COMP9727: Recommender Systems Project Design

Multi-Platform Video Game Recommendation - Z5247852 Daniel Lawson

1. Introduction

This document acts as the Project Design Document for Term 2 of *COMP9727: Recommender Systems*. The document covers the project design of Version 1.0 of the *Multi-Platform Video Game Recommender* System including both the technical and business requirements, as well as high-level design. This document has been developed as a hybrid of a Software Requirements Specification (SRS; Ref A), Software Design Document (SDD; Ref B) and the Project Design Assignment Specification. A project pitch was completed on 01/07/2025, with feedback incorporated into the document. The format of the document is as follows:

1.1. Structure of Document

1. **Introduction:** This section outlines the purpose of the document, structure of the document and table of contents.
2. **Project Scope**: This section outlines the purpose of the system, history of the problem, intended audience and usage, and business model. This is the first half of the assignment specifications scope section.
3. **System Scope**: This section outlines the system architecture and boundaries, and the proposed user interface for the system.
4. **Dataset Design**: This section outlines the project's requirements for data including the proposed datasets (for both data and users), why they were chosen and any underlying assumption and constraints for this data.
5. **Recommender Method Design**: This section outlines the requirements for the recommendation methods, the general outline of each method including the baseline and why were chosen. The section also outlines the use of hybrid methods and underlying assumptions and constraints.
6. **Evaluation**: This section outlines how the recommender will be evaluated from a qualitative and quantitative perspective. This includes the choice of metrics for both the model and system, and how the metrics are weighted for evaluating overall performance. The section also outlines how a proposed user study will be conducted.
7. **Project Timeline**: This section outlines a rough estimate of the project timeline including the deliverables required for the project and a list of key milestones and risks.
8. **References**: Contains any references used in this document.
9. **Glossary**: Contains a list of common acronyms and definitions.

1.2 Table of Contents

1.1.1 Table of Figures

1.1.2 Table of Tables

2. Project Scope

2.1. Purpose

The purpose of the *Multi-Platform Video Game Recommendation System* is to provide users recommendations across multiple video game ecosystems that answer the following questions:

1. *From my backlog of owned but unplayed games, what video game should I play next?*
2. *From the entire gaming marketplace, which new game should I buy next?*

These questions are answered using the players various game libraries from multiple ecosystems (Steam, Xbox and PlayStation) such that a recommendation for one platform should not include any games the user has **played** on another platform. Thus, this recommender serves the video game subdomain from the larger entertainment sector.

2.1.1. History of the Problem

The current video game market is dominated by major technology companies, with each console requiring the user have an account allowing users to purchase games digitally via a marketplace as well as tracking several aspects of the user's activity including:

- Each game the user has played including total playtime.
- What games the user has purchased.
- Any achievements the player has earned within each game.
- A list of the user's friends (or groups)
- Any downloadable content (DLC) the play has purchased for a game.

A summary of major gaming platforms is shown in Table 1 which shows the devices users of that platform can play games on, who owns the ecosystem and whether they include a personalised recommender within the ecosystem. Whilst other gaming platforms exist including *Epic Games*, *EA Play, Ubisoft Plus and GOG Galaxy,* these are often also playable through one of the other major platforms and often require an account within an account. As such these are out of scope for this Project. Whilst all the platforms have a dedicated marketplace, encouraging users to purchase new games (many of which are available on multiple platforms) through their own store, only Xbox and Steam have a recommender. Instead, PlayStation and Nintendo simply show users what is popular. The two sections below provide a brief overview of these two recommenders, which only know the user's activity within that ecosystem and will often recommend a game the user has played on another system. This is the primary gap that this project seeks to address.

Table 1: Summary of Major Platforms and Ecosystems

Platform	Company	Devices Useable	Recommender
Xbox	Microsoft	Xbox, PC (Windows) Handheld (ROG Ally)	Yes
Steam	Valve	PC (Windows or Linux) Handheld (Steam Deck, ROG Ally)	Yes
PlayStation	Sony	PlayStation PC (via Steam)	No
Switch / Switch 2	Nintendo	Switch / Switch 2	No

2.1.2. Existing Domain Recommenders

2.1.2.1. *Xbox Recommendation*

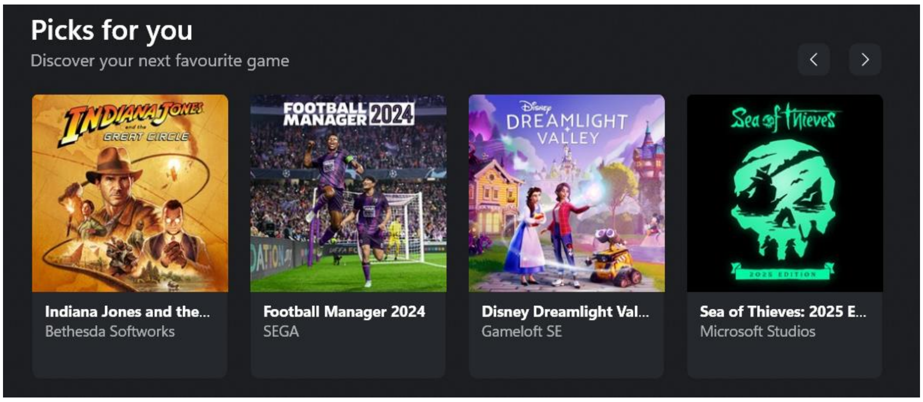


Figure 1: Xbox Game Pass Recommendations

Microsoft has developed a simple recommendation service as part of the Xbox Marketplace. Xbox is unique in that it has a presence on two platforms (the various Xbox Consoles, and Windows PC), as well as offering a subscription service known as *Xbox Game Pass*. Thus. Microsoft's goal is to get users to both try new Game Pass games (Figure 1) and to purchase games outright. This recommendation service is poor as there is no transparent way for the user to influence the service. Microsoft also offers a recommendation service within the Marketplace (Figure 2), however does not remove games the user has played (and finished) through the Subscription service.

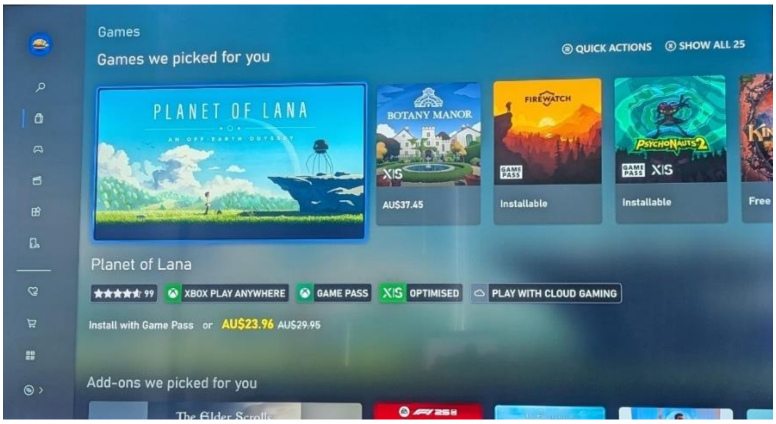


Figure 2: Xbox Store Recommendations

2.1.1.1. Steam Recommendation

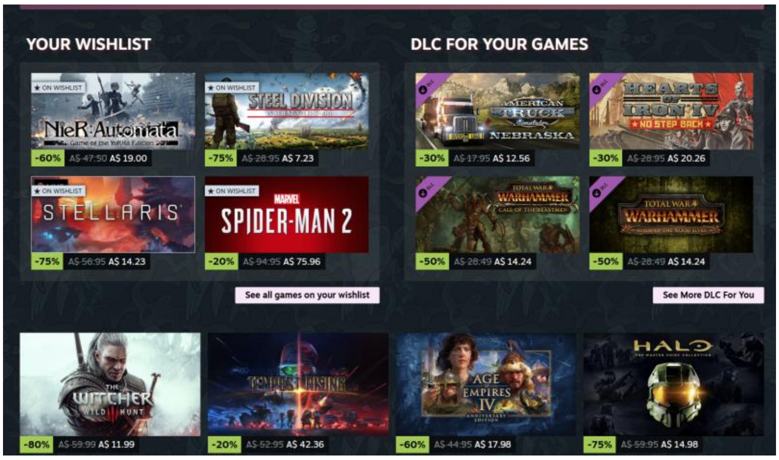
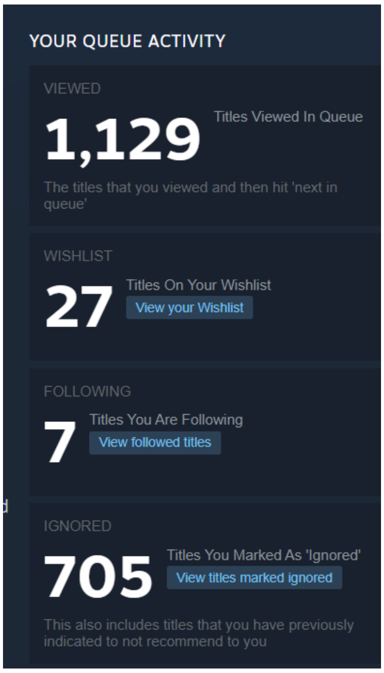


Figure 3: Steam Marketplace Front Page

Steam has a recommendation system known as the *Steam Discovery Queue.* As per Ref C, the queue, first released in 2014 is described as an alternative to the front page of the marketplace (Figure 3), aiming to be a "mix of products that are new, top-selling, and similar to what you play and use on Steam". Thus, the aim of the recommender is to get users to either buy new games or wish list it. The performance of the recommender has generally been mixed, with Ref D noting that the system was overhauled in 2022 due to poor user experience and often only used during major sales. Figure 4 reinforces this viewpoint with the authors Steam activity showing that only 27 titles out of 1,129 recommendations were added to the wish list.



YOUR QUEUE ACTIVITY

VIEWED

1,129 Titles Viewed In Queue

The titles that you viewed and then hit 'next in queue'

WISHLIST

27 Titles On Your Wishlist

View your Wishlist

FOLLOWING

7 Titles You Are Following

View followed titles

IGNORED

705 Titles You Marked As 'Ignored'

View titles marked ignored

This also includes titles that you have previously indicated to not recommend to you

Figure 4: Steam Discovery Queue Stats

2.2. Intended Audience

The intended userbase of this project are gamers who play and own games across multiple platforms. Whilst the initial use case comes from the project author who primarily uses both Xbox and Steam, the userbase could comprise of any combination of two or more supported systems. It is noted that users of only a single system could still benefit, especially if the system in question does not offer (or provides a poor) recommendation service. The major constraint put on the user of the system is they must have an account (with accessible profile data) for a system to be supported. This use case is not unreasonable as a study in 2023 (Ref W) showed that 47% of gamers in a study of 74,000 people used multiple platforms. Even excluding mobile games, this leaves 19% playing across both a console and PC.

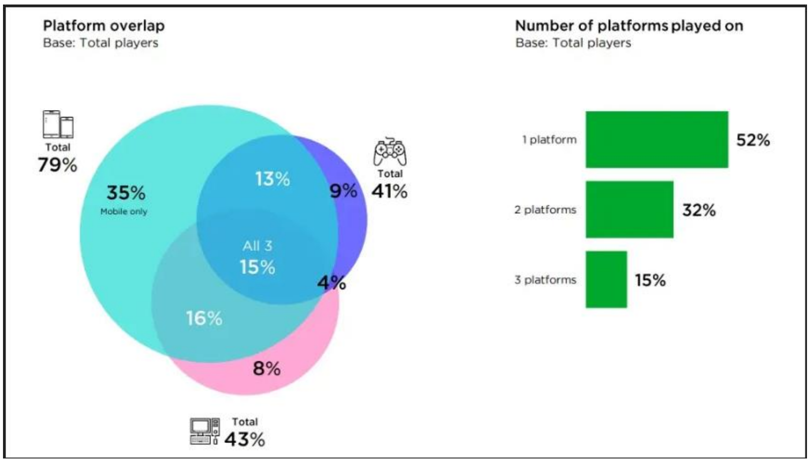


Figure 5: Overlap of Users Across Multiple Platforms (Ref W)

2.2.1. Use Case Diagram

A use case diagram is provided in Figure 6, with users performing one of the following tasks:

1. Register their profile to connect their platform account to the system.
2. Update their profile to account for missing games and active subscriptions (i.e. Game Pass)
3. Asking for recommendations to play or buy.

The recommender system then uses the profile information to provide two types of outputs:

1. Recommend a game to purchase and which platforms the game is available on.
2. Recommend a game to play from the list of games the user owns but has not played.

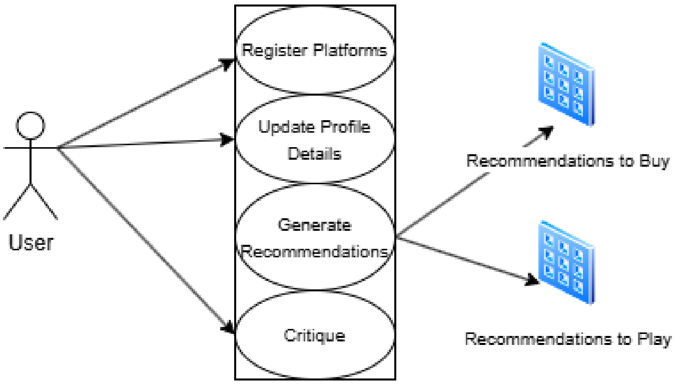


Figure 6: Use Case Diagram

2.3 Intended Use

The intended use of the recommender system is shown by the User Sequence Flow Diagram in Figure 7. The primary flow of user interaction is as follows:

1. A User connects to the system via the front-end application. Whilst likely out of scope for Version 1.0 of the system, a future goal would be to utilise an account for the recommender service to store the user's preferences and previously connected accounts. The mock-up of the front-end application is shown in Section 3.3.
2. Next the user registers their profiles. This involves selecting the platforms they own, the required account information as well as additional metadata such as if they are an *Xbox Game Pass Subscriber.*
3. The user then selects which of the recommendation services they wish to use and whether they wish to get results for games to buy or play.
4. If the user wishes to see recommendations based on similar games they have played, then **Content-Based** recommendations are provided as per Section 5.2.1.
5. If the user wishes to see recommendations based on games other users have liked, then either **Item-Based Collaborative Filter** recommendations or **User-Based Collaborative Filter** recommendations are provided as per Sections 5.2.2 and **Error! Reference source not found.**.
6. If the user wishes to see recommendations based on adjustable criteria, then the **Knowledge-Based** recommendation will be used as per Section 5.2.4.
7. If the user wishes to see recommendations based on a hybrid approach, then both the **Content-Based** and the **Item-Based Collaborative** recommenders will be used.
8. Regardless of algorithm used, the user will be provided with a list of recommended games. An example of the mock-up is shown in Section 3.3, whilst the number of recommendations is shown in Section 2.3.1 below.

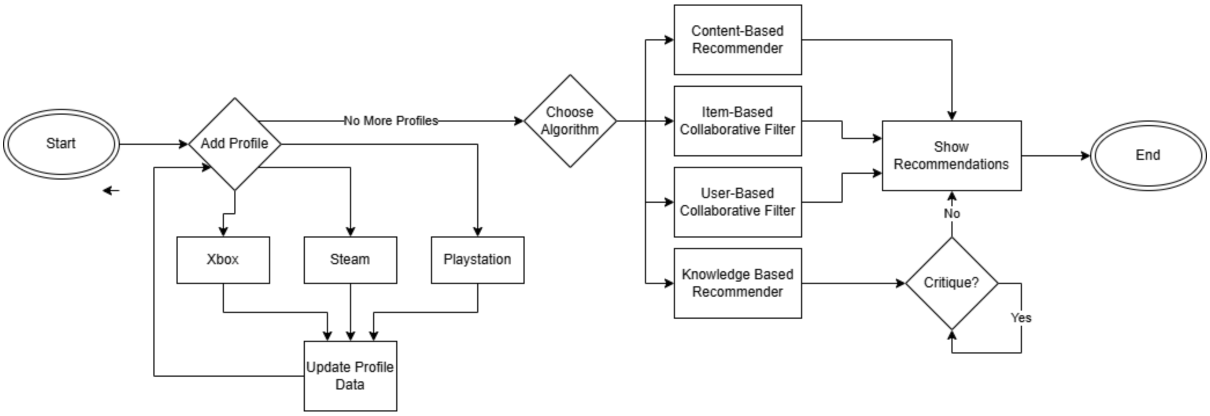


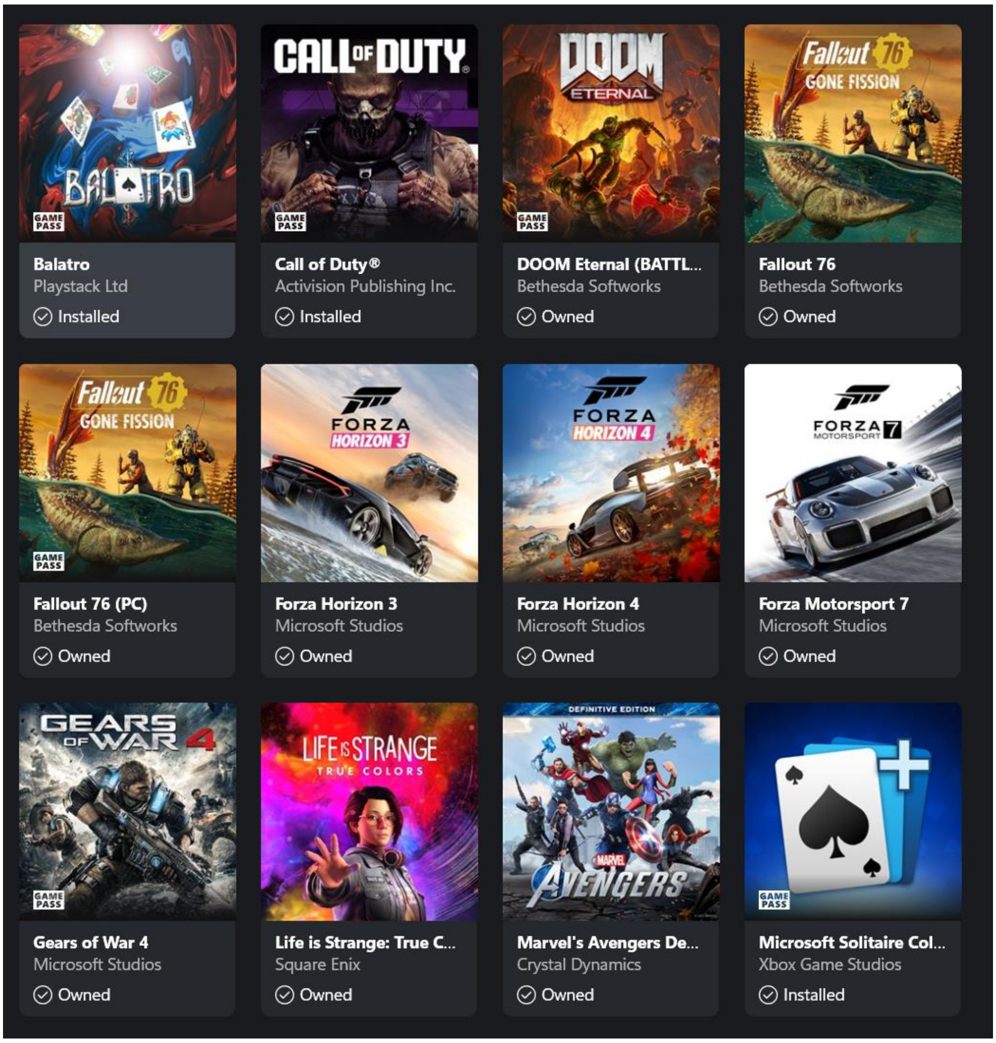
Figure 7: User Sequence Flow Diagram

2.3.1 How Many Items to Recommend

One of the key design decisions is to determine how many items to show to the user. This is a trade-off between providing sufficient choice for the user against information overload and must also compensate for several hidden factors the model is unable to determine such as the user's mood. For example, if the user just finished a 200-hour RPG, it might be reasonable to expect that they don't want to be recommended another 200-hour game, despite the similarity metrics showing it is a good choice. The two recommenders showcased in Section 2.1.2, provide some real-world comparisons to help determine this choice. As seen in Figure 2, the

Microsoft Recommender provides 25 titles, whilst Steam provides 12 in the *Discovery Queue*. Additionally, as per Miller's Law (Ref E), "the average person can only keep 7 (plus or minus 2) items in their working memory. Based on these factors, as well as user experience from Assignment 1, this project has decided to show **12 titles to the user** for the following reasons:

1. Based on user feedback during the Assignment, the user wanted recommendations for several genres. As described above, some gamers may like to switch genre after completing a game, so 12 titles provide sufficient coverage to provide 3 recommendations for up to 4 genres (or 2 titles for 6 or 4 titles for 3, based on user feedback during design).
2. Unlike the *Steam Discovery Queue*, the goal is to provide games like the user's tasks, not "to provide a mix of top sellers and new games". However, their reasoning for 12 makes sense as they want "every time you launch Steam you can explore through your queue and feel like you've gotten a good idea of products that may be interesting to you. "
3. The final reason, is that the intended interface is web-based, with titles ideally displayed using box art. As such based on an examination of the Xbox PC application (Figure 8) where 3 rows of 4 titles provides a good usage of space on a website.



Balatro
Playstack Ltd
Installed

Call of Duty®
Activision Publishing Inc.
Installed

DOOM Eternal (BATTL...
Bethesda Softworks
Owned

Fallout 76
Bethesda Softworks
Owned

Fallout 76 (PC)
Bethesda Softworks
Owned

Forza Horizon 3
Microsoft Studios
Owned

Forza Horizon 4
Microsoft Studios
Owned

Forza Motorsport 7
Microsoft Studios
Owned

Gears of War 4
Microsoft Studios
Owned

Life is Strange: True C...
Square Enix
Owned

Marvel's Avengers De...
Crystal Dynamics
Owned

Microsoft Solitaire Col...
Xbox Game Studios
Installed

Figure 8: Microsoft Library View

2.4. Business Model

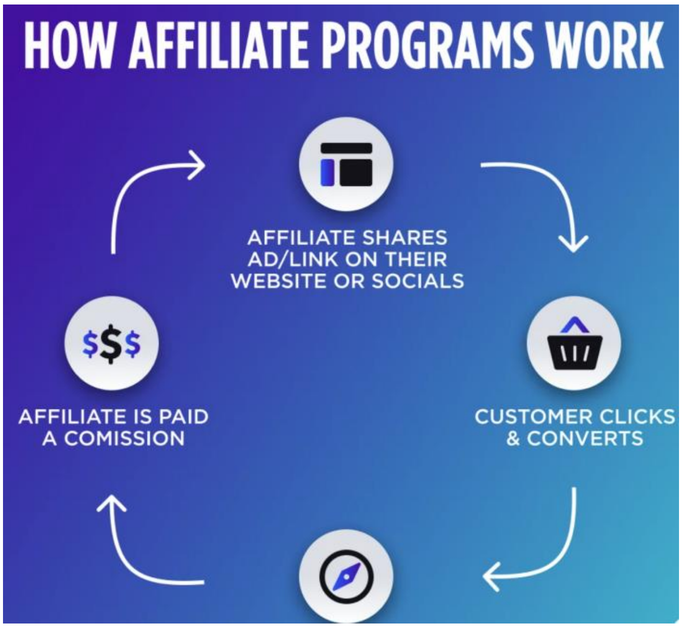

HOW AFFILIATE PROGRAMS WORK
AFFILIATE SHARES AD/LINK ON THEIR WEBSITE OR SOCIALS
CUSTOMER CLICKS & CONVERTS
AFFILIATE IS PAID A COMISSION
$$$

Figure 9: How Affiliate Programs Work (Source: Ref F)

As the system aims to provide curated recommendations for a user to **purchase** new games, the most likely business model is through *affiliate links*. An affiliate link is where a system "earns a commission for marketing another person's or company's product" (Ref F). This is often done via a cookie that tracks the referee when linking from your system to the affiliate system as seen in Figure 10. Such purchases could often be up to 5% of the purchase price and hence a viable business model. Whilst the user data (recommendations, ratio of recommendations to purchase) would likely be of interest to partner platforms to help target personalised marketing, it is not a planned aspect of the business model. Thus, the recommender would provide storefront links for each game, allowing an affiliate link to be generated.

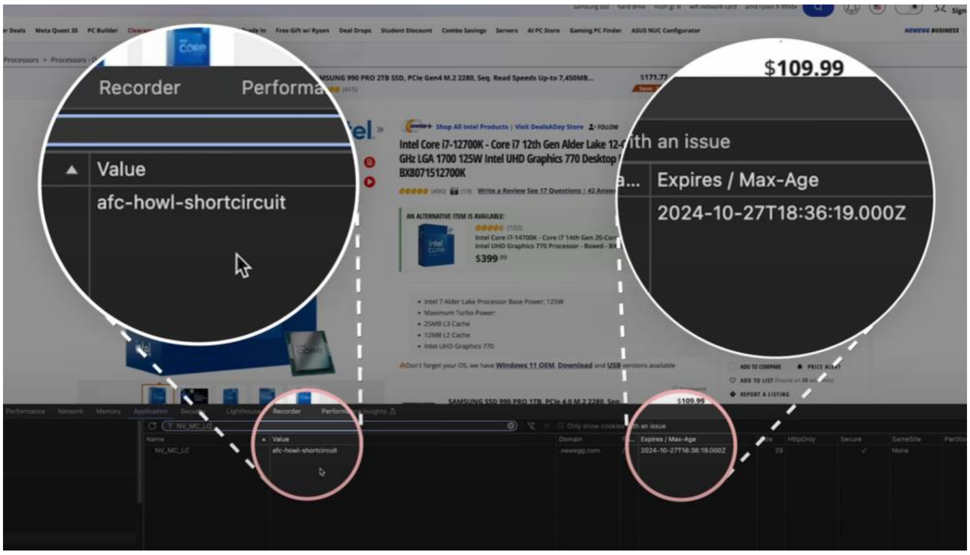


Figure 10: Affiliate Link Example. (Source Ref G)

2.5.1 Continuous Update

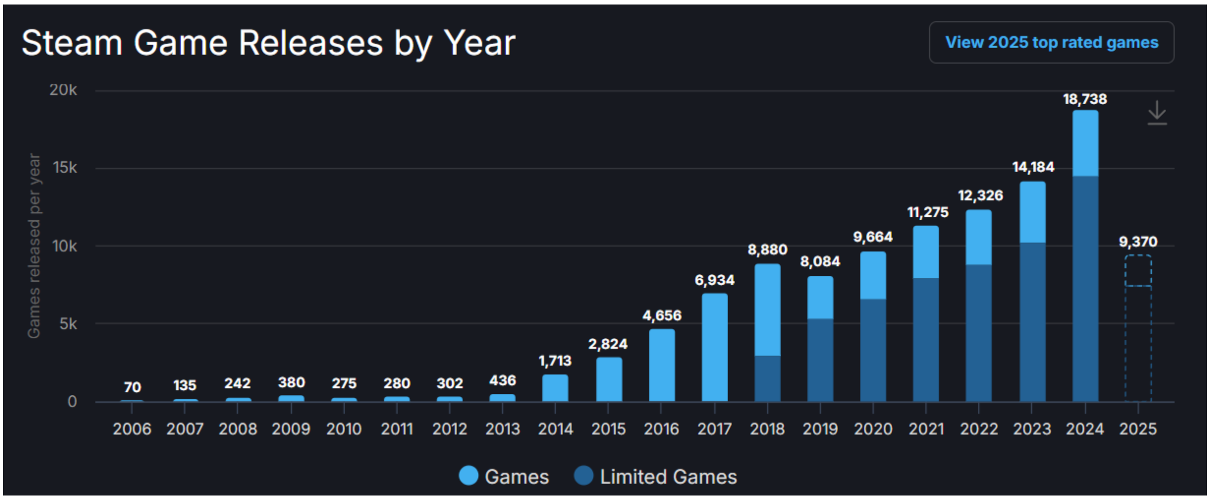


Figure 11: Steam Game Releases by Year (Source: Ref H)

The video games industry is a dynamically evolving space, with Steam alone releasing 18,738 games in 2025, up 32% to the previous year (Ref H). As later discussed in Section 4.2, the primary dataset for this project is the Internet Games Database (IGDB). As of the 28th of June 2025, the dataset claims to have 321,302 games (Ref I). To handle updates to the recommendation system the following procedures are proposed:

1. The dataset for the project is provided by scraping the IGDB dataset periodically. Unlike many other data sources, the IGDB specifically was designed with scraping in mind[1] and encourages downloading the entire dataset to a local database and updating through "webhooks".
2. On each login (or recommendation if the account service is not functional), the user can update their profile data using the platform APIs discussed in Section 4.2.2.
3. If the user does not have any connected platforms, hence does not have a user profile, they will only be able to use the Knowledge-Based Recommendation service.

[1] https://discuss.dev.twitch.com/t/igdb-store-all-games-in-local-database/47241

3. System Scope

3.1. System Architecture

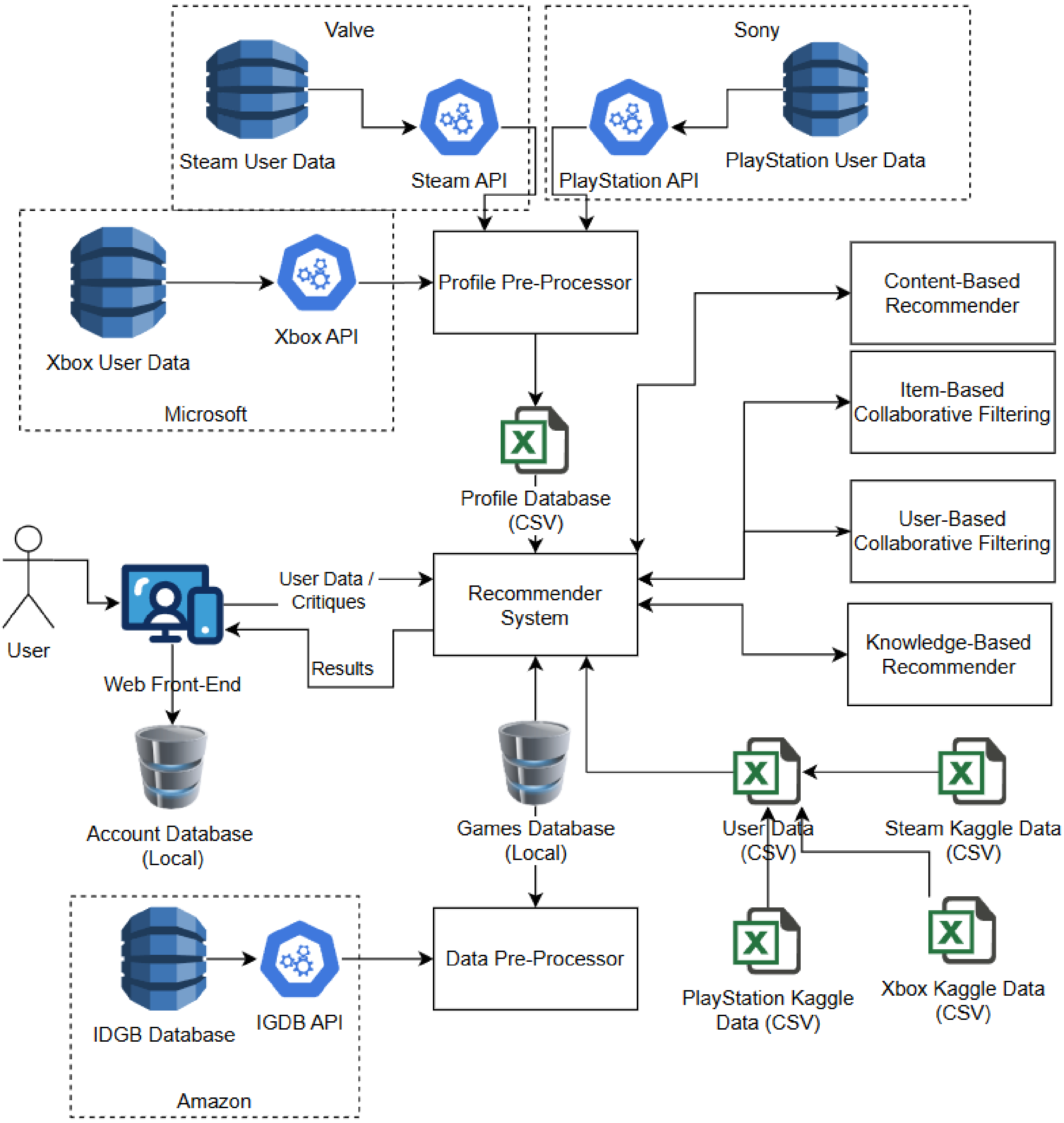


Figure 12: System Architecture Diagram

A System Architecture Diagram is provided in Figure 12 above, whilst Table 2 below describes the key components of the system.

Table 2: System Components

Component	Subsystem Owner	Framework	Description
Web-Front End[2]	Internal	JavaScript	Allows the user to provide profile information for the system and which recommender they wish to use.
Account Database[3]	Internal	TBD - Microsoft SQL, CSV or Web Cookie	Allows the user to store settings for future usage.
Recommender Back-End	Internal	Python	Implements the four Recommendation algorithms to provide results to the user.
GamesDB	Internal	CSV (SQL as an extension)	Provides the dataset of all games and features used by the Recommender.
User Data	Internal	CSV	A common dataset comprised of user activity (games past users have purchased). This is based on the Kaggle dataset (Ref L)
Data Pre-Processor	Internal	Python	Connects to the IGDB to download and pre-process the dataset for use with the Recommender
IGDB Database	Amazon	External API	Provides the underlying dataset to the Data Pre-Processor
Xbox Profile Data	Microsoft	External API	Provides the users profile data to the Recommender.
Steam Profile Data	Valve	External API	Provides the users profile data to the Recommender.
PlayStation Profile Data	Sony	External API	Provides the users profile data to the Recommender

3.2 System Boundaries

The system boundary is defined by the Architecture diagram in Figure 12. However, the following aspects are out of scope for this project (or the initial roll-out of the system).

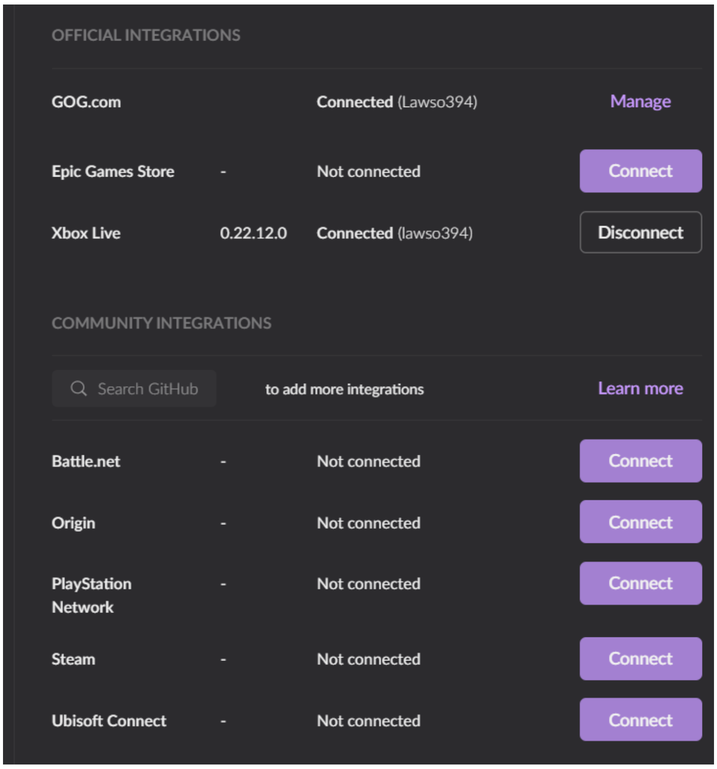
1. Authentication to external data providers. Except for a Twitch account needed to connect to the IGDB database, no user authentication to data providers is required. Thus, the project is limited to what profile information is provided to a public user.
2. Affiliate Links to the store platforms. Whilst discussed as a potential business model in Section 2.4, no affiliate links will be implemented for this project.
3. Deployment will not be handled as part of the project. As per the Project Specification, the system will be completed via a Jupiter Notebook and as such will not include any deployment activities shown in Figure 12. This includes the user account database and web-front end.

3.3 User Interface Design

A mock-up of the user design is shown below. Upon first entering the application, the user would connect their profiles in a similar way to Figure 13 which shows the integration process of GOG Galaxy (a unified game launcher and achievement tracker, but unfortunately not a recommender). Upon selecting a recommender, Figure 14 shows the expected layout of the

[2] The planned Web-Front End is an optional part of the project and will only be completed if time permits.
[3] The Account Database is an extension part of the project and unlikely to be developed in this phase.

results page. This is what the user would see when using methods 1-3 (i.e. Content-based, Item-Based or User-Based Collaborative Filters) or the hybrid method. Finally, Figure 15 shows the user's perspective when using the Knowledge-based recommendation algorithm. As this algorithm requires critiquing by the user to refine the results, this is similar in appearance to the generic results page but includes several UI elements to help narrow the choices. Feedback would be generated via collecting user telemetry data such as which games the user clicks on after being provided with recommendations.



OFFICIAL INTEGRATIONS

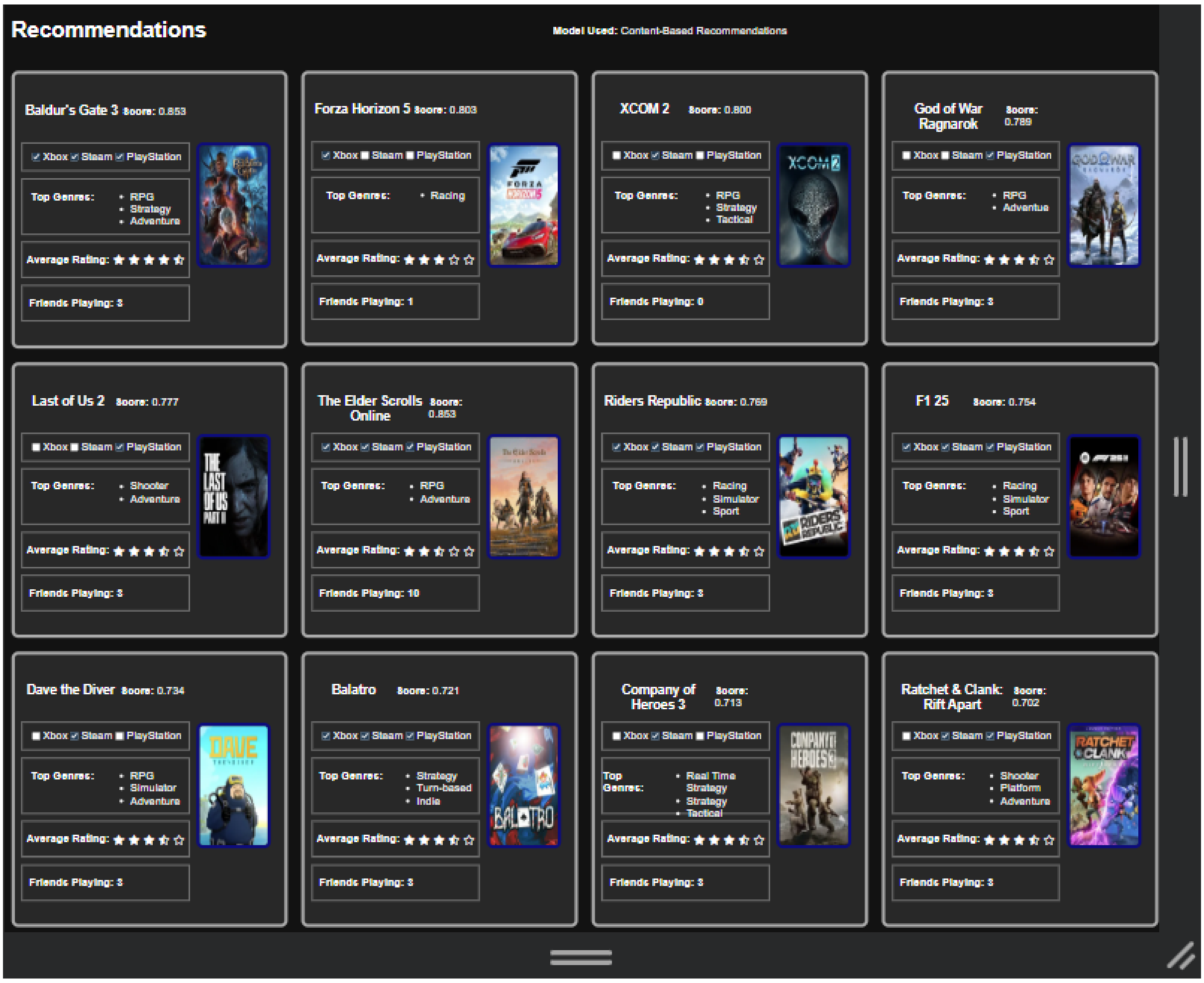
GOG.com		Connected (Lawso394)	Manage
Epic Games Store	-	Not connected	Connect
Xbox Live	0.22.12.0	Connected (lawso394)	Disconnect

COMMUNITY INTEGRATIONS

Search GitHub to add more integrations Learn more

Battle.net	-	Not connected	Connect
Origin	-	Not connected	Connect
PlayStation Network	-	Not connected	Connect
Steam	-	Not connected	Connect
Ubisoft Connect	-	Not connected	Connect

Figure 13: Mock-Up Main Page



Recommendations

Model Used: Content-Based Recommendations

Baldur's Gate 3 **Score:** 0.853
☑ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- RPG
- Strategy
- Adventure

Average Rating: ★★★★⯪
Friends Playing: 3

Forza Horizon 5 **Score:** 0.803
☑ Xbox ☐ Steam ☐ PlayStation
Top Genres:
- Racing

Average Rating: ★★★☆☆
Friends Playing: 1

XCOM 2 **Score:** 0.800
☐ Xbox ☑ Steam ☐ PlayStation
Top Genres:
- RPG
- Strategy
- Tactical

Average Rating: ★★★⯪☆
Friends Playing: 0

God of War Ragnarok **Score:** 0.789
☐ Xbox ☐ Steam ☑ PlayStation
Top Genres:
- RPG
- Adventue

Average Rating: ★★★⯪☆
Friends Playing: 3

Last of Us 2 **Score:** 0.777
☐ Xbox ☐ Steam ☑ PlayStation
Top Genres:
- Shooter
- Adventure

Average Rating: ★★★⯪☆
Friends Playing: 3

The Elder Scrolls Online **Score:** 0.853
☑ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- RPG
- Adventure

Average Rating: ★★⯪☆☆
Friends Playing: 10

Riders Republic **Score:** 0.769
☑ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- Racing
- Simulator
- Sport

Average Rating: ★★★⯪☆
Friends Playing: 3

F1 25 **Score:** 0.754
☑ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- Racing
- Simulator
- Sport

Average Rating: ★★★⯪☆
Friends Playing: 3

Dave the Diver **Score:** 0.734
☐ Xbox ☑ Steam ☐ PlayStation
Top Genres:
- RPG
- Simulator
- Adventure

Average Rating: ★★★⯪☆
Friends Playing: 3

Balatro **Score:** 0.721
☑ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- Strategy
- Turn-based
- Indie

Average Rating: ★★★⯪☆
Friends Playing: 3

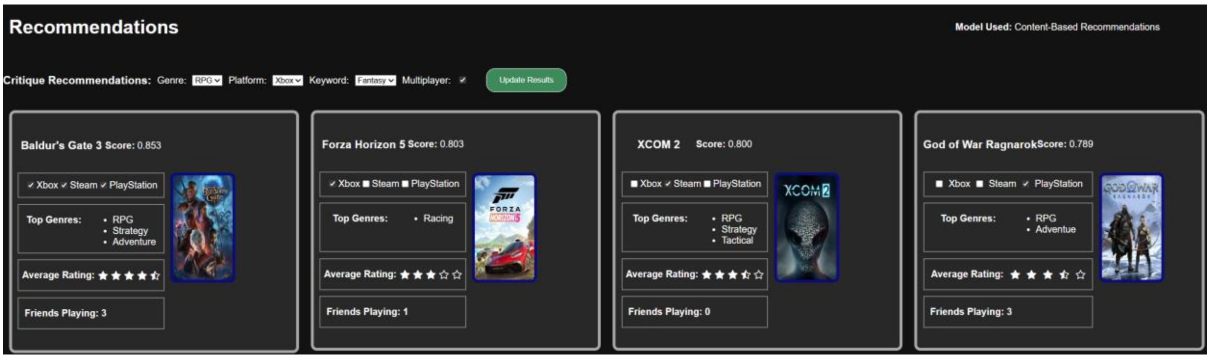
Company of Heroes 3 **Score:** 0.713
☐ Xbox ☑ Steam ☐ PlayStation
Top Genres:
- Real Time Strategy
- Strategy
- Tactical

Average Rating: ★★★⯪☆
Friends Playing: 3

Ratchet & Clank: Rift Apart **Score:** 0.702
☐ Xbox ☑ Steam ☑ PlayStation
Top Genres:
- Shooter
- Platform
- Adventure

Average Rating: ★★★⯪☆
Friends Playing: 3

Figure 14: Mock-Up of Results Page



Recommendations

Model Used: Content-Based Recommendations

Critique Recommendations: Genre: RPG Platform: Xbox Keyword: Fantasy Multiplayer: ✓

Baldur's Gate 3 **Score:** 0.853
✓ Xbox ✓ Steam ✓ PlayStation
Top Genres:
- RPG
- Strategy
- Adventure

Average Rating: ★★★★⯪
Friends Playing: 3

Forza Horizon 5 **Score:** 0.803
✓ Xbox ☐ Steam ☐ PlayStation
Top Genres:
- Racing

Average Rating: ★★★☆☆
Friends Playing: 1

XCOM 2 **Score:** 0.800
☐ Xbox ✓ Steam ☐ PlayStation
Top Genres:
- RPG
- Strategy
- Tactical

Average Rating: ★★★⯪☆
Friends Playing: 0

God of War Ragnarok**Score:** 0.789
☐ Xbox ☐ Steam ✓ PlayStation
Top Genres:
- RPG
- Adventue

Average Rating: ★★★⯪☆
Friends Playing: 3

Figure 15: Mock-up of the Knowledge-based Recommender Page

4. Dataset Design

This section contains the evaluation and design decisions for the datasets used in training and testing the recommender system. There are three types of datasets required for this project, the first containing the underlying video game data (features) shown under Section 4.2.1. The second is the user history data containing what games past users have co-owned, discussed in Section 4.2.2. Finally, the third is the target user profile data (for evaluation) and discussed in Section 4.2.3.

4.1 Dataset Requirements

Based on the intended usage described in Section 2.3, several requirements can be identified for the dataset. These requirements are shown in Table 3.

Table 3: Dataset Requirements

ID	Name	User Story	Example	Priority
D-1	Games by Platform	As a user, I want to see what games are available on platforms I own.	If I own an Xbox, I want to be recommended games available on Xbox.	Must
D-2	Games by Feature	As a user, I want to categorise games by various features (such as genre) to identify games based on similarity to my interests.	I enjoy racing games, so I want to be recommended racing games.	Must
D-3	Played Games	As a user, I want to filter played games out of the list of available games, so I am not recommended something I own,	I have played “Riders Republic” on Xbox, so I don’t want to be recommended the Steam version.	Must
D-4	Owned Games	As a user, I want to be recommended games that I own (but have not played).	I have a lot of games in my ‘backlog’, I want to know which one I should play next.	Must
D-5	Game Statistics	As a user, I want games I have played to be indirectly rated.	If I played a game for 1 hour, I don’t want it to have the same weighting as one I have played for 100.[4]	Must
D-5	Games Played by Friends	As a user, I want to be recommended games that my friends like.	If my friend and I both like RPGs, then I would like to be recommended any RPGs they like that I have not played.	Should
D-6	Subscriptions	As a user, I want my included subscriptions to be counted as ‘owned games’	If I pay for Game Pass, I want to be recommended Game Pass games even though I don’t own them.	Should
D-7	Extensibility	As a user, I want to easily add new platforms that I own.	If I buy a PS5, I want to update my recommendation space to include PS5 games.	Should
D-8	Price	As a user, I want to know how much a recommended game will cost.	If I am recommended games I don’t own, I want to know their cost (or filter by price)	Extension

[4] The idea of how long I have played a game for is more complex and is discussed further in Section 19

4.2 Chosen Datasets

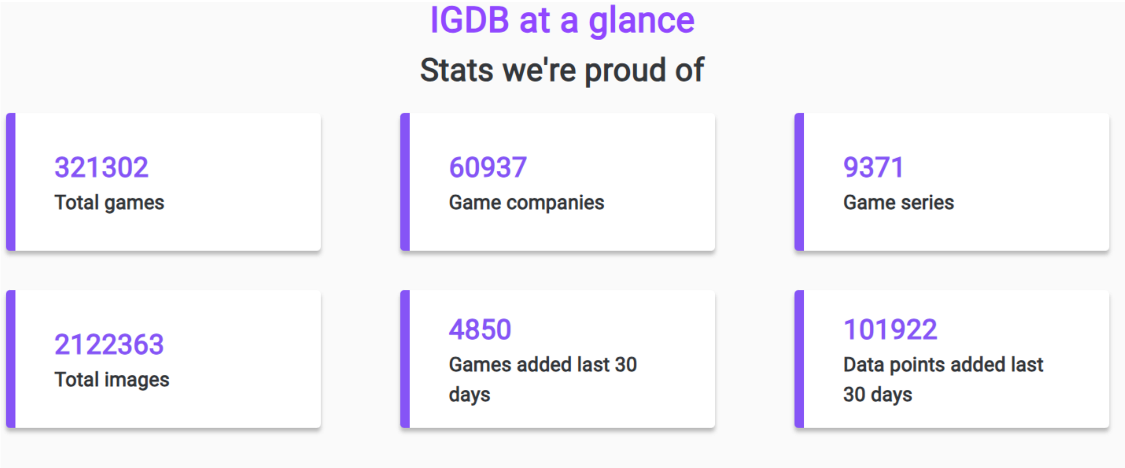
Based on the requirements in Table 3, real-word datasets were examined to see if they could be met. In the end, two primary datasets were identified for training data, one containing the underlying game data (i.e. all available games) and one containing the past user history data for users at scale. Finally, user profile data is required for evaluation purposes.

4.2.1 Game Data

It was identified early in the project requirements phase that the primary data source needs to contain the following properties:

1. A large list of games across multiple platforms.
2. Have many accessible and easily queryable fields.
3. Be free to access.
4. Be accessible in an easy-to-use manner.

Upon searching for a dataset that meets these criteria, the Internet Games Database (IGDB) was identified. As seen in Figure 16, the IGDB contains over 300,000 games from 60,000 game companies so easily meets criteria 1. In addition, the IGDB documentation (Ref J), contains over 50 features access via API endpoints and is accessible for free, only requiring a Twitch Account. Finally, APIs are easy to access through Python using a wrapper (Ref K), addressing criteria 4.



IGDB at a glance

Stats we're proud of

321302
Total games

60937
Game companies

9371
Game series

2122363
Total images

4850
Games added last 30 days

101922
Data points added last 30 days

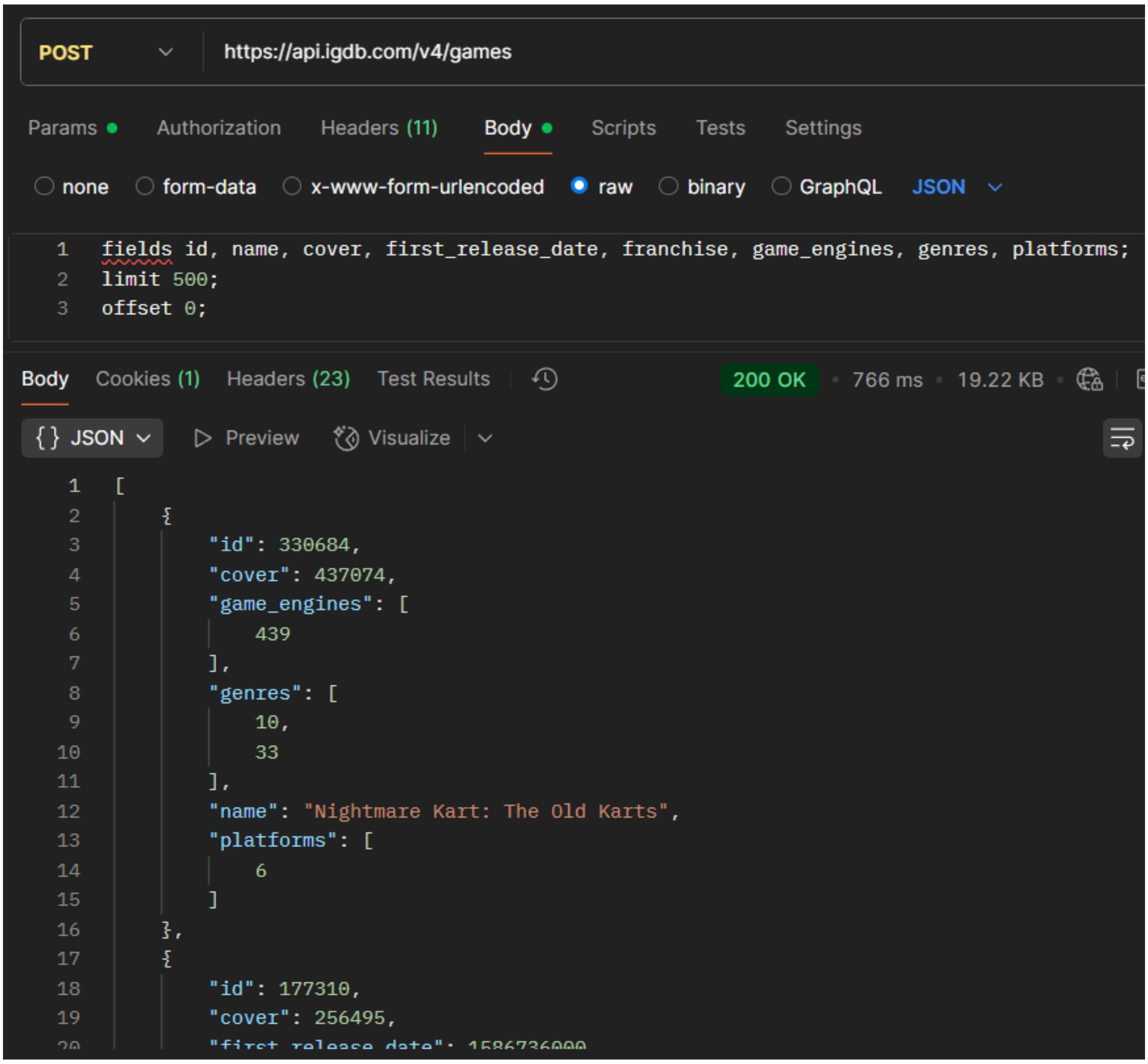
Figure 16: IGDB Statistics (Ref I)

4.2.1.1 Example of API Query to IGDB

As stated in Ref J, the IGDB provides easy access to the data via API. As such it is possible (and encouraged) to scrape the dataset 500 elements at a time. The primary route used is:

https://api.igdb.com/v4/games

Figure 17 below shows an example API call, where the first 500 games are provided and several fields. The selection of which fields from the route are required is shown in Section 4.2.1.2.



POST https://api.igdb.com/v4/games

```
fields id, name, cover, first_release_date, franchise, game_engines, genres, platforms;
limit 500;
offset 0;
```

200 OK • 766 ms • 19.22 KB

```
[
    {
        "id": 330684,
        "cover": 437074,
        "game_engines": [
            439
        ],
        "genres": [
            10,
            33
        ],
        "name": "Nightmare Kart: The Old Karts",
        "platforms": [
            6
        ]
    },
    {
        "id": 177310,
        "cover": 256495,
        "first_release_date": 1586736000
```

Figure 17: Paginated API Route to IGDB

4.2.1.2 Proposed Features from IGDB

As per Ref J, the primary game route provides 59 fields, though not all of them are appropriate for use in the Recommender system. Table 4 shows which fields are the most promising and are used to build the dataset. The primary field for identification is Name (as well as Id), whilst the some of the primary features include keywords (essentially folksonomies), genres and the various game modes. Fields such as hypes, rating and aggregated rating count can be used to represent popularity. Additional features are left for further exploration during implementation.

Table 4: Fields used from IGDB

Field	Type	Description
age_ratings	Array of Age Rating IDs	PEGI Age Rating
aggregated_rating	Double	Rating based on external critic scores
aggregated_rating_count	Integer	Number of external critic scores
alternative_names	Array of Alternative Name Ids	Alternative names for this game
collections	Array of Collection Ids	The collections that this game is in.
cover	Reference Id for Cover	Cover of this game
first_release_date	Unix Time Stamp	The first release date for this game
franchise	Reference Id for Franchise	The main franchise
franchises	Array of Franchise Ids	Other franchises the game belongs to
game_engines	Array of Game Engine Ids	The game engine used in this game
game_modes	Array of Game Mode Ids	Modes of gameplay
game_type	Reference Id for Game Type	The type of game
genres	Array of Genre Ids	Genres of the game
hypes	Integer	Number of follows a game gets before release
involved_companies	Array of Involved Company Ids	Companies who developed this game
Keywords	Array of Keyword Ids	Associated Keywords
language_supports	Array of Language Support Ids	Supported Languages for this game
multiplayer_modes	Array of Multiplayer Mod eIds	Multiplayer modes for this game
name	String	Name of Game
platforms	Array of Platform Ids	Platforms this game was released on
player_perspectives	Array of Player Perspective Ids	The main perspective of the player
rating	Double	Average IGDB user rating
rating_count	Integer	Total number of IGDB user ratings
release_dates	Array of Release Date IDs	Release dates of this game
similar_games	Array of Game IDs	Similar games
storyline	String	A short description of a games story
summary	String	A description of the game
tags	Array of Tag Numbers	Related entities in the IGDB API
themes	Array of Theme IDs	Themes of the game
total_rating	Double	Average rating based on both IGDB user and external critic scores
total_rating_count	Integer	Total number of user and external critic scores

4.2.2 Historical User Data

The next dataset needed to form the recommender is the user training data. This is where the games from the IGDB database are linked to purchases made by users at scale. The primary user data comes from Kaggle (Ref L), with several of the Kaggle Datasets used as per Table 5. The main complexity comes from needing to **link** the user data to the IGDB dataset by title (with release data or company helping to delineate similar names). For each of the three platforms (Xbox, Steam and PlayStation), the primary data is the *purchased_games.csv* which contains a list of users and the games they have purchased. Steam also provides a friends list and review list to help determine rating as later discussed in 4.2.1, with all three also providing a history file which contains a timestamped list of when players earned an achievement.

Table 5: Kaggle User Data Fields

Dataset	Description	Priority
purchased_games.csv	Contains a list of User Ids and Game Ids that they have purchased	High
friends.csv (Steam)	Contains a list of Friend Ids for each User Id	Medium
games.csv	Contains a list of Game Ids, Titles and Publishers	High
reviews.csv (Steam)	Contain a list of User Ids, Game Ids, Review and Helpfulness	Low
history.csv	Contains a link of Player Ids and Achievement Ids	Low
achievments.csv	Contains a list of Game Ids and Achievements	Low

The entity relation diagram for how the user data connects is shown in Figure 18.

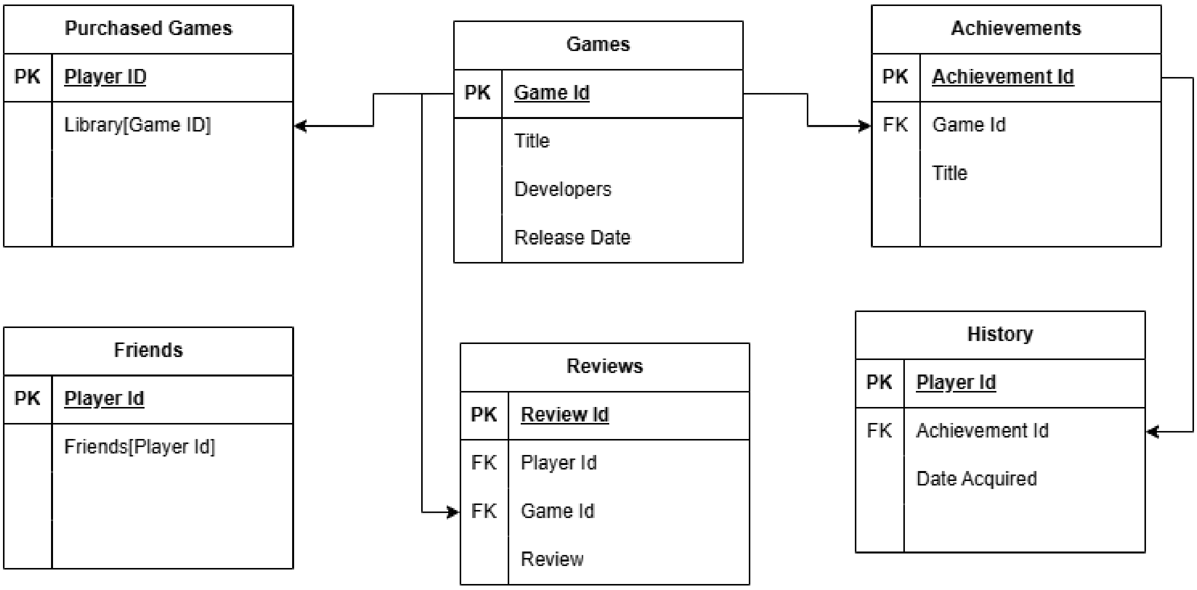


Figure 18: Entity Relation Diagram

4.2.3 User Profile (Evaluation) Data

The final dataset provides real user data for final evaluation and testing. As the Kaggle dataset (Ref L) provides a list of historical purchases, the recommender can be built and tested. However, to provide a final test of quality (such as during the user study), an evaluation dataset must be built. The intention is to use metadata from the target user's real profile. This means that for each supported ecosystem, the user's profile must provide sufficient metadata to link to the game dataset. This includes:

1. What games the user owns.
2. What games the user has played.
3. How much of a game the user has played (either in hours or percentage or achievements completed).
4. What friends the user has.

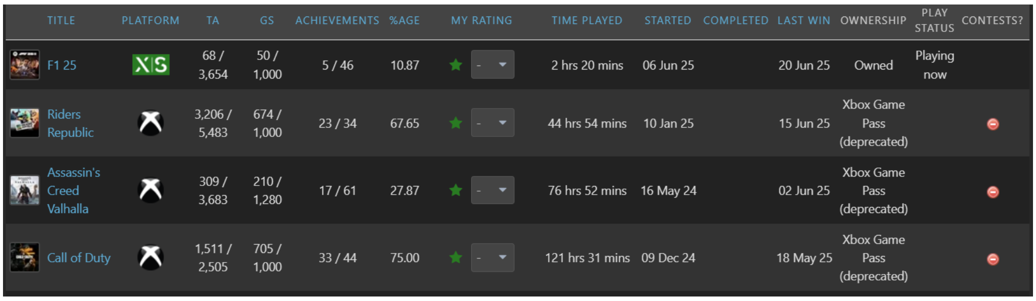
Fortunately, for the three primary supported platforms, Xbox, Steam and PlayStation, this information is available through free-to-use APIs or 3rd party websites (using official APIs). As APIs are already used for the Game Data, it is known that this can be easily supported from a technical viewpoint. Whilst the data available from each service is discussed below, the result is the same as all three services must end up writing to a common *profile database* (likely as a CSV file) that can be passed to the algorithm. If the API services do not work or are too time consuming, this data can be manually recreated using the 3rd Party sites.

4.2.3.1 Example of Xbox Profile Data

An unofficial and open-source API, known as *OpenXBL* exists for accessing user data from Xbox (Ref M) as well as many third-party sites for tracking statistics. The data in Table 6 shows the user data information available through the OpenXBL routes, where most of the required information can be accessed. The alternative data source is the 3rd Party achievement tracking website known as TrueAchievements[5]. This has access to the official Xbox API and can collate further information including Time Played, as seen in Figure 19.

Table 6: OpenXBL Route Data from /api/v2/player/titleHistory

Field	Value
Title Id	Unique Id of a Game
Name	Name of the Gmae
Devices	Array of Supported Devices (Xbox One, Xbox Series…)
Achivment.ProgressPercentage	Integer of Achievement Completion



TITLE	PLATFORM	TA	GS	ACHIEVEMENTS	%AGE	MY RATING	TIME PLAYED	STARTED	COMPLETED	LAST WIN	OWNERSHIP	PLAY STATUS	CONTESTS?
F1 25	X|S	68 / 3,654	50 / 1,000	5 / 46	10.87	★ -	2 hrs 20 mins	06 Jun 25		20 Jun 25	Owned	Playing now	
Riders Republic		3,206 / 5,483	674 / 1,000	23 / 34	67.65	★ -	44 hrs 54 mins	10 Jan 25		15 Jun 25	Xbox Game Pass (deprecated)		⊖
Assassin's Creed Valhalla		309 / 3,683	210 / 1,280	17 / 61	27.87	★ -	76 hrs 52 mins	16 May 24		02 Jun 25	Xbox Game Pass (deprecated)		⊖
Call of Duty		1,511 / 2,505	705 / 1,000	33 / 44	75.00	★ -	121 hrs 31 mins	09 Dec 24		18 May 25	Xbox Game Pass (deprecated)		⊖

Figure 19: True Achievements Xbox User Data

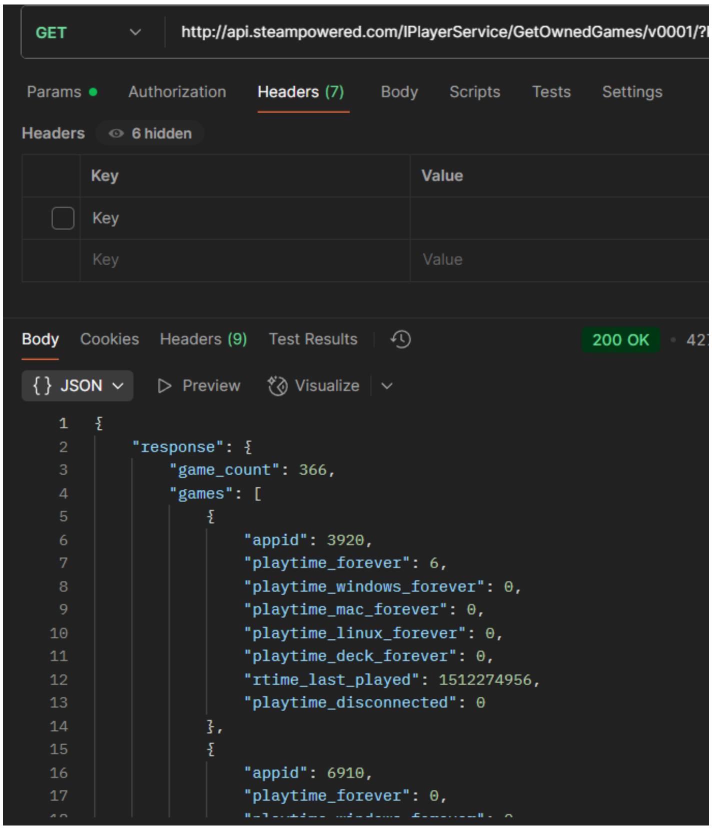
[5] https://www.trueachievements.com/

4.2.3.2 Example of Steam Profile Data

Valve (the owner of Steam) has opened up access to Steam via an API so “website developers can use the data from Steam in new and interesting ways” (Ref N), with one example already shown in this document in Figure 11 where Ref H uses Steam API’s to get access to game release data. The Valve Developer Community provides the documentation on how to access the Steam Web API (Ref O). The primary routes used for Steam are shown in Table 7, with an example of the output shown in Figure 20.

Table 7: Steam API Routes Used

Route	Description	Fields Used
Get Player Summaries	Returns basic profile information for a given user	Steam Id of Player
Get Friend List	Returns the friend list of any Steam user, provided their Steam Community profile visibility is set to "Public".	SteamId of Friend
Get App List	Returns all the Application Names and Id	AppId Name
Get Owned Games	Returns a list of games a player owns along with some playtime information[6]	AppId PlayTime_Forever
Get Player Achievements	Returns a list of achievements for this user by app id	Boolean for each achievement in the game
Get Global Achievement Percentages For App	Returns on global achievements overview of a specific game in percentages.	Achievement Percent



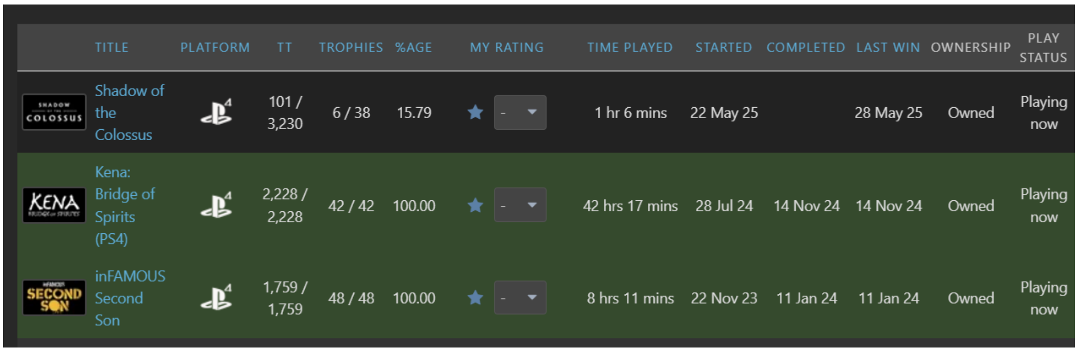
```
{
    "response": {
        "game_count": 366,
        "games": [
            {
                "appid": 3920,
                "playtime_forever": 6,
                "playtime_windows_forever": 0,
                "playtime_mac_forever": 0,
                "playtime_linux_forever": 0,
                "playtime_deck_forever": 0,
                "rtime_last_played": 1512274956,
                "playtime_disconnected": 0
            },
            {
                "appid": 6910,
                "playtime_forever": 0,
                "playtime_windows_forever": 0
```

Figure 20: Steam API Route Results

[6] Only provided if the profile is publicly visible.

4.2.3.3 Example of PlayStation Profile Data

Sony does not provide a well-documented official API; however, an open-source project does provide one instead (Ref P). This API provides the required information for PlayStation titles. However, it was decided to use TrueAchievments sister site TrueTrophies[7] which provides the required information as seen in Figure 21 due to easy of access.



	TITLE	PLATFORM	TT	TROPHIES	%AGE	MY RATING	TIME PLAYED	STARTED	COMPLETED	LAST WIN	OWNERSHIP	PLAY STATUS
	Shadow of the Colossus		101 / 3,230	6 / 38	15.79	★ -	1 hr 6 mins	22 May 25		28 May 25	Owned	Playing now
	Kena: Bridge of Spirits (PS4)		2,228 / 2,228	42 / 42	100.00	★ -	42 hrs 17 mins	28 Jul 24	14 Nov 24	14 Nov 24	Owned	Playing now
	inFAMOUS Second Son		1,759 / 1,759	48 / 48	100.00	★ -	8 hrs 11 mins	22 Nov 23	11 Jan 24	11 Jan 24	Owned	Playing now

Figure 21: TrueTrophies Player Information

4.1.1.1 Evaluation Dataset Structure

The final aspect of the evaluation dataset is to put all the inputs together into a single data. This can be seen in Table 8.

Table 8: Evaluation Dataset Structure

Field	Description	Example
Name	String Name of the Game	Shadow of the Colossus
Progress	Percentage of Game Complete (Double)	25
Time Played	Hours spend in game (Double)	1.25
Platform	String Platform Game is Owned In	PlayStation

4.1.2 Alternative Datasets

Before settling on the IGDB, an alternative dataset was considered called MobyGames (Ref Q). Whilst similar in appeal, MobyGames was not considered due to the entry-level user (or “Hobbyist”) being required to pay $9.99 / month to access the database.

Additionally, several other platforms were considered to feed in data. Nintendo was considered but it was not possible to find a public facing API (instead requiring registering as a developer)[8]. This information may still be used but would need to be manually added to the profile dataset.

4.2 Dataset Assumptions and Dependencies

Several assumptions and dependencies exist for both the Game and Profile Datasets described above. The major risk mitigation is that profile data can be manually provided.

1. Access to profile APIs may require authentication to the platform service.
2. Access to official APIs may require an API key to be registered for the service.
3. Unofficial APIs may not work correctly.
4. API requests may be rate limited.

[7] https://www.truetrophies.com/gamer/lawso394/gamecollection
[8] https://www.reddit.com/r/gamedev/comments/1c0m2sf/how_to_get_switch_api/

5. Recommender Methods

As shown in the architectural diagram (Figure 12), several recommendation methods are considered for this project. While each method is implemented using different techniques, most of these recommendation methods have similar user requirements which are described in Section 5.1, with Section 5.2 describing the individual implementation for each method.

5.1 Recommender System Requirements

Table 9: Recommender System Requirements

ID	Name	User Story	Example	Priority
M-1	Ranked Top N Recommendations	As a user I want the Ranked Top N recommendations to provide variety.	The system must give the player multiple recommendations as several hidden factors might affect the players choice.	Must
M-2	Diversity	As a user I want N recommendations to come from multiple genres.	By providing multiple genres, this promotes diversity of the recommendation set.	Should
M-3	Exclude Library Data from New List	As a user I want N recommendations for new games only.	The system must exclude the players library from the recommendation set.	Must
M-4	Exclude Completed Games from Library List	As a user I want N recommendations for next game to play excluding those I have already played.	The system must exclude the players played games list from the recommendation set.	Must
M-5	Scoring	As a developer, I want my recommendations to provide a score for evaluation.	As the system provides ranked recommendations, the raw scores should be provided for evaluation.	Must
M-6	Pseudo-Randomness	As a developer, I want any randomness used in the algorithms to be seeded for repeatability.	As the system must be evaluated, true randomness must be removed for repeated evaluation.	Must
M-7	Hardware Constrained	As a developer, I want any algorithms to be runnable on desktop PC hardware.	As the development is limited to a single machine, any algorithms should require minimal hardware resources (i.e. ML training may be constrained).	Must
M-8	Modular	As a developer, I want any recommendation algorithm to be modular to allow for potential hybrid systems.	As multiple recommenders are used, they should be built in a modular fashion to allow for further experimentation.	Should

5.2 Methods Chosen

As per the Assignment Specification, several methods are proposed that should work with the chosen dataset. Additionally, one of the methods should be a simple 'baseline' method from which the more complex methods can be compared against, with both single-type and hybrid methods considered. The scale of algorithms must fit within the project timeline in Section 6.2. Based on these requirements, five methods are proposed, with each discussed below. It is noted that not all of these may be implemented based on experimentation results.

5.2.1 Content-Based Recommendation (Baseline)

The first method is a standard Content-Based Recommendation. As described by Ref R, Content-Based recommendation is primarily a machine learning based method that treats the dataset as a collection of attributes or features. The system then aims to recommend games that have strong 'similarity' to the user's profile data. The content-based recommendation is constrained as to not use the same code from the Assignment, however, acts as the 'baseline' method. A flow chart of the Content-Based Recommendation approach is shown in Figure 22 below, where the main stages are:

1. **Feature Engineering**: As stated in Ref J, there are more than 50 potential features in the IGDB dataset. Thus, the first stage of implementing a content-based recommender is to conduct a deep dive into the various features to determine which ones are relevant.
2. **Data Cleansing**: As per the Lecture slides, the second stage is to cleanse the data to reduce noise in the dataset. Details of this step are determined during implementation.
3. **TF-IDF Features Extracted:** Here the game data set is converted into appropriate features for defining similarity with.
4. **Build the User Profiles:** Next the user profiles (from the evaluation set or the training) are applied to the dataset, with the following filters:
    a. User Played games are filtered out.
    b. If only looking at backlog, anything not from the user's library filtered out.
5. **Determine Similarity:** The profile is then compared against the dataset using cosine similarity to find the games that are most like the user's profile.
6. **Rank the Results and Provide to User:** Finally, the results from Step 5 are ranked and the top N are provided to the user.

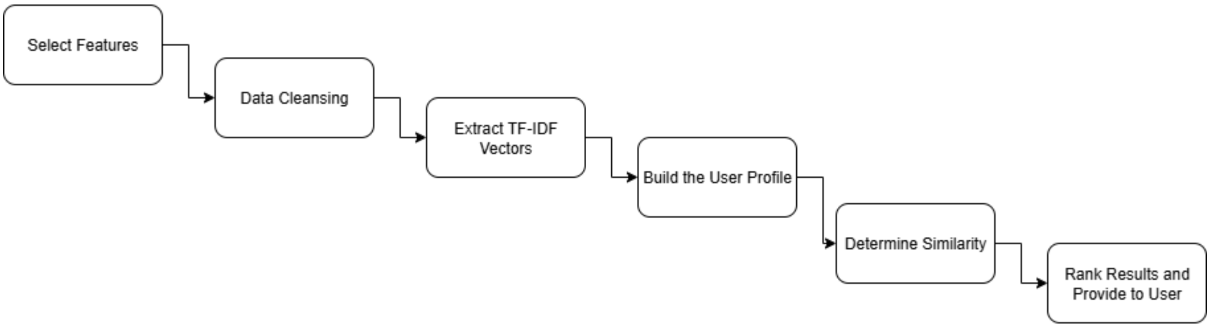


Figure 22: High Level Process for Content-Based Recommendation

5.2.2 Item Based Collaborative Filtering Recommendation

As per Ref R, Collaborative-Based filtering uses crowd-source ratings where items are recommended based on similar 'users' where if you rate items similarly to others then you are recommended games those user like that you haven't played. The process for Collaborative-Based Recommendation is shown in the Flow Diagram in Figure 23, with the major steps being:

1. **Initial Data Analysis:** The game data from the IGDB is pre-processed for appropriate features as per Content-Based recommendation.
2. **Link the User Training Data:** The next stage is to link the training data. Here, the games that all users have played are linked to the original dataset with rating provided as one of three potential methods for providing an implicit rating. Note the idea of rating is discussed deeper in Section 4.2.1.
    a. Binary representation: Has the user played the game or not?

   b. Rated by Playtime: How long has the user played the game for? This may be normalised by how long the average user takes to beat a game. This is provided from the service *HowLongToBeat*[9] which can be sourced from another API as shown in Ref S or through an alternative route in the IGDB dataset.
   c. Rated by Percentage of Achievements: As each game service has their own achievements, the user's library could be rated against the percentage of total achievements they have collected.

3. **Add the Test User Profile*:*** Next, the user in question (either from the test set or the evaluation set) is passed to the recommender, with their ratings similarly built.
4. **Build a Neighbourhood:** A neighbourhood of similarly rated items are generated for the user's profile (again either the library of unplayed games or all games). As per Ref R, this is known as Item-*Based Collaborative Filtering*. Potential techniques that could be used here include **K-***Nearest Neighbourhood*.
5. **Rank the Results and Provide to User:** Here the results from Step 4 are ranked and the top N are provided to the user.

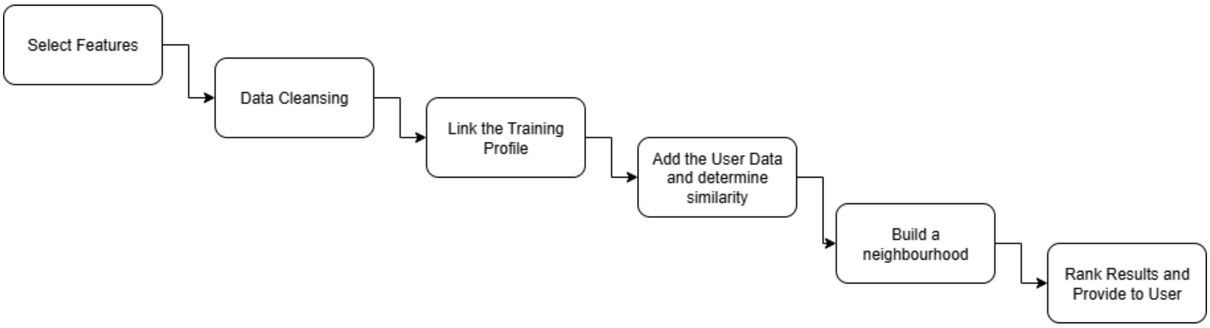

Select Features
Data Cleansing
Link the Training Profile
Add the User Data and determine similarity
Build a neighbourhood
Rank Results and Provide to User

Figure 23: Item-Based Collaborative Filtering Flow Diagram

5.2.3 User-Based Collaborative Filtering

The third method proposed is an alternative form of Memory Based Collaborative Filtering. Here instead of finding items that are similarly to what the user has rated, the users' real friends (or the larger test set) are instead used. This requires gaining access to the users' friend lists for each platform using the APIs in Section 4.2.2 or the friends.csv in the Kaggle Data from Section 4.2.2. The general process for this recommender, is mostly the same except that now the neighbourhoods are of similar users, not similar items. The final recommendations come from the games the highest rated games similar users have played that the target user has not.

5.2.4 Knowledge-Based Recommendation

The next recommendation method is a Knowledge-Based Recommender. As per Ref R this type of recommender focuses on a selection of useful attributes which the user interactively critiques to narrow down the recommendation set. Rather than use a profile to determine what the user likes, here the only use of the profile is to exclude already owned (or played) games. Figure 24 provides the general flow of the Knowledge-Based Recommender, with the key stages:

1. **Determine the Rules for Critiquing:** Here a set of associate rules are generated based on the underling IGDB dataset. As described by Ref R these rules are *constraint-based* in which a user sets constraints which can be critiqued against. This may include features such as Genre, Released Date or Platform.

[9] https://howlongtobeat.com/

2. **Generate a small consideration set:** Here the initial consideration set of recommendations are presented for the user to critique against. How this consideration set is determined is left as a design decision to allow for exploration.
3. **Navigation-Based Recommendation:** Here the user provides several changes to the consideration set based on the design rules. This will refine the consideration set to a series of better recommendations.
4. **Iterate on Step 3:** Until the user has found recommendations they are happy with.

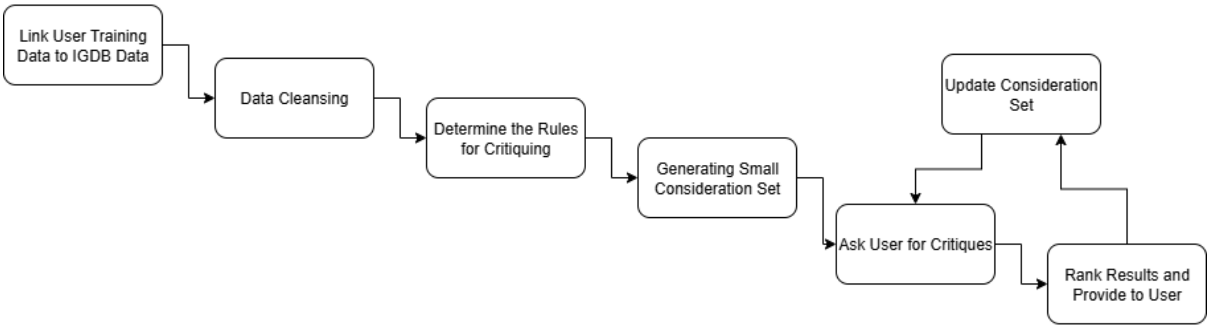

Link User Training Data to IGDB Data
Data Cleansing
Determine the Rules for Critiquing
Generating Small Consideration Set
Ask User for Critiques
Rank Results and Provide to User
Update Consideration Set

Figure 24: Knowledge-Based Recommendation Flow Diagram

4.1.1 Hybrid Recommendation

The final recommendation method (and potential alternative to Knowledge-Based Filtering) is to use a Hybrid approach. Here the baseline method (Content-Based Filtering) and the second method (Item-Based Collaborative Filtering) is combined to provide the best of both approaches.

1. **Build the Dataset as per Content-Based Filtering:** Here the IGDB dataset is again built and TF-IDF features are extracted as per Method 1.
2. **Link the User Training Set:** Next the User Training set which contains all the items that each user has purchased is linked into the IGDB data.
3. **Build Neighbourhood against Training Profile:** Next the neighbourhood similarity of item-based collaborative filtering is repeated as per Method 2.
4. **Generate both Recommendations and Merge:** Finally, the results of both recommenders are generated, and an average is combined to provide more refined results to the user.

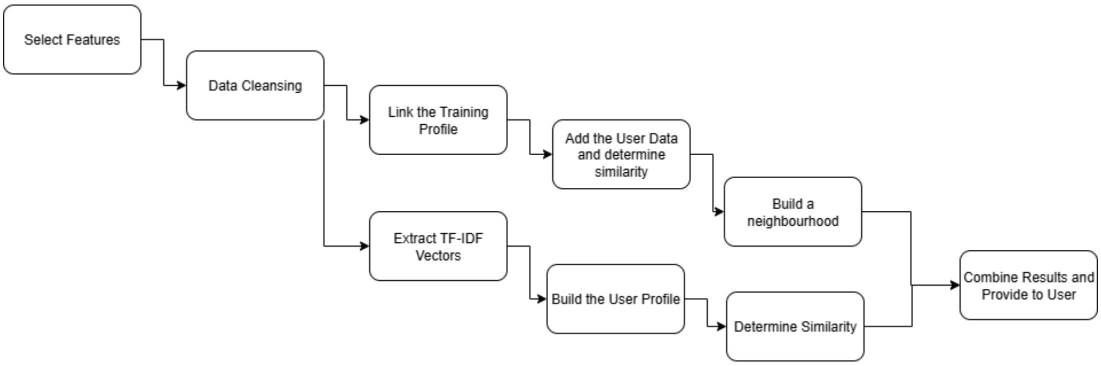


Figure 25: Hybrid Recommender Flow Diagram

Note: A potential alternative to simply combining the results is to pass the results of the content-based recommender to the collaborative filter, ensuring that the CF method has a smaller dataset to work with. This is left as a design decision through experimentation.

4.2 Method Assumptions and Constraints

4.2.1 Discussion on Implicit Ratings

Several of the recommendation algorithms require a ‘rating’ of what the user likes. As discussed in Section 5.2.1, an several possible implicit ratings could be used, each with their own trade-offs. The three main ones are:

4.2.1.1 Binary Representation

This is simply a yes or no regarding whether the user has played the game or not. This is the simplest form of implicit rating as it is represented by a binary variable. However, this is misleading as it doesn’t show whether a user liked a game. As such it is generally considered a poor metric for determining a rating but is used for indicating the list of eligible games.

4.2.1.2 Rated by Playtime

A slightly more advanced way of determining whether a user liked a game or not is playtime. It can be assumed that if a player has spent a significant time in a game, then they must have liked it. However, this can be misleading when considering how long a game is. Consider the case of two games, one taking 10 hours to complete and one taking 100 hours. By simply measuring rating as a timescale, this implies that the 100-hour game is rated 10x better than the 10-hour game which is misleading. The best use of playtime is to keep the binary representation and threshold a nominal value (maybe 2 hours) to exclude games the player tried but did not like. This is reasonable as evidenced by Steam’s Refund policy (Ref T) being limited to 2 hours.

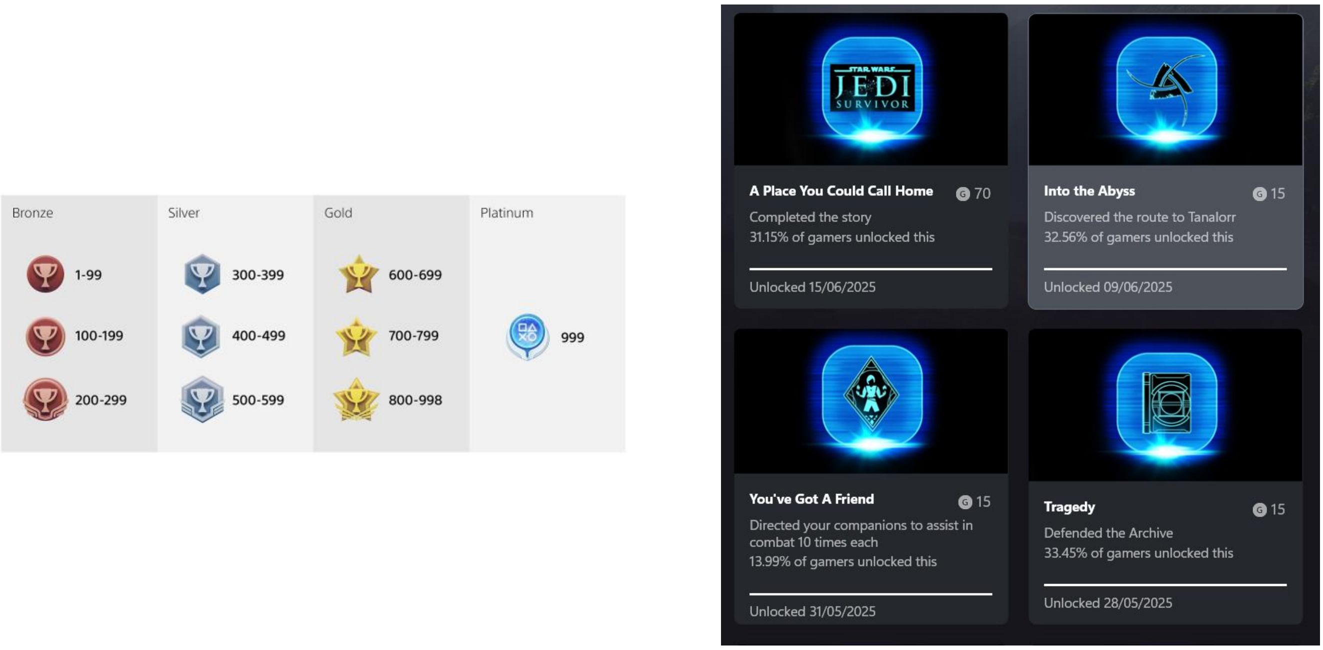
4.2.1.3 Normalised Rated by Playtime

The next method for implicitly rating a game is by normalised playtime. Instead of using playtime to threshold a binary representation for liking a game, the playtime can instead be normalised against the global average. A measure can be taken from *HowLongToBeat* which is a crow-sourced way for determining how long the average player takes to complete a game. Then the 10-hour and 100-hour games can be normalised with a value of 1.0 meaning the player has taken the average completion time. This means the following equation can be used for player rating (allowing an error factor):

$$Rating = \frac{Time\ User\ Spent\ in\ Game}{Average\ Time\ User\ Spent\ in\ Game} = \begin{cases} < 0\ 8 \rightarrow User\ did\ not\ like\ the\ game \\ \geq 0.8 < 1.2 \rightarrow The\ user\ liked\ the\ game \\ \leq 1.2 \rightarrow The\ user\ really\ liked\ the\ game \end{cases}$$

4.2.1.4 Rated by Completion Percentage

As most modern games now include achievements, an alternative measure of rating how much a player liked a game is by measuring how far the player got through it. Here the assumption is that players who enjoy a game are more likely to complete more of the game. This can be done through the percentage of achievements completed in the game. This model does not consider achievement difficulty (something that PlayStation account for with trophy rarity, whilst Xbox and Steam account for with global achievement attainment statistics in Figure 26). While a more advanced model using achievement rarity could be determined it is beyond the scope of this assignment.



Bronze	Silver	Gold	Platinum
1-99	300-399	600-699	
100-199	400-499	700-799	999
200-299	500-599	800-998	

A Place You Could Call Home 70
Completed the story
31.15% of gamers unlocked this
Unlocked 15/06/2025

Into the Abyss 15
Discovered the route to Tanalorr
32.56% of gamers unlocked this
Unlocked 09/06/2025

You've Got A Friend 15
Directed your companions to assist in combat 10 times each
13.99% of gamers unlocked this
Unlocked 31/05/2025

Tragedy 15
Defended the Archive
33.45% of gamers unlocked this
Unlocked 28/05/2025

Figure 26: PlayStation Trophy Levels[10] (Left) and Xbox Achievement Statistics[11] (Right)

[10] https://www.playstation.com/en-au/support/games/how-to-earn-trophies-on-playstation--consoles/?#grades

[11] Own screenshot

5. Evaluation

As per the assignment specification, the recommender system must be evaluated in two different ways. The first is through quantitative metrics as used in the Assignment. These metrics are discussed in Section 5.1 and are used to compare the four recommendation algorithms relative performance. The second set of metrics are qualitative metrics as shown in Section 5.2. These metrics help determine whether both the 'correctness' of the recommendation model when used against the historic user purchases (as a held back test split). This can also be used to determine whether a real user would want to use the system in production, helping to support the User Study in Section 5.4.

5.1 Quantitative Metrics

As discussed in Section 2.3.1, a design decision was made that 12 recommended items should be provided to the user. This allows for Top N metrics to be used. As highlighted by Ref U, most of these metrics rely on the idea of relevance, which in the case of this project is whether the user thinks a recommendation is valuable. This can be either binary (the item is something they would play), or a ranked list of games they would play. Due to the large amount of data in the dataset, a ranked list is more viable when only using user's library for the 'next game to play' recommendation. The primary quantitative metrics used are

5.1.1 Precision At N

The first quantitative metric is Precision at K. This is a simple calculation as shown below that calculates how many of the N-Recommendations are relevant to the user.

$$Precision\ @\ N = \frac{\#\ of\ Games\ the\ User\ would\ Play}{N}$$

5.1.2 Mean Reciprocal Rank (MRR)

The second quantitative metric is the mean reciprocal rank. This is defined by the below equation with the reciprocal rank determining how many recommendations the use must filter through before finding the first relevant one. This is averaged over all users to provide a more generalised score.

$$MRR\ @\ N = \frac{1}{U}\sum_{u \in U} \frac{1}{Rank\ of\ First\ Relevant\ Item}$$

5.1.3 Mean Average Precision (MAP)

The third quantitative metrics is the mean average precision. Unlike standard precision, this metric accounts for the rank of relevant recommendations by penalising each relevant recommendation by how many irrelevant ones appear higher in the ranked list. It is given by the below equation:

$$MAP\ @\ N = \frac{1}{U}\sum_{u \in U} \frac{1}{I}\sum_{i \in I} \frac{i}{Rank(i)}\ \ where\ I\ \ is\ the\ set\ of\ relevant\ items$$

5.1.4 Hit Rate

The fourth quantitative metric is Hit Rate. This is a simple binary measure asking if at least one relevant item was in the list of recommendations and increases in probability as the value of N increases.

5.1.5 Discounted Cumulative Gain (DCG)

The fifth metric is Discounted Cumulative Gain. This is a measure the total relevance of the entire list. Unlike Cumulative Gain, it discounts the gain by the rank of the relevant items, penalising lower ranked items in the list. It is given by the equation:

$$DCG\ @\ N = \sum_{i=1}^{N} \frac{rel(r_i)}{\log_2(i+1)}$$

5.2 Qualitative Metrics

In addition to the quantitative metrics, several qualitative metrics are used. These metrics are better for the user study (and the evaluation against real users) as they are better able to represent the feedback provided by the user. These metrics help quantify the question of "would a real person want to use this recommendation", hence providing validation compared to the verification performed with the quantitative metrics.

5.2.1 Diversity

The first qualitative metric used is diversity. As stated in the Table 9, one of the requirements of the system was to provide different genres of game. This was one of the driving reasons for providing 12 recommendations in Section 2.3.1 as it allows for anywhere from 2-4 genres. This metric is simply driven by the number of genres presented in the list. It is considered qualitative rather than quantitative as there is no 'ideal' value for diversity, instead being up to the user to determine if there are a sufficient variety in recommendations.

5.2.2 Serendipity

The second qualitative metric used is serendipity. This is defined in Ref U as the system suggesting "items beyond the user's typical preferences or expectations". Whilst this is potentially measurable with values such as cosine similarity against the user's profile, the real measure of serendipity is left up to the user and how 'surprised' they are by the results. The goal here is for the system to recommend a non-obvious (i.e. not a popular AAA) game.

5.2.3 Satisfaction

The final qualitative measure is satisfaction. As defined in Lecture 4, this is a measure of how satisfied the user is with the recommendations. A perfect score here is that every item in the list is a good recommendation and the lowest possible score is that the user is unhappy with every single recommendation. It is qualitative because every user has their own definition of how good the ratings are.

5.3 Weighting of Metrics

As per the assignment specification, when multiple metrics are used, a method for weighting the metrics is used to determine the best model as well as consideration for how each metric is a trade-off. For example, whilst Precision @ N is listed as one of the primary metrics, it is understood that this is a trade-off against Recall @ N, with the Precision-Recall curve shown in Figure 27. For this project Precision is favoured over Recall which calculates the entire list of relevant items as per the equation below. As this is hard to determine considering the dataset, Recall is not calculated.

$$Recall\ @\ N = \frac{\#\ of\ Relevant\ Items\ at\ N}{\#\ of\ Relevant\ Items}$$

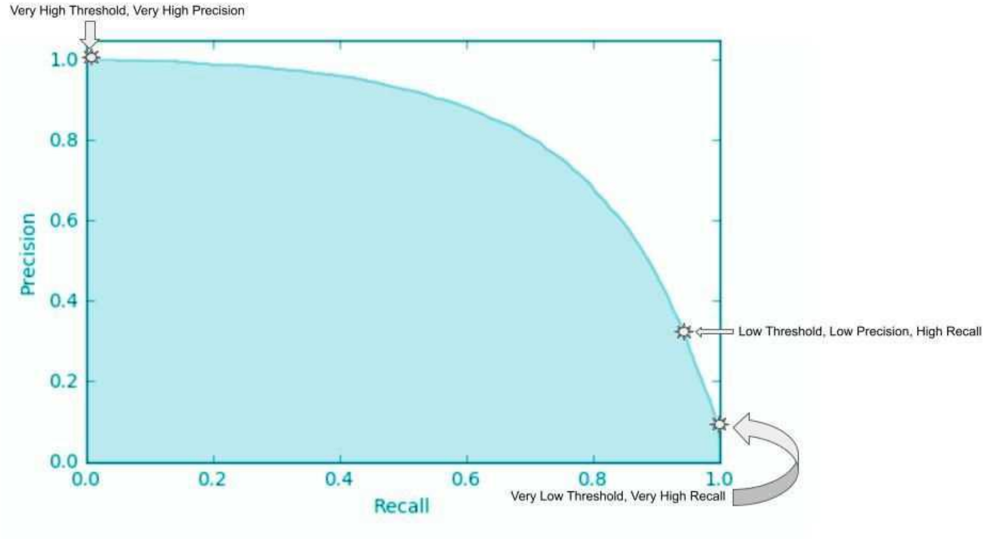


Figure 27: Precision-Recall Curve

When determining performance against the historical users, the quantitative metrics in Section 5.1 are used with the primary metric Precision at N (as the order of recommendations are not known based only on a library). When performing the user study and evaluation, the Qualitative metrics are the primary metrics alongside more ‘user focused’ quantitative metrics such as Hit Rate, MRR and MAP. Finally, another metric used for comparing the different algorithms is as Time to Recommendation below.

5.3.1 Time to Recommendation

Whilst not a measure of how good the recommendations are, time to recommendation is a performance measure which helps determine whether the system is fit for purpose. It is simply how long it takes from the time the user submits their profile information until the user is presented with the N recommendations. As the system needs to run in near-real time, it would imply a poor user experience if the recommender was slow. This is not defined by a particular ‘cut-off’ value for slowness, instead being whether the system is noticeably slow to the user (or when compared to other methods).

5.4 User Study

The final aspect of the evaluation is a user study. The purpose of the user study is to both test the system against real world users as well as generate qualitative feedback on the recommenders themselves to answer the fundamental question “would somebody use this”? The user study is broken into the following phases:

5.4.1 Phase 0: Evaluation Dataset

Before the user study is conducted, the recommender system is used by the development team internally. This is to simulate how the system would perform for a real user and acts as integration testing. Whilst not actually a user study, the results here can still be used from a qualitative perspective to evaluate performance. This is conducted using the Evaluation dataset as described in 4.2.3.

5.4.2 Phase 1: Basic User Study

Here an actual user study is conducted using associates of the development team. This is where several distinct users are provided the system and asked to use it. Unlike traditional user testing this will not involve an undirected approach as no UI is planned for the initial release. Instead, the user will be asked questions whilst the developer inputs the responses to drive the recommender system in a Jupyter notebook. At the end a series of recommendations will be provided for each of the models and the user asked which games were relevant. This will help generate both the qualitative and quantitative metrics.

5.4.3 Phase 2: User Survey

The final part of the user study is to provide a series of questions to the user to help generate more detailed feedback. Whilst the survey will likely be refined during implementation a sample set of questions are provided below. Most of the questions are directed towards the Knowledge-Based recommender, which is harder to generate quantitative recommendations for, instead stopping when the user is satisfied.

1. Were you recommended games that you knew you wanted to play?
2. Did you find any surprise recommendations that you now want to play?
3. Did the recommender exceed your initial expectations?
4. Which model did you find had the most valuable recommendations?
5. Were the features presented in the final recommendation set meaningful?
6. Were there any features that you thought added no value?
7. Were there any features that you felt were missing during critiquing?
8. Would you use this system if it was developed towards a production standard (i.e. addition of user interface).?
9. Was the use of friend-based recommendation more useful than general popularity?
10. Are there any unsupported platforms you would like to see added?

5.5 Metric Assumptions and Constraints

The use of these metrics relies on several assumptions and constraints including:

1. Time to Recommend is related to hardware performance and will run differently on different devices. This is why it is considered qualitative or comparative only.
2. The definition of relevance is subjective and can affect metric quality. It is expected that relevance could generate different results depending on the user's mood.
3. The number of friends the user has will affect the quality of social based recommendations including the idea of how much a user 'trust' the network. Note that trust propagation may be used if the model includes friends of friends.
4. General Popularity metrics depend on users of the IGDB or Steam. This reduces the quality of popularity as larger userbases ratings are not available (only games they have purchased).

6. Project Implementation

This section deals with the proposed timeline for the project. The deadline for the project is defined in the Project Specification as Sunday Week 10 (17:00, 10th August 2025), whilst a presentation is required on Tuesday Week 10 (5th August). This section is based on these deadlines. Finally, an individual Project Evaluation is due on Friday Week 11 (17:00 15th August 2025)

6.1 Project Deliverables

The required deliverables for the project (as per the Project Specification) are shown in Table 10.

Table 10: List of Project Deliverables

Deliverable	Format	Description
Main Notebook	Jupyter Notebook	Contains the primary code for the system including generating inputs and presenting outputs to the user.
Additional Modules	Jupyter Notebook	1 Notebook per recommendation algorithm which takes the inputs (provided as external data) and generates recommendations into an external format (i.e. JSON)
Presentation	Jupyter Notebook, PowerPoint or PDF File	Presentation delivered on Tuesday Week 10.
Project Evaluation	PDF	Individual Report as Per the Project Evaluation Specification
Team Member Contributions	Moodle Form	A peer assessment of each users contribution.

6.2 Project Milestones

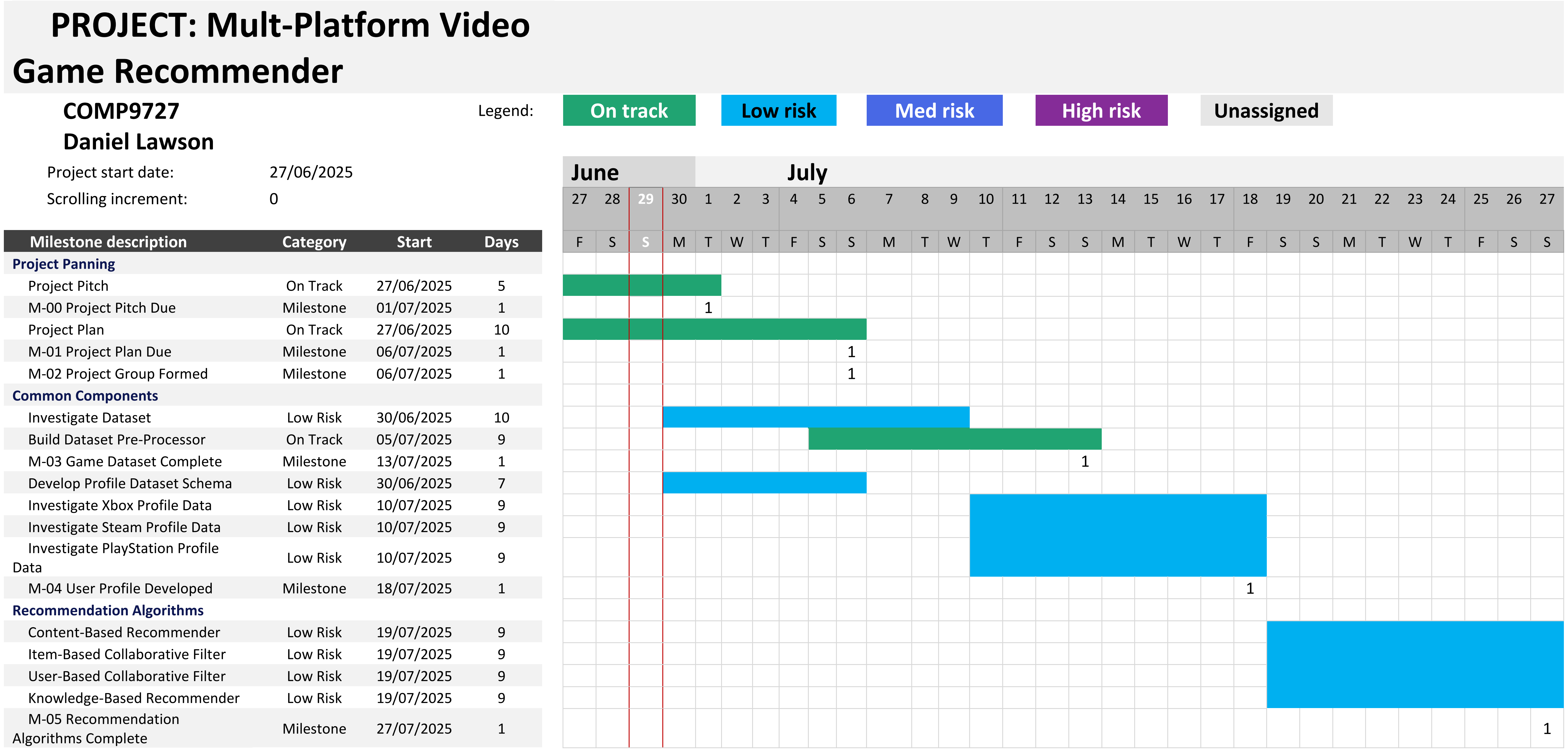
A list of Project Milestones is provided in Table 11. These are then used along with the deliverables above to generate the Project Timeline in Section 6.3

Table 11: Project Milestones

ID	Name	Date	Description
M-00	Pitch Compete	01 July 2025	Pitch delivered to class
M-01	Project Group Formed	06 July 2025	Group Formed for Project
M-02	Project Plan Submitted	06 July 2025	Submission of this document.
M-03	Game Dataset Complete	13 July 2025	Local dataset visualised with features extracted.
M-04	User Profiles Developed	18 July 2025	User Profile process complete and data added
M-05	Recommendation Algorithms Complete	27 July 2025	Recommendations algorithm complete
M-06	User Study Complete	3 August 2025	Users have completed feedback against completed model
M-07	Presentation Complete	5 August 2025	Presentation delivered to class
M-08	Implementation Submitted	10 August 2025	Code submitted
M-09	Evaluation Compete	15 August 2025	Final report submitted

6.3 Project Timeline

A project timeline has been generated using the milestones listed in Table 11, as displayed in Figure 28 and Figure 29.



PROJECT: Mult-Platform Video Game Recommender

COMP9727

Daniel Lawson

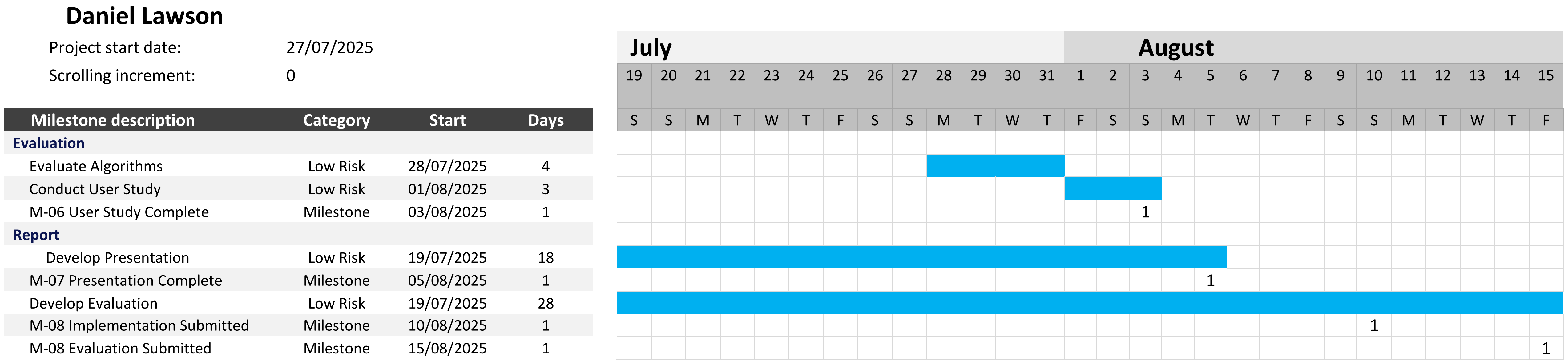
Legend: On track | Low risk | Med risk | High risk | Unassigned

Project start date: 27/06/2025

Scrolling increment: 0

Milestone description	Category	Start	Days
Project Panning			
Project Pitch	On Track	27/06/2025	5
M-00 Project Pitch Due	Milestone	01/07/2025	1
Project Plan	On Track	27/06/2025	10
M-01 Project Plan Due	Milestone	06/07/2025	1
M-02 Project Group Formed	Milestone	06/07/2025	1
Common Components			
Investigate Dataset	Low Risk	30/06/2025	10
Build Dataset Pre-Processor	On Track	05/07/2025	9
M-03 Game Dataset Complete	Milestone	13/07/2025	1
Develop Profile Dataset Schema	Low Risk	30/06/2025	7
Investigate Xbox Profile Data	Low Risk	10/07/2025	9
Investigate Steam Profile Data	Low Risk	10/07/2025	9
Investigate PlayStation Profile Data	Low Risk	10/07/2025	9
M-04 User Profile Developed	Milestone	18/07/2025	1
Recommendation Algorithms			
Content-Based Recommender	Low Risk	19/07/2025	9
Item-Based Collaborative Filter	Low Risk	19/07/2025	9
User-Based Collaborative Filter	Low Risk	19/07/2025	9
Knowledge-Based Recommender	Low Risk	19/07/2025	9
M-05 Recommendation Algorithms Complete	Milestone	27/07/2025	1

Figure 28: Project Timeline Part 1



Daniel Lawson

Project start date: 27/07/2025

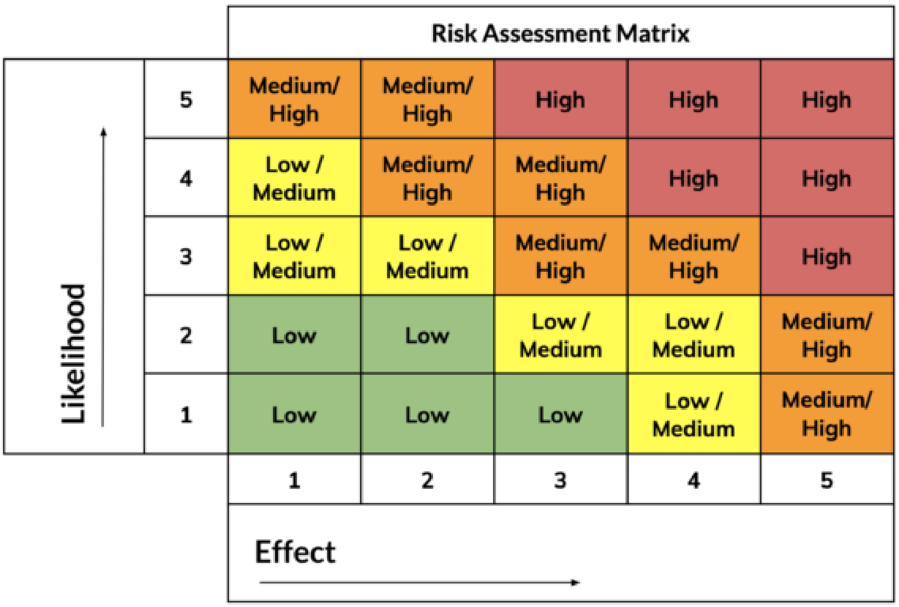
Scrolling increment: 0

Milestone description	Category	Start	Days
Evaluation			
Evaluate Algorithms	Low Risk	28/07/2025	4
Conduct User Study	Low Risk	01/08/2025	3
M-06 User Study Complete	Milestone	03/08/2025	1
Report			
Develop Presentation	Low Risk	19/07/2025	18
M-07 Presentation Complete	Milestone	05/08/2025	1
Develop Evaluation	Low Risk	19/07/2025	28
M-08 Implementation Submitted	Milestone	10/08/2025	1
M-08 Evaluation Submitted	Milestone	15/08/2025	1

Figure 29: Project Gantt Chart Part 2

6.4 Risks

Several risks have been identified for the successful completion of the assignment, these are listed in Table 12, along with a severity, likelihood and mitigation. The likelihood and severity are rated against the 5x5 matrix in Figure 30.



Risk Assessment Matrix

Likelihood					
5	Medium/ High	Medium/ High	High	High	High
4	Low / Medium	Medium/ High	Medium/ High	High	High
3	Low / Medium	Low / Medium	Medium/ High	Medium/ High	High
2	Low	Low	Low / Medium	Low / Medium	Medium/ High
1	Low	Low	Low	Low / Medium	Medium/ High
	1	2	3	4	5
	Effect				

Figure 30: Risk Matrix (Ref V)

Table 12: Project Risks

ID	Description	Likelihood	Severity	Mitigation
R-01	Unable to Access IGDB	1	5	Download IGDB database to local format (i.e. CSV)
R-02	Unable to Access Profile API	3	3	Generate the data manually via alternative sources.
R-03	Unable to Access Friend Data	2	4	Replace the model with an alternative model.
R-04	Inability to Identify Suitable Users	2	4	Identify users early in project phase to schedule in available time. Can use other groups if needed.
R-05	Inability to Generate Metrics	3	2	Replace metrics with more suitable calculates.
R-06	Insufficient Hardware to perform ML Calculations	2	2	Scale back the dataset or identify a suitable machine (this may include Google Collab)
R-07	Group Member Unable to perform Contributions	2	3	Scope Project to allow for independent and parallel work.
R-08	Unable to Get Suitable Recommendations	3	3	Re-scope requirements to ensure suitable recommendations are possible.

7. References

A. United States Department of Defense (1999). *Data Item Description: Software Requirements Specification*. DI-IPSC-814331.

B. United States Department of Defense (1999). *Data Item Description: Software Design Description*. DI-IPSC-814351

C. Steam (2014). *The Discovery Update*. Available: https://store.steampowered.com/about/newstore. Date Accessed: 28/06/2025

D. Wolens, J (2022). *Valve trials a new take on Steams Discovery Queue*. Available: https://www.pcgamer.com/valve-trials-a-new-take-on-steams-discovery-queue/ Date Accessed: 28/06/2025.

E. Laws of UX (2025). *Miller's Law*. Available: https://lawsofux.com/millers-law/. Date Accessed: 28/06/2025.

F. BigCommerce (2024). *Affiliate Marketing 101: What it is and How to Get Started*. Available: https://www.bigcommerce.com.au/articles/ecommerce/affiliate-marketing/. Date Accessed: 28/06/2025.

G. MegaLag (2025). *Exposing the Honey Influence Scam (Video)*. Available: https://www.youtube.com/watch?v=vc4yL3YTwWk. Date Accessed: 28/06/2025.

H. SteamDB (2025). *Steam Stats – Releases*. Available: https://steamdb.info/stats/releases/. Date Accessed: 28/06/2025.

I. Amazon (2025). *IGDB*. Available: https://www.igdb.com/about. Date Accessed: 28/06/2025.

J. Amazon (2025). *IGDB API*. Available: https://api-docs.igdb.com/#getting-started. Date Accessed: 28/06/2025.

K. Pykes, K (2024). *Getting Started with Python HTTP Requests for REST APIs*. Available: https://www.datacamp.com/tutorial/making-http-requests-in-python. Date Accessed: 28/06/2025.

L. *Kruglov, A (2025). Gaming Profiles 2025 (Steam, PlayStation, Xbox)*. Available: https://www.kaggle.com/datasets/artyomkruglov/gaming-profiles-2025-steam-playstation-xbox. Date Accessed: 01/07/2025

M. OpenXbox (2025). *OpenXBL API Console*. Available: https://xbl.io/console. Date Accessed: 29/06/2025.

N. Valve (2025). *Steam Web API Documentation*. Available: https://steamcommunity.com/dev. Date Accessed: 29/06/2025.

O. Valve (2025) *Steam Web API*. Available: https://developer.valvesoftware.com/wiki/Steam_Web_API. Date Accessed: 29/06/2025.

P. Andshrew (2024). *PlayStation Trophies API v2*. Date Accessed: 29/06/2025.

Q. MobyGames (2025). *MobyGames API*. Available: https://www.mobygames.com/api/subscribe/. Date Accessed: 28/06/2025.

R. Aggarwal, C (2016) *Recommender Systems. Vol 1: Cham: Springer International Publishing*.

S. *Ckatzorke (2023). HowLongToBeat API*. Available: https://github.com/ckatzorke/howlongtobeat. Date Accessed: 29/06/2025.

T. *Valve (2025). Steam Refunds*. Available: https://store.steampowered.com/steam_refunds/ Date Accessed: 29/06/2025.

U. *Evidently (2025)*. *10 Metrics to Evaluate Recommender and Ranking Systems*. Available: https://www.evidentlyai.com/ranking-metrics/evaluating-recommender-systems. Date Accessed: 29/06/2025.

V. *Veriforce (2023)*. *How to Use a 5x5 Risk Assessment Matrix*. Available: https://www.chas.co.uk/blog/how-to-use-a-5x5-risk-assessment-matrix. Date Accessed: 29/06/2025.

W. Sinclair, B (2023). *Study: 47% of Gamers Play on Multiple Platforms*. Available: https://www.gamesindustry.biz/study-47-of-gamers-play-on-multiple-platforms. Date Accessed: 02/07/2025.

8. Glossary

AAA Game	Major release with a large budget
API	Application Programming Interface
CF	Collaborative Filtering
CNF	Content-Based Filtering
DCG	Discounted Cumulative Gain
DLC	Downloadable Content
IGDB	Internet Games Database
MAP	Mean Average Precision
ML	Machine Learning
MRR	Mean Reciprocal Rank