommendation system employs a different approach to game recommendations: recommendations are based on research, analysis, and handpicked titles (Games Finder – About Us, n.d.). Games Finder mentions that they use the user review ratings made on their platform in the recommendation process. They, however, do not mention how much weight they assign to these review ratings. They also do not provide any further information about the recommendation process.



YOU SEARCHED FOR GAMES LIKE "GRAND THEFT AUTO V"

Grand Theft Auto V

62 results found

Mafia: The City of Lost Heaven

Mafia: The City of Lost Heaven is the first game of the Mafia series. It features third person shooter gameplay with some free roam elements similar to that of the GTA series. The game was originally released in 2002 for Windows and eventually made its way to Xbox and PlayStation 2 with console game ports. As the name sugge...[Read Review]

PC PS2 Xbox

Similar 10 7

9.2 (35 votes)

The Saboteur

The Saboteur mixes up the standard setting that most open world action adventures flock to by using a World War II theme instead. It's a setting that the genre hasn't touched before and for the most part it pulls off an impressive and unique experience. The game is available on Windows, Xbox 360 and PlayStation 3. The game ...[Read Review]

PC PS3 Xbox 360

Similar 10 1

8.9 (44 votes)

Figure 4 - Search result for an example query (You searched from games like "Grand Theft Auto V", n.d.)

Game Finder's post-recommendation experience is neither better than Quantic Foundry's or Steam's implementation. The site provides a written review by an author for each game, and its own scoring system and two YouTube videos (Franklin, 2020). Although there is an integration of embedded YouTube videos, which is great for variation, it is lacking in the other aspects. The review provided by an author makes learning about the game lengthy as it does not contain all of the description that the publisher or developer has produced for the game. Furthermore, the site's scoring system would be otherwise excellent were it not for the extremely small sample sizes. In the case of the latest addition to Game Finder's library, "Splitgate: Arena Warfare" only has two votes at the time of writing (Franklin, 2020). However, it does offer more purchasing choices than both Quantic Foundry (Amazon) and Steam.

2.2.4 Differences table

Table 1 - Differences between existing applications

Web Application	Advantages	Disadvantages
Steam	ML integration based on wealth of data Adjustable preferences with slider	No integration with Twitch No integration with YouTube
Quantic Foundry	Custom recommendation algorithm Adjustable preferences Links to IGDB for additional game information if desired	No integration with Steam
Games Finder	Custom recommendation algorithm Integration with YouTube	No ML integration No integration with Twitch No integration with Steam No adjustable preferences
This Project	Custom recommendation algorithm Integration with YouTube Integration with Twitch Integration with Steam Integration with IGDB	No ML integration No adjustable preferences

2.2.5 Conclusion

It is clear that these applications are missing key integrations with other major platforms. However, Steam utilises superior technology when it comes to a recommender system: ML. This would be an excellent asset for this project and would have largely negated the utility gap between this project and these applications. However, it is out of the scope of this project. As to build such a system that is not just useful, but also competitive, would make for an extremely challenging task. A custom recommendation algorithm, as seen in both Quantic Foundry and Game Finder's implementation, is better suited for this project. As this type of algorithm would allow for greater flexibility in terms of its complexity. If there were to be a great amount of time remaining near the end of this project, then it would allow for further development on the complexity and effectiveness of the algorithm and vice versa.

In addition to, this project will be able to make better use of third-party platforms that neither Steam nor Quantic Foundry employ. These, as mentioned earlier, are YouTube and Twitch's APIs.

2.3 Implementation Tools

This chapter will focus on evaluating what are the best implementation tools for this project.

2.3.1 Web Application Framework

A WAF (web application framework) is a software framework that is designed to aid in the development of web applications. These include, but are not limited to, web APIs, web services and web resources. The main benefit of employing the use of a WAF is the existing implementation of overhead that would be needed for common activities in web development. Examples include protocols, sessions, database management, security and testing.

By selecting a suitable WAF to use for this project, also locks the project into the programming language to be/that will be used for development. This is as WAF's are written with specific programming languages.

The WAFs that will be evaluated are Django and Rails, as these are among the most "starred" WAFs on GitHub. Where Django is second at 56,299, and Rails fourth at 47,811 (Web application frameworks, 2021)*. The purpose of stars on GitHub is for the ease of any given GitHub user to return to a repository, or a topic, later (Saving repositories with stars, n.d.). This makes it a useful metric to show the interest level in a given WAF.

*There may be confusion when investigating the source (Web application frameworks, 2021). Please see Appendix 2 for clarifications.

2.3.1.1 Django

Django is a MVT (Model-View-Template) based open-source WAF. It is written in the programming language Python. The main features of Django are speed, comes with exhaustive libraries, security, scalability and versatility (Why Django?, n.d.). Django is also the most popular Python WAF as of 2020, with 49% of Python WAF users opting for Django as their web framework of choice (Python, 2020). This is from a pool of 19,696 developers that JetBrains surveyed in their yearly "The State Of Developer Ecosystem" survey (The State of Developer Ecosystem 2020, 2020).

2.3.1.2 Rails

Rails, also known as "Ruby on Rails", is an open-source WAF that is based on the MVC (Model-View-Controller) architectural pattern. It written in the programming language Ruby. According to JetBrain's 2020 State Of Developer Ecosystem survey, from the pool of participants that use Ruby, 83% use Ruby On Rails (Ruby, 2020). Rails is built on the philosophy of DRY (Don't Repeat Yourself) and Convention Over Configuration (Getting Started with Rails, n.d.).

DRY is a principle of software development that states, "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system" (Smith, n.d.). The "Convention Over Configuration" concept is a software design paradigm that seeks to apply defaults that can be implied from the structure of the code as opposed to requiring explicit code (Miller, 2009).

2.3.1.3 Differences Table

Table 2 - Difference table for Web Application Frameworks

WAF	Advantages	Disadvantages
Django	Great documentation Security Exhaustive libraries Flexible framework Fast development REST frameworks for APIs ML capabilities Scalability	No conventions Monolithic framework Steep learning curve
Rails	Large number of libraries Automation in testing Security High speed development Diverse tools and pre-sets	Improper documentation (Vyas, 2020) Heavy time cost with mistakes Lack of flexibility

2.3.1.4 Final Choice

It is not surprising why both Django and Rails are popular WAF choices for software developers (Top Web Development Frameworks in 2021, 2021). They both have advantages that make them excellent choices for a web application project. However, whilst experienced developers may be able to mitigate some of the disadvantages with experience. For example, the steep learning curve in Django and the improper documentation with Rails, I will not be able to.

As this is the first large project that is being undertaken by myself, the problems with Rails, namely the time costs with mistakes and improper documentation, makes Django an already great choice for the project. Even if this were to be ignored, the advantages Django provides are substantial. The REST frameworks available for APIs, that it comes fully loaded and that it has ML capabilities makes it an excellent candidate for this project. Although this project will not use ML, there are native packages, as discussed later, that will aid in the development of the custom recommendation algorithm. These same packages would be more difficult to integrate and develop on Rails. Lastly, Django comes with exhaustive libraries, so there is no need to add unnecessary workload to the project by understanding what modules are required and then locating and installing them. Therefore, for the aforementioned reasons, Django is the WAF of choice for this project.

2.4 Custom Recommendation Algorithm

As mentioned previously, this project will have a custom recommendation algorithm to recommend games. This recommendation algorithm will be based on how similar games are, and then recommend the most similar game.

This section should also be prefaced with the message that although the project does not incorporate ML, it will use some methods that would be commonly used in such systems. These are methods that are carried out for prepossessing data for the purpose of preparing the data for an ML model. They are still of great use even without the ML model in place, as will be seen and explained in the following sub sections.

2.4.1 Features of the Algorithm

Researching for existing similarity algorithms proved unfruitful: all solutions for recommendation algorithms comprise of some form of an ML model. It is not surprising that this dominated my research into similarity algorithms, as a recommender system within ML is seen–as the absolute answer to a recommendation system. Furthermore, custom recommendation algorithms outside of this spectrum most likely are proprietary and therefore are not discussed.

However, there is a useful resource provided by Google in their Google Developers Machine Learning Crash Course, that, although eventually defaults to the use of a ML model, mentions “manual similarity measure” (Create a Manual Similarity Measure, 2020).

A manual similarity measure is the calculation of similarity between two examples where the feature data between these two examples are combined in a single numeric value (Blockgeni, 2019). Features, in machine learning, are individual independent variables that act like input in a system (What are Features in Machine Learning and Why it is Important?, 2019). In other words, they are the attributes of an object that are represented as a numerical value that is then used as input in a system.

As mentioned in the Google resource, as data becomes more complex, creating a manual similarity measure becomes harder. When it becomes complex enough, you cannot create a manual measure (Create a Manual Similarity Measure, 2020). In their following page, Google provides an example of a dataset that comprises of seven attributes to demonstrate a manual similarity measure through the use of an exercise (Manual Similarity Measure Exercise, 2021). These features, in this example, are in relation to houses, where the features are price, size, postal code, number of bedrooms, type of house, garage and colours. This demonstrates the rudimentary nature of the characteristics of an object that can used in a manual similarity measure. This leads onto the features I have chosen for games that will be used in the algorithm.

The features of a game that will be used are the following:

- Description
- Genre
- Category
- Developers
- External critic review score

2.4.1.1 Description

This will be the most complicated feature to implement as it requires NLP (Natural Language Processing). NLP would be used by comparing two video game descriptions with the end goal of attaining a numeric value that represents the similarity of the two texts. Specifically, this will be achieved by vectorising each description such that the numeric values can be used with a feature extraction methodology. Feature extraction refers to the process of transforming raw data into numerical features that can then be further processed (Feature extraction for machine learning and deep learning, n.d.). The models that will be discussed are the Bag of Words and Term Frequency – Inverse Document Frequency model.

2.4.1.1.1 Bag of Words

The BoW (bag-of-words) model is a way of representing text data when modelling text with machine learning algorithms. It is called a "bag" of words, as any information known about the order, or the structure, of the words in the document is discarded. The model focuses primarily on whether known words occur it the document(s) and not where (Brownlee, 2017).

BoW works by creating a vocabulary of unique words within a given document or corpus (collection of documents), then proceeds to score and vectorize the words in each document within the corpus.

The scoring is calculated by checking documents for each word in the vocabulary, if the word occurs, the word will be assigned a value 1, and 0 if it does not occur. This is also how the vectorization occurs, the values 0 or 1 are placed in an array as a representation of each word in the vocabulary.

2.4.1.1.2 Term Frequency – Inverse Document Frequency

The TF-IDF (Term Frequency – Inverse Document Frequency) model is where each term is weighted by dividing the term frequency by the number of documents in the corpus containing the word (Lavin, 2019).

The "term frequency" in TF-IDF works by measuring the frequency of a word in a given document and then dividing by the total number of words in the document. This eliminates the problem where, in a document that is arbitrarily large, there is greater value assigned to a word as it occurs more frequently as opposed to a smaller document. The result is a normalisation of each document in a corpus.

The "inverse document frequency" in TF-IDF means inversing the value assigned to the occurrence of a word in a corpus. This is where the most common words have a value closer to 0 whereas the less common will approach the value 1. The calculation that is required to attain these values for each word is carried out by taking the total number of documents in the corpus, dividing it by the number of documents that contain that word and then calculating the logarithm. The logarithm is what allows for the scaling down of the value for the most common words, whilst scaling up the value of the rarer words. This is the inversion. In terms of how this is vectorized, the variables required for calculation must of course be numeric, these are easy to obtain when the required input is how many documents are in a corpus and how many times a word occurs in a document.

Finally, the values produced from multiplying the TF with IDF results in the TF-IDF score of a word in a document. This is how a word is rated for its relevancy.

2.4.1.1.3 Differences

Table 3 - Differences between Natural Language Processing methods

Model	Advantages	Disadvantages
Bag of Words	Easy to implement Efficient	Rarity of words not considered Semantics of a word not considered Syntax of a word not considered Synonyms not considered
TF-IDF	Less weight given to common words Rarity of words considered	Semantics of a word not considered Syntax of a word not considered Synonyms not considered

2.4.1.1.4 Model chosen

As seen in the table above, TF-IDF is a superior variation of the BoW model. Whilst it has one major advantage, that the rarity of words is considered, it surpasses the two advantages that BoW has: that it is easy to implement and its efficiency. Whilst the ease of implementation saves time in the project and the efficiency results in a more responsive application, words that are rare, thereby more meaningful, is a useful measure to incorporate into the recommendation algorithm. Therefore, it makes TF-IDF the model of choice for the description part of the recommendation algorithm.

2.4.1.1.5 Similarity Measure

The next part of the NLP for the game description similarity value, is to select a similarity measure. This is required as what the TF-IDF produces is a vector per document in the corpus. These vectors contain values of each word produced by its relation to words in other documents, not a singular value of similarity between documents. A similarity measure will be able to take these vectors and calculate the similarity value in vector space between one document and another. This will provide a singular value that will be used in the recommendation algorithm for the game description feature.

A similarity measure is a measure in distance of how alike two data objects are, where dimensions represent the feature of the objects (Polamuri, 2015). The distance in this context is the vector space between vectors. If this distance is small, then this signifies that the features, which are the vectors, have a high degree of similarity. If the distance is large, then this means the features have a high degree of dissimilarity.

2.4.1.1.5.1 Similarity Measure 1: Euclidean distance

Euclidean distance is the measurement of distance in vector space between the end of vectors (Measuring Similarity from Embeddings, 2020). The greater the distance, the less similar the vectors are, and vice versa. A value of 0, for the Euclidean distance between two vectors, means the two vectors are similar but not necessarily identical. Whereas a value greater than 0, denotes a greater degree of dissimilarity, the higher the value.

2.4.1.1.5.2 Similarity Measure 2: Cosine similarity

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle produced between two vectors and thereby determining whether the two vectors are pointing in the same direction (Han, Kamber and Pei, 2012). Considering the value produced by the cosine similarity, if the value is 1, then the vectors are similar. Whereas if it is 0, then it is dissimilar.

2.4.1.1.5.3 Similarity Measures: Differences

As Euclidean distance is calculated based on the distance between vectors, it presents an issue as a game description is unlikely to have the exact same length as the one it is being compared to. The longer a game description (document) is, the greater variation of words it is likely to contain, this thereby increases the number of words in a vocabulary, therefore increasing the magnitude of the vector. This results in an ever-increasing sparse vector, which is a vector with few non-zero elements, as other documents are increasingly less likely to contain the same words. Consequently, this results in an increased distance between the two vectors, when they may have otherwise been similar had it not been for the length of a vector.

This can be negated if the vectors were normalised in terms of magnitude. This can be achieved by dividing a given vector by its magnitude. This would in consequence allow the comparisons to be of an accurate representation of the text when considering the Euclidean distance calculation.

However, this is not necessary when considering an alternative that is Cosine similarity. Cosine similarity already factors in the magnitude of the vectors in question whilst also considering the angle between the two vectors. This is one of the greatest assets of cosine similarity: the disregard of the level of sparseness of the vectors in question. In terms of the equivalence of the two measures, when comparing the normalization component, they are equivalent. As mentioned by Zhao and Chellappa in section 7.4.1 of their book Face Processing: “A cosine similarity measure is equivalent to length-normalizing the vectors prior to measuring Euclidean distance when doing nearest neighbor” (Zhao and Chellappa, 2006). This marks the normalisation mentioned earlier as not necessary.

2.4.1.1.5.4 Similarity Measures: Measure chosen

This leads on the choice of a similarity measure. Considering the differences outlined in the section above, the Cosine similarity measure would be most appropriate for the recommendation algorithm. This is as the vectors do not need to be normalized and so avoids adding further steps when it is not needed, thereby making this measurement a better choice out of the two.

2.4.1.2 Genre, Category, Developers, and External critic review score

These attributes have been grouped together as they are simple in terms of their logic, whilst most share the same procedure.

The genre, category, and developer features of a game will be used in a comparison based on how many genres, categories, and developers a game and its comparing game share.

For example, in the case of genre, if game A has genres {"Platformer", "Stealth", "Survival"}, and game B {"Platformer", "Stealth", and "Shooter"} – then the shared games are {"Platformer", "Stealth"}. Out of the total of four unique genres, {"Platformer", "Stealth", "Survival", "Shooter"}, they share two of them. From this, the value produced would be 0.5.

The same logic applies to the "Category" and "Developers" features. The difference is the type of words being compared. For example, in game categories, words such as "Single-player", "multiplayer", "co-op" etc are present. For Developers, this could be any word(s), as they would be names for a development studio.

Finally, the external critic score will be used by calculating how close their critic score ratings are. This would be achieved by dividing the smallest score over the larger one.

Chapter 3: Methodology

This chapter will evaluate three software development methodologies. This will then provide the reasons for the chosen methodology for this project. At the end of this chapter, the risk register will be provided as well as a project plan that will be followed until the completion of this project.

3.1 Software Development Methodologies

3.1.1 Overview

A methodology, by definition, is a system of methods used in a particular area of study or activity. In terms of software engineering, these methods are followed in order to provide greater control over the software development processes (Wong, Tshai and Lee, 2013).

The three distinct software development methodologies that will be evaluated in this chapter are Waterfall, Agile and Spiral.

3.1.2 Waterfall

The waterfall software development methodology is where each phase of a product's life cycle takes place in sequence. This is why it can be described as a waterfall, as progress "flows" downwards towards the end of the development cycle. This methodology is one of the oldest methodologies mentioned in this paper as it was first described, but not mentioned as the waterfall method until a later date by others, by Dr. Winston W. Royce in his 1970 paper "Managing the Development of Large Software Systems", originally published by TRW.

There are multiple variations of the waterfall methodology regarding the steps that can be followed. Generally, they can be condensed into the following order: requirements, design, implementation, verification, and maintenance.

3.1.3 Agile

Agile software development is a software engineering methodology that, at its core, is based on early and continuous delivery of software (Beck, et al., 2001).

It is considered an umbrella term (Guchhait, 2018) that encompasses multiple frameworks, such as Scrum, Crystal, Kanban, XP and others. These frameworks are based on the values and principles expressed in the "Agile Manifesto" (Beck, et al., 2001).

Agile development is to break development down into small blocks where it is then worked on during a designated amount of time. This is usually a small amount of time in order to allow for greater flexibility, more feedback, and a faster turnaround time.

3.1.4 Spiral

Spiral development is a family of software development processes characterized by repeatedly iterating a set of elemental development processes and managing risk, so it is actively being reduced (Boehm, B. and Hansen, W.J., 2000.).

The spiral development model can be considered as a mix between both iterative and waterfall development (Sethi, n.d.). This is as improvements are made iteratively (iterative development) and software released gradually (waterfall development).

3.1.5 Comparison table

The following table encompasses the advantages and disadvantages of the aforementioned methodologies.

Table 4 - Advantages and disadvantages of software development methodologies

	Advantage	Disadvantage
Waterfall	Timescales are easily followed Goal of project determined early Minimal client intervention	Changes are difficult to make Testing is deferred until completion Delivery of working software is slow
Agile	Working software is delivered quicker Greater adaptability to change Flexibility to experiment and test ideas	A lack of designing and documentation Difficult to measure overall progress Prone to scope creeping
Spiral	Major risks avoided Deploy working software faster Focus on documentation control	Possibility spiral may continue indefinitely Success may depend on risk analysis Multiple intermediate stages required

3.1.6 Final Choice

After evaluating these methodologies, it would be best to adopt a combination of both the Agile and Waterfall methodologies for this project.

By setting out the major stages that must be completed, these are the initial analysis, design, development, and testing. It aligns with the milestones that would otherwise be seen in a project with a waterfall development model. For the smaller features, by adopting the agile methodology, it will allow for greater responsiveness to any changes that may be required in order to keep the project on target for completion. This is in addition to providing the flexibility to add in more advance features should time permit.

3.2 Risk Register

The following table contains the risks that this project entails. These are not guaranteed to occur but however have the potential to, so therefore are receiving a mention here.

Table 5 - Risk register

Risk	Likelihood	Impact	Control Measures
New/Changing requirements	Low	Medium	Develop in Iteration
Project too complex	Medium	High	Descope requirements as and when needed
Not complete on time	Medium	High	Anticipate early and divert more resources to the project
Requirements lack clarity	Low	Medium	When encountered, adapt requirements
Loss of access to an API	Low	High	Try to find similar data through another platform's API. If cannot, then re-evaluate requirements and structure of application

3.3 Project Plan

The project plan is located in Appendix 1. This is the timeline of the expected progression of the project.

Chapter 4: Analysis

This chapter will outline the details of the initial analysis of the project. The purpose is to add further depth to the high-level objectives that were listed in 1.4.

4.1 Technical Objectives

As a result of the findings in the literature review, the high-level objectives listed in 1.4 have been developed into the following technical objectives.

1. Front end
    a. Game recommendation page:
        i. Create front end page for entry of video game for game recommendation
        ii. After game entry, display recommended video game with details from IGDB, YouTube, Twitch and Steam
            1. Specifically:
                a. Attain information from IGDB detailing the recommended game
                b. Embedded YouTube videos, based on relevance to the game name
                c. Embedded Twitch Streams of the most popular channels currently streaming the recommended game
                d. Information about the game from steam and a link to the purchase page
    b. Game search page:
        i. Create front end page for entry of a game for further information about the inputted game
        ii. All of what is listed above, under 1. a. ii., for the inputted game, as opposed to the recommended game.
2. Back end
    a. Implement similarity algorithm using:
        i. TF-IDF (Term-Frequency-Inverse-Document-Frequency)
            1. Compute TF-IDF
        ii. Cosine similarity
            1. Compute the similarity between two video game descriptions using value produced by TF-IDF
    b. Integrate Twitch's API in order to:
        i. Receive most watched streams for the inputted/recommended video game
    c. Integrate IGDB's API in order to:
        i. Receive video game data relevant to the inputted/recommended game for the user

d. Integrate YouTube's API in order to:
   i. Receive three most relevant videos, according to YouTube when based on the search term, for the inputted/recommended game

e. Integrate Steam's API in order to:
   i. Receive data, from Steam's storefront, for the inputted/recommended game

4.2 Functional Requirements

Here are the functional requirements for this project.

Key:

WA – Web Application

Table 6 - Functional requirements

	Requirement	
Web Application	1	WA must be accessible online
	2	Users must be able to manually enter a game into the WA in order to retrieve information about inputted game
	3	Users must be able to manually enter a game into the WA in order to retrieve a recommended game
	4	WA must be able to display information from IGDB for the recommended game, if available
	5	WA must be able to display information from IGDB for the inputted game, if available
	6	WA must be able to display the three most relevant videos from YouTube for the recommended game, if available
	7	WA must be able to display the three most relevant videos from YouTube for the inputted game, if available
	8	WA must be able to display top three live streams on Twitch for the recommended game, if available
	9	WA must be able to display top three live streams on Twitch for the inputted game, if available
	10	WA must be able to display information for Steam for recommended game, if available
	11	WA must be able to display information for Steam for inputted game, if available
Recommendation Algorithm	12	Recommendation algorithm must take a single game title as input
	13	Recommendation algorithm must be able to provide a single game as output

Chapter 5: Design

This chapter aims to show the details of the software through design diagrams. This will be demonstrated with the use of a context diagram, to show the overview of the software, and a sequence diagram to show the flow of a request and the process behind a user's request for a recommendation.

5.1 Context Diagram

Figure 5 shows the internal and external components of the web application.

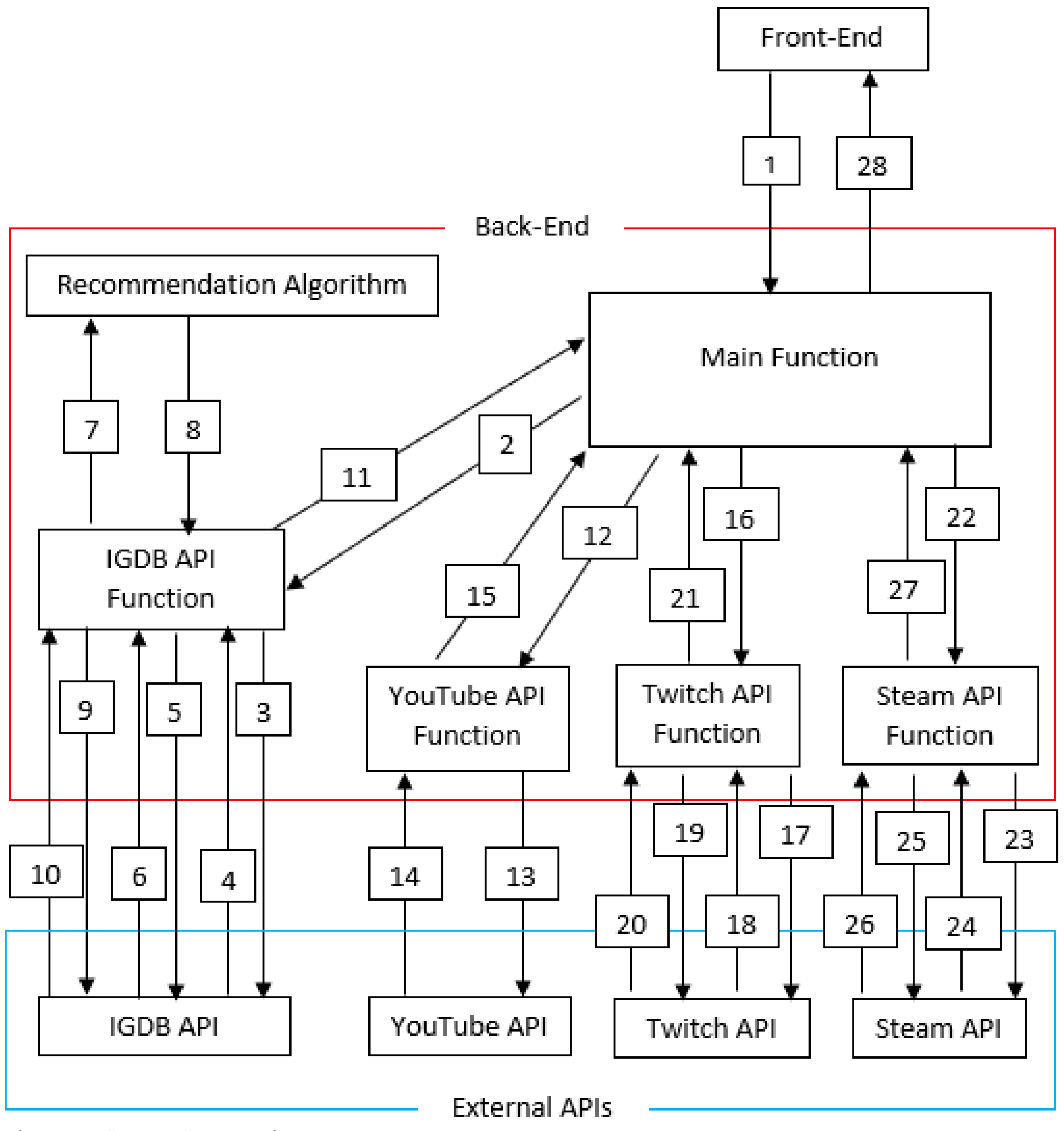


Figure 5 – System Context Diagram

There are six internal, the textboxes within the red box, and five external components, the textboxes within the blue box and the front-end, that make up the web application. The flow of the application is demonstrated by the numbering on the arrows. Note that this is the process behind a recommendation request, if it were not, then steps 5, 6, 7 and 8 are not required. This is as these steps are unique to a recommendation request. With this in mind, the summary of what happens at each number is as follows:

1. A game name is sent from the front end
2. This game name is sent to the IGDB API function, alongside a "HTML string builder". This is string that contains some HTML code that will continuously have more HTML and inline CSS code concatenated to it as the program executes.
3. Information about this game is requested from IGDB's API
4. IGDB's API sends back a JSON response with all information they have about this game
5. Another API query is made, this time to get a list of 500 games that share the same genres as the game that was received from the front end. This call can be made multiple times depending on how many genres the original game has.
6. The list is sent back as a JSON response
7. The data gathered from steps 3 and 5 is sent onto the recommendation algorithm function
8. The most similar game is returned from the list of games gotten from step 5.
9. A request is made to get back a game cover for the game that was recommended from step 8
10. A URL to a cover hosted on IGDB's platform is provided
11. The HTML string builder is what is sent back to the main function as that now contains all of the HTML code to display all of the information that has been found thus far.
12. The YouTube API function is given the recommended game's name
13. A request is made to YouTube's API to get back three videos that is the most relevant to this search term (the game name)
14. The information about these three videos is returned
15. Repeat/Same as step 11
16. The Twitch API function is given the recommended game's name
17. A request is sent to Twitch to get the game ID using the recommended game's name
18. If Twitch has this game ID, this is what is returned back to the application
19. Using Twitch's game ID, the application requests the top three streamers that are streaming this game
20. JSON response is received with this data
21. Repeat/Same as step 11
22. The recommended game name is passed into the Steam API function
23. A request is made to Steam's API asking for a list of all Steam games
24. A list of games is returned as a JSON response
25. From this list, if the game is found using the recommended game name, then Steam's game ID is found. This Steam game ID is sent with a request to get details about what Steam is currently showing on their storefront for this game
26. The details are received from Steam as a JSON response
27. Repeat/Same as step 11
28. Finally, the HTML string builder is sent back to the front end, generating all the information gathered from all the APIs.

5.2 Sequence Diagram

Whilst the context diagram shows an overview of the system, it does not easily show the complete flow of information. The sequence diagram found in figure 6 will show this, as well as other aspects such as the looping done in the IGDB function that was difficult to show in the context diagram. The process outlined is a user requesting a game recommendation from the frontend.

Similar to the context diagram, if the request made by the user was for more information about a specific game, then in figure 6, steps 1.1.3, 1.1.4, 1.1.5 and 1.1.6 are not required. Additionally, any mention of “RecommendedGame” would also be replaced with “InputtedGame”.

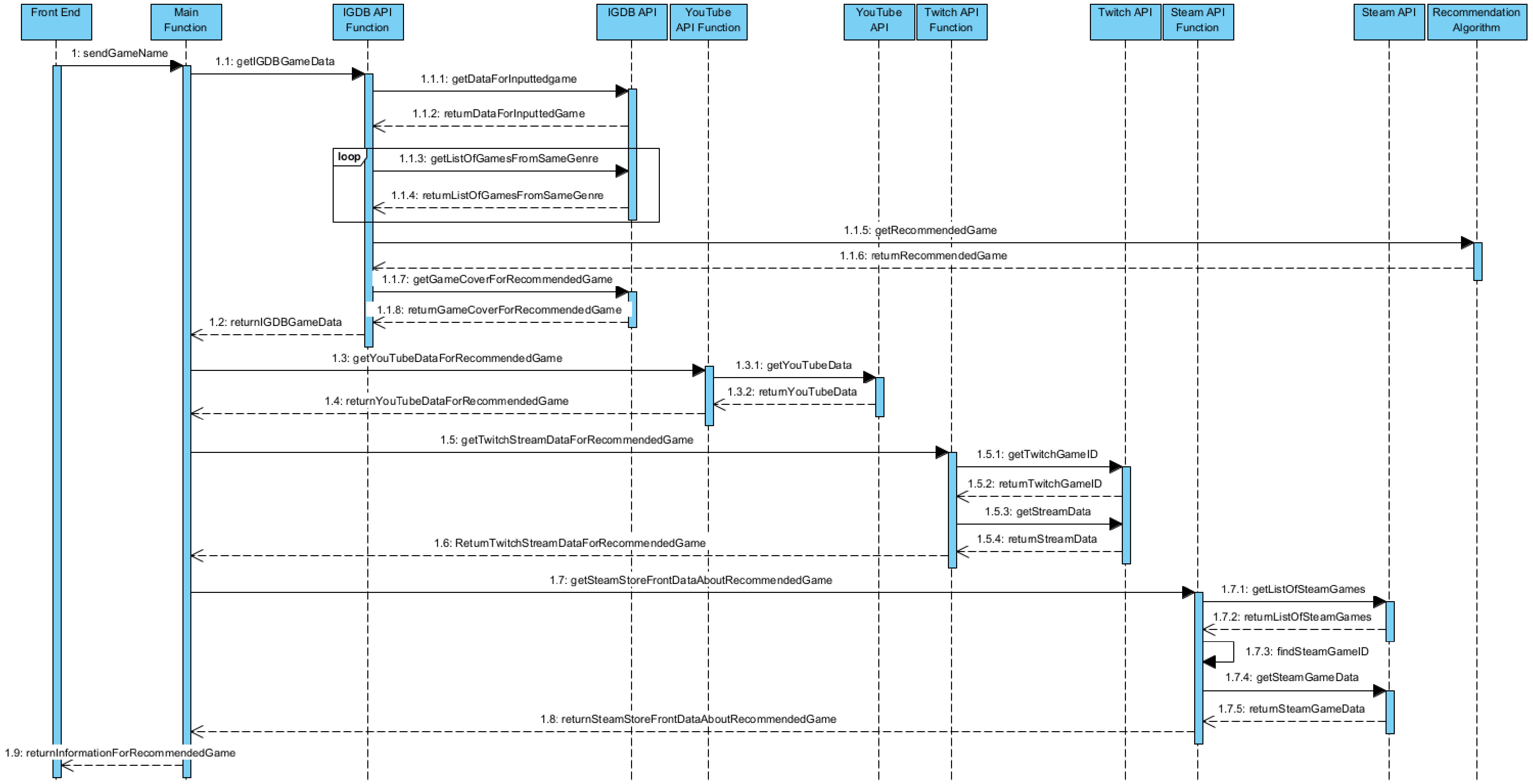

Front End
Main Function
IGDB API Function
IGDB API
YouTube API Function
YouTube API
Twitch API Function
Twitch API
Steam API Function
Steam API
Recommendation Algorithm
1: sendGameName
1.1: getIGDBGameData
1.1.1: getDataForInputtedgame
1.1.2: returnDataForInputtedGame
loop
1.1.3: getListOfGamesFromSameGenre
1.1.4: returnListOfGamesFromSameGenre
1.1.5: getRecommendedGame
1.1.6: returnRecommendedGame
1.1.7: getGameCoverForRecommendedGame
1.1.8: returnGameCoverForRecommendedGame
1.2: returnIGDBGameData
1.3: getYouTubeDataForRecommendedGame
1.3.1: getYouTubeData
1.3.2: returnYouTubeData
1.4: returnYouTubeDataForRecommendedGame
1.5: getTwitchStreamDataForRecommendedGame
1.5.1: getTwitchGameID
1.5.2: returnTwitchGameID
1.5.3: getStreamData
1.5.4: returnStreamData
1.6: ReturnTwitchStreamDataForRecommendedGame
1.7: getSteamStoreFrontDataAboutRecommendedGame
1.7.1: getListOfSteamGames
1.7.2: returnListOfSteamGames
1.7.3: findSteamGameID
1.7.4: getSteamGameData
1.7.5: returnSteamGameData
1.8: returnSteamStoreFrontDataAboutRecommendedGame
1.9: returnInformationForRecommendedGame

Figure 6 - Sequence Diagram

Chapter 6: Implementation

This chapter will discuss the details of the designs outlined in Chapter 5. The order will be as follows: third-party dependencies used, the details of the main function, the API functions, the use of APIs, the recommendation algorithm, and finally where it is hosted.

6.1 Third Party Dependencies

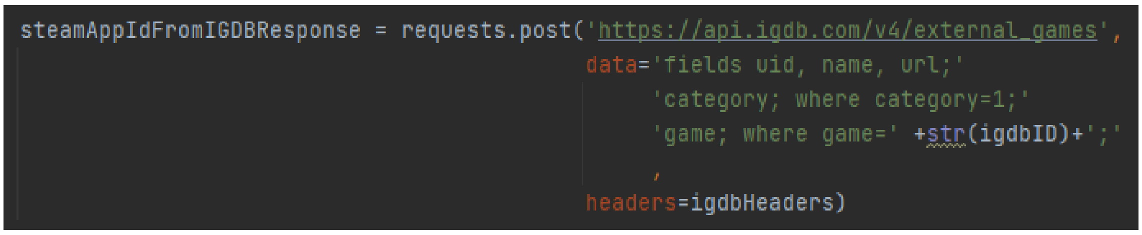
As the backend was developed with Python, the following, Python specific, third-party dependencies were used to aid in the development of the software:

- Requests (v. 2.25.1)
- Scikit-learn (v. 0.24.1)
- Pandas (v. 1.1.5)

6.1.1 Requests

The Requests dependency (requests 2.25.1, 2020) is a HTTP library that simplifies the process of making HTTP/1.1 requests. The utilisation of this dependency removes the requirement of manual form-encoding of PUT and POST data, or the addition of query strings to URLs.

The most common occurrence of the use of this dependency within the project was the use of the POST function. An example of a use can be seen in figure 7 below, where the program is making a call to the IGDB API to request the ID of a game on Steam's platform.



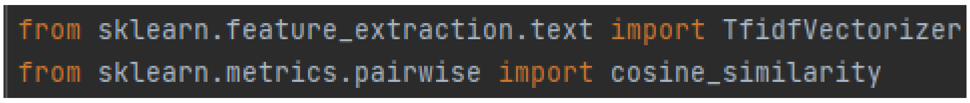
```
steamAppIdFromIGDBResponse = requests.post('https://api.igdb.com/v4/external_games',
                                           data='fields uid, name, url;'
                                                'category; where category=1;'
                                                'game; where game=' +str(igdbID)+';'
                                                ,
                                           headers=igdbHeaders)
```

Figure 7 – Example of a Request POST call from a request object used in the project

6.1.2 Scikit-learn

Scikit-learn (scikit-learn 0.24.1, 2021) is a library for Python that encompasses many tools to aid in the development of machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction (Jain, 2015).

This dependency was necessary for the project as it contains the tools for the natural language processing segment of the recommendation algorithm. In particular, the "TfidfVectorizer" class and the "cosine_similarity" function from the "sklearn.feature_extraction.text" and "sklearn.metrics.pairwise" modules. The snippet has been provided below (figure 8).

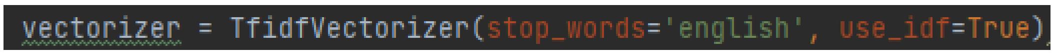


```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Figure 8 – Sklearn functions imported from their respective modules

These two imports already contain the code required for the TF_IDF and cosine similarity calculations, so there was no need to write new code. These were used as follows: "TfidfVectorizer" was required to initialise a "vectorizer", which is an object that will convert a collection of raw documents into a matrix of TF-IDF values. The parameters used were "stop_words" (common words in a language removed) and "use_idf" as "True". The use of IDF as "True" enables the use of inverse-document-frequency reweighting in the final calculation.

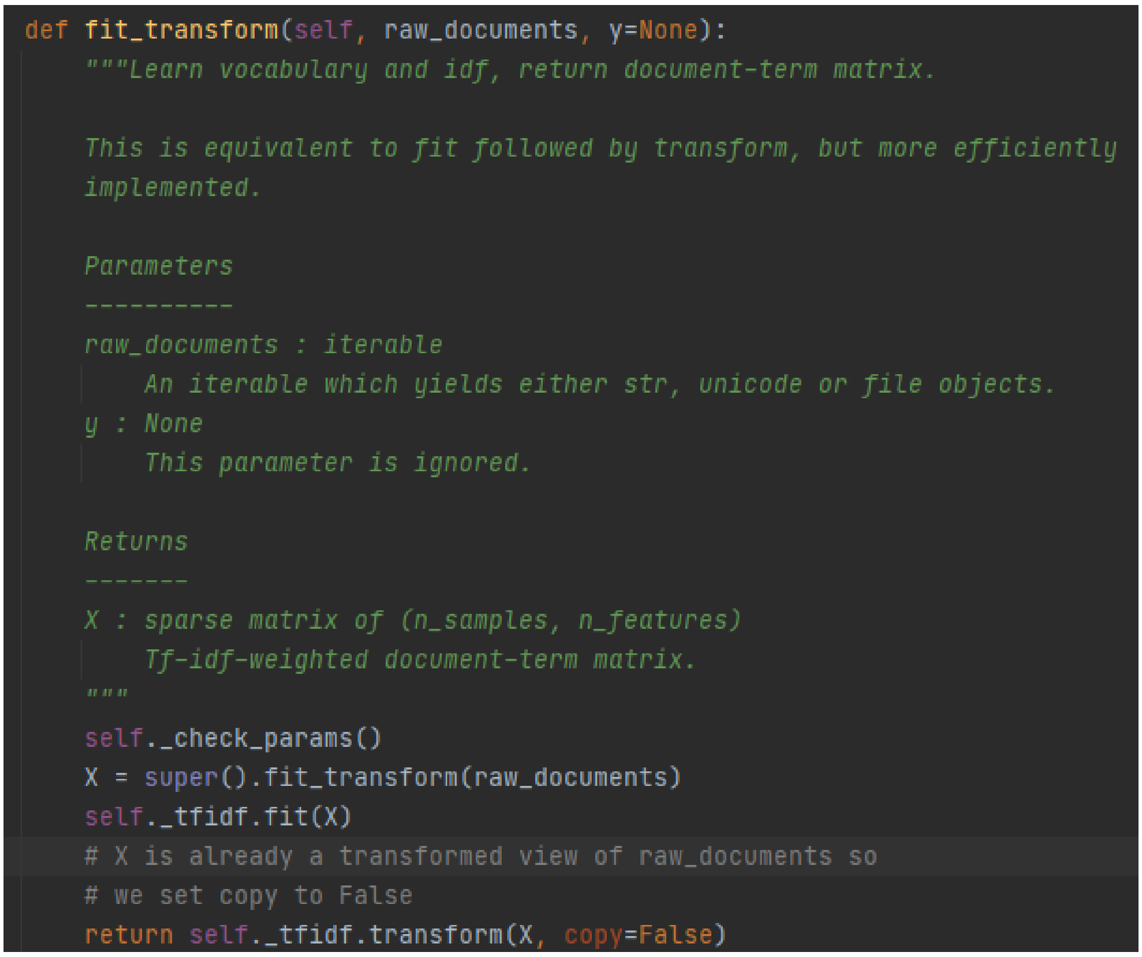
The figure below will help clarify how the vectorizer object is initialized.



```python
vectorizer = TfidfVectorizer(stop_words='english', use_idf=True)
```

Figure 9 – TfidfVectorizer initialisation

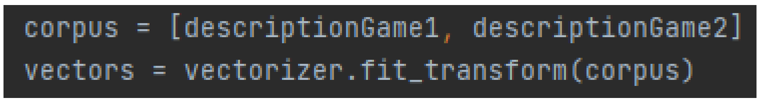
The next part where these imports from Sklearn are used, is the learning of the vocabulary and the IDF of the corpus, and then the return of this matrix. This is achieved by the "fit_transform()" function that is part of the "TfidfVectorizer" class:



```python
def fit_transform(self, raw_documents, y=None):
    """Learn vocabulary and idf, return document-term matrix.

    This is equivalent to fit followed by transform, but more efficiently
    implemented.

    Parameters
    ----------
    raw_documents : iterable
        An iterable which yields either str, unicode or file objects.
    y : None
        This parameter is ignored.

    Returns
    -------
    X : sparse matrix of (n_samples, n_features)
        Tf-idf-weighted document-term matrix.
    """
    self._check_params()
    X = super().fit_transform(raw_documents)
    self._tfidf.fit(X)
    # X is already a transformed view of raw_documents so
    # we set copy to False
    return self._tfidf.transform(X, copy=False)
```

Figure 10 – "fit_transform" function within "TfidfVectorizer" class (code snippet taken from Scitkit-learn's "sklearn.feature_extraction.text.TfidfVectorizer .fit_transform()" function)

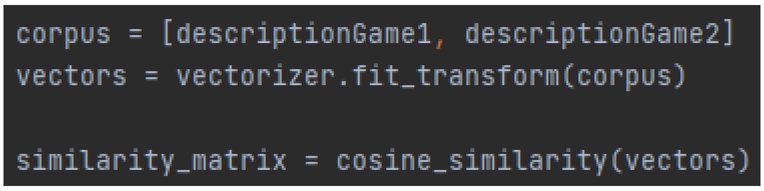
This function was implemented in the following manner:



```
corpus = [descriptionGame1, descriptionGame2]
vectors = vectorizer.fit_transform(corpus)
```

Figure 11 – "TfidfVectorizer.fit_transform" function implementation

This produces the matrices that represent each game description as a vector, thereby leading onto the use of the "cosine_similarity" function. This is where the cosine similarity value is produced for the two game descriptions. This was implemented soon after figure 11:



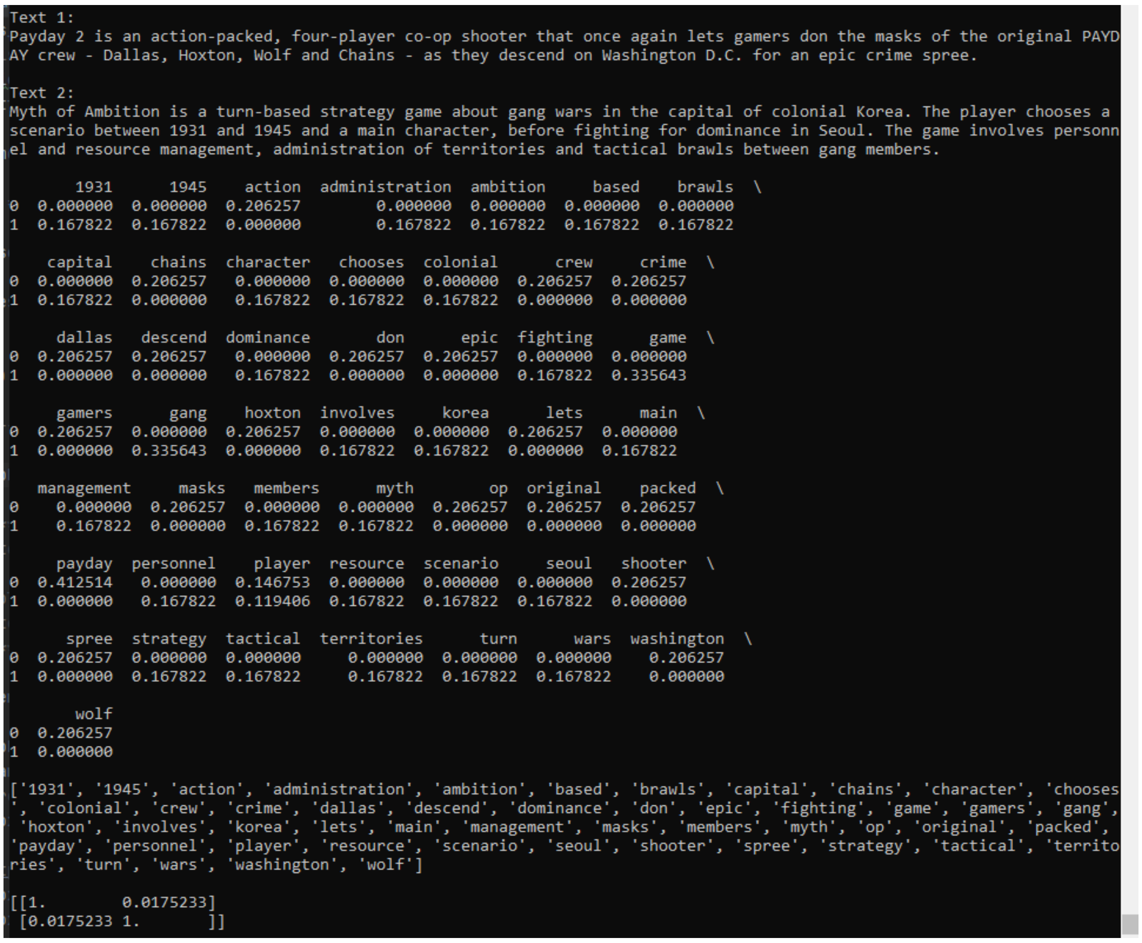
```
corpus = [descriptionGame1, descriptionGame2]
vectors = vectorizer.fit_transform(corpus)

similarity_matrix = cosine_similarity(vectors)
```

Figure 12 – Cosine similarity function utilisation

6.1.3 Pandas

Pandas (pandas, 2021) is a Python package that includes additional data structures that are designed to make working with structured data (tabular, multidimensional, potentially heterogeneous) easier and intuitive (pandas 1.1.5, 2020).

This dependency was required for the project as it made interpreting the data produced from the vectorization and cosine similarity process easier. The following example will demonstrate this, it is a game that was chosen at random during the execution of the recommendation algorithm. The game that was inputted into the frontend for a recommendation to be based off was "Payday 2", this game's description can also be seen under "Text 1" from the console output below in figure 13.



```
Text 1:
Payday 2 is an action-packed, four-player co-op shooter that once again lets gamers don the masks of the original PAYD
AY crew - Dallas, Hoxton, Wolf and Chains - as they descend on Washington D.C. for an epic crime spree.

Text 2:
Myth of Ambition is a turn-based strategy game about gang wars in the capital of colonial Korea. The player chooses a
scenario between 1931 and 1945 and a main character, before fighting for dominance in Seoul. The game involves personn
el and resource management, administration of territories and tactical brawls between gang members.

       1931      1945    action  administration  ambition     based    brawls  \
0  0.000000  0.000000  0.206257        0.000000  0.000000  0.000000  0.000000
1  0.167822  0.167822  0.000000        0.167822  0.167822  0.167822  0.167822

    capital    chains  character   chooses  colonial      crew     crime  \
0  0.000000  0.206257   0.000000  0.000000  0.000000  0.206257  0.206257
1  0.167822  0.000000   0.167822  0.167822  0.167822  0.000000  0.000000

     dallas   descend  dominance       don      epic  fighting      game  \
0  0.206257  0.206257   0.000000  0.206257  0.206257  0.000000  0.000000
1  0.000000  0.000000   0.167822  0.000000  0.000000  0.167822  0.335643

     gamers      gang    hoxton  involves     korea      lets      main  \
0  0.206257  0.000000  0.206257  0.000000  0.000000  0.206257  0.000000
1  0.000000  0.335643  0.000000  0.167822  0.167822  0.000000  0.167822

   management     masks   members      myth        op  original    packed  \
0    0.000000  0.206257  0.000000  0.000000  0.206257  0.206257  0.206257
1    0.167822  0.000000  0.167822  0.167822  0.000000  0.000000  0.000000

     payday  personnel    player  resource  scenario     seoul   shooter  \
0  0.412514   0.000000  0.146753  0.000000  0.000000  0.000000  0.206257
1  0.000000   0.167822  0.119406  0.167822  0.167822  0.167822  0.000000

      spree  strategy  tactical  territories      turn      wars  washington  \
0  0.206257  0.000000  0.000000     0.000000  0.000000  0.000000    0.206257
1  0.000000  0.167822  0.167822     0.167822  0.167822  0.167822    0.000000

       wolf
0  0.206257
1  0.000000

['1931', '1945', 'action', 'administration', 'ambition', 'based', 'brawls', 'capital', 'chains', 'character', 'chooses
', 'colonial', 'crew', 'crime', 'dallas', 'descend', 'dominance', 'don', 'epic', 'fighting', 'game', 'gamers', 'gang',
 'hoxton', 'involves', 'korea', 'lets', 'main', 'management', 'masks', 'members', 'myth', 'op', 'original', 'packed',
'payday', 'personnel', 'player', 'resource', 'scenario', 'seoul', 'shooter', 'spree', 'strategy', 'tactical', 'territo
ries', 'turn', 'wars', 'washington', 'wolf']

[[1.        0.0175233]
 [0.0175233 1.       ]]
```

Figure 13 – tf-idf function output with use of pandas DataFrame object

The first two blocks of text seen in Figure 13 are "Text 1" and "Text 2". These are the descriptions for the games, where, as mentioned earlier, Text 1 is the description of the original inputted game, and Text 2 is the description of the game it is currently being compared against.

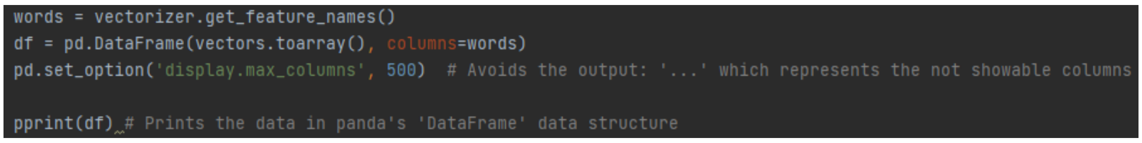
The next set of text that can be seen is a representation of the vectors in a tabular form, with the column names denoting the word, and rows the TF-IDF weightings for each document. Rows labelled "0" are a representation of words from Text 1 and rows labelled "1" are a representation of words from Text 2.

The next block of text in the console output are all the words in the corpus. Note that stop words have been used, so the most common words that are seen initially in Text 1 and 2 are not featured here, this was the primary reason for using the stop words as using TF-IDF devalues common occurring words naturally, as mentioned in 2.4.1.1.2. This meant the data was easier to navigate and investigate as there were less words to look through in the table of vectors that were mentioned in the prior paragraph.

Finally, the matrix seen at the end is the similarity matrix produced after performing the cosine similarity calculation on the vectors, seen in the last line of figure 12.

Where pandas proved the most use was table with the TF-IDF weightings, seen in the centre of Figure 13. This was only achievable with the "DataFrame" class from the pandas package (pandas.DataFrame, 2021). This made it significantly easier to see what values were being calculated for each word in the corpus in relation to the two provided texts.

The code required to receive this output is below, note that “pandas” has been imported as “pd”:

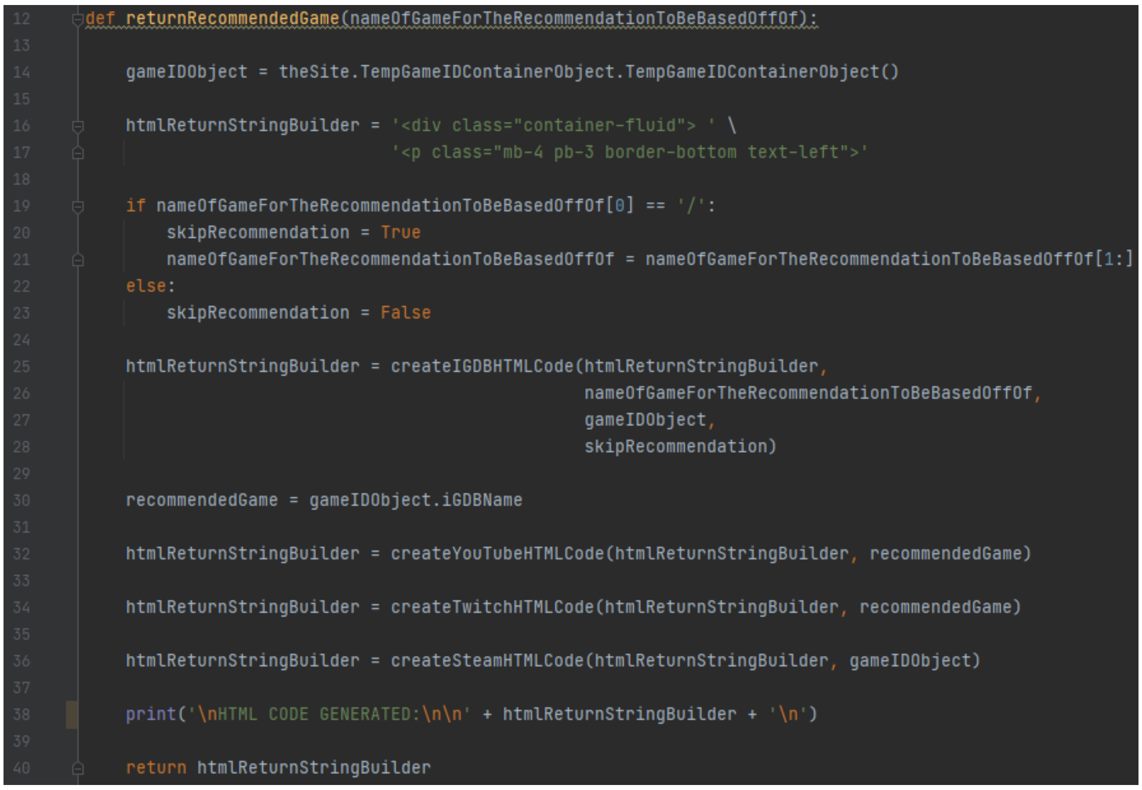


```
words = vectorizer.get_feature_names()
df = pd.DataFrame(vectors.toarray(), columns=words)
pd.set_option('display.max_columns', 500)  # Avoids the output: '...' which represents the not showable columns

pprint(df) # Prints the data in panda's 'DataFrame' data structure
```

Figure 14 – Pandas DataFrame class in use

6.2 Main function

The main function is an internal component, as seen in figure 5, within the software that handles the calling of the other functions, and then returns the combined result of these as a string that contains HTML and inline CSS. This is what will render the results on the front end, once returned. The code for this function can be seen below.



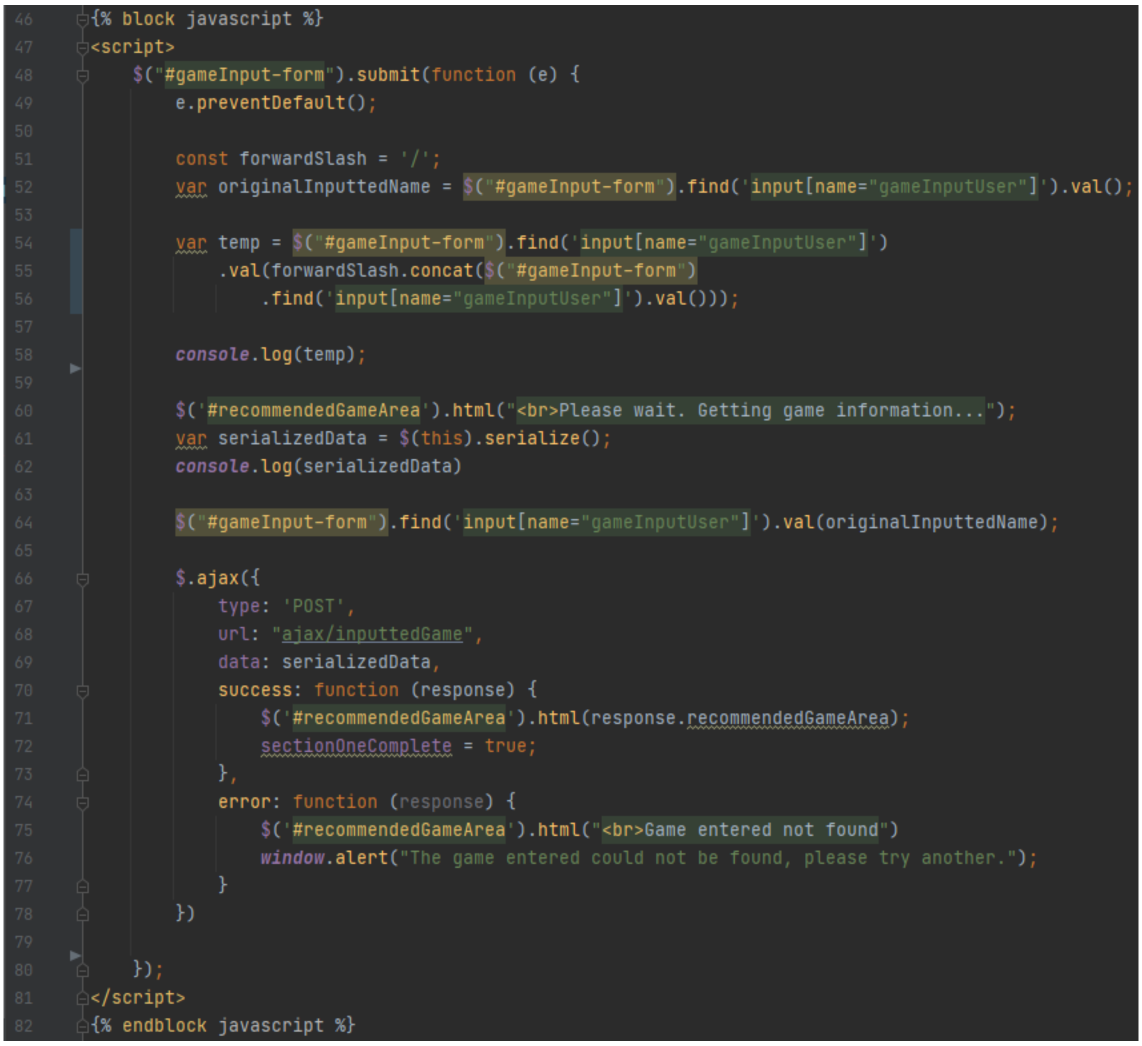
```
def returnRecommendedGame(nameOfGameForTheRecommendationToBeBasedOffOf):

    gameIDObject = theSite.TempGameIDContainerObject.TempGameIDContainerObject()

    htmlReturnStringBuilder = '<div class="container-fluid"> ' \
                              '<p class="mb-4 pb-3 border-bottom text-left">'

    if nameOfGameForTheRecommendationToBeBasedOffOf[0] == '/':
        skipRecommendation = True
        nameOfGameForTheRecommendationToBeBasedOffOf = nameOfGameForTheRecommendationToBeBasedOffOf[1:]
    else:
        skipRecommendation = False

    htmlReturnStringBuilder = createIGDBHTMLCode(htmlReturnStringBuilder,
                                                 nameOfGameForTheRecommendationToBeBasedOffOf,
                                                 gameIDObject,
                                                 skipRecommendation)

    recommendedGame = gameIDObject.iGDBName

    htmlReturnStringBuilder = createYouTubeHTMLCode(htmlReturnStringBuilder, recommendedGame)

    htmlReturnStringBuilder = createTwitchHTMLCode(htmlReturnStringBuilder, recommendedGame)

    htmlReturnStringBuilder = createSteamHTMLCode(htmlReturnStringBuilder, gameIDObject)

    print('\nHTML CODE GENERATED:\n\n' + htmlReturnStringBuilder + '\n')

    return htmlReturnStringBuilder
```

Figure 15 - Main function Code

The first task that the main function carries out is the initialisation of the object “gameIDObject”. This initialised object belongs to a class whose purpose is to keep a track of the IDs of IGDB and Steam for the recommended or inputted game. The main purpose of this object is that sometimes IGDB has a steam ID in the data that is sent back for a game. This means the program would not need to call steam’s API for a list of games on these occasions and can skip to calling Steam’s API for the store front details of the game.

Next, from lines 16-17, you see the first occurrence of the “htmlReturnStringBuilder”. This is the object that contains the HTML and inline CSS code mentioned in the opening paragraph of this section. Its purpose is to be continuously passed into functions to have additional HTML/CSS code concatenated to it, and then for it to be returned, as seen in line 40 in figure 15.

Lines 19 to 23 contain an if else block that’s purpose is to determine if the incoming game is to be run through the recommendation algorithm or not. The way this has been structured, is that in the front end, depending on which page the game was inputted (the pages available are the “Game Recommendation Page” and “Game Search Page”), that if the input was entered in the “Game Search Page”, then the jQuery at the frontend will concatenate a forward slash to the input without the user knowing. Lines 54 to 56 in the figure below shows this.



```
46  {% block javascript %}
47  <script>
48      $("#gameInput-form").submit(function (e) {
49          e.preventDefault();
50
51          const forwardSlash = '/';
52          var originalInputtedName = $("#gameInput-form").find('input[name="gameInputUser"]').val();
53
54          var temp = $("#gameInput-form").find('input[name="gameInputUser"]')
55              .val(forwardSlash.concat($("#gameInput-form")
56                  .find('input[name="gameInputUser"]').val()));
57
58          console.log(temp);
59
60          $('#recommendedGameArea').html("<br>Please wait. Getting game information...");
61          var serializedData = $(this).serialize();
62          console.log(serializedData)
63
64          $("#gameInput-form").find('input[name="gameInputUser"]').val(originalInputtedName);
65
66          $.ajax({
67              type: 'POST',
68              url: "ajax/inputtedGame",
69              data: serializedData,
70              success: function (response) {
71                  $('#recommendedGameArea').html(response.recommendedGameArea);
72                  sectionOneComplete = true;
73              },
74              error: function (response) {
75                  $('#recommendedGameArea').html("<br>Game entered not found")
76                  window.alert("The game entered could not be found, please try another.");
77              }
78          })
79
80      });
81  </script>
82  {% endblock javascript %}
```

Figure 16 - Front end JavaScript and jQuery code for “Game Search Page”

When the request comes through, it will be in the format “/<Game Name>”, to which the forward slash will be removed in line 21 in figure 15. This would not occur when the user enters input on the “Game Recommendation Page”.

After this if else block, begins the section where all the calls to the API functions are made. The first function called is “createIGDBHTMLCode”, which is the IGDB API Function seen in figure 5. This is the most complex internal component as it calls the recommendation algorithm and has the most calls to any API within the project, this can also be seen in figure 5. This function will be explained in the next section of this chapter.

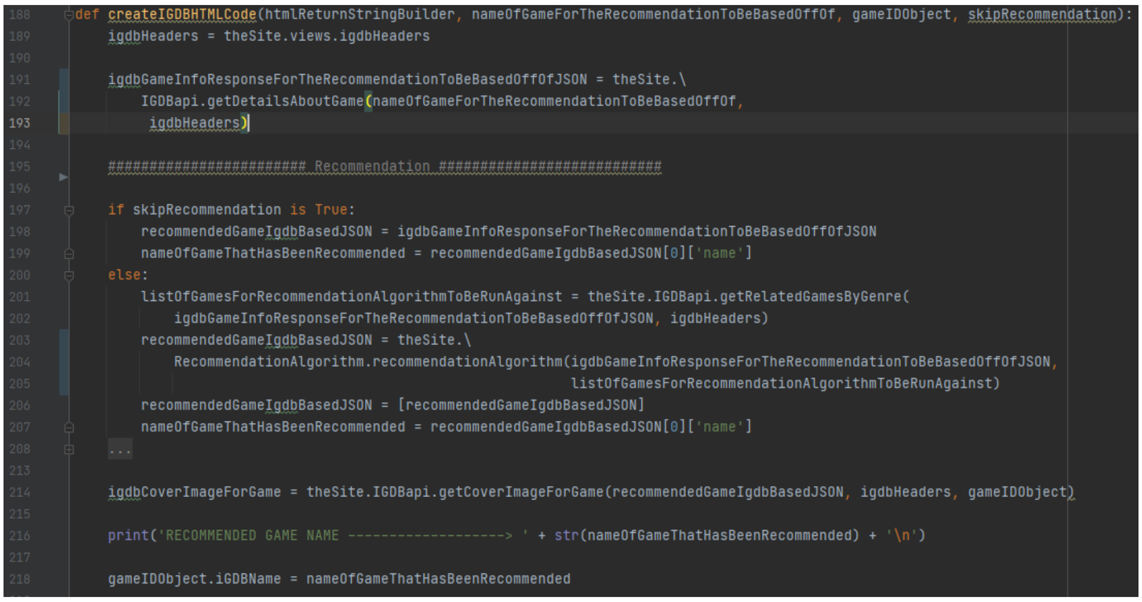
This concludes the code for this function. To see an example of what the code returned to the front end would look like when printed, see Appendix 3. This is the code printed from the print statement on line 38 in figure 15.

6.3 API Functions

As a reminder, there are four API functions: IGDB, YouTube, Twitch and Steam. As the IGDB function is the most complex, this will be the function discussed. The similarities between the functions will also be mentioned explicitly to help ease the issue of their absence.

6.3.1 IGDB Function

The purpose of this function is to get all the details for the specified or recommended game, depending on what the user request is. It also calls the recommendation algorithm function as it is easier to make the subsequent call to the IGDB cover art endpoint within the same function.

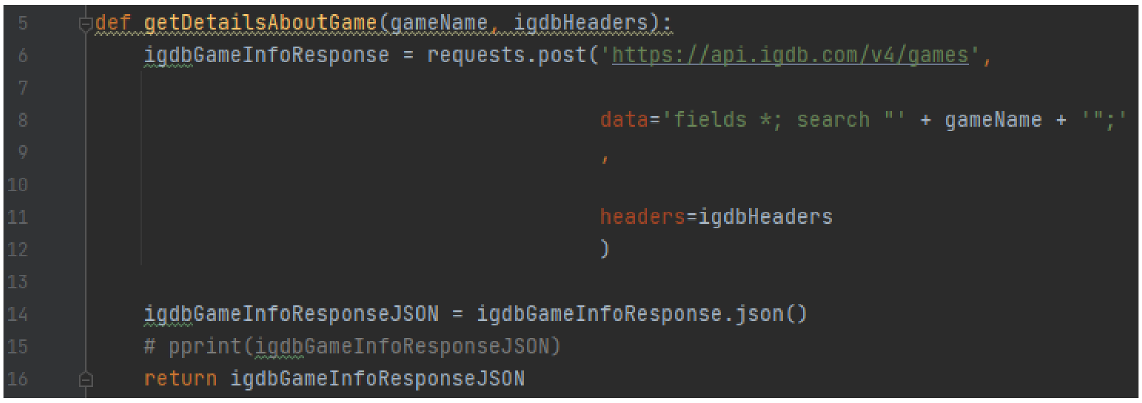


```
188 def createIGDBHTMLCode(htmlReturnStringBuilder, nameOfGameForTheRecommendationToBeBasedOffOf, gameIDObject, skipRecommendation):
189     igdbHeaders = theSite.views.igdbHeaders
190 
191     igdbGameInfoResponseForTheRecommendationToBeBasedOffOfJSON = theSite.\
192         IGDBapi.getDetailsAboutGame(nameOfGameForTheRecommendationToBeBasedOffOf,
193          igdbHeaders)
194 
195     ######################## Recommendation ###########################
196 
197     if skipRecommendation is True:
198         recommendedGameIgdbBasedJSON = igdbGameInfoResponseForTheRecommendationToBeBasedOffOfJSON
199         nameOfGameThatHasBeenRecommended = recommendedGameIgdbBasedJSON[0]['name']
200     else:
201         listOfGamesForRecommendationAlgorithmToBeRunAgainst = theSite.IGDBapi.getRelatedGamesByGenre(
202             igdbGameInfoResponseForTheRecommendationToBeBasedOffOfJSON, igdbHeaders)
203         recommendedGameIgdbBasedJSON = theSite.\
204             RecommendationAlgorithm.recommendationAlgorithm(igdbGameInfoResponseForTheRecommendationToBeBasedOffOfJSON,
205                                                             listOfGamesForRecommendationAlgorithmToBeRunAgainst)
206         recommendedGameIgdbBasedJSON = [recommendedGameIgdbBasedJSON]
207         nameOfGameThatHasBeenRecommended = recommendedGameIgdbBasedJSON[0]['name']
208     ...
213 
214     igdbCoverImageForGame = theSite.IGDBapi.getCoverImageForGame(recommendedGameIgdbBasedJSON, igdbHeaders, gameIDObject)
215 
216     print('RECOMMENDED GAME NAME ------------------> ' + str(nameOfGameThatHasBeenRecommended) + '\n')
217 
218     gameIDObject.iGDBName = nameOfGameThatHasBeenRecommended
```

Figure 17 - IGDB Function (part 1 of 2)

Line 189 gets the headers, which contain all the authentication details that are required to make requests to IGDB's API.

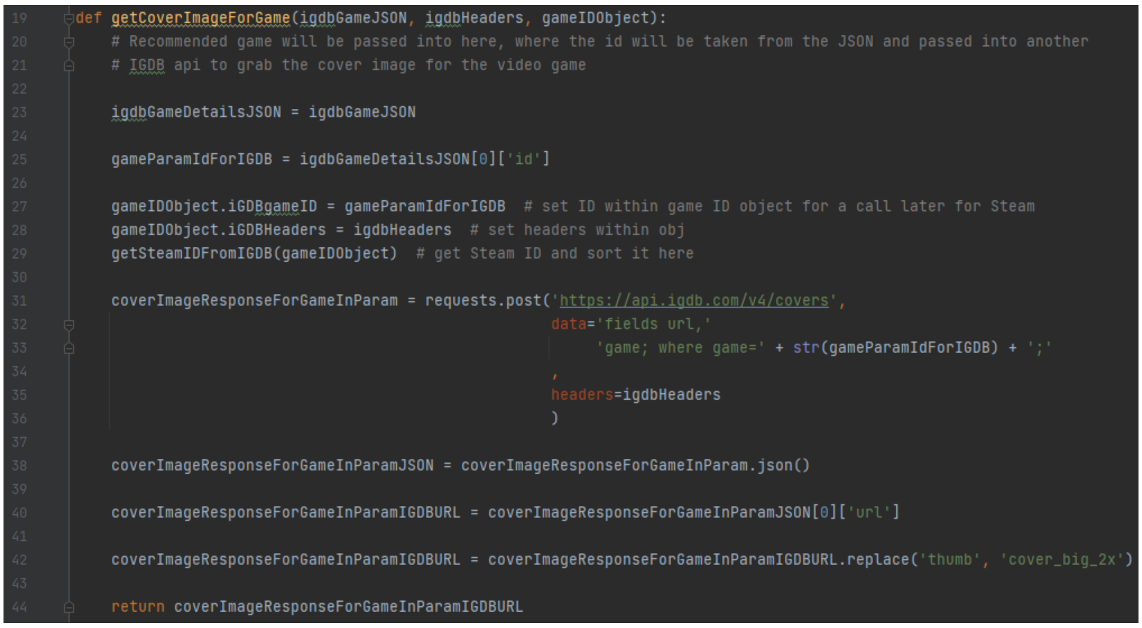
Lines 191 to 193 contain the code for the first request to the IGDB's API through another function that's purpose is to send and receive data about the queried game. This function can be seen below.



```python
def getDetailsAboutGame(gameName, igdbHeaders):
    igdbGameInfoResponse = requests.post('https://api.igdb.com/v4/games',

                                         data='fields *; search "' + gameName + '";'
                                         ,

                                         headers=igdbHeaders
                                         )

    igdbGameInfoResponseJSON = igdbGameInfoResponse.json()
    # pprint(igdbGameInfoResponseJSON)
    return igdbGameInfoResponseJSON
```

Figure 18 - "getDetailsAboutGame" IGDB request function

From lines 197 to 207, in figure 17, there is an if else block that determines if the recommendation algorithm is called or not. The “skipRecommendation” variable was set in figure 15 on line 20. The outcome of this block of code can either be a recommended game, or the original inputted game in the front end. The cover of the game is retrieved next on line 214. The function that is called can be seen below.

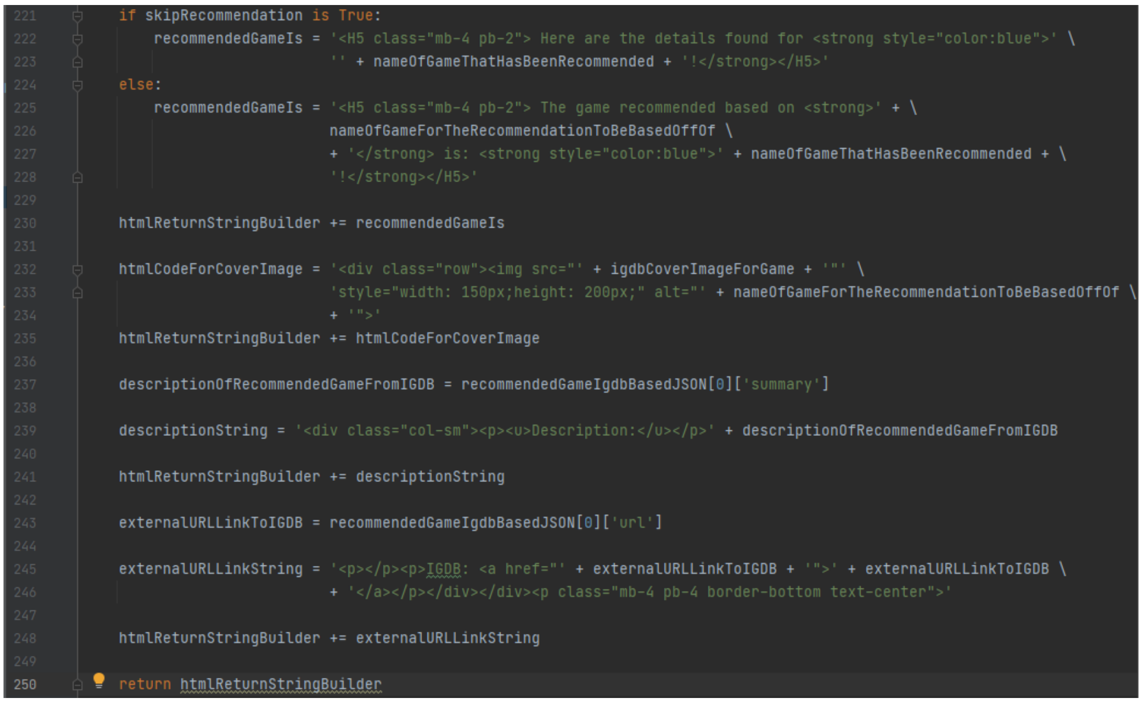


```python
def getCoverImageForGame(igdbGameJSON, igdbHeaders, gameIDObject):
    # Recommended game will be passed into here, where the id will be taken from the JSON and passed into another
    # IGDB api to grab the cover image for the video game

    igdbGameDetailsJSON = igdbGameJSON

    gameParamIdForIGDB = igdbGameDetailsJSON[0]['id']

    gameIDObject.iGDBgameID = gameParamIdForIGDB  # set ID within game ID object for a call later for Steam
    gameIDObject.iGDBHeaders = igdbHeaders  # set headers within obj
    getSteamIDFromIGDB(gameIDObject)  # get Steam ID and sort it here

    coverImageResponseForGameInParam = requests.post('https://api.igdb.com/v4/covers',
                                                     data='fields url,'
                                                          'game; where game=' + str(gameParamIdForIGDB) + ';'
                                                     ,
                                                     headers=igdbHeaders
                                                     )

    coverImageResponseForGameInParamJSON = coverImageResponseForGameInParam.json()

    coverImageResponseForGameInParamIGDBURL = coverImageResponseForGameInParamJSON[0]['url']

    coverImageResponseForGameInParamIGDBURL = coverImageResponseForGameInParamIGDBURL.replace('thumb', 'cover_big_2x')

    return coverImageResponseForGameInParamIGDBURL
```

Figure 19 - "getCoverImageForGame" function using IGDB's API

Line 218, in figure 17, sets the name of the game, exactly as previously returned by IGDB, to the game ID object. This is so it can be used to search the list of steam games later if needed. This is as a game is more likely to be found with the format of the official entry in IGDB. The formatting of the text is important and is case and symbol sensitive when searching Steam’s game list.

The next figure below is the remainder of the IGDB function.



```
if skipRecommendation is True:
    recommendedGameIs = '<H5 class="mb-4 pb-2"> Here are the details found for <strong style="color:blue">' \
                        '' + nameOfGameThatHasBeenRecommended + '!</strong></H5>'
else:
    recommendedGameIs = '<H5 class="mb-4 pb-2"> The game recommended based on <strong>' + \
                        nameOfGameForTheRecommendationToBeBasedOffOf \
                        + '</strong> is: <strong style="color:blue">' + nameOfGameThatHasBeenRecommended + \
                        '!</strong></H5>'

htmlReturnStringBuilder += recommendedGameIs

htmlCodeForCoverImage = '<div class="row"><img src="' + igdbCoverImageForGame + '"' \
                        'style="width: 150px;height: 200px;" alt="' + nameOfGameForTheRecommendationToBeBasedOffOf \
                        + '">'
htmlReturnStringBuilder += htmlCodeForCoverImage

descriptionOfRecommendedGameFromIGDB = recommendedGameIgdbBasedJSON[0]['summary']

descriptionString = '<div class="col-sm"><p><u>Description:</u></p>' + descriptionOfRecommendedGameFromIGDB

htmlReturnStringBuilder += descriptionString

externalURLLinkToIGDB = recommendedGameIgdbBasedJSON[0]['url']

externalURLLinkString = '<p></p><p>IGDB: <a href="' + externalURLLinkToIGDB + '">' + externalURLLinkToIGDB \
                        + '</a></p></div></div><p class="mb-4 pb-4 border-bottom text-center">'

htmlReturnStringBuilder += externalURLLinkString

return htmlReturnStringBuilder
```

Figure 20 - IGDB Function (part 2 of 2)

The first if else block seen is present to determine which text to prepare for concatenation to the htmlReturnStringBuilder, as the output needs to be different if the game is a recommended one or the same as the inputted one.

What is seen from thereon after is very similar to other API functions. This is where the values are taken from the JSON files returned from each API and then concatenated to the htmlReturnStringBuilder.

6.4 API Use

The details of how APIs were used will be found in the appendices 5, 6, 7, 8. This is where the appendix numbers are:

5. For IGDB
6. For YouTube
7. For Twitch
8. For Steam

These are not mandatory to read but do contain detailed information on what endpoints were used to attain what specific data.

Note that if reading any of the aforementioned appendices, the structure of lists provided when outlining the use of each API is as follows: the header denotes the goal behind all of the items in the list below it. Thereafter, are the endpoint(s) used to obtain the list of values underneath it. The words within single quotations are what the keys were in order to attain the value (key, value) from the JSON responses, with the description on the right summarising each item in the list.

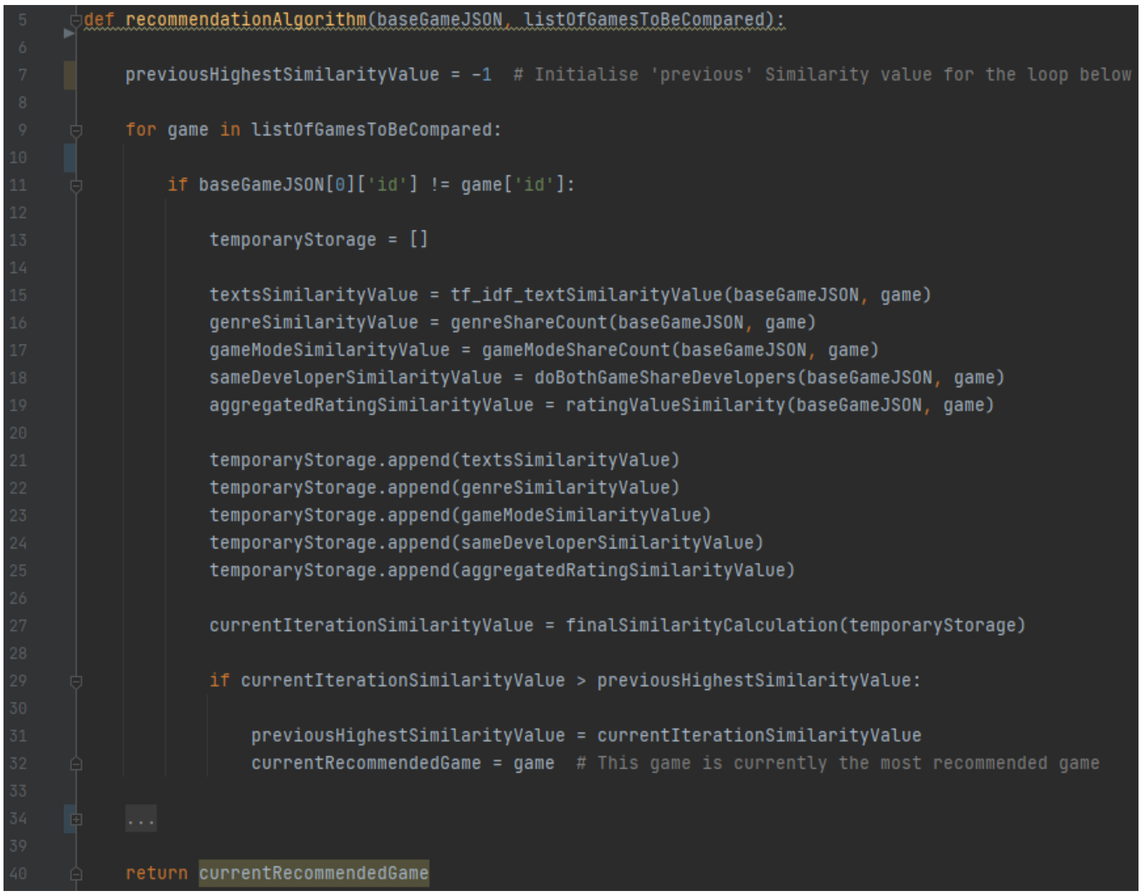
6.5 Recommendation Algorithm

By the time the recommendation algorithm is called, the data due for processing is as complete as it can be. As a reminder, the algorithm returns the game with the greatest overall score that is averaged from the following categories:

- Game description similarity
- Genres shared
- Categories shared (game modes)
- Developers shared
- External critic scores

The overall flow of the function is that it will continuously iterate over a list of games that were provided by IGDB, as seen in the loop in figure 6, where the number of games in the list is produced by how many genres the original game has multiplied by 500. This is as 500 is the maximum number of games IGDB's API will send back per call.

All the functions that calculate each part of the listed values above, are called. The final similarity value is then calculated from the values produced from each function. If this overall value is higher than the previous highest, it will take the position of most similar. Once the iterations are complete, the game that is in this position is returned.

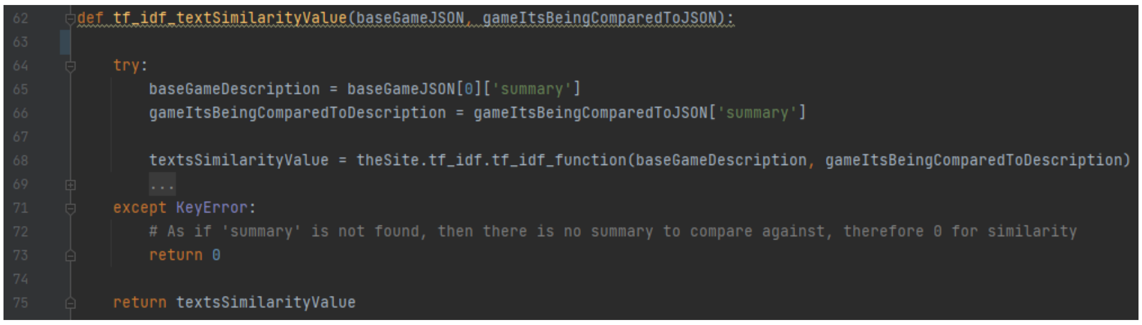


```python
def recommendationAlgorithm(baseGameJSON, listOfGamesToBeCompared):

    previousHighestSimilarityValue = -1  # Initialise 'previous' Similarity value for the loop below

    for game in listOfGamesToBeCompared:

        if baseGameJSON[0]['id'] != game['id']:

            temporaryStorage = []

            textsSimilarityValue = tf_idf_textSimilarityValue(baseGameJSON, game)
            genreSimilarityValue = genreShareCount(baseGameJSON, game)
            gameModeSimilarityValue = gameModeShareCount(baseGameJSON, game)
            sameDeveloperSimilarityValue = doBothGameShareDevelopers(baseGameJSON, game)
            aggregatedRatingSimilarityValue = ratingValueSimilarity(baseGameJSON, game)

            temporaryStorage.append(textsSimilarityValue)
            temporaryStorage.append(genreSimilarityValue)
            temporaryStorage.append(gameModeSimilarityValue)
            temporaryStorage.append(sameDeveloperSimilarityValue)
            temporaryStorage.append(aggregatedRatingSimilarityValue)

            currentIterationSimilarityValue = finalSimilarityCalculation(temporaryStorage)

            if currentIterationSimilarityValue > previousHighestSimilarityValue:

                previousHighestSimilarityValue = currentIterationSimilarityValue
                currentRecommendedGame = game  # This game is currently the most recommended game

    ...

    return currentRecommendedGame
```

Figure 21 – Recommendation algorithm base loop

6.5.1 Game Description Similarity

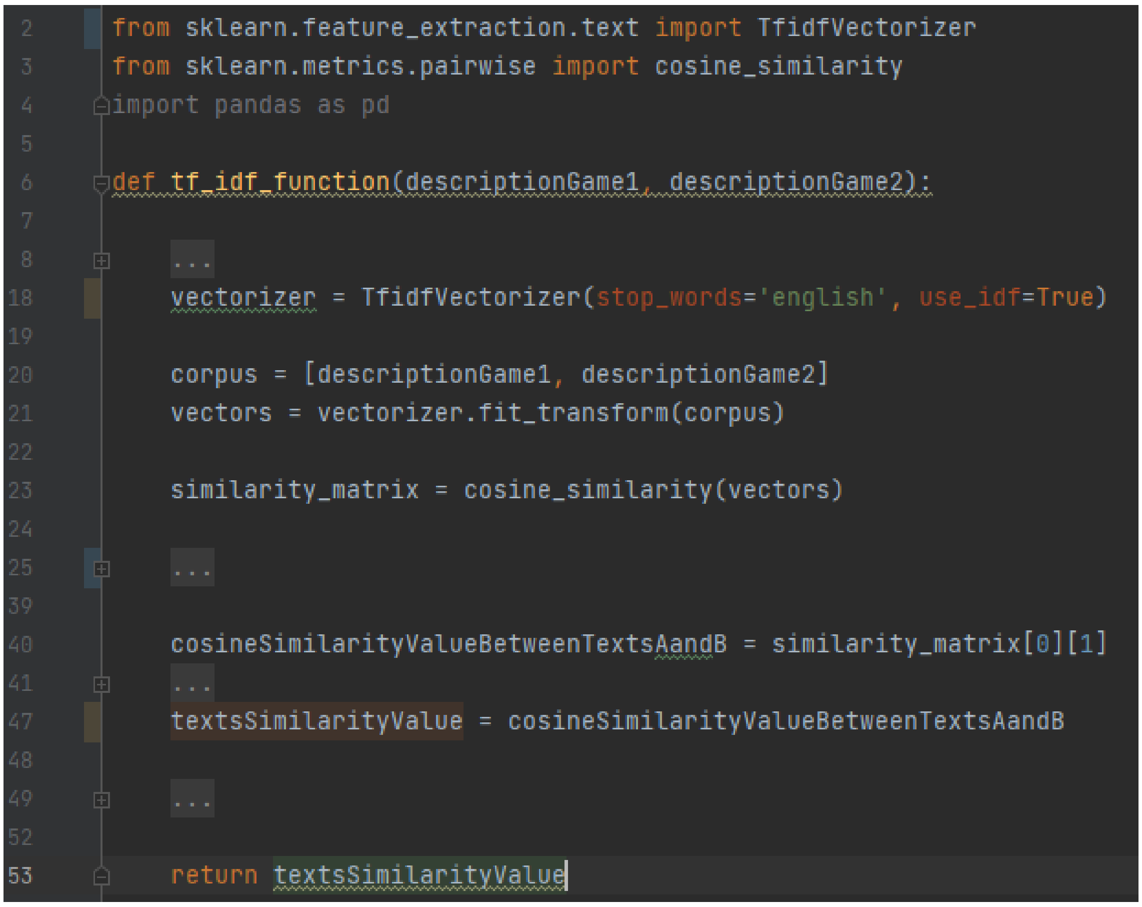
The function named “tf_idf_textSimilarityValue”, as seen in figure 22 below, is the first function to be called when the loop starts. The bulk of the function is in a try and except block. This is as there are games that may not have a description, if any game does not have a description, the “KeyError” exception is caught, then the value 0 is returned.



```python
def tf_idf_textSimilarityValue(baseGameJSON, gameItsBeingComparedToJSON):

    try:
        baseGameDescription = baseGameJSON[0]['summary']
        gameItsBeingComparedToDescription = gameItsBeingComparedToJSON['summary']

        textsSimilarityValue = theSite.tf_idf.tf_idf_function(baseGameDescription, gameItsBeingComparedToDescription)
        ...
    except KeyError:
        # As if 'summary' is not found, then there is no summary to compare against, therefore 0 for similarity
        return 0

    return textsSimilarityValue
```

Figure 22 – tf_idf_textSimilarity function

This function also calls the tf_idf_function, as seen below.



```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

def tf_idf_function(descriptionGame1, descriptionGame2):

    ...
    vectorizer = TfidfVectorizer(stop_words='english', use_idf=True)

    corpus = [descriptionGame1, descriptionGame2]
    vectors = vectorizer.fit_transform(corpus)

    similarity_matrix = cosine_similarity(vectors)

    ...

    cosineSimilarityValueBetweenTextsAandB = similarity_matrix[0][1]
    ...
    textsSimilarityValue = cosineSimilarityValueBetweenTextsAandB

    ...

    return textsSimilarityValue
```

Figure 23 – tf_idf_function code

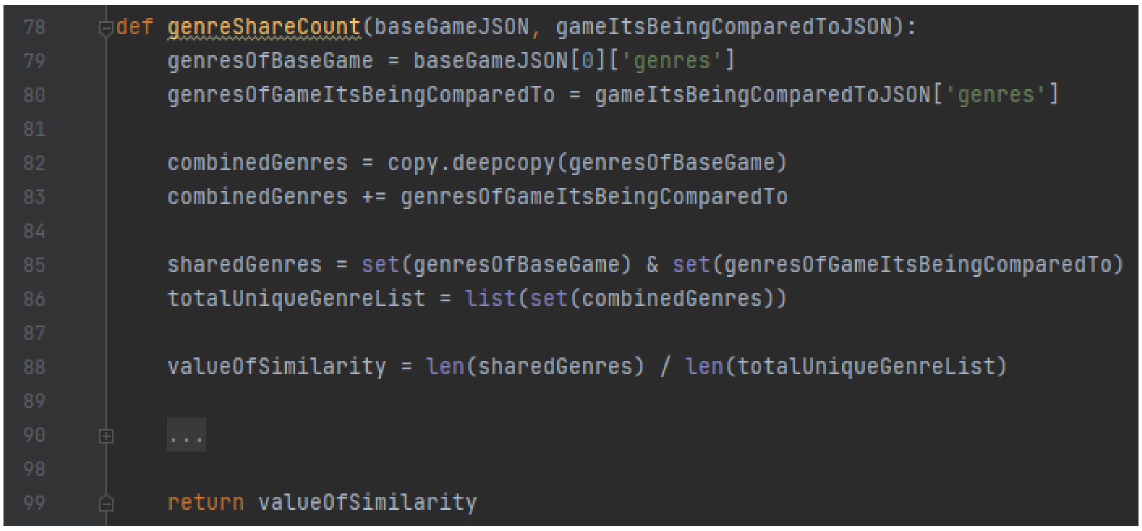
The majority of this function was already explained in detail in 6.1.2. What was omitted was the commentary on some of the variables as well the view of the entire function which is seen above.

The variable named "corpus" contains the descriptions, or the documents as they were referred to in 2.4.1.1.2 of the games.

Lastly, the similarity value was obtained from an array that has the structure seen at the bottom of figure 13 in 6.1.3, hence the referencing "[0][1]" as seen at the end of line 40 in figure 23.

6.5.2 Genres Shared

The second function to be called within the main recommendation algorithm loop is "genreShareCount". The function's body is presented below.



```python
def genreShareCount(baseGameJSON, gameItsBeingComparedToJSON):
    genresOfBaseGame = baseGameJSON[0]['genres']
    genresOfGameItsBeingComparedTo = gameItsBeingComparedToJSON['genres']

    combinedGenres = copy.deepcopy(genresOfBaseGame)
    combinedGenres += genresOfGameItsBeingComparedTo

    sharedGenres = set(genresOfBaseGame) & set(genresOfGameItsBeingComparedTo)
    totalUniqueGenreList = list(set(combinedGenres))

    valueOfSimilarity = len(sharedGenres) / len(totalUniqueGenreList)

    ...

    return valueOfSimilarity
```

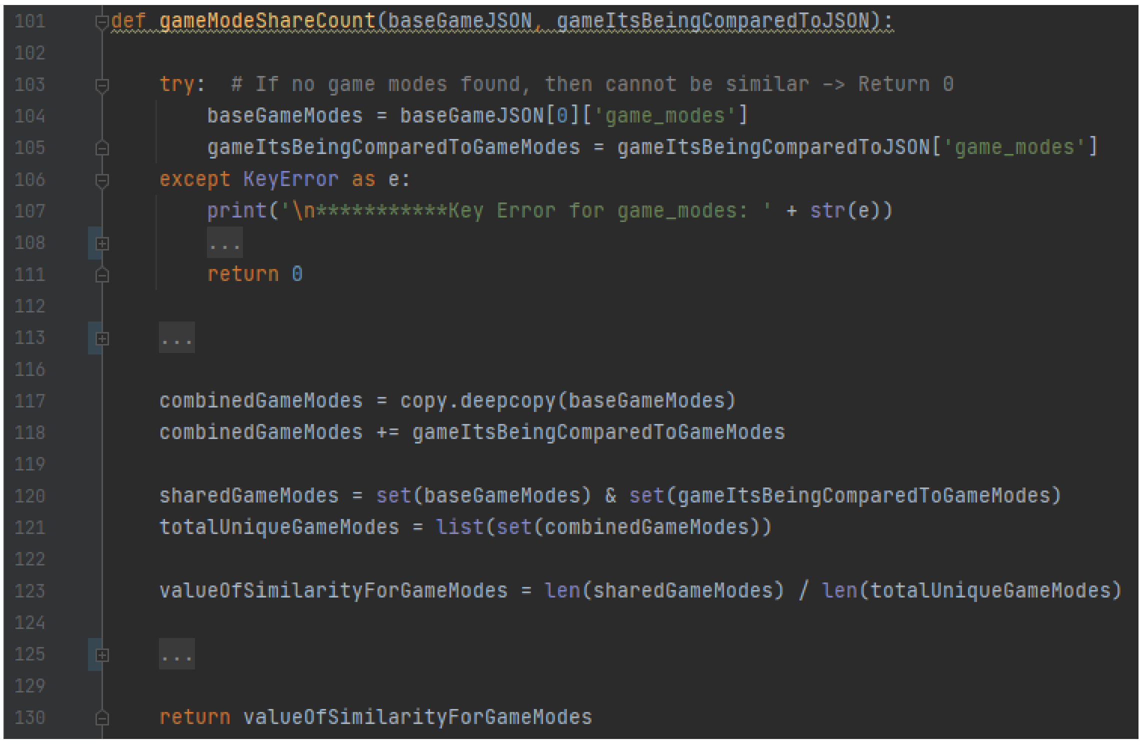
Figure 24 – genreShareCount function code

The function first gets the list of genres from each game's JSON files. It then gets the set of both game's genres in order to find the genres they share. The set of the combined genres is also taken, to get a singular representation of each genre that both games have. Finally, the size of the shared genres is divided by the size of the total amount of unique genres.

With an implementation like this, there is no need to know the contents of each list – so long as they are standardised. This is the case as all game data comes from IGDB.

6.5.3 Categories Shared (game modes)

The categories shared function is third to be called. Note that it is referred to as "GameModes" in the code. The code presented here shares much of the same implementation as 6.5.2 above, however has a try and except block initially.



```
101  def gameModeShareCount(baseGameJSON, gameItsBeingComparedToJSON):
102
103      try:  # If no game modes found, then cannot be similar -> Return 0
104          baseGameModes = baseGameJSON[0]['game_modes']
105          gameItsBeingComparedToGameModes = gameItsBeingComparedToJSON['game_modes']
106      except KeyError as e:
107          print('\n***********Key Error for game_modes: ' + str(e))
108          ...
111          return 0
112
113      ...
116
117      combinedGameModes = copy.deepcopy(baseGameModes)
118      combinedGameModes += gameItsBeingComparedToGameModes
119
120      sharedGameModes = set(baseGameModes) & set(gameItsBeingComparedToGameModes)
121      totalUniqueGameModes = list(set(combinedGameModes))
122
123      valueOfSimilarityForGameModes = len(sharedGameModes) / len(totalUniqueGameModes)
124
125      ...
129
130      return valueOfSimilarityForGameModes
```

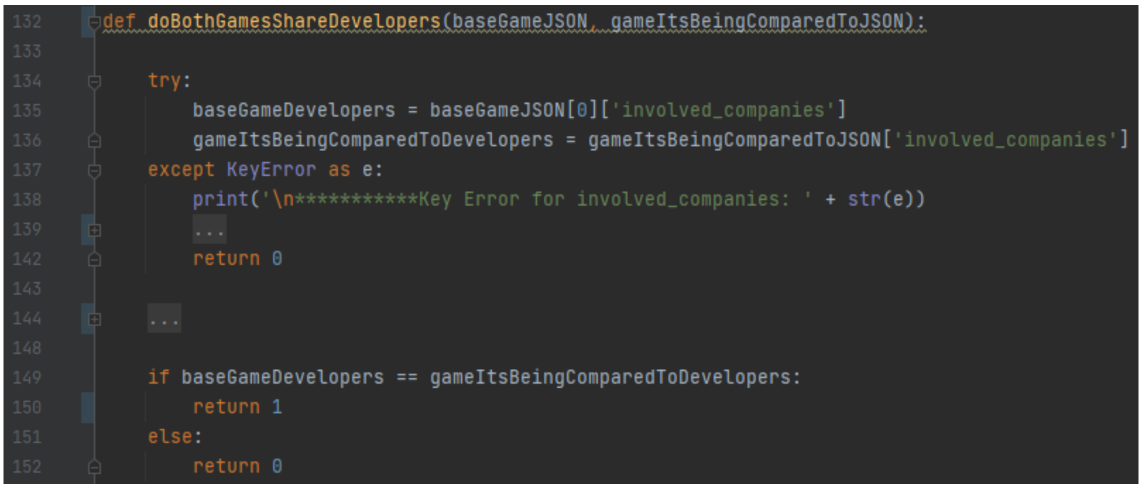
Figure 25 – gameModeShareCount function code

The try and except block is present as during testing there were multiple instances where games were missing game modes. These were games that were not complete with all the important information on IGDB. When a game mode is not found for either game, the value 0 is returned, as the game modes cannot be compared.

From line 117 onwards in figure 25, the code has an identical structure and explanation to that of 6.5.2, except it is in the context of game modes.

6.5.4 Developers Shared

Next is the developers shared function as the fourth function to be called. It is referred to as "doBothGamesShareDevelopers", as seen below.



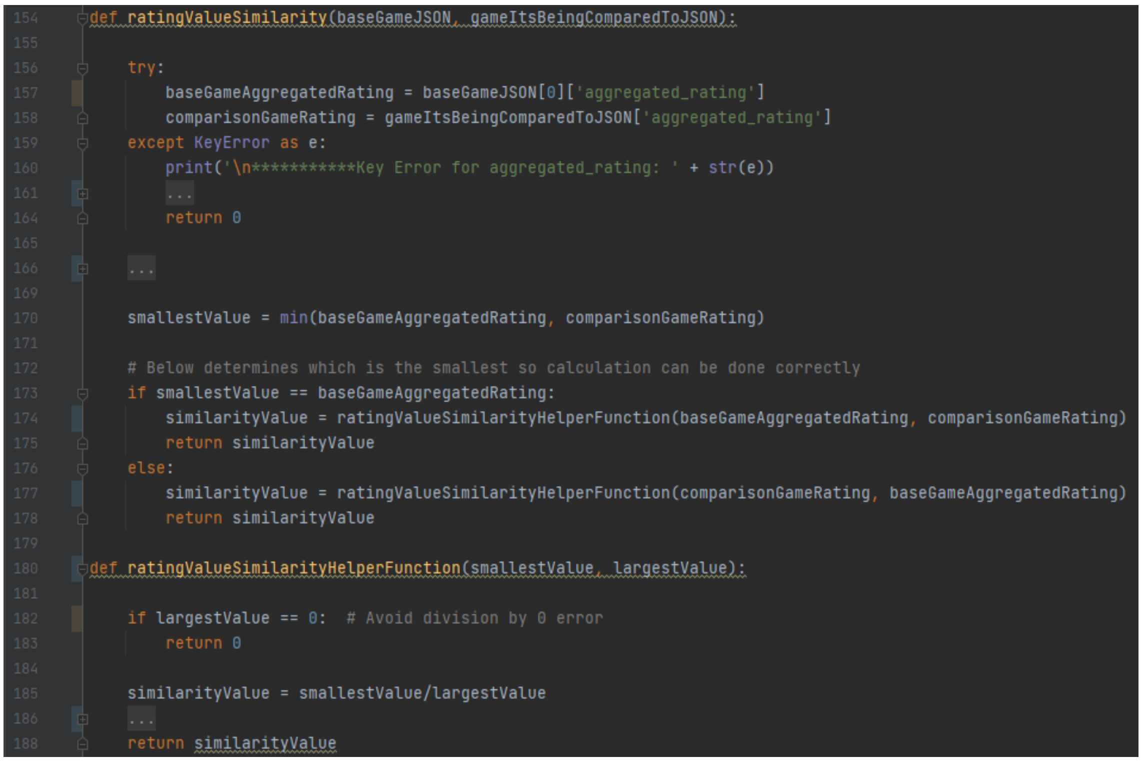
```
def doBothGamesShareDevelopers(baseGameJSON, gameItsBeingComparedToJSON):

    try:
        baseGameDevelopers = baseGameJSON[0]['involved_companies']
        gameItsBeingComparedToDevelopers = gameItsBeingComparedToJSON['involved_companies']
    except KeyError as e:
        print('\n***********Key Error for involved_companies: ' + str(e))
        ...
        return 0

    ...

    if baseGameDevelopers == gameItsBeingComparedToDevelopers:
        return 1
    else:
        return 0
```

Figure 26 – doBothGamesShareDevelopers function code

The implementation of this function is the simplest in relation to the others: if the developers are identical then return 1, otherwise a 0. The try and except block also makes an appearance here as it was surprising to see that so many entries within IGDB's database that did not contain this information.

6.5.5 External Critic Scores

The external critic score calculation function will calculate how close the two games currently being compared are in relation to their external critic scores.



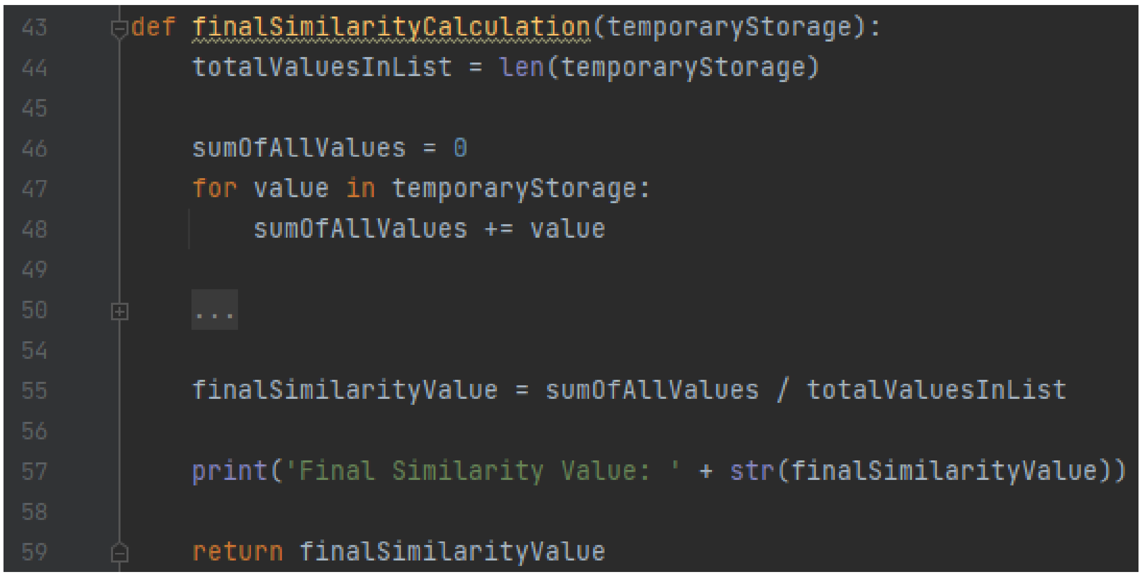
```
def ratingValueSimilarity(baseGameJSON, gameItsBeingComparedToJSON):

    try:
        baseGameAggregatedRating = baseGameJSON[0]['aggregated_rating']
        comparisonGameRating = gameItsBeingComparedToJSON['aggregated_rating']
    except KeyError as e:
        print('\n***********Key Error for aggregated_rating: ' + str(e))
        ...
        return 0

    ...

    smallestValue = min(baseGameAggregatedRating, comparisonGameRating)

    # Below determines which is the smallest so calculation can be done correctly
    if smallestValue == baseGameAggregatedRating:
        similarityValue = ratingValueSimilarityHelperFunction(baseGameAggregatedRating, comparisonGameRating)
        return similarityValue
    else:
        similarityValue = ratingValueSimilarityHelperFunction(comparisonGameRating, baseGameAggregatedRating)
        return similarityValue

def ratingValueSimilarityHelperFunction(smallestValue, largestValue):

    if largestValue == 0:  # Avoid division by 0 error
        return 0

    similarityValue = smallestValue/largestValue
    ...
    return similarityValue
```

Figure 27 – ratingValueSimilarity function with its helper function

The first task is finding the smallest rating value of the two, so that it can then be used as the numerator. The helper function is called to determine if the denominator is 0, in order to avoid a division error, and then subsequently return the final value.

6.5.6 Final similarity calculation

Finally, with all the values calculated for all the listed categories in 6.5, the similarity value is produced.



```python
def finalSimilarityCalculation(temporaryStorage):
    totalValuesInList = len(temporaryStorage)

    sumOfAllValues = 0
    for value in temporaryStorage:
        sumOfAllValues += value

    ...

    finalSimilarityValue = sumOfAllValues / totalValuesInList

    print('Final Similarity Value: ' + str(finalSimilarityValue))

    return finalSimilarityValue
```

Figure 28 – Final similarity calculation

As seen in lines 21 to 25 in figure 21, the values were added to a temporary array that serves the purpose of containing the values and a method of keeping track of the number of values inside of it. This is what allows to conveniently retrieve a sum of all the values and correctly divide by the total number of values using the “len()” function.

The role this function plays is seen in Figure 21, from lines 27 to 32. The game that has the highest score once the loop is finished executing, is the game that is returned as the recommended game.

6.6 Web Hosting

The project is hosted on a platform called Heroku (What is Heroku, 2021), under the URL “https://ugprojectsite.herokuapp.com/”.

Chapter 7: Testing

This section will cover the testing done for the project.

7.1 Manual Black-box testing

The first method of testing used will be manual black box testing. The purpose of carrying out this test is to ensure that the application can successfully execute with a list of random inputs. The following tables will show the input (as a string), the application output for the "Game Search" page, the result of the input for this page, the output for "Recommendation page" and the result of the input for the recommendation page. The results will consist of either a pass or fail.

Table 7 - Manual Blackbox testing results

Input	Game Search Output	Result	Recommendation Output	Result
"."	"Game entered not found"	PASS	"Game entered not found"	PASS
"%"		PASS		PASS
"limbo"	All APIs return expected details for game called 'Limbo'	PASS	All APIs return details about the recommended game "Forgotten Anne"	PASS
"%limbo"		PASS		PASS
".limbo"		PASS		PASS
"celeste"	All APIs return expected details for game called 'Celeste'	PASS	All APIs return details about the recommended game "Narita Boy"	PASS
"randomentry123"	"Game entered not found"	PASS	"Game entered not found"	PASS
"randomentry"		PASS		PASS
"random entry"		PASS		PASS

7.2 Sanity Testing

The second method used for testing the application is sanity testing. Carrying out this test will allow us to see how rational the recommendations received are. The testing is not carried out for the "game search" part of the application. This is because the results from table 7, shown under the column "Game Search Output", shows that we receive the correct game information for the game that was inputted. This is precisely what is expected and desired, so does not make sense to carry out sanity tests for “game search”.

To provide external data that confirms the rationality and accuracy of the recommended games produced from the algorithm, Steam’s game tags feature will be used (Steam Game Tags, 2021). This is where the most popular tags, selected by users, is shown. If they share at least six identical tags, then it will be enough to be considered a pass in the test.

Table 8 - Sanity tests results, with Steam's game tags

Input	Inputted game’s tags	Recommendation Output	Recommended game’s tags	Tags shared	Result
“Limbo”	Appendix 4, figure 33	“Forgotten Anne”	Appendix 4, figure 34	1. Puzzle Platformer 2. Indie 3. Puzzle 4. Platformer 5. Atmospheric 6. Adventure, 7. 2D 8. Singleplayer 9. Action 10. Casual	PASS
“Celeste”	Appendix 4, figure 35	“Narita Boy”	Appendix 4, figure 36	1. Action, Platformer 2. Adventure 3. 2D 4. Singleplayer 5. Pixel Graphics 6. 2D Platformer 7. Story Rich 8. Atmospheric 9. Retro 10. Exploration	PASS
“Risk of Rain 2”	Appendix 4, figure 37	“I Hate Running Backwards”	Appendix 4, figure 38	1. Co-op 2. Action Roguelike 3. Shooter 4. Roguelite 5. Difficult 6. Bullet Hell 7. Action 8. Singleplayer 9. Indie 10. Roguelike	PASS

"Resident Evil 5"	Appendix 4, figure 39	"Plants vs. Zombies: Battle for Neighborville™"	Appendix 4, figure 40	1. Action 2. Zombies 3. Online Co-Op 4. Singleplayer 5. Shooter 6. Third-Person Shooter 7. Co-op 8. Multiplayer	PASS

Chapter 8: Discussion

This chapter will provide my view on this project by talking about what went well, the challenges faced and what I would have done next.

8.1 Achievements

Overall, as the project stands, I am proud of what has been accomplished. There are some major improvements that can be made, as you will see near the end of this chapter, that would have made this into an excellent project. However, unfortunately I did not have the time.

The project meets the outlined aims (1.3), objectives (1.4), and functional requirements (4.2) that were set out in the earlier stages of the project. This means what was promised has been delivered.

This project has pushed me out of my comfort zone multiple times. As an individual with little confidence in web technologies, my goal was to become more rounded, and I believe I have achieved this. I am glad to have undertaken this project.

8.2 Challenges

The biggest challenge was a personal one. It was overcoming my fear when it came to implementing the big features. I have not done a project on my own that is as much as a third as the size of this one. I was concerned if I could not complete "this", and it did set me back at times. Nonetheless, I had no choice but to push through and I leave this project with something I am proud of.

As for the technical challenges, the greatest was understanding how to integrate some of the APIs. I have found that the APIs with poor documentation were the ones that I struggled a lot more with to integrate as I did not know how to properly format the headers or the bodies of the requests. For these, I had to rely on a lot of trial and error. Twitch's API was the most time consuming and technically difficult to implement for this reason.

8.3 Future Improvements

The most meaningful improvement for this project would be the use of a database. The data retrieved from the APIs could be reused when each recommendation request is made. This also means these games could then be used in future comparisons. This bypasses the issue where the application only gets a limited number of games from IGDB. In addition to this, it would be ideal to develop a user account system. This would be where users have an option to make an account, and all searches would be saved so they have access to a history of their recommended games. Figure 29 below shows what all of the above would look like.

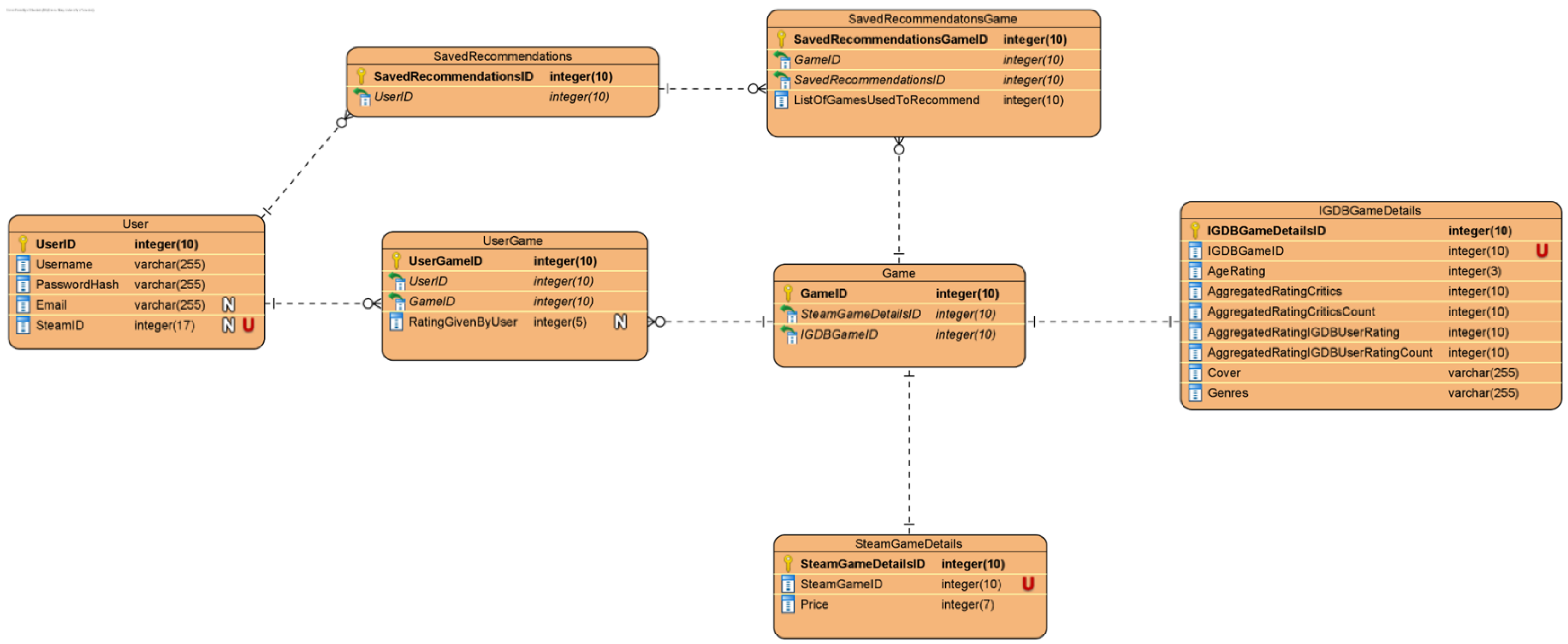

SavedRecommendations
SavedRecommendationsID integer(10)
UserID integer(10)
SavedRecommendatonsGame
SavedRecommendationsGameID integer(10)
GameID integer(10)
SavedRecommendationsID integer(10)
ListOfGamesUsedToRecommend integer(10)
User
UserID integer(10)
Username varchar(255)
PasswordHash varchar(255)
Email varchar(255) N
SteamID integer(17) N U
UserGame
UserGameID integer(10)
UserID integer(10)
GameID integer(10)
RatingGivenByUser integer(5) N
Game
GameID integer(10)
SteamGameDetailsID integer(10)
IGDBGameID integer(10)
IGDBGameDetails
IGDBGameDetailsID integer(10)
IGDBGameID integer(10) U
AgeRating integer(3)
AggregatedRatingCritics integer(10)
AggregatedRatingCriticsCount integer(10)
AggregatedRatingIGDBUserRating integer(10)
AggregatedRatingIGDBUserRatingCount integer(10)
Cover varchar(255)
Genres varchar(255)
SteamGameDetails
SteamGameDetailsID integer(10)
SteamGameID integer(10) U
Price integer(7)

Figure 29 - Entity Relationship Diagram for the layout of a future database

References

api-docs.igdb. 2021. IGDB API Rate Limits. [online] Available at: <https://api-docs.igdb.com/#rate-limits> [Accessed 25 April 2021].

Ayodele, T., 2010. Types Of Machine Learning Algorithms. [online] Pdfs.semanticscholar.org. Available at: <https://pdfs.semanticscholar.org/c4ae/802491724aee021f31f02327b9671cead3dc.pdf> [Accessed 22 November 2020].

Beck, et al., 2001. Manifesto for Agile Software Development [online] Available at: <https://agilemanifesto.org/principles.html> [Accessed 13 March 2021].

Boehm, B. and Hansen, W.J., 2000. Spiral development: Experience, principles, and refinements. CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST. Available at: < https://apps.dtic.mil/sti/pdfs/ADA382590.pdf> [Accessed 13 March 2021]

Brownlee, J., 2017. A Gentle Introduction to the Bag-of-Words Model. [online] machinelearningmastery. Available at: <https://machinelearningmastery.com/gentle-introduction-bag-words-model/> [Accessed 21 March 2021].

Blockgeni, T., 2019. Machine Learning: Creating a Similarity Measure. [online] Blockgeni. Available at: <https://blockgeni.com/machine-learning-creating-a-similarity-measure/> [Accessed 20 March 2021].

cogitotech. 2019. What are Features in Machine Learning and Why it is Important?. [online] Available at: <https://cogitotech.medium.com/what-are-features-in-machine-learning-and-why-it-is-important-e72f9905b54d> [Accessed 20 March 2021].

django. n.d. Why Django?. [online] Available at: <https://www.djangoproject.com/start/overview/> [Accessed 18 March 2021].

Franklin, S., 2020. Splitegate: Arena Warfare. [online] GamesFinder. Available at: <https://gameslikefinder.com/review/splitgate-arena-warfare/> [Accessed 16 March 2021].

Fumo, D., 2017. Types Of Machine Learning Algorithms You Should Know. [online] Towards Data Science. Available at: <https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861> [Accessed 22 November 2020].

Games Finder. n.d. Games Finder [online] Available at: <https://gameslikefinder.com/> [Accessed 16 March 2021].

Games Finder. n.d. Games Finder – About Us. [online] Available at: <https://gameslikefinder.com/about/> [Accessed 16 March 2021].

Games Finder. n.d. You searched from games like "Grand Theft Auto V". [online] Available at: <https://gameslikefinder.com/?s=Grand+Theft+Auto+V> [Accessed 16 March 2021].

GitHub. 2021. Web application frameworks. [online] Available at: <https://github.com/showcases/web-application-frameworks?s=stars> [Accessed 20 March 2021].

GitHub Docs. n.d. Saving repositories with stars. [online] Available at: <https://docs.github.com/en/github/getting-started-with-github/saving-repositories-with-stars> [Accessed 20 March 2021].

Google Console Help. 2020. Google Support - Developer Program Policy: September 16, 2020 announcement. [online] Available at: <https://support.google.com/googleplay/android-developer/answer/10065487?hl=en> [Accessed 25 April 2021].

Google Developers. 2020. Create a Manual Similarity Measure. [online] Available at: <https://developers.google.com/machine-learning/clustering/similarity/manual-similarity> [Accessed 20 March 2021].

Google Developers. 2021. Manual Similarity Measure Exercise. [online] Available at: <http://Manual Similarity Measure Exercise> [Accessed 20 March 2021].

Google Developers. 2020. Measuring Similarity from Embeddings. [online] Available at: <https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity> [Accessed 22 March 2021].

Google Developers. 2021. YouTube Data API Reference. [online] Available at: <https://developers.google.com/youtube/v3/docs> [Accessed 25 April 2021].

Google Support. 2021. Google Support - Verify your site ownership. [online] Available at: <https://support.google.com/webmasters/answer/9008080?hl=en&visit_id=637549672979677858-1633334593> [Accessed 25 April 2021].

Google Support. 2021. Verify your domain for Google Workspace. [online] Available at: <https://support.google.com/a/answer/60216?hl=en> [Accessed 25 April 2021].

Guchhait, S., 2018. Agile – An Umbrella view. [online] Sudeepta Guchhait. Available at: <https://deepgnosis.me/agile-an-umbrella-view/> [Accessed 13 March 2021].

Han, J., Kamber, M. and Pei, J., 2012. Cosine Similarity. [online] ScienceDirect. Available at: <https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=Cosine%20similarity%20measures%20the%20similarity,document%20similarity%20in%20text%20analysis.> [Accessed 22 March 2021].

Heroku. 2021. What is Heroku. [online] Available at: <https://www.heroku.com/what> [Accessed 27 April 2021].

IGDB. 2021. API: Video Game database on demand!. [online] Available at: <https://www.igdb.com/api> [Accessed 24 April 2021].

IGDB. 2021. What is IGDB.com?. [online] Available at: <https://www.igdb.com/about> [Accessed 24 April 2021].

Jain, K., 2015. Scikit-learn(sklearn) in Python – the most important Machine Learning tool I learnt last year!. [online] analyticsvidhya. Available at: <https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/> [Accessed 24 April 2021].

JetBrains. 2020. The State of Developer Ecosystem 2020. [online] Available at: <https://www.jetbrains.com/lp/devecosystem-2020/> [Accessed 18 March 2021].

JetBrains. 2020. Python. [online] Available at: <https://www.jetbrains.com/lp/devecosystem-2020/python/> [Accessed 18 March 2021].

JetBrains. 2020. Ruby. [online] Available at: <https://www.jetbrains.com/lp/devecosystem-2020/ruby/> [Accessed 19 March 2021].

Lavin, M., 2019. Analyzing Documents with TF-IDF. [online] The Programming Historian. Available at: <https://programminghistorian.org/en/lessons/analyzing-documents-with-tfidf> [Accessed 21 March 2021].

lvivity. 2021. Top Web Development Frameworks in 2021. [online] Available at: <http://Top Web Development Frameworks in 2021> [Accessed 20 March 2021].

MathWorks. n.d. Feature extraction for machine learning and deep learning. [online] Available at: <https://uk.mathworks.com/discovery/feature-extraction.html> [Accessed 21 March 2021].

Miller, J., 2009. Patterns in Practice - Convention Over Configuration. [online] Microsoft. Available at: <https://docs.microsoft.com/en-us/archive/msdn-magazine/2009/february/patterns-in-practice-convention-over-configuration> [Accessed 19 March 2021].

pandas. 2021. pandas. [online] Available at: <https://pandas.pydata.org/> [Accessed 24 April 2021].

pandas. 2021. pandas.DataFrame. [online] Available at: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html> [Accessed 24 April 2021].

Perez, S., 2019. Twitch acquires gaming database site IGDB to improve its search and discovery features. [online] techcrunch. Available at: <https://techcrunch.com/2019/09/17/twitch-acquires-gaming-database-site-igdb-to-improve-its-search-and-discovery-features/> [Accessed 24 April 2021].

Prescott, S., 2020. *The Most Popular Desktop Gaming Clients, Ranked*. [online] pcgamer. Available at: <https://www.pcgamer.com/uk/the-most-popular-desktop-gaming-clients-ranked/> [Accessed 22 November 2020].

Polamuri, S., 2015. FIVE MOST POPULAR SIMILARITY MEASURES IMPLEMENTATION IN PYTHON. [online] Dataasprint. Available at: <https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python/> [Accessed 22 March 2021].

pypi. 2020. Pandas 1.1.5. [online] Available at: <https://pypi.org/project/pandas/1.1.5/> [Accessed 24 April 2021].

pypi. 2020. requests 2.25.1. [online] Available at: <https://pypi.org/project/requests/2.25.1/> [Accessed 24 April 2021].

pypi. 2021. scikit-learn 0.24.1. [online] Available at: <https://pypi.org/project/scikit-learn/0.24.1/> [Accessed 24 April 2021].

Quantic Foundry. n.d. Video Game Recommendation Engine. [online] Available at: <https://apps.quanticfoundry.com/recommendations/gamerprofile/videogame/> [Accessed 15 March 2021].

RailsGuide. n.d. Getting Started with Rails. [online] Available at: <https://guides.rubyonrails.org/getting_started.html> [Accessed 19 March 2021].

Romano, S., 2021. Exclusives. [online] gematsu. Available at: <https://www.gematsu.com/exclusives> [Accessed 15 March 2021].

SimilarWeb. 2020. *Top Websites Ranking*. [online] Available at: <https://www.similarweb.com/top-websites/> [Accessed 22 November 2020].

Steam. 2018. Celeste. [online] Available at: <https://store.steampowered.com/app/504230/Celeste/> [Accessed 30 April 2021].

Steam. 2018. Forgotton Anne. [online] Available at: <https://store.steampowered.com/app/542050/Forgotton_Anne/> [Accessed 30 April 2021].

Steam. 2021. Narita Boy. [online] Available at: <https://store.steampowered.com/app/1069530/Narita_Boy/> [Accessed 30 April 2021].

Steam. 2021. I Hate Running Backwards. [online] Available at: <https://store.steampowered.com/app/575820/I_Hate_Running_Backwards/> [Accessed 30 April 2021].

Steam. n.d. Interactive Recommender. [online] Available at: <https://store.steampowered.com/recommender> [Accessed 15 March 2021].

Steam. 2011. LIMBO. [online] Available at: <https://store.steampowered.com/app/48000/LIMBO/> [Accessed 30 April 2021].

Steam. 2019. Plants vs. Zombies: Battle for Neighborville™. [online] Available at: <https://store.steampowered.com/app/1262240/Plants_vs_Zombies_Battle_for_Neighborville/> [Accessed 30 April 2021].

Steam. 2009. Resident Evil 5. [online] Available at: <https://store.steampowered.com/app/21690/Resident_Evil_5/> [Accessed 30 April 2021].

Steam. 2020. Risk of Rain 2. [online] Available at: <https://store.steampowered.com/app/632360/Risk_of_Rain_2/> [Accessed 30 April 2021].

Steam Game Tags. 2021. Steam Game Tags. [online] Available at: <https://store.steampowered.com/tag/> [Accessed 30 April 2021].

Stephan, B., 2020. *The Lockdown Live-Streaming Numbers Are Out, And They'Re Huge*. [online] The Verge. Available at: <https://www.theverge.com/2020/5/13/21257227/coronavirus-streamelements-arsenalgg-twitch-youtube-livestream-numbers> [Accessed 22 November 2020].

Sethi, N., n.d. Software Development Models in SDLC Process – Waterfall, Iterative, Spiral, V & Agile. [online] Electricalfundablog. Available at: <https://electricalfundablog.com/software-development-models/> [Accessed 13 March 2021].

Smith, S., n.d. Chapter 30. Don't Repeat Yourself. [online] O'Reilly. Available at: <https://www.oreilly.com/library/view/97-things-every/9780596809515/ch30.html> [Accessed 19 March 2021].

Vyas, A., 2020. Pros and Cons of Ruby on Rails for Web Development. [online] botreetechnologies. Available at: <https://www.botreetechnologies.com/blog/pros-and-cons-of-ruby-on-rails-for-web-development/> [Accessed 20 March 2021].

Wong, W., Tshai, K. and Lee, C., 2013. The Importance of a Software Development Methodology in IT Project Management: An Innovative Six Sigma Approach - A Case Study of a Malaysian SME Organization. University of Nottingham, Semenyih, Malaysia. Available at: <https://www.researchgate.net/publication/264623326_The_Importance_of_a_Software_Development_Methodology_in_IT_Project_Management_An_Innovative_Six_Sigma_Approach_-_A_Case_Study_of_a_Malaysian_SME_Organization> [Accessed 12 March 2021]

Zhao, W. and Chellappa, R., 2006. Cosine Similarity. [online] sciencedirect. Available at: <https://www.sciencedirect.com/topics/computer-science/cosine-similarity#:~:text=A%20cosine%20similarity%20measure%20is,doing%20nearest%20neighbor%3A%20(13)> [Accessed 22 March 2021].

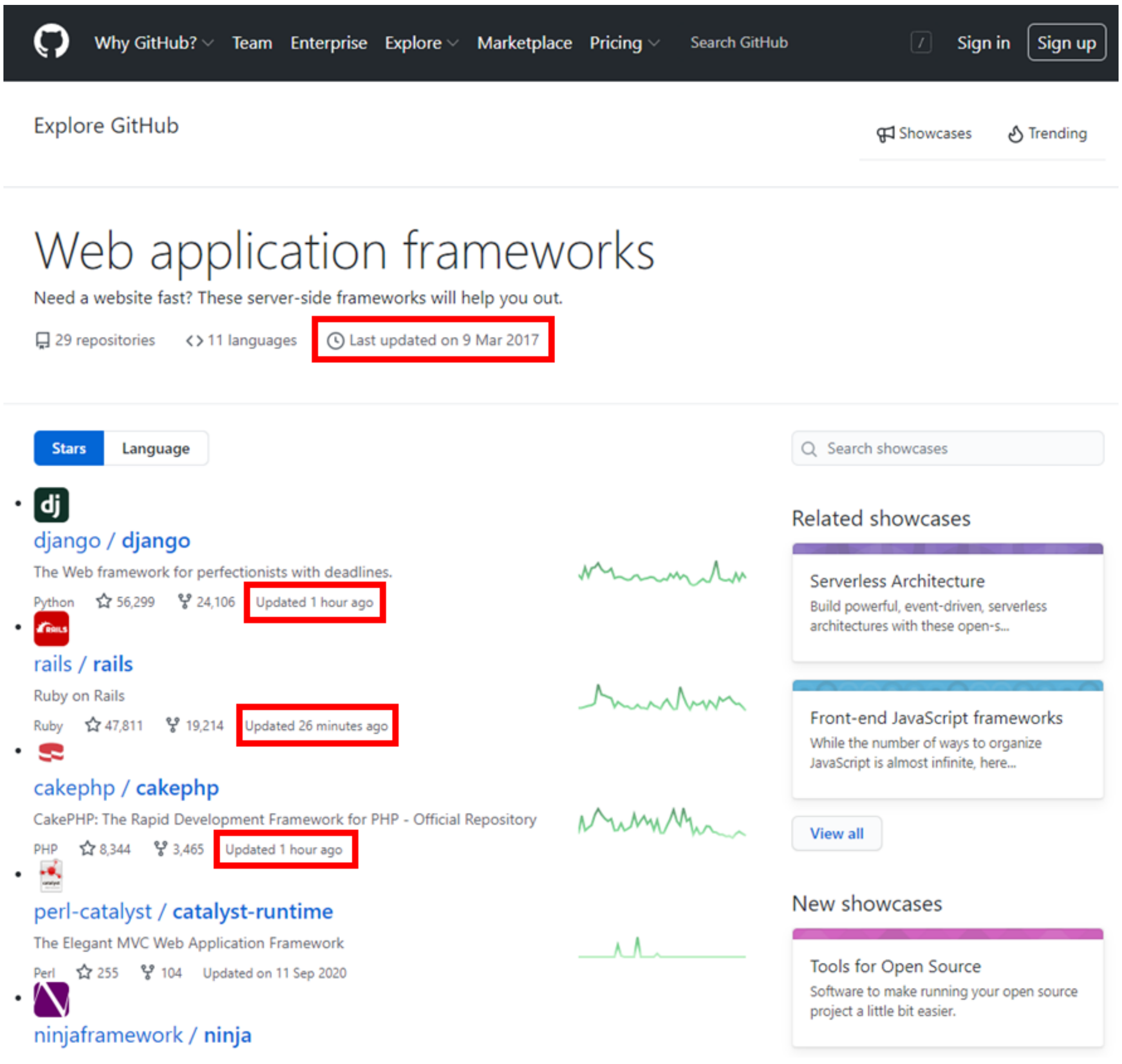
Appendix 1

Table 9 - Project plan

Phase	Task	Start date	End date	Duration
Initial research	Problem definition	30/10/20	1/11/20	3 days
	Investigate existing web applications	04/11/20	18/11/20	14 days
	Investigate and evaluate resources required to deliver the software.			
Analysis	Define technical objectives from research conducted in the initial research phase	19/11/20	27/11/20	8 days
	Define functional requirements			
Design	Convert technical objectives into components	20/01/21	04/02/21	15 days
	Design System context diagram			
	Design Sequence diagram			
Implementation	Develop front end	05/02/21	21/04/21	75 days
	Integrate APIs			
	Develop custom recommendation algorithm			
Testing	Conduct testing	22/04/21	29/04/21	7 days
Report				
Documentation	Interim	10/11/20	22/11/20	12 days
	Draft	05/03/21	21/03/21	16 days
	Final	01/04/21	01/05/21	30 days

Appendix 2

The confusion may occur where the web page mentions that it has been last updated on "9 Mar 2017". As seen below in figure 30, there is an inconsistency between what has been updated on the web page and the date GitHub provides as "Last updated". The source has been dated with 2021, instead of 2017. This as a result of an inquiry with GitHub ,"the 'Last updated' date is the last time the curated collection of repositories was edited by us.". This email is produced after figure 30.



Why GitHub? Team Enterprise Explore Marketplace Pricing Search GitHub Sign in Sign up

Explore GitHub

Showcases Trending

Web application frameworks

Need a website fast? These server-side frameworks will help you out.

29 repositories 11 languages Last updated on 9 Mar 2017

Stars Language

Search showcases

- django / django
  The Web framework for perfectionists with deadlines.
  Python 56,299 24,106 Updated 1 hour ago
- rails / rails
  Ruby on Rails
  Ruby 47,811 19,214 Updated 26 minutes ago
- cakephp / cakephp
  CakePHP: The Rapid Development Framework for PHP - Official Repository
  PHP 8,344 3,465 Updated 1 hour ago
- perl-catalyst / catalyst-runtime
  The Elegant MVC Web Application Framework
  Perl 255 104 Updated on 11 Sep 2020
- ninjaframework / ninja

Related showcases

Serverless Architecture
Build powerful, event-driven, serverless architectures with these open-s...

Front-end JavaScript frameworks
While the number of ways to organize JavaScript is almost infinite, here...

View all

New showcases

Tools for Open Source
Software to make running your open source project a little bit easier.

Figure 30 – web application frameworks "last updated" clarification (Web application frameworks, 2021).

The official response received from GitHub Support is outlined below.

"Hi <redacted>,

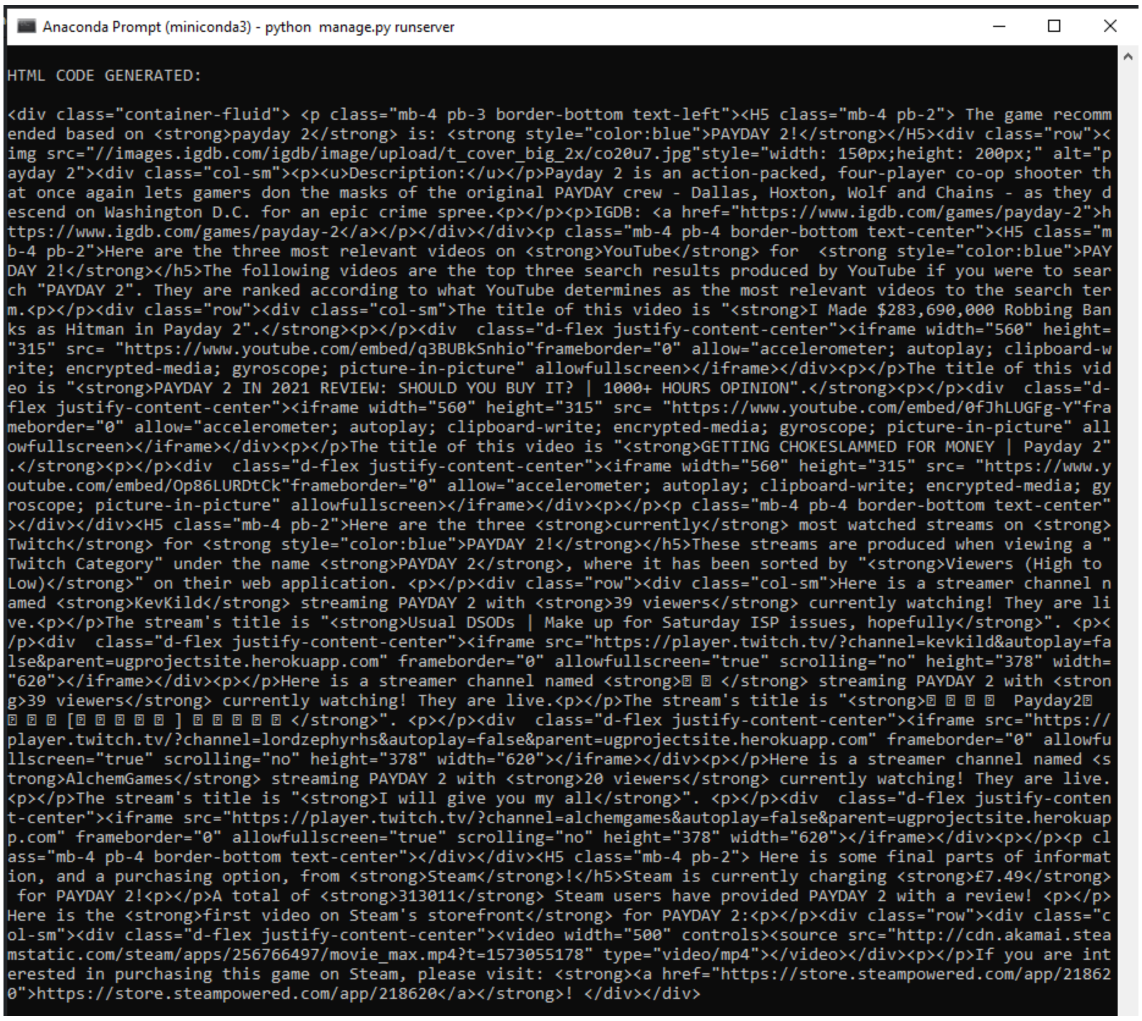
Thanks for reaching out! You are correct, the "Last updated" date is the last time the curated collection of repositories was edited by us. It is definitely expected that the language repositories themselves would be quite recently updated.

I hope this helps! Please let us know if you have further questions and we will be happy to assist.

Thanks,

<redacted>
GitHub Support”

Appendix 3

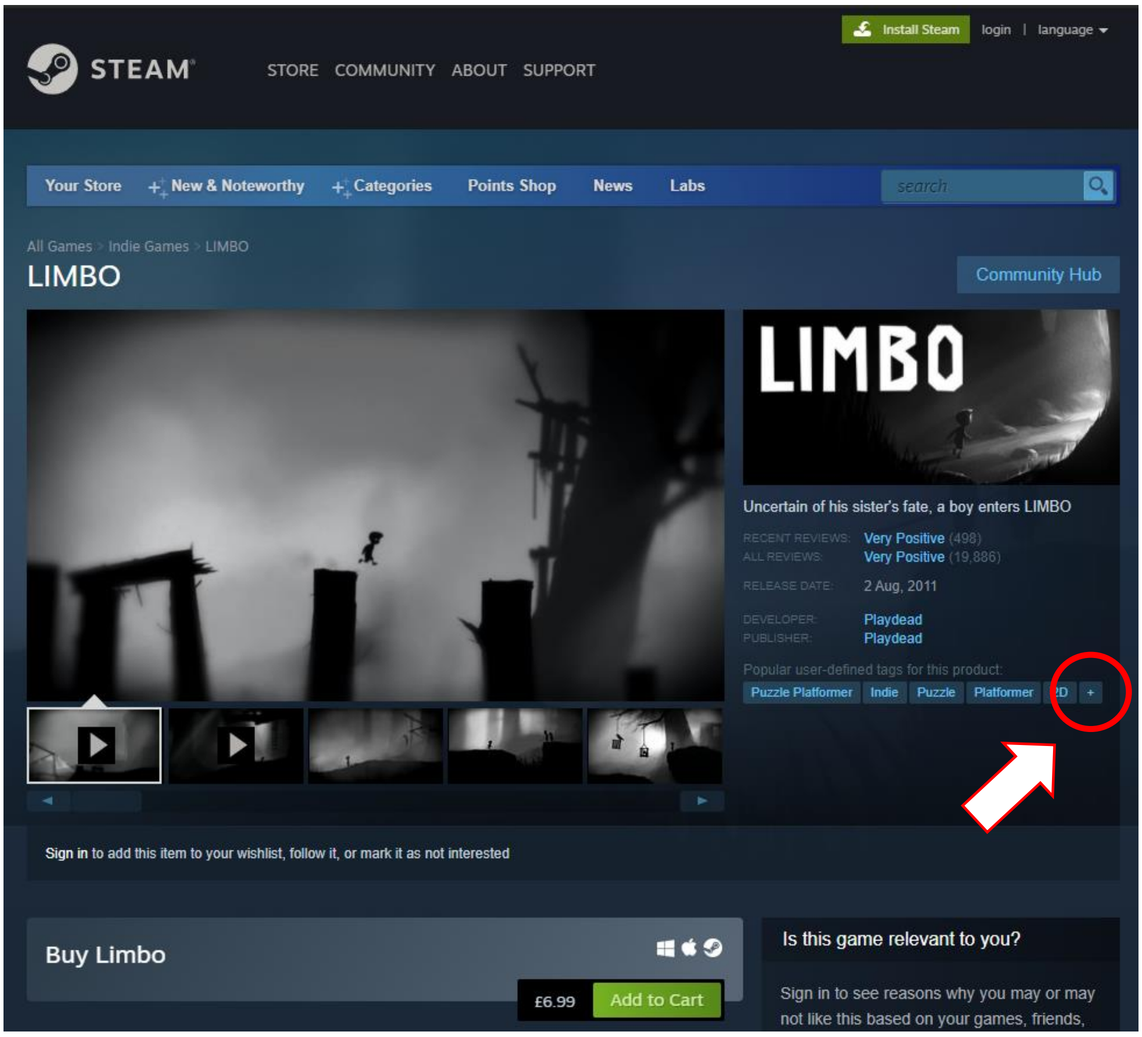


```
HTML CODE GENERATED:

<div class="container-fluid"> <p class="mb-4 pb-3 border-bottom text-left"><H5 class="mb-4 pb-2"> The game recomm
ended based on <strong>payday 2</strong> is: <strong style="color:blue">PAYDAY 2!</strong></H5><div class="row"><
img src="//images.igdb.com/igdb/image/upload/t_cover_big_2x/co20u7.jpg"style="width: 150px;height: 200px;" alt="p
ayday 2"><div class="col-sm"><p><u>Description:</u></p>Payday 2 is an action-packed, four-player co-op shooter th
at once again lets gamers don the masks of the original PAYDAY crew - Dallas, Hoxton, Wolf and Chains - as they d
escend on Washington D.C. for an epic crime spree.<p></p><p>IGDB: <a href="https://www.igdb.com/games/payday-2">h
ttps://www.igdb.com/games/payday-2</a></p></div></div><p class="mb-4 pb-4 border-bottom text-center"><H5 class="m
b-4 pb-2">Here are the three most relevant videos on <strong>YouTube</strong> for  <strong style="color:blue">PAY
DAY 2!</strong></h5>The following videos are the top three search results produced by YouTube if you were to sear
ch "PAYDAY 2". They are ranked according to what YouTube determines as the most relevant videos to the search ter
m.<p></p><div class="row"><div class="col-sm">The title of this video is "<strong>I Made $283,690,000 Robbing Ban
ks as Hitman in Payday 2".</strong><p></p><div  class="d-flex justify-content-center"><iframe width="560" height=
"315" src= "https://www.youtube.com/embed/q3BUBkSnhio"frameborder="0" allow="accelerometer; autoplay; clipboard-w
rite; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe></div><p></p>The title of this vid
eo is "<strong>PAYDAY 2 IN 2021 REVIEW: SHOULD YOU BUY IT? | 1000+ HOURS OPINION".</strong><p></p><div  class="d-
flex justify-content-center"><iframe width="560" height="315" src= "https://www.youtube.com/embed/0fJhLUGFg-Y"fra
meborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" all
owfullscreen></iframe></div><p></p>The title of this video is "<strong>GETTING CHOKESLAMMED FOR MONEY | Payday 2"
.</strong><p></p><div  class="d-flex justify-content-center"><iframe width="560" height="315" src= "https://www.y
outube.com/embed/Op86LURDtCk"frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gy
roscope; picture-in-picture" allowfullscreen></iframe></div><p></p><p class="mb-4 pb-4 border-bottom text-center"
></div></div><H5 class="mb-4 pb-2">Here are the three <strong>currently</strong> most watched streams on <strong>
Twitch</strong> for <strong style="color:blue">PAYDAY 2!</strong></h5>These streams are produced when viewing a "
Twitch Category" under the name <strong>PAYDAY 2</strong>, where it has been sorted by "<strong>Viewers (High to
Low)</strong>" on their web application. <p></p><div class="row"><div class="col-sm">Here is a streamer channel n
amed <strong>KevKild</strong> streaming PAYDAY 2 with <strong>39 viewers</strong> currently watching! They are li
ve.<p></p>The stream's title is "<strong>Usual DSODs | Make up for Saturday ISP issues, hopefully</strong>". <p><
/p><div  class="d-flex justify-content-center"><iframe src="https://player.twitch.tv/?channel=kevkild&autoplay=fa
lse&parent=ugprojectsite.herokuapp.com" frameborder="0" allowfullscreen="true" scrolling="no" height="378" width=
"620"></iframe></div><p></p>Here is a streamer channel named <strong>� � </strong> streaming PAYDAY 2 with <stron
g>39 viewers</strong> currently watching! They are live.<p></p>The stream's title is "<strong>� � � �  Payday2�
� � � [� � � � � � ] � � � � � � </strong>". <p></p><div  class="d-flex justify-content-center"><iframe src="https://
player.twitch.tv/?channel=lordzephyrhs&autoplay=false&parent=ugprojectsite.herokuapp.com" frameborder="0" allowfu
llscreen="true" scrolling="no" height="378" width="620"></iframe></div><p></p>Here is a streamer channel named <s
trong>AlchemGames</strong> streaming PAYDAY 2 with <strong>20 viewers</strong> currently watching! They are live.
<p></p>The stream's title is "<strong>I will give you my all</strong>". <p></p><div  class="d-flex justify-conten
t-center"><iframe src="https://player.twitch.tv/?channel=alchemgames&autoplay=false&parent=ugprojectsite.herokuap
p.com" frameborder="0" allowfullscreen="true" scrolling="no" height="378" width="620"></iframe></div><p></p><p cl
ass="mb-4 pb-4 border-bottom text-center"></div></div><H5 class="mb-4 pb-2"> Here is some final parts of informat
ion, and a purchasing option, from <strong>Steam</strong>!</h5>Steam is currently charging <strong>£7.49</strong>
 for PAYDAY 2!<p></p>A total of <strong>313011</strong> Steam users have provided PAYDAY 2 with a review! <p></p>
Here is the <strong>first video on Steam's storefront</strong> for PAYDAY 2:<p></p><div class="row"><div class="c
ol-sm"><div class="d-flex justify-content-center"><video width="500" controls><source src="http://cdn.akamai.stea
mstatic.com/steam/apps/256766497/movie_max.mp4?t=1573055178" type="video/mp4"></video></div><p></p>If you are int
erested in purchasing this game on Steam, please visit: <strong><a href="https://store.steampowered.com/app/21862
0">https://store.steampowered.com/app/218620</a></strong>! </div></div>
```

Figure 31 - HTML and CSS code sent to front end with AJAX

Appendix 4

If you would like to investigate the tags, please visit see the reference within a figure's caption and then click the following button seen below in figure 32 on each games Steam's storefront page.



Uncertain of his sister's fate, a boy enters LIMBO

RECENT REVIEWS: Very Positive (498)
ALL REVIEWS: Very Positive (19,886)
RELEASE DATE: 2 Aug, 2011
DEVELOPER: Playdead
PUBLISHER: Playdead

Popular user-defined tags for this product:

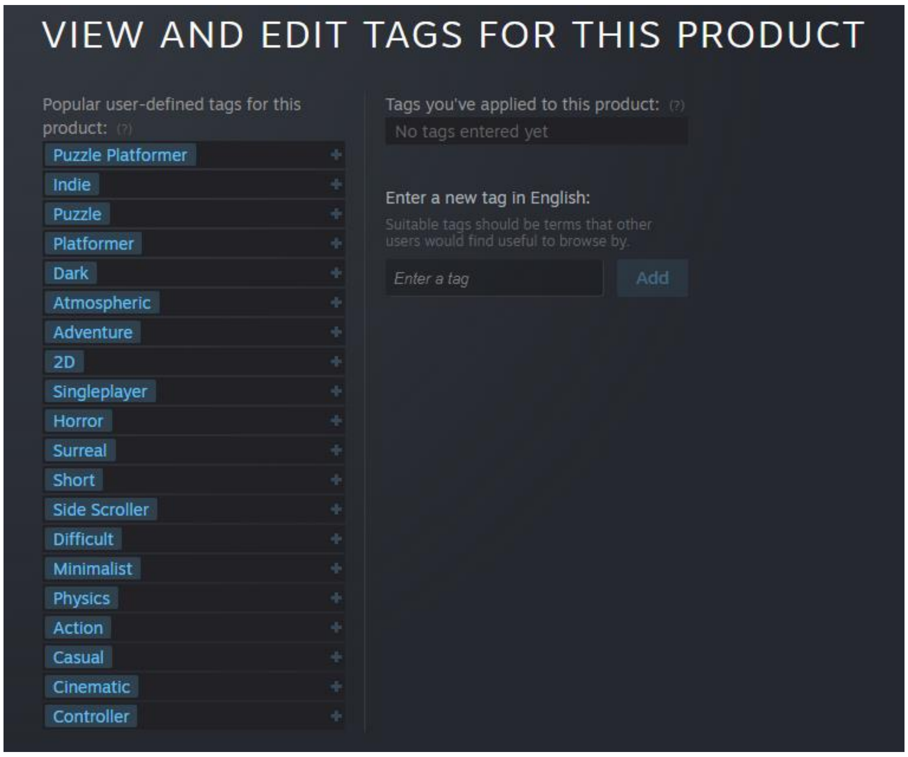
Sign in to add this item to your wishlist, follow it, or mark it as not interested

Buy Limbo

Is this game relevant to you?

Sign in to see reasons why you may or may not like this based on your games, friends,

Figure 32 – Where to find "all tags" (LIMBO, 2011)


VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

- Puzzle Platformer
- Indie
- Puzzle
- Platformer
- Dark
- Atmospheric
- Adventure
- 2D
- Singleplayer
- Horror
- Surreal
- Short
- Side Scroller
- Difficult
- Minimalist
- Physics
- Action
- Casual
- Cinematic
- Controller

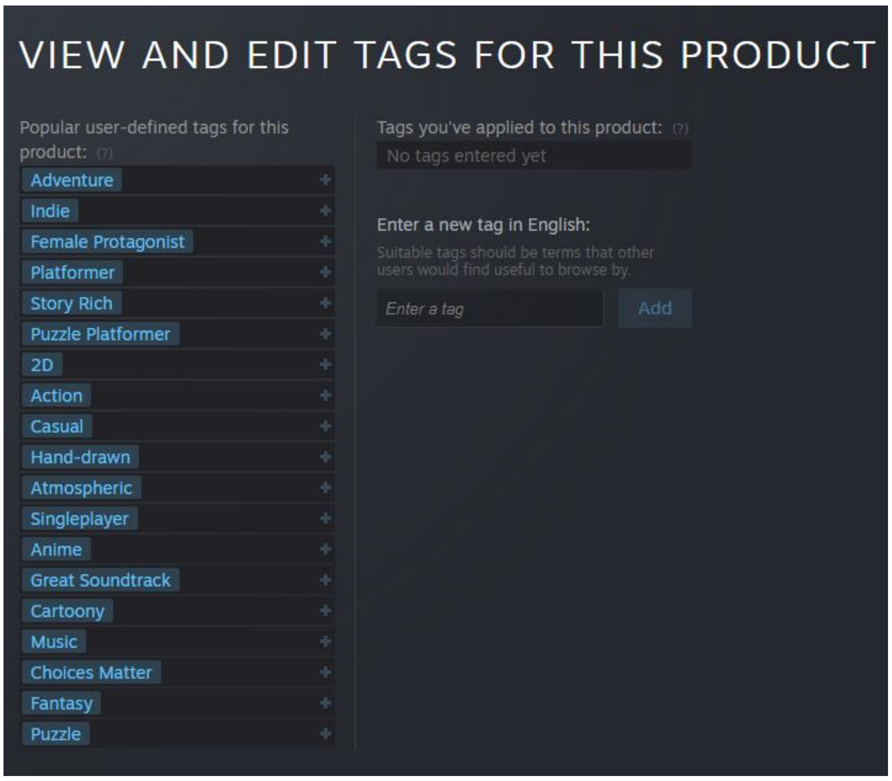
Tags you've applied to this product: (?)

No tags entered yet

Enter a new tag in English:

Suitable tags should be terms that other users would find useful to browse by.

Figure 33 – "Limbo" tags (LIMBO, 2011)


VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

- Adventure
- Indie
- Female Protagonist
- Platformer
- Story Rich
- Puzzle Platformer
- 2D
- Action
- Casual
- Hand-drawn
- Atmospheric
- Singleplayer
- Anime
- Great Soundtrack
- Cartoony
- Music
- Choices Matter
- Fantasy
- Puzzle

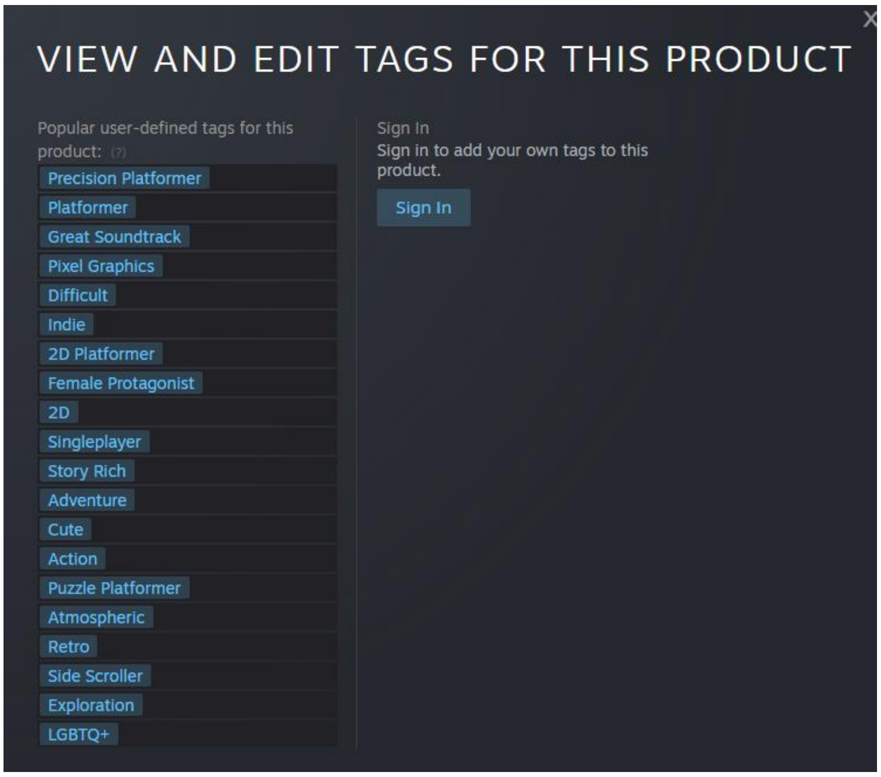
Tags you've applied to this product: (?)

No tags entered yet

Enter a new tag in English:

Suitable tags should be terms that other users would find useful to browse by.

Figure 34 - "Forgotten Anne" tags (Forgotton Anne, 2018)



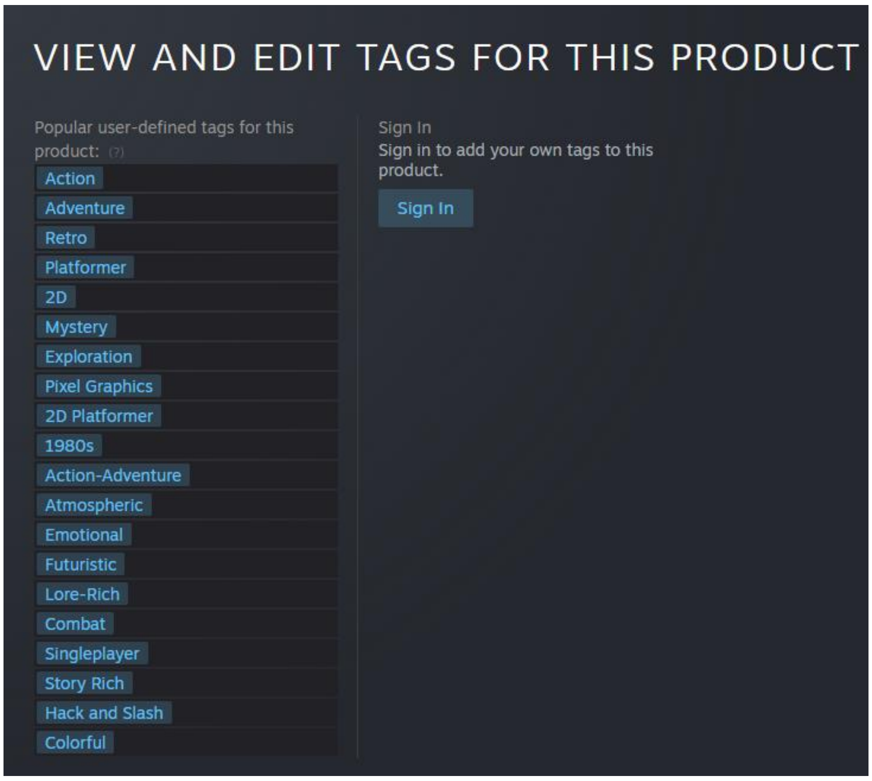
VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

Precision Platformer
Platformer
Great Soundtrack
Pixel Graphics
Difficult
Indie
2D Platformer
Female Protagonist
2D
Singleplayer
Story Rich
Adventure
Cute
Action
Puzzle Platformer
Atmospheric
Retro
Side Scroller
Exploration
LGBTQ+

Sign In
Sign in to add your own tags to this product.

Figure 35 – "Celeste" tags (Celeste, 2018)



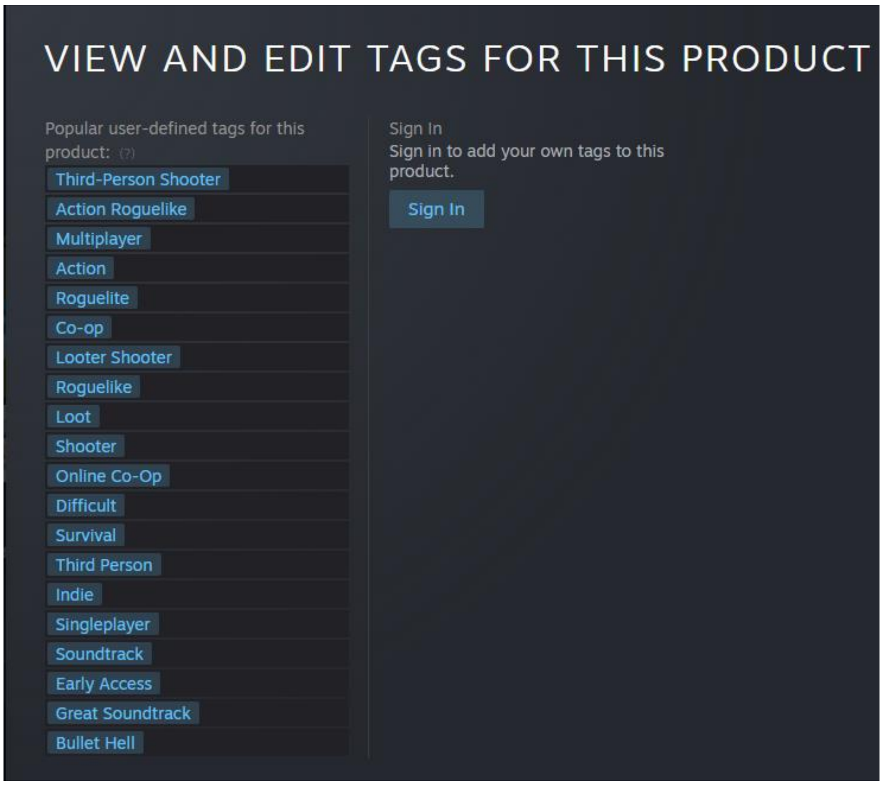
VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

Action
Adventure
Retro
Platformer
2D
Mystery
Exploration
Pixel Graphics
2D Platformer
1980s
Action-Adventure
Atmospheric
Emotional
Futuristic
Lore-Rich
Combat
Singleplayer
Story Rich
Hack and Slash
Colorful

Sign In
Sign in to add your own tags to this product.

Figure 36 – "Narita Boy" tags (Narita Boy, 2021)

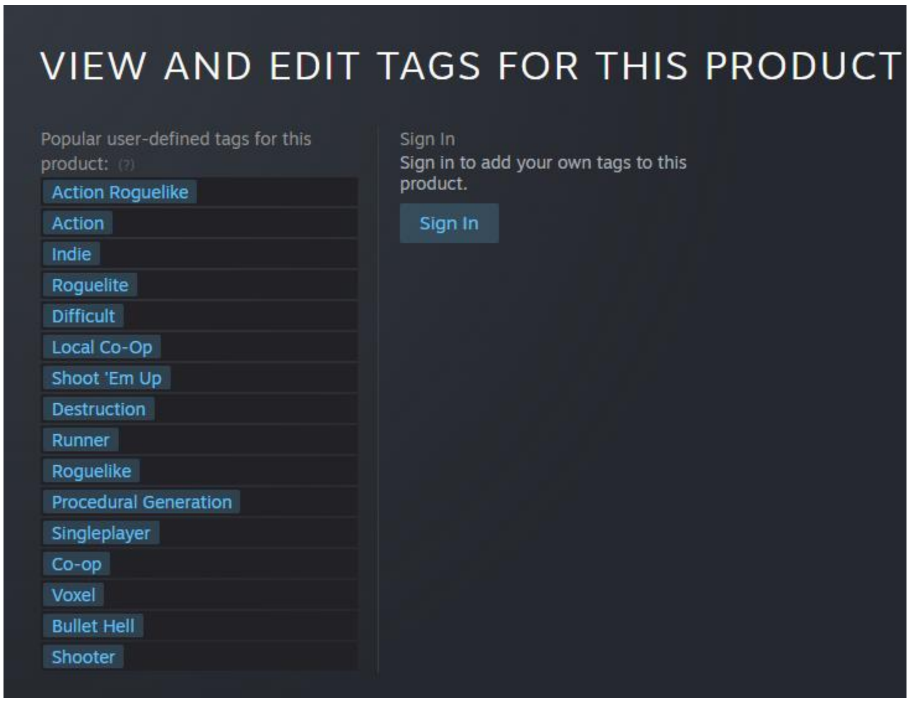

VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

- Third-Person Shooter
- Action Roguelike
- Multiplayer
- Action
- Roguelite
- Co-op
- Looter Shooter
- Roguelike
- Loot
- Shooter
- Online Co-Op
- Difficult
- Survival
- Third Person
- Indie
- Singleplayer
- Soundtrack
- Early Access
- Great Soundtrack
- Bullet Hell

Sign In
Sign in to add your own tags to this product.

Figure 37 – "Risk of rain 2" tags (Risk of Rain 2, 2020)

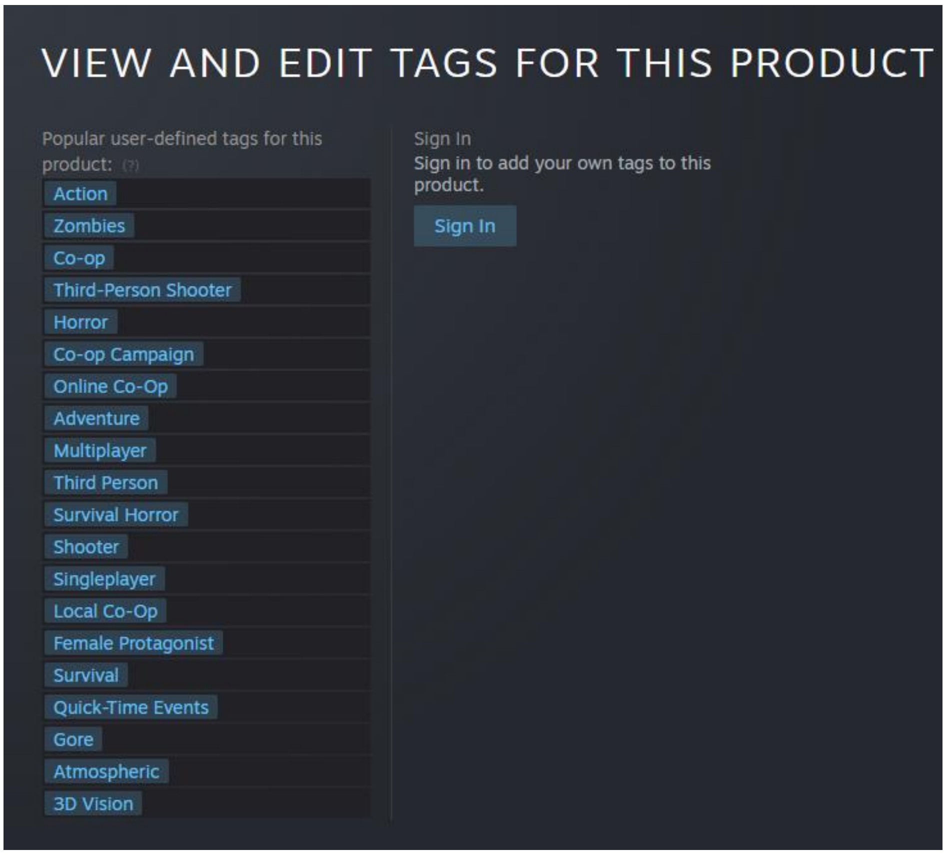

VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

- Action Roguelike
- Action
- Indie
- Roguelite
- Difficult
- Local Co-Op
- Shoot 'Em Up
- Destruction
- Runner
- Roguelike
- Procedural Generation
- Singleplayer
- Co-op
- Voxel
- Bullet Hell
- Shooter

Sign In
Sign in to add your own tags to this product.

Figure 38 – "I Hate Running Backwards" tags (I Hate Running Backwards, 2021)

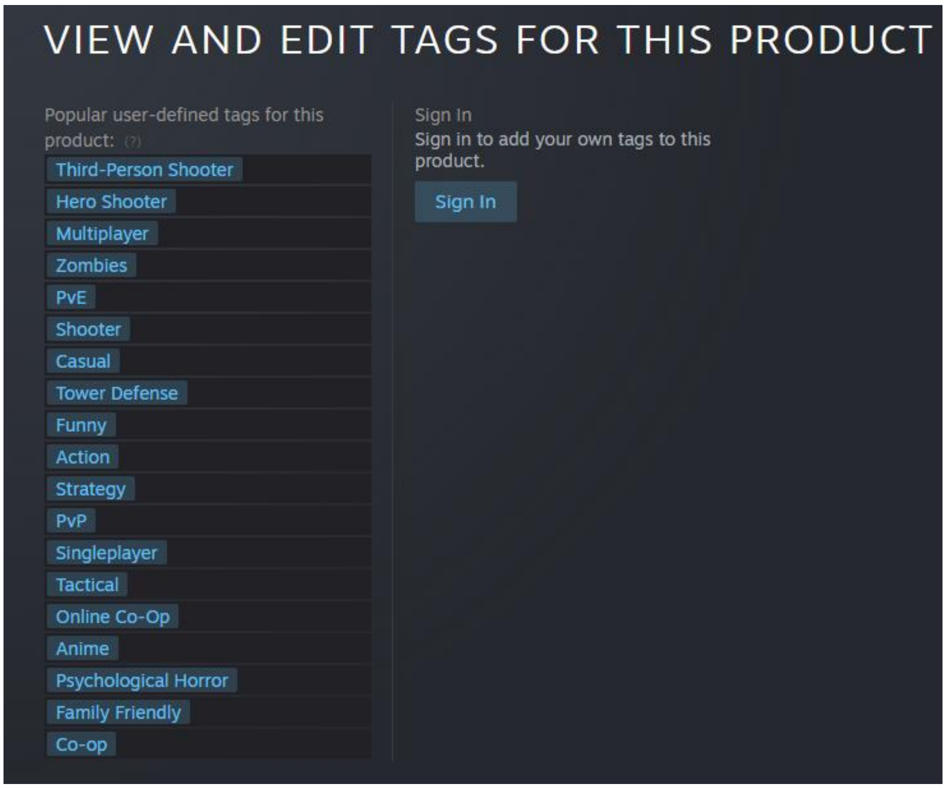

VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

Action
Zombies
Co-op
Third-Person Shooter
Horror
Co-op Campaign
Online Co-Op
Adventure
Multiplayer
Third Person
Survival Horror
Shooter
Singleplayer
Local Co-Op
Female Protagonist
Survival
Quick-Time Events
Gore
Atmospheric
3D Vision

Sign In
Sign in to add your own tags to this product.
Sign In

Figure 39 – "Resident Evil 5" tags (Resident Evil 5, 2009)


VIEW AND EDIT TAGS FOR THIS PRODUCT

Popular user-defined tags for this product: (?)

Third-Person Shooter
Hero Shooter
Multiplayer
Zombies
PvE
Shooter
Casual
Tower Defense
Funny
Action
Strategy
PvP
Singleplayer
Tactical
Online Co-Op
Anime
Psychological Horror
Family Friendly
Co-op

Sign In
Sign in to add your own tags to this product.
Sign In

Figure 40 – "Plants vs. Zombies: Battle for Neighborville™" tags (Plants vs. Zombies: Battle for Neighborville™, 2019)

Appendix 5

IGDB’s API

In order to first attain access to IGDB’s API (API: Video Game database on demand!, 2021), authorization was first required from Twitch. Once this was obtained, the API was used for the following purposes:

1. Get details about a game
    a. Using end point “https://api.igdb.com/v4/games”:
        i. ‘id’ – IGDB’s internal game ID
        ii. ‘name’ – Name as entered in IGDB’s database
        iii. ‘genres’ – Genres said game is associated with when concerning the queried game
        iv. ‘game_modes’ – Game modes (single player, multiplayer, etc.) the queried game has
        v. ‘involved_companies’ – Developers of queried game
        vi. ‘aggregated_rating’ – External critic score for queried game
        vii. ‘summary’ – Descriptions of queried game

    b. Using end point “https://api.igdb.com/v4/covers”:
        i. ‘url’ – URL for game cover of queried game

    c. Using end point “https://api.igdb.com/v4/external_games”:
        i. ‘uid’ – Steam’s ID of queried game

2. Receive a list of games
    a. Using end point “https://api.igdb.com/v4/games”:
        i. Additional call(s) for list for 500 (max limit) of games within a specific genre
            1. The call amount is dependent on how many genres the original inputted game on the front end has

Appendix 6

YouTube’s API

YouTube’s API (YouTube Data API Reference, 2021) was used in order to receive a list of the three most relevant videos in relation to the search term. Then, YouTube’s embed endpoint is used to get the links for the embedded variation of these videos onto the web application.

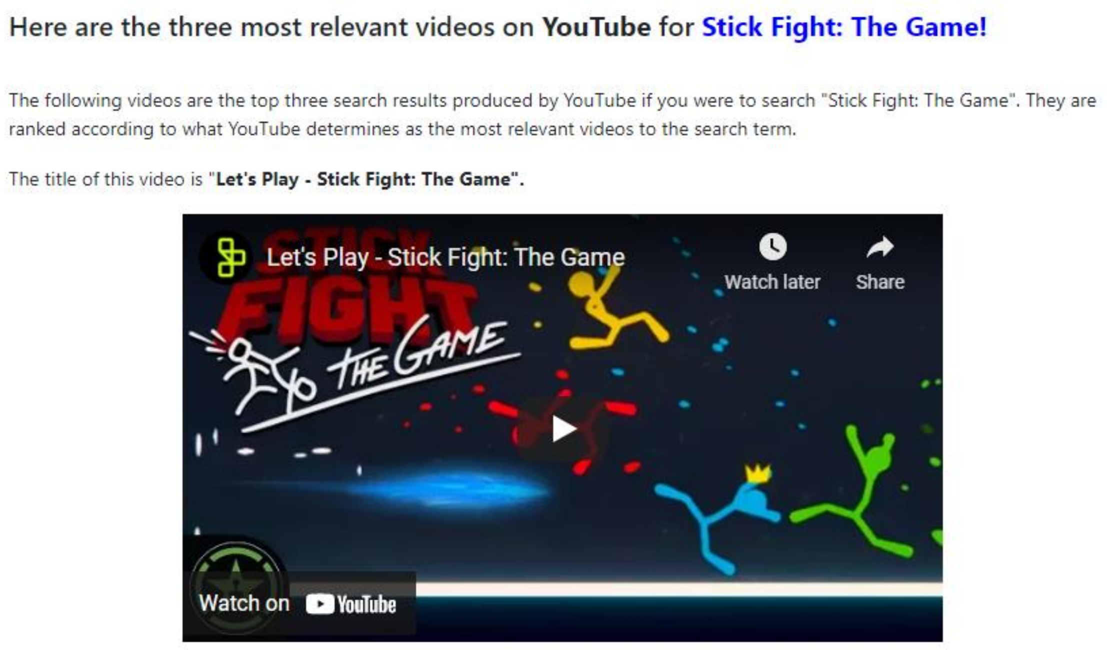
The use of this API requires Oauth authentication from Google, part of which requires the project to have a URL to a verification record that must be accessible at any time to Google. This is to not only verify the domain of the web application (Verify your domain for Google Workspace, 2021) but continue to verify site ownership (Google Support – Verify your site ownership, 2021), otherwise access to YouTube’s API will be revoked after at least a 15-day grace period (Google Support – Developer Program Policy: September 16, 2020 announcement, 2020).

The outline of the use of the API is as follows:

1. Receive data about most relevant videos for the game (using game name as the query)
    a. Using endpoint “https://www.googleapis.com/youtube/v3/search”, to attain:
        i. ‘snippet’, ‘title’ – YouTube video title
            1. Used to display video title on project front end
        ii. ‘id’, ‘videoId’ – Video ID on YouTube
            1. Used to then request the correct video via YouTube’s embed endpoint

2. Receive embed link to video resource using aforementioned video ID
    a. Using endpoint “https://www.youtube.com/embed/”:
        i. Link to video on YouTube as an embed

Additionally, the search/query term is always the outputted game name. This is provided in blue, seen in figure 41 below, so that the user will have the option to continue the search on YouTube with the correct search term. Doing so, will provide results that are within the same cluster of videos as the recommendation provided to them from the front end of the web application.



Here are the three most relevant videos on **YouTube** for **Stick Fight: The Game!**

The following videos are the top three search results produced by YouTube if you were to search "Stick Fight: The Game". They are ranked according to what YouTube determines as the most relevant videos to the search term.

The title of this video is "**Let's Play - Stick Fight: The Game".**

Figure 41 - Blue text denoting search term used to query YouTube's API

Appendix 7

Twitch's API

The integration with Twitch's API not only allowed for the access to the most popular streams for a given game, but also the authorisation required to access IGDB's API.

The layout of the use of the API is as follows:

1. Authorisation for access to Twitch and IGDB APIs
   a. Using endpoint "https://id.twitch.tv/oauth2/token":
      i. 'access_token' – to be used for any and every query to Twitch's and IGDB's API

2. Receive Twitch's internal game data
   a. Using endpoint "https://api.twitch.tv/helix/games":
      i. 'data', 'id' – Contains Twitch's internal ID for queried game

3. Receive list of streams using Twitch's internal game ID
   a. Using endpoint "https://api.twitch.tv/helix/streams":
      i. 'data' – returns list of top three streams based on viewer count
         1. For each iteration (max three iterations) in the JSON file get the following:
            a. 'user_login' – Actual username of the channel
            b. 'user_name' – Displayed username of the channel (a user can have different user login and username on Twitch)
            c. 'game_name' – Name of game they are streaming
            d. 'viewer_count'- Current number of viewers watch the stream live at the moment of the call
            e. 'type' – Type of playback. Can be live or video on demand
            f. 'title'- Title of the stream

Appendix 8

<u>**Steam's API**</u>

The use behind Steam's API was to find the Steam ID of a game, if it could not be found on IGDB's database, and to retrieve relevant information from their storefront.

The outline of use is as follows:

1. Receive list of all games in steam's database:
    a. Using endpoint "http://api.steampowered.com/ISteamApps/GetAppList/v0002/":
        i. 'applist', 'apps' – List of all steam games
            1. Find ID based on name of game

2. Receive steam storefront information about queried game:
    a. Using endpoint "https://store.steampowered.com/api/appdetails":
        i. 'is_free' – denotes if a game is free or not
        ii. 'release_date', 'coming_soon' – If a game has a release date and if it is coming soon
        iii. 'recommendations', 'total' – total amount of recommendations given for the game on Steam's platform
        iv. 'movies', [0], 'mp4', 'max' – First video on the store page for that game