Tribhuvan University

Academia International College

Final Year Project Report
On
Game Recommendation System (GRS)
[CSC 412]

Under the supervision of

"Er. Ganesh Ram Suwal"

Submitted by

Ritic Raj Byanjankar (T.U. Exam Roll No. 26506/077)

Ronil Maharjan (T.U. Exam Roll No. 26509/077)

Raj Budha (T.U. Exam Roll No. 26503/077)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

Game Recommendation System (GRS)

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Ritic Raj Byanjankar (T.U. Exam Roll No. 26506/077)

Ronil Maharjan (T.U. Exam Roll No. 26509/077)

Raj Budha (T.U. Exam Roll No. 26503/077)

Submitted to

Department of Computer Science and Information Technology Academia International College

Institute of Science and Technology

Tribhuvan University

January, 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Mr. Ritic Raj Byanjankar (26506/077), Mr. Ronil Maharjan (26509/077), and Mr. Raj Budha(26503/077) entitled "Game Recommendation System (GRS)" be accepted as fulfilling in partial requirements for the degree of Bachelors of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

.....

Er. Ganesh Ram Suwal

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Gwarko, Lalitpur

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Mr. Ritic Raj Byanjankar, Mr. Ronil Maharjan, and Mr. Raj Budha entitled "Game Recommendation System (GRS)" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Er. Ganesh Ram Suwal Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We owe our most profound appreciation to Academia International College for giving us a chance to work on this project as part of our syllabus.

Special thanks to our supervisor, Er. Ganesh Ram Suwal(Academia International College), for his consistent guidance, support, and feedback throughout the report's creation. We are generously obligated to him for providing this excellent opportunity to expand our knowledge. It helped us a lot to realize what we studied for.

We would like to express our sincere gratitude to all those individuals, families, friends, colleagues, and teachers for supporting and helping us a lot in finalizing this project within the limited time frame by providing valuable insights and feedback on the report.

Thanking You,

Ritic Raj Byanjankar (T.U. Exam Roll No. 26506/077)
Ronil Maharjan (T.U. Exam Roll No. 26509/077)
Raj Budha (T.U. Exam Roll No. 26503/077)

Abstract

The Game Recommendation System (GRS) is a system that recommends the game similar to the user's query or based on the games added to the user's library. The exponential growth of the gaming industry has created a dilemma for gamers to choose games that suit their preference. GRS addresses this issue by using Porter Stemming Algorithm, TF-IDF vectorization and Cosine Similarity algorithm to recommend games, tailored based on the user-criteria such as game titles, descriptions and ratings.

The system has been developed using Python-Django and ReactJS utilizing PostgreSQL to provide dynamic and user-friendly interface. The GRS is a valuable tool that enhances the gaming experience of the users by simplifying the decision-making process for gamers.

GRS is developed using the Agile Software Development methodology emphasizing on collaboration, customer satisfaction and continuous improvement. With a dynamic and user friendly interface, the GRS aims to familiarize gamers with the recommendation system.

Keywords: Game Recommendation System, Python-Django, ReactJS, Cosine Similarity, Porter-Stemming Algorithm, Cosine Similarity Algorithm

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
DFD	Data Flow Diagram
DRF	Django Rest Framework
EDA	Exploratory Data Analysis
ER	Entity Relationship
GRS	Game Recommendation System
IDF	Inverse Document Frequency
NLTK	Natural Language Toolkit
TF	Term Frequency
UI	User Interface
UX	User Experience
VS Code	Visual Studio Code

Chapter 1: Introduction

1.1. Introduction

The Game Recommendation System (GRS) is a system focusing on the game enthusiasts out there. The gaming industry has been growing rapidly making it incredibly difficult for both the players and the game company to keep track of player's preference of the games. The project mainly focuses on the part of the player's dilemma. Often players encounter a hard to make decision; i.e. Which game should I play?

The GRS aims to help players gain a keen insight on the game's that match their preferences. The project will allow the user to find games that match the user's preferences. The user will be able to get a tailored set of game recommendations based on their own custom interests and will also be able to search for genre-specific games. Thus, the project aims to facilitate gamers, game-enthusiasts, and online content creators in making the decision of which games to play.

1.2. Problem Statement

The GRS is mainly aimed to help gamers and potential content creators. While it is convenient for its users it may not be useful for a casual user. The system focuses on the following problems:

- With the growth of the gaming industry there are millions of games available for players to pick from.
- Casual gamers feel frustrated while trying to find games they enjoy.
- Choosing a paid game can end up being a letdown.
- Help streamers and content creators find a popular game for their audience

1.3. Objectives

The objective of GRS include:

- To provide a platform where users can find games based on their preferences.
- To provide a user-friendly interface that allows users to easily interact with recommendation systems.
- To allow users to register and log in to the system.
- To provide recommendations based on the user query.

1.4. Scope and Limitation

1.4.1. Scopes

The scopes of GRS include:

- User Registration: User's will be able to register and create a profile.
- User-Friendly Interface: The interface will be simple and easy to navigate through.
- Popular games: The user's will be able to see the popular games
- Description Based Recommendation: User will be able to get recommendation based on description of game.
- Title Based Recommendation: User will be able to get recommendation based on the title of game.
- Rating Based Recommendation: User will be able to get the recommendation based on the rating (average playtime) of the game.
- Genre-based filter: User will be able to choose genre that match their tastes or whims.

1.4.2. Limitations

The limitations of GRS include:

- The system needs a continuous internet connection to function.
- The system will only be able to provide the recommendations based on the games present in the dataset.
- The system cannot record user's play history so the recommendation is only made via manual input.

1.5. Development Methodology

The development methodology used for this project is Agile Software Development. Agile is a flexible and iterative approach to project management and software development that emphasizes collaboration, customer satisfaction and continuous improvement.[1] It is based on the Agile manifesto which is a set of principles for software development that prioritizes individuals and interactions, working software, customer collaboration and responding to change.

We adopted Agile methodology for our project due to its flexibility and iterative development approach. We implemented Kanban which is a visual project management

framework within Agile. It helped us to visualize tasks and progress using Kanban board divided into columns representing different stages of task completion like “To Do”, “In Progress” and “Completed” to ensure smooth workflow.

In the initial phase, we discussed project objective and identified different requirements with the help of our supervisor. We then developed the design plan based on the necessities and the requirements of the project. After which the functionality of the system was designed and implemented. Various testing methods were also used to test the system.

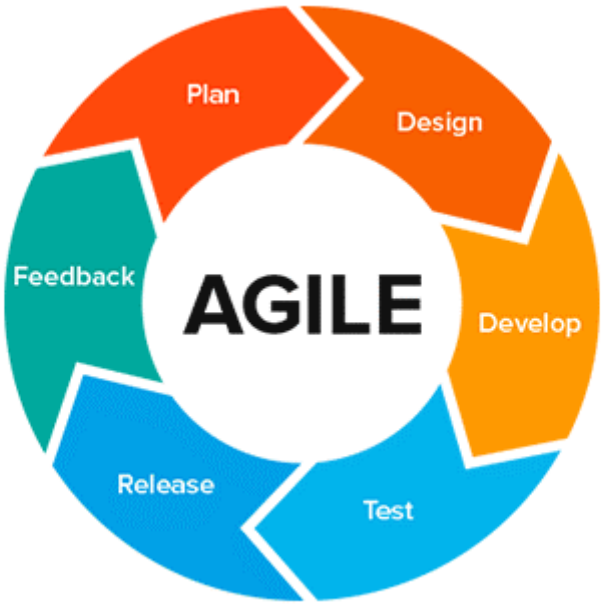


Figure 1.1. Agile Development Methodology

1.6. Report Organization

The main report is organized into 6 chapters, aligning with their respective headings and content, which include:

Chapter 1: Introduction

The chapter: introduction gives a general overview of the project along with the problem statement, objective, scope, limitation and the methodologies used.

Chapter 2: Background Study and Literature Review

It addresses the detailed background of the project and performed the analysis of the existing literature and similar systems building the knowledge and connecting it to the current technology.

Chapter 3: System Analysis

It covers the detailed analysis of the system including requirement analysis and the feasibility analysis. Various functional and non-functional requirements are studied. Gantt chart is used to visualize the schedule and progress of the project. The Exploratory Data Analysis (EDA) is also performed on the dataset.

Chapter 4: System Design

It covers the design of the architecture, the user interface, forms and the database schema. This chapter also covers the algorithms used for the system.

Chapter 5: Implementation and Testing

It provides the implementation details of the system using specific tools and technologies. This chapter also covers the testing process highlighting various testing methods used such as unit testing, integration testing as well as system testing.

Chapter 6: Conclusion

It summarizes the project highlighting the key achievements and suggesting future enhancements for the project.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In recent years, the gaming industry has seen substantial growth with online gaming platforms like Steam and Epic Games Store becoming central hub for gamers worldwide. These platforms provide a vast and growing library of games across multiple genres. With the increase in number of games, it becomes a challenge for gamers to choose suitable game aligning with their interest.

To tackle this issue, Game Recommendation System have been introduced as a critical tool to help gamers choose suitable games across vast selection of games which suit their taste[2]. These systems employ various technique such as collaborative filtering, gamification, machine learning algorithms and data analysis techniques to improve accuracy of recommendation and enhance the dynamic user experience. These algorithms consider various factors such as player count, ratings, genres and user feedback to generate personalized recommendations.

This project uses publicly available dataset from Kaggle that includes detailed information about games including attributes like game IDs, title, reviews and descriptions to recommend games to the user implementing cosine similarity algorithm. It measures the relationship between games and users' preference to provide personalized recommendations. The system also integrates React based frontend with a Django based backend to create an interactive user experience.

In conclusion, the Game Recommendation System provides numerous benefits for both gamers and game companies as the gamers can save time searching for new games that suit their interests and game companies can use these systems to increase user engagement while creating better games that suit the interest of the users.

2.2 Literature Review

Improving the Accuracy of Online Game Recommendations: A Content-Based Filtering Approach Utilizing Cosine Similarity Matrix.[3] study explores the use of content-based game recommendation system utilizes user preferences and game attributes to recommend similar games. It utilizes cosine similarity matrix to enhance accuracy, aligning

recommendations with users' interests based on previously enjoyed games. This helped improve the overall user satisfaction.

Game Recommendation Using Content-based Algorithm Title[4] uses user’s input like his/her previously played games in order to create a user profile. It identifies similar games based on the categories, genre, etc. achieving an accuracy of 82% in fitting user interest.

A Comparative Analysis of Collaborative Filtering Models for Game Recommendation Using Cosine Similarity, SVD, K-Means Clustering and Real-Time Game Insights[5] GRS is a game recommendation system utilizing collaborative filtering techniques like Cosine Similarity, SVD, and K-means clustering to deliver personalized game recommendations based on user preferences and game attributes.

Game Recommendation System[6] This paper discusses on a game recommendation system designed to assist users in selecting suitable games from online platforms. It focuses on challenges in data collection, utilizing machine learning and data visualization for effective recommendation. The collection of data was performed by scraping all the URLs of video games.

Movie Recommendation System Using Content Based Filtering[7] This paper while not for game recommendation system utilized techniques like Count Vectorizer and TF-IDF Vectorizer to convert textual information into numeric values. Also utilizing the similarity measurement like cosine similarity. It suggests the development of hybrid system to improve accuracy and efficiency.

Recommendation System from Microsoft News Data using TF-IDF and Cosine Similarity Methods[8] The study shows that TF-IDF and Cosine similarity methods are effective in recommending news articles. Based on it and other paper that utilized these techniques in not just movies and news articles it is evident that it can be utilized to process descriptions for Game Recommendation system.

Collaborative filtering recommender systems[9] This paper shows that collaborative filtering is a powerful technique for building recommender system but faces challenges like scalability, sparsity and cold start. It suggests utilization of other techniques like content-based filtering, demographic and knowledge-based filtering to create a hybrid approach.

2.3. Study of Existing System

There are a number of existing game recommendation systems. Some of the existing systems are:

- Games Finder: The website is one of the game recommendation systems that allows users to search for similar games on their database. The website does not provide personalize game recommendation system.
- Quantic Foundry: By entering 3 game titles, this website will show a list of games that are similar with those you enter. There is also no personalized game recommendation system.
- 50gameslike: It is also one of the existing game recommendation systems that allows the user to search for similar games or filter games based on the genres. This website is using outdated dataset and takes a long time to find recommendations.

Chapter 3: System Analysis

3.1. System Analysis

In preparation for the development of the project, a detailed list of tasks was compiled, taking into account the functional and non-functional requirements of the system. This allowed for all the critical factors to be put in place from the onset. After the establishment of the requirements, the desired operation and behavior of the system were critically examined. This includes understanding the user interactions, system performance criteria, and overall architecture in relation to aligning the system design with the project's objectives.

3.1.1. Requirement Analysis

The requirements can be functional and non-functional.

i. Functional Requirements: The project considers following functional requirements.

- User registration and authentication
- Login option for registered user
- Edit profile details
- User’s game library
- Update games in game library
- Recommend game based on user query
- Recommend game based on user library
- List of popular games
- Filter based on category

ii. Non-Functional Requirements: The project considers the following non-functional requirements.

- The system was supposed to be available 24×7, in the presence of a reliable internet connection.
- The system was supposed to have an easy to navigate user interface.

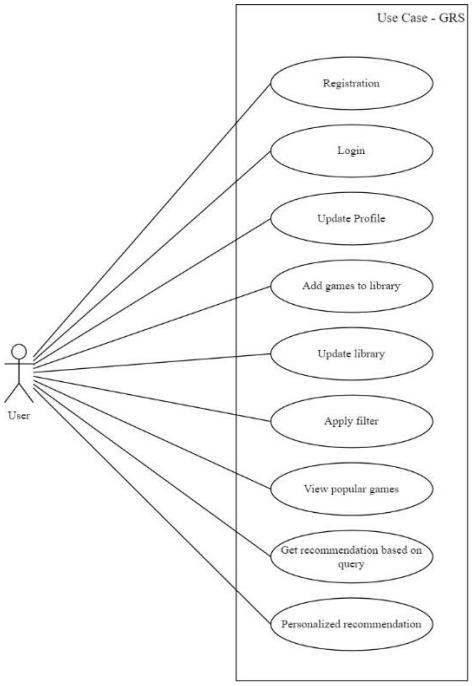


Figure 3.1. Use-Case Diagram

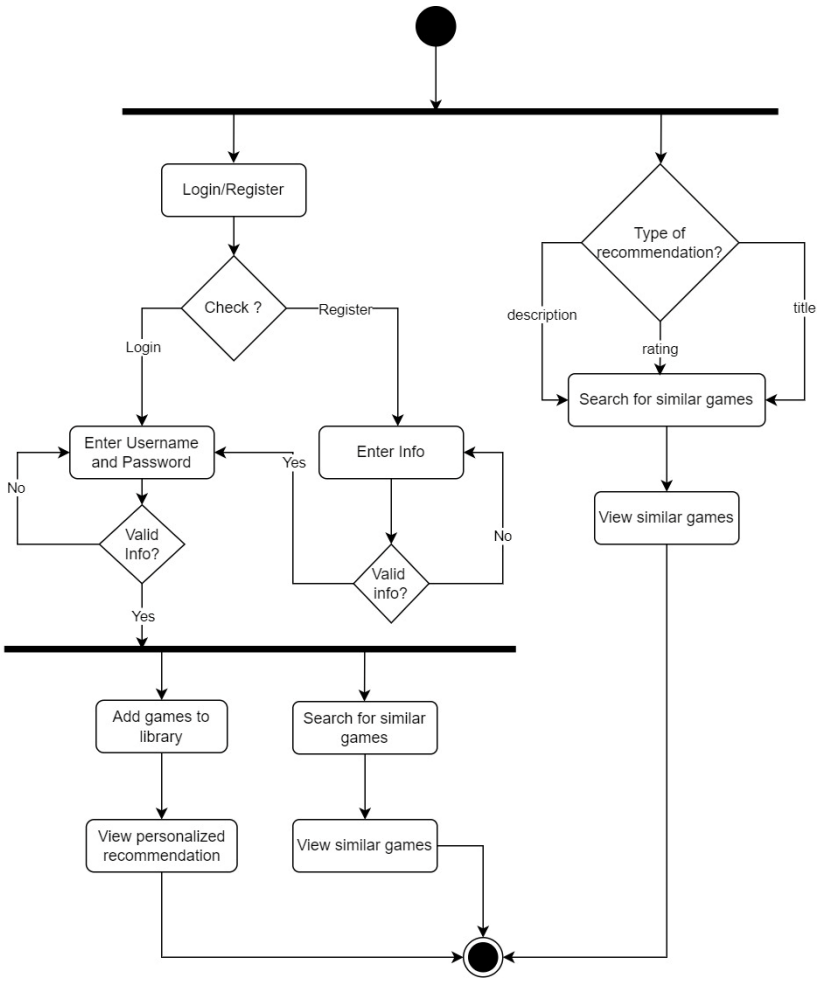


Figure 3.2. Activity Diagram

3.1.2. Feasibility Analysis

Feasibility analysis is a critical step conducted to determine whether a project is technically, economically, operationally, and schedule-wise feasible to guarantee successful project completion.

i. Technical Feasibility

The project is technically feasible because it is in line with the latest technology, including both hardware and software elements. The project utilizes Python for calculating similarities and generating logic for recommendations, Django framework for backend services and React as the frontend library all of which are free and open source. React's component-based architecture allows creation of dynamic and responsive user interfaces whilst Django provides a robust backend framework and seamless integration with DRF for API development.

ii. Operational Feasibility

The operational feasibility of the project is promising, as it aligns with both user expectations and organizational capabilities. The intuitive and responsive user interface created using React and the powerful functionality provided by Django backend can provide a smooth user experience. The project involved a team of three individuals working together as a software developer, researcher, and technical writer to ensure the project's success. Furthermore, the project can be scaled as required, allowing for future feature additions or performance optimization based on user feedback and evolving business requirements.

iii. Economic Feasibility

The developed project is very economically viable as all the software and frameworks used are free and open source (i.e., VS Code, React and Python-Django) and the hardware requirements are minimal and economically sound.

iv. Schedule Feasibility

The project consists of multiple tasks like data analysis, UI design, Backend development, Frontend development and integration. The project schedule is created with consideration to all the relevant tasks.

The following Gantt chart depicts the schedule established for the development of the system

Table 3.1. Task Schedule

Tasks	Start Date	End Date
Project Planning	20-Aug	31-Aug
Data Analysis	31-Aug	21-Sep
Research Algorithms	10-Sep	28-Sep
Backend Design	20-Sep	10-Nov
Frontend Design	10-Oct	29-Nov
Integration and Testing	15-Nov	20-Dec
Documentation	10-Sep	5-Jan

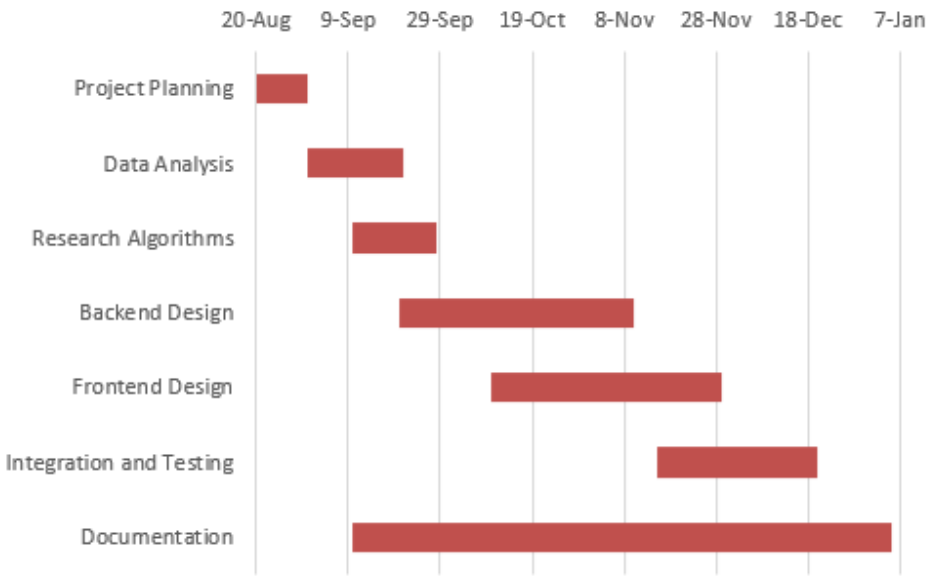


Figure 3.3. Gantt Chart

3.1.3. Data Modeling using ER Diagrams

The below Entity-Relationship (ER) diagram visually represent the relationships between entities or concepts within the developed system, providing a conceptual view of a database.

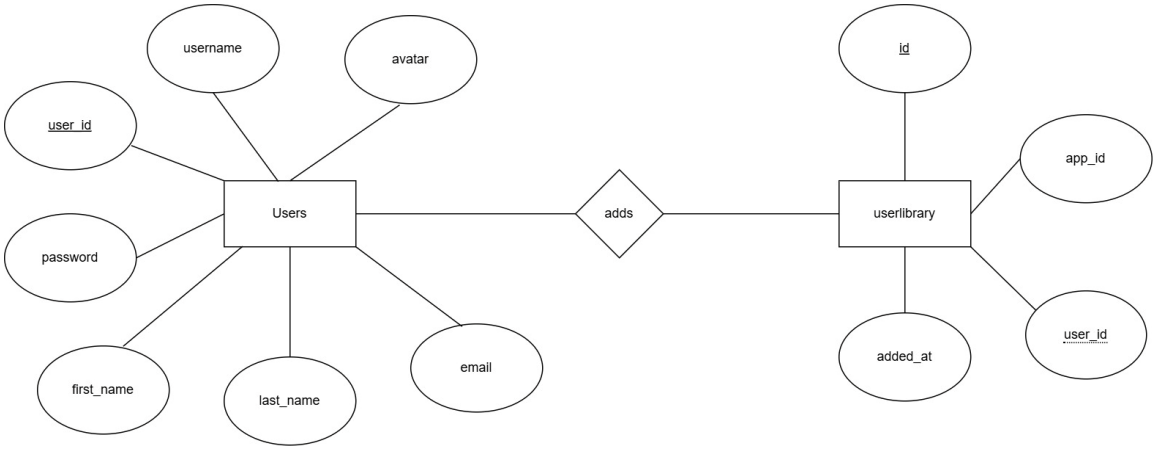


Figure 3.4. ER Diagram of GRS

3.1.4. Process Modeling using DFD

The below Data Flow Diagram (DFD) graphically depict the flow of processes used to capture, manipulate, store, and distribute data between the system and among its components.

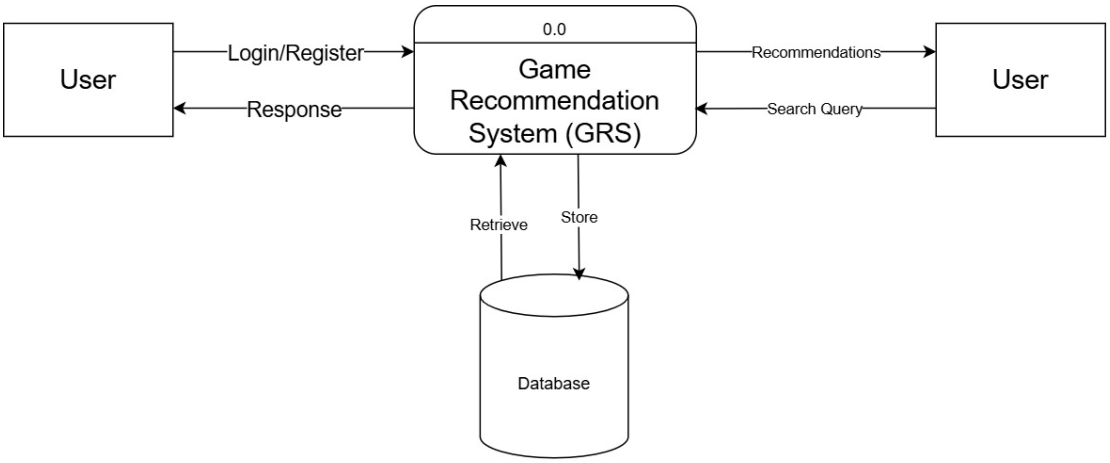


Figure 3.5. Level 0 DFD

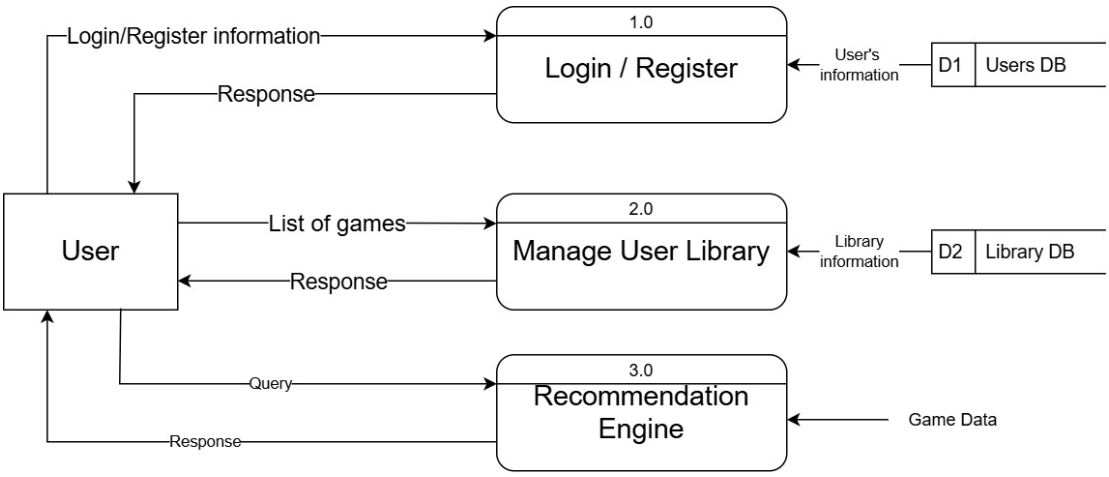


Figure 3.6. Level 1 DFD

3.1.5. Exploratory Data Analysis (EDA)

One of the major requirements of the system is having a good dataset. It is an important step that helps to understand the relations and characteristics of the dataset. The dataset was obtained from Kaggle. Its key steps include the following:

1. Understanding the data and problems:

This step is to understand the problem we are trying to solve and the data we have. Know about the attributes and variables in the data and what they represent. What type of data does it consists?

Our dataset was mostly text and consists of attributes like genre, descriptions, tags etc.

2. Importing and Inspecting the Data:

The next step is to load the dataset into a suitable data structure like DataFrames and inspect it. This involves reviewing the dataset's initial rows, checking the names of the columns, identifying the kinds of data contained within each, and looking for any visible problems.

```
games_df=reduce_memory(pd.read_csv('./final_games_data_forDesc.csv'))
games_df
```

Python

	app_id	title	About the game	Categories	Genres	Tags
0	2525270	The House	' The House ' is a short psychological horror ...	Single-player,Steam Workshop	Adventure,Casual,Indie	Adventure,Casual,Action-Adventure,Walking Simu...

Figure 3.7. Inspecting the data

3. Data cleaning:

Major operations include cleaning noisy data, removing duplicates. It ensures that the data is reliable and ready for analysis.

```
games_df['Genres']=games_df['Genres'].apply(lambda x:x.split())
games_df['Categories']=games_df['Categories'].apply(lambda x:x.split())
games_df['Tags']=games_df['Tags'].apply(lambda x:x.split())
games_df['About the game']=games_df['About the game'].apply(lambda x:x.split())
games_df['Categories']=games_df['Categories'].apply(lambda x:[i.replace(" ","") for i in x])
games_df['Genres']=games_df['Genres'].apply(lambda x:[i.replace(" ","") for i in x])
games_df['Tags']=games_df['Tags'].apply(lambda x:[i.replace(" ","") for i in x])

games_df
```

Figure 3.8. Performing Data Cleaning

```
(variable) preprocessed_games_df: DataFrame
r suffix removal
games_df['Categories'] + games_df['Genres'] + games_df['Tags'
preprocessed_games_df=games_df[['app_id','title','tags']]

preprocessed_games_df
```

	app_id	title	tags
0	2525270	The House	[', The, House, ', is, a, short, psychological...
1	2414640	Bite Size Terrors: Erobos Heaven	[Bite, Size, Terrors:, Are, Short, Experimenta...
2	2486670	TD Tower Defense	[Td, Tower, Defense, is, a, fun, tower, defens...

Figure 3.9. Combining dataset

Converting to string and lowering Case

```
#Converting to string and lowering Case
preprocessed_games_df['tags']=preprocessed_games_df['tags'].apply(lambda x:" ".join(x))
preprocessed_games_df['tags']=preprocessed_games_df['tags'].apply(lambda x:x.lower())
preprocessed_games_df['tags']
```

Figure 3.10. Data cleaning

```
0        ' the house ' is a short psychological horror ...
1       bite size terrors: are short experimental horr...
2       td tower defense is a fun tower defense shoote...
3       the scrap is an independent third-person shoot...
```

Figure 3.11. Cleaned Data

4. Perform Data Transformation:

Preparing the data to ensure it is in correct format. In this case ensuring the descriptions/tags used for the system is consistent. The goal is to have the data be used effectively for building the recommendation system.

It can involve steps like stemming using tools like Porter-Stemmer.

```
from nltk.stem.porter import PorterStemmer
ps= PorterStemmer()

def stem(words):
    porterStemmed_words=[]
    for word in words.split():
        porterStemmed_words.append(ps.stem(word))
    return " ".join(porterStemmed_words)



preprocessed_games_df['tags']=preprocessed_games_df['tags'].apply(stem)
preprocessed_games_df
```

Figure 3.12. Using Porter Stemmer

	app_id	title	tags
0	2525270	The House	' the hous ' is a short psycholog horror indi ...
1	2414640	Bite Size Terrors: Erobos Heaven	bite size terrors: are short experiment horror...
2	2486670	TD Tower Defense	td tower defens is a fun tower defens shooter ...
3	2304650	The Scrap	the scrap is an independ third-person shooter ...
4	2519670	Wind Love	gameplay wind love - is a japanese-styl visual...
...	...	...	...

Figure 3.13. Transformed Data

```
from sklearn.feature_extraction.text import TfidfVectorizer
cv=TfidfVectorizer(max_features=5000,stop_words='english')



vectors=cv.fit_transform(preprocessed_games_df['tags'])
# vectors=cv.fit_transform(games_df['tags'])

with open('vectors_final_forDesc_tfid.pkl', 'wb') as f:
    pickle.dump(vectors, f)
```

Figure 3.14. Using TF-IDF Vectorizer

Chapter 4: System Design

4.1. Design

The Game Recommendation System (GRS) is built to allow the users to find similar games based on search input or based on the games in their library. The system design involves detailed system components and their connections. The system design was detailed outlined with the help of requirement analysis and project plan.

4.1.1. Architectural Design

The GRS uses the three-tier architecture utilizing client-server pattern. The presentation layer is designed to provide a user-friendly interface to users ensuring easy usability and accessibility. The application layer is responsible for authorizing the user, managing the library and generating the recommendation using the database.

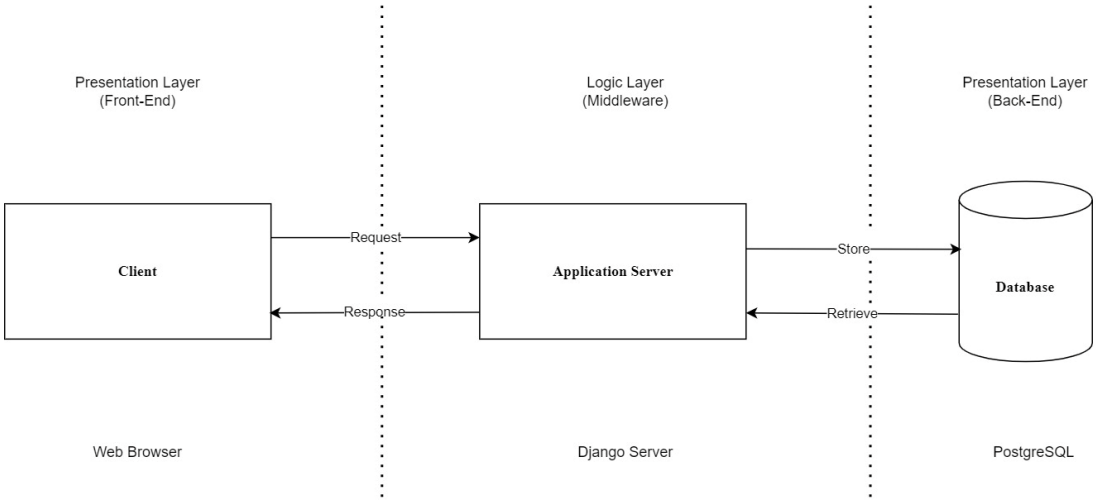


Figure 4.1. Three-Tier Architecture of GRS

4.1.2. Database Design

The GRS uses PostgreSQL database for storing and retrieving information. The effective database design is necessary to ensure that the system works as intended.

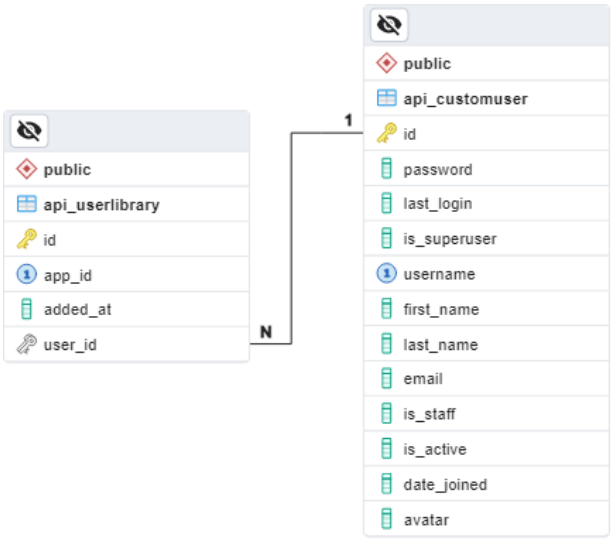


Figure 4.2. Database Schema

4.1.3. Forms and Interface Design

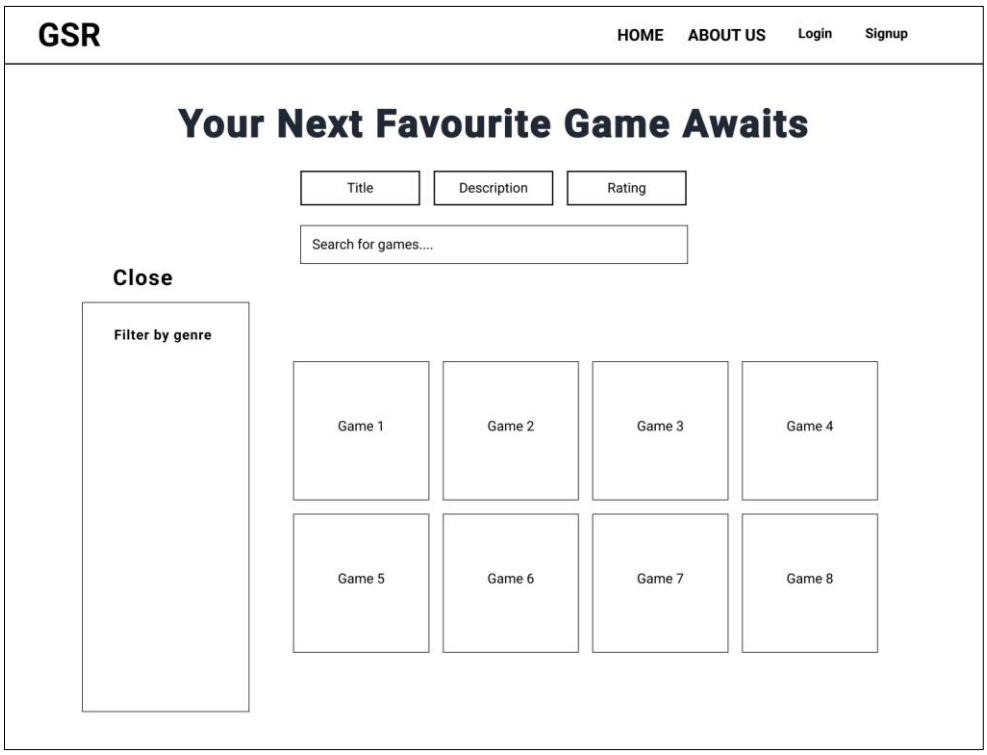


Figure 4.3. Home Page

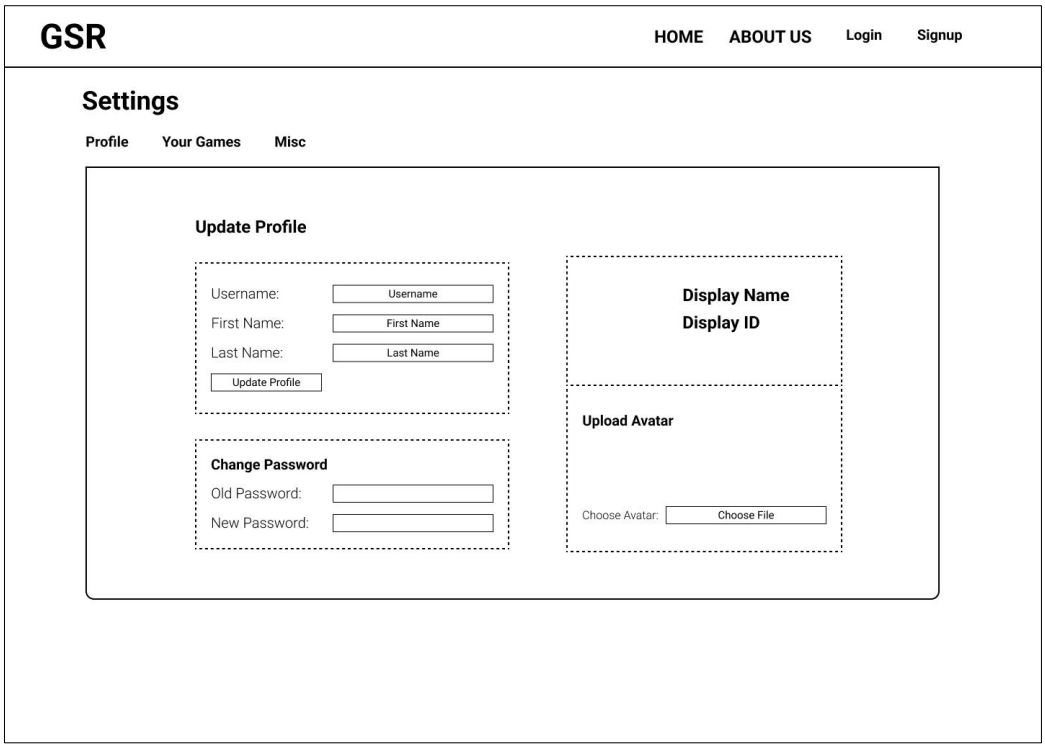


Figure 4.4. Profile Setting Page

Figure 4.5. Signup Form

Figure 4.6. Login Form

4.2. Algorithm Details

4.2.1. Porter Stemming Algorithm with TF-IDF vectorizer

We used Porter Stemming Algorithm to process the game description and tags to increase the accuracy of game recommendation. The Porter Stemming Algorithm is a process for removing suffixes from words in English in order to bring them to their basic or root forms. The main goal of stemming is to simplify and normalize vocabulary. Typically, a document is represented by a vector of words or terms. Terms with a common stem will usually have similar meanings. For example:

CONNECT

CONNECTED

CONNECTING

CONNECTIONS

The performance of system improves if similar terms are conflated into a single term. This may be done by removal of various suffixes -ED, -ING, -ION, -IONS to leave a single term CONNECT.

Algorithm:

A consonant in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. So, in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a vowel.

A consonant will be denoted by c, a vowel by v. A list ccc... of length greater than 0 will be denoted by C, and a list vvv... of length greater than 0 will be denoted by V.

Any word, or part of a word, therefore has one of the four forms:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

These may all be represented by the single form

[C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents.

Using,

(VC){m}

to denote VC repeated m times, this may again be written as

C{m}[V].

m will be called the measure of any word or word part when represented in this form. The case m = 0 covers the null word. Here are some examples:

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES, IVY.

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

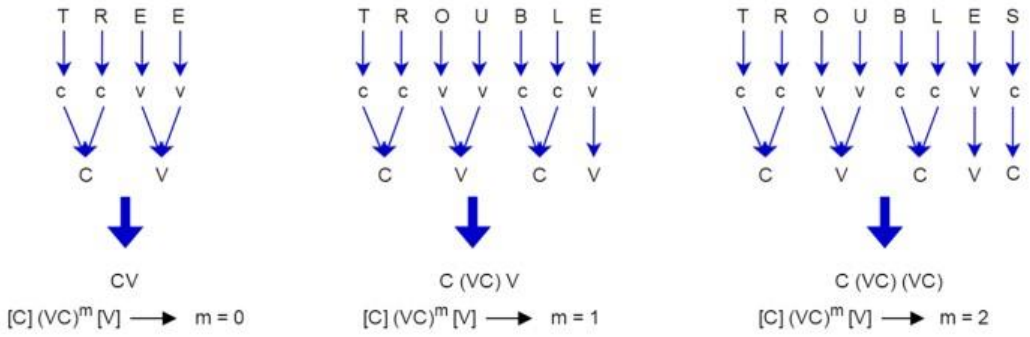


Figure 4.7. Porter Stemming Algorithm

The rules for removing a suffix will be given in the form

(condition) S1 -> S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of m, e.g.

(m > 1) EMENT ->

Here S1 is `EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which m = 2.

4.2.2. Term Frequency-Inverse Document Frequency (TF-IDF)

Term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.[10] It was invented for document search and information retrieval. We first calculated the TF-IDF matrix on the stemmed text by converting the text description to numerical vectors that shows the importance of each word in the game description.

Term Frequency (TF):

Term Frequency measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in longer documents than shorter ones.

Thus, the term frequency is calculated as:

$$TF(t,d) = \frac{Number\ of\ times\ term\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

Inverse Document Frequency (IDF):

Inverse Document Frequency measures how important a term is. While computing TF, all terms are considered equally important.

However, certain terms, like "is", "of", and "that", may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scaling up the rare ones, by computing the following:

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D: t \in d\}|}\right)$$

where:

- N is the total number of documents in the corpus D.
- $|d \in D: t \in d|$ is the number of documents where the term t appears.
- If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1+|\ d \in D: t \in d\ |$.

TF-IDF Calculation:

TF-IDF score is calculated by multiplying these two statistics:

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

4.2.3. Cosine Similarity Algorithm

Cosine Similarity is a mathematical technique used to measure the similarity between two non-zero vectors widely applied in many machine learning and data analysis application. It measures the cosine of the angle between two vectors to give an idea about how far the two vectors point in the same direction irrespective of their magnitudes.[11] The cosine similarity is calculated as:

$$\cos(x, y) = \frac{x.y^T}{||x||.||y||} = \frac{\sum_{i=1}^{n} x_i.y_i^T}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

Where,

- x.y = dot product of vectors x and y.
- ||x|| and ||y|| = magnitude of the two vectors x and y
- ||x||.||y|| = regular product of the two vectors x and y

The cosine of the angle formed by the two non-zero vectors, x and y, is what determines how comparable the two are measured. For example: If we have a game-user matrix for review, the cosine similarity would provide us a matrix in which each game's similarity score is obtained with every other game. A content-based filtering strategy constructs a cosine similarity matrix to measure game similarities based on attributes, significantly improving recommendation accuracy compared to traditional methods.[3]

Chapter 5: Implementation and Testing

5.1. Implementation

Implementation of the Game Recommendation System required accomplishing various stages from planning, algorithm research, designing UI/UX, frontend development, implementing backend with frontend, testing. For this, various tools and software were utilized.

5.1.1. Tools Used

Draw.io: Draw.io is a free and open-source diagramming tool which allows you to create diagrams like flowcharts, mockups, and network diagrams. We primarily used it to create ER-Diagrams, Use Case Diagrams, Flowcharts, etc.

Python: The core logics for the Recommendation System is implemented using python programming language. Its support for data visualization, data preprocessing and natural language processing allowed us to test and experiment various algorithms.

ReactJS: Powerful JavaScript library used to create interactive user interface.

Scikit-learn: A powerful machine learning python library used to implement TF-IDF (Term Frequency-Inverse Document Frequency) vectorization, cosine similarity,

Pandas: A popular, easy to use open-source data analysis and manipulation tool used for organizing and manipulation the game dataset.

NumPy: NumPy is a general-purpose array-processing package. It facilitates efficient similarity calculations and data manipulation.

Jupyter Notebook: Used during the development and data analysis phase to visualize data, implement prototypes of the recommendation system. Allowed us to have a proper development environment where recommendation algorithms could be improved iteratively.

GitHub: A developer platform that allows developers to create, store, manage and share code. Used for version control, collaboration and tracking changes in the development of the system.

Postman: Used to test and debug the API endpoints. It simplified the process of testing APIs by providing an interface for making requests, viewing responses, and debugging issues.

VS Code: The code editor developed by Microsoft used to write and edit the code for the project.

5.1.2. Implementation Details of Modules

Profile Manager: The Profile Manager/ User Management Module is responsible for handling user interactions with the game recommendation system. It manages the users account like registration, login, profile management, personal game library management used for personal recommendations.

Data Preprocessing Module: Main responsibility of this module is to prepare the games dataset like its title, descriptions, tags for further processing. Cleaning the dataset like removing stop words like (and, the, as) which doesn't carry significant meaning is also a part of this module. It can involve steps like tokenization and stemming using Porter Stemmer from the NLTK library to reduce words to its base form. This ensures that different forms of the same word are processed as a same token. After preprocessing, the **TF-IDF** (Term Frequency-Inverse Document Frequency) method is used to convert the text descriptions into numerical vectors. TF-IDF assigns weights to words in game descriptions, allowing the system to identify significant terms that resonate with user interests.[8]

Recommendation Module: The Recommendation module is the core of the system which is responsible for generating recommendations based on the similarity. User Inputs a game title, the recommendation algorithm calculates the **cosine similarity** between the TF-IDF vector of the input game and all other games in the dataset. By calculating the cosine similarity between TF-IDF vectors of games, the system can recommend games that are contextually similar to those a user has previously enjoyed.[12]

5.2. Testing:

Testing is an iterative process carried out in conjunction with implementation. It takes important measures to check the quality, performance, or reliability of software products before it is released. It is a common way to check if the software product meets the user requirements, Business Requirement Specification, and System Requirement Specification. It can be manual or automated.

Some important Levels of Testing include Unit Testing, Integration Testing, System Testing, etc. These tests are performed during software development for proper implementation of the required features. Various test cases are made and the results are analyzed, returning those found with issues back to the development team.

5.2.1. Unit Testing:

In unit testing, we test the smallest functional unit of code and ensure they behave as intended.

In our game recommendation system project, we used the following Test Cases:

Table 5.1. Test Cases for User Registration

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	User Tries to register with invalid inputs(email) & leaves the required field blank.	Username: john FirstName: john LastName:____ Email: john@gmail Pass: john Confirm Pass: john	Signup Failed. The user is made aware of the required field and necessary changes to the email.	Same as the Expected Result	Pass
2	The user tries to register with valid inputs.	Username: john FirstName: John LastName: Doe	Signup Successful.	Same as the	Pass

		Email:john@gmail.com Pass: john Confirm Pass: john	The User is redirected to the login page.	Expected Result	

Table 5.2. Test Cases for User Login

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	User Login with Valid Inputs	Email: root@gmail.com Password: root	Login successfully and redirect to Homepage	Same as the Expected Result	Pass
2	User Login with Invalid Inputs	Email: rot@gmail.com Password: root	Login Unsuccessful and display the appropriate error message	Same as the Expected Result	Pass

Table 5.3. Test Case for Mode Selection

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	The user selects the mode of recommendation	Chooses Description.	Selection Changes to Description	Same as the Expected Result	Pass

Table 5.4. Test Case for getting recommendations

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	The user inputs the game title to get recommendations based on it.	Counter Str	Suggestions Matching what the user has typed should pop up.	Same as the Expected Result	Pass
2	The user selects one of the suggestions and presses the search icon.	Counter-Strike	Recommendations based on Counter-Strike are listed.	Same as the Expected Result	Pass

Table 5.5. Test Cases for the filters

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	The user selects the available filter and presses the filter button	Strategy check box	Strategy Games should be filtered and displayed to the user	Same as Expected Result	Pass
2	The user presses the reset button		Filter should be removed and all games should be displayed.	Same as Expected Result	Pass

5.2.2. Integration Testing:

Testing the interface between two software modules or units is known as integration testing. Its main goal is to determine whether the interface is correct.

Table 5.6. Test Cases for Integration Testing to get and filter recommendations

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	The user inputs the game title to get recommendations based on it.	Call of	Suggestions Matching what the user has typed should pop up.	Same as the Expected Result	Pass
2	The user selects one of the suggestions and presses the search icon.	Call of Duty	Recommendations based on Call of Duty are listed.	Same as the Expected Result	Pass
3	The user selects the available filter and presses the filter button	Adventure check box	Adventure Games should be filtered and displayed to the user	Same as Expected Result	Pass

Integration testing of the system to log in, view user profile settings, and add played games to his/her library:

Table 5.7. Test Cases for Integration testing of logging in and adding games to library

Test ID	Test Scenario	Test Data	Expected Result	Observed Result	Test Status
1	User Login with Valid Inputs	Email: root@gmail.com Password: root	Login successfully and redirect to HomePage	Same as the Expected Result	Pass
2	The user hovers over their avatar icon and selects the settings		The user should be redirected to the Settings page	Same as the Expected Result	Pass
3	The user switches to the Your Games Tab and adds played games	Names of played games	Shows list of games to be added and upon clicking add to library saves the game to his/her profile.	Same as the Expected Result	Pass

5.2.3. System Testing:

System testing is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements. Since one of the objectives of our project is to have a responsive site, we made sure to test it on various device resolutions.

Table 5.8. Test Cases for Responsiveness of the site

Test ID	Test Scenario	Steps to Execute	Expected Result	Observed Result	Test Status
1	Verify that the layout adapts to desktop screens.	1. Open the application on a desktop. 2. Resize the browser window to 1920x1080.	Layout should render correctly with no overlapping or misalignment.	Same as the Expected Result.	Pass
2	Verify that the layout adapts to tablet screens.	1. Open the application on a tablet or emulate a tablet screen size (e.g., 768x1024).	Layout should be adjusted with proper alignment of components.	Same as the Expected Result.	Pass
3	Verify that the layout adapts to mobile screens.	1. Open the application on a mobile or emulate a mobile screen size (e.g., 375x667).	Layout should stack elements vertically with proper spacing and use alternative ways to show content.	Same as the Expected Result.	Pass

5.3. Result Analysis

Result Analysis involves evaluating the effectiveness of the game recommendation system and see if the system meets the project's goals and objectives. This section sees the outcomes of our game recommendation system through metrics like precision, recall and F1 score.

Evaluation Metrics:

Standard evaluation metrics such as Precision, Recall, and F1 Score were used for this analysis. Since our system is unsupervised, we create a ground truth for the games based on the relevance of the game to the description of the title. To calculate this, we have the following:

True Positive: If the system recommends games that are relevant and exists in ground truth it is true positive

False Positive: Games that were recommended by the system but were irrelevant and not in the ground truth.

False Negative: Games that the system did not recommend but were in ground truth. (i.e. System failed to recommend relevant games.)

Precision

It measures the proportion of recommended games that were relevant.

$$Precision = \frac{Total\ TP}{Total\ TP + Total\ FP}$$

Recall:

It is proportion of relevant games that were successfully recommended.

$$Recall = \frac{Total\ TP}{Total\ TP + Total\ FN}$$

F1 Score:

F1 score computes the average of precision and recall.

$$F1\ Score = \frac{2\ \times (Precision\ \times Recall)}{Precision\ + Recall}$$

Steps for Calculation:

1. Preparation of Ground Truth

Ground truth consists of a list of relevant games for a specific input game and is based on their descriptions. It is based on domain knowledge.

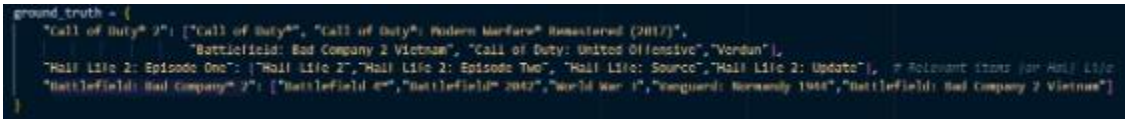


Figure 5.1. Ground Truth based on Relevance

2. Get the recommendations:

The system provides a list of recommended games.

```
def recommend(game):
    index = games[games['title'] == game].index[0]

    item_vector = vectors[index]
    similarities = cosine_similarity(item_vector, vectors).flatten()
    recommended_indices = similarities.argsort()[::-1]  # Get the indice
    game_lists=[]
    for i in recommended_indices[1:n_recommendation]:
        game_lists.append(games.iloc[i].title)
    return (game_lists)
```

Figure 5.2. Function for getting Recommendation

```
recommend("Call of Duty® 2")
✓ 0.1s
['Call of Duty®',
 'Call of Duty: United Offensive',
 'Call of Duty®: Modern Warfare® Remastered (2017)',
 'Warrior Paint - 2005 GOTY Edition',
 'Verdun',
 'Red Orchestra: Ostfront 41-45',
 'Fog Of War - Free Edition',
 'Day of Defeat: Source',
 'Battlefield: Bad Company 2 Vietnam',
```

Figure 5.3. Recommendations

3. Calculate Precision Recalls:

```python
def calculate_precision_recall(input_title, recommended_titles, ground_truth, top_n=3):
    """
    Calculate precision, recall, and other metrics for a single input title.
    """
    # Get the relevant items from the ground truth for this test item
    relevant_items = ground_truth.get(input_title, [])
    if not relevant_items:
        return {"precision": 0, "recall": 0, "true_positives": 0, "false_positives": 0, "false_negatives": 0}
    # Consider only the top N recommended items
    top_recommended = recommended_titles[:top_n]
    # Calculate true positives, false positives, and false negatives
    true_positives = len(set(relevant_items) & set(top_recommended))
    false_positives = len(set(top_recommended) - set(relevant_items))
    false_negatives = len(set(relevant_items) - set(top_recommended))
    # Calculate precision and recall
    precision = true_positives / len(top_recommended) if len(top_recommended) > 0 else 0
    recall = true_positives / len(relevant_items) if len(relevant_items) > 0 else 0
    return {
        "precision": precision,
        "recall": recall,
        "true_positives": true_positives,
        "false_positives": false_positives,
        "false_negatives": false_negatives,
    }
```

Figure 5.4. Calculation of Precision and Recall

4. Micro F1 score for multiple games:

```python
all_titles = ["Call of Duty® 2", "Half-Life 2: Episode One","Battlefield: Bad Company™ 2"]
f1_scores = []
total_tp, total_fp, total_fn = 0, 0, 0  # For micro-average
for input_title in all_titles:
    # Get recommendations
    recommended_titles = recommend(input_title)
    # Calculate precision and recall
    metrics = calculate_precision_recall(input_title, recommended_titles, ground_truth, top_n=5)
    precision = metrics['precision']
    recall = metrics['recall']
    # Calculate F1 score for this title
    if precision + recall > 0:
        f1 = 2 * (precision * recall) / (precision + recall)
    else:
        f1 = 0.0
    # Add F1 score to list for macro-average
    f1_scores.append(f1)
    # Aggregate TP, FP, FN for micro-average
    total_tp += metrics['true_positives']
    total_fp += metrics['false_positives']
    total_fn += metrics['false_negatives']
# Micro-Averaged F1
if total_tp + total_fp > 0 and total_tp + total_fn > 0:
    micro_precision = total_tp / (total_tp + total_fp)
    micro_recall = total_tp / (total_tp + total_fn)
    micro_f1 = 2 * (micro_precision * micro_recall) / (micro_precision + micro_recall)
else:
    micro_f1 = 0.0
# print(f"Macro-Averaged F1: {macro_f1}")
print(f"Precision: {micro_precision}" )
print(f"Recall: {micro_recall}" )
print(f"Averaged F1 Score: {micro_f1}")
```

Figure 5.5. Average F1 score for multiple games

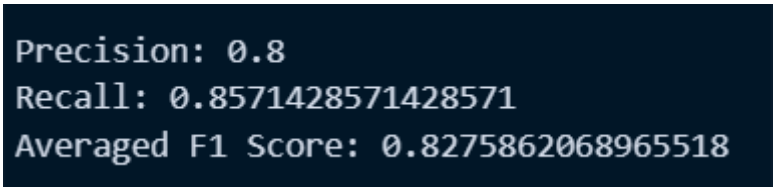


Figure 5.6. Output for average F1 Score

Chapter 6: Conclusion and Future Recommendations

6.1. Conclusion

The Game Recommendation System(GRS) achieves its goals by providing a platform that makes the process of game selection easier for the users. Using content-based filtering and complete data analysis, the system really gives accurate and relevant recommendations tailored to individual preferences. Its user-focused design, in addition to using modern web technologies, guarantees a responsive and engaging experience. While it has achieved a great deal at this time, some limitations—for instance, the dependency on a predefined data set and the absence of user play history—point to future development. Future efforts may include real-time user feedback, extending the dataset, and incorporating collaborative filtering to enhance the accuracy of recommendation. In a nutshell, GRS represents a great stride toward revolutionizing the discovery and curation of games for a heterogeneous audience.

6.2. Future Recommendations

- Integrate Collaborative Filtering algorithms to help identify patterns based on user's behaviors and preferences and improve the system accuracy.
- Feature to allow users to rate and review recommendations to improve the relevance and quality of future suggestions dynamically.
- Develop mechanisms to track and analyze user play history for personalized and evolving recommendations.
- Implement a system to update the dataset with latest games.
- Add support for multiple languages

References

[1] P. Zhigulev, “AGILE METHODOLOGY FOR MANAGING DEVELOPMENT TEAMS (EXAMPLE FROM THE GAMING INDUSTRY),” Sep. 2024, doi: 10.14293/PR2199.001064.v1.

[2] M.-C. Yuen *et al.*, “Game Recommendation System,” 2023. doi: 10.3233/FAIA231096.

[3] A. H. Imran Abd Rouf, N. Musa, and I. Ismail, “Improving the Accuracy of Online Game Recommendations: A Content-Based Filtering Approach Utilizing Cosine Similarity Matrix,” in *2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, IEEE, Dec. 2023, pp. 1–6. doi: 10.1109/ICRAIE59459.2023.10468396.

[4] Z. B. Pragusma, V. Kwandinata, J. Natthannael, Meiliana, and S. Achmad, “Game Recommendation Using Content-based Algorithm Title,” in *2023 International Conference on Informatics, Multimedia, Cyber and Informations System (ICIMCIS)*, IEEE, Nov. 2023, pp. 313–317. doi: 10.1109/ICIMCIS60089.2023.10349063.

[5] V. S. Krishna Sundaran, “A Comparative Analysis of Collaborative Filtering Models for Game Recommendation Using Cosine Similarity, SVD, K-Means Clustering and Real-Time Game Insights.” [Online]. Available: www.ijfmr.com

[6] M. C. Yuen *et al.*, “Game Recommendation System,” in *Frontiers in Artificial Intelligence and Applications*, IOS Press BV, Dec. 2023, pp. 843–857. doi: 10.3233/FAIA231096.

[7] S. Rakesh, “Movie Recommendation System Using Content Based Filtering,” *Al-Bahir Journal for Engineering and Pure Sciences*, vol. 4, no. 1, Dec. 2023, doi: 10.55810/2313-0083.1043.

[8] G. Yunanda, D. Nurjanah, and S. Meliana, “Recommendation System from Microsoft News Data using TF-IDF and Cosine Similarity Methods,” *Building of Informatics, Technology and Science (BITS)*, vol. 4, no. 1, Jun. 2022, doi: 10.47065/bits.v4i1.1670.

[9] M. Nilashi, K. Bagherifard, O. Ibrahim, H. Alizadeh, L. A. Nojeem, and N. Roozegar, “Collaborative filtering recommender systems,” *Research Journal of Applied Sciences, Engineering and Technology*, vol. 5, no. 16, pp. 4168–4182, 2013, doi: 10.19026/rjaset.5.4644.

[10] Jeevan chavan, “NLP: Tokenization , Stemming , Lemmatization , Bag of Words ,TF-IDF , POS.” Accessed: Jan. 17, 2025. [Online]. Available: https://medium.com/@jeevanchavan143/nlp-tokenization-stemming-lemmatization-bag-of-words-tf-idf-pos-7650f83c60be

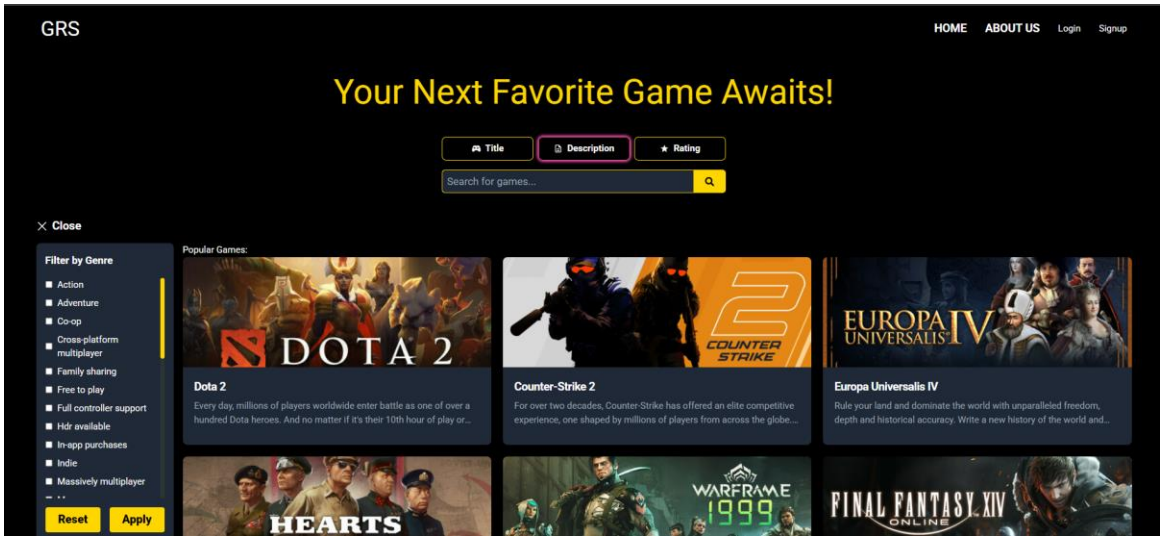
[11] geeksforgeeks, “Cosine Similarity.” Accessed: Jan. 17, 2025. [Online]. Available: https://www.geeksforgeeks.org/cosine-similarity/

[12] PV. Snigdha, M. Naveen, S. Rahul, Dr. C. N. Sujatha, and Mr. P. Pradeep, “Movie Recommendation System Using TF-IDF Vectorization and Cosine Similarity,” *Int J Res Appl Sci Eng Technol*, vol. 10, no. 8, pp. 1128–1134, Aug. 2022, doi: 10.22214/ijraset.2022.46367.

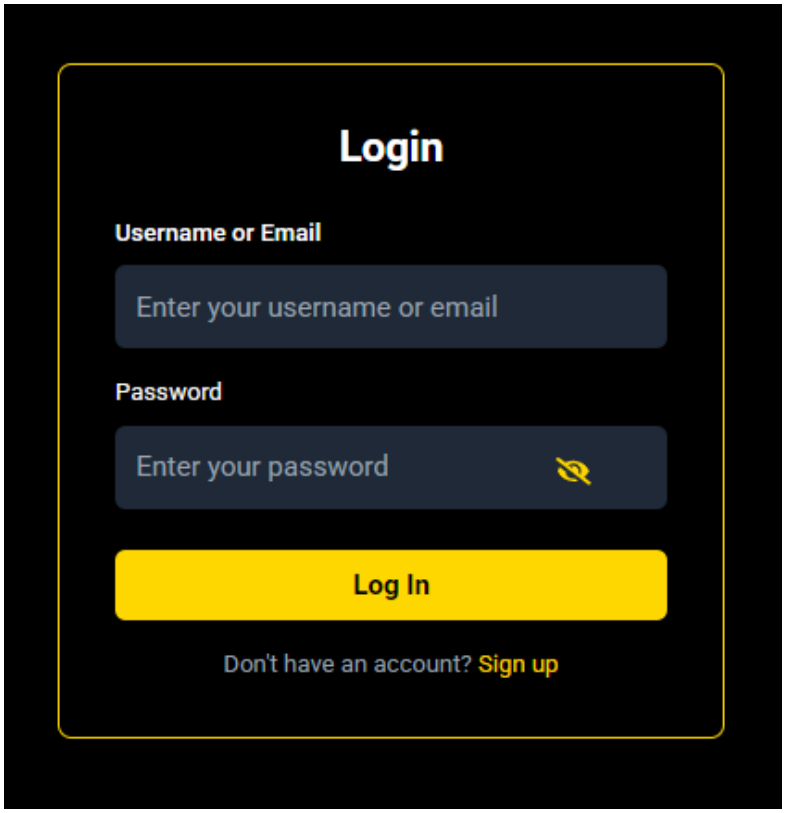
Appendices

Screenshot of the website

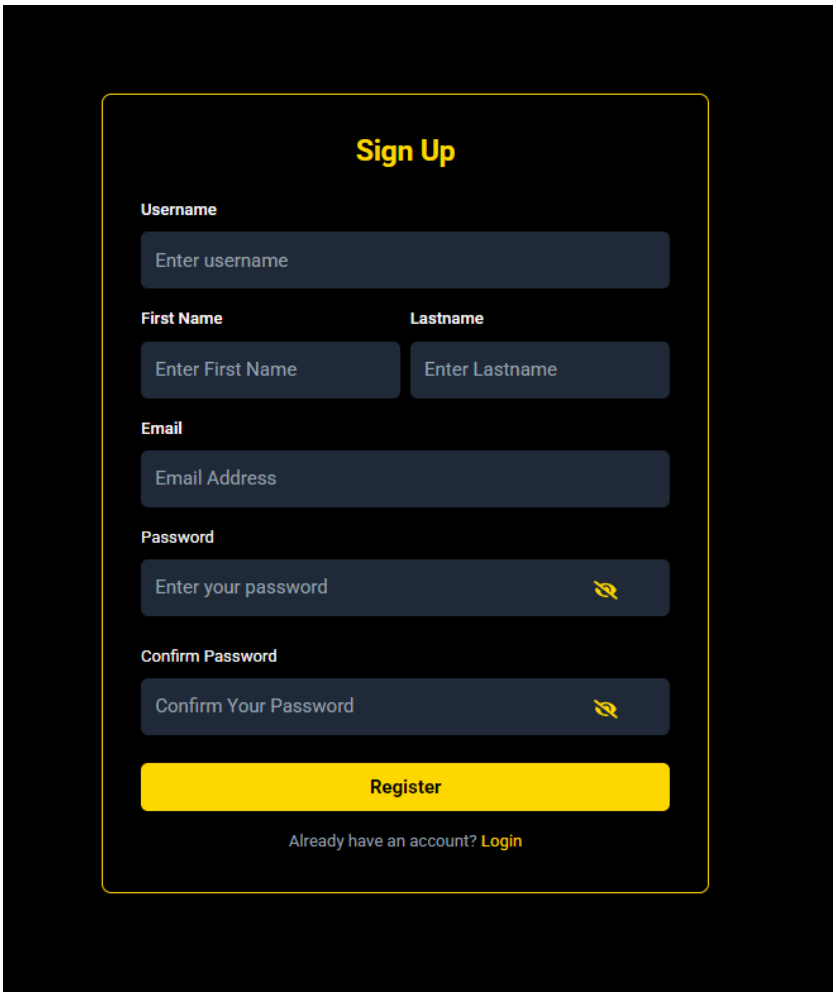
a. HomePage



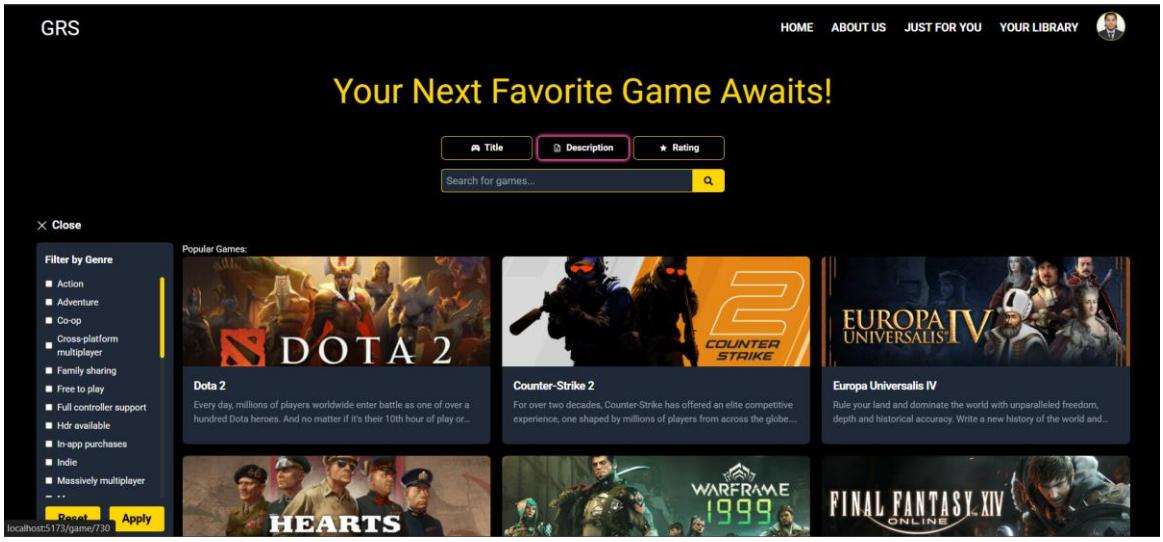
b. Login Page



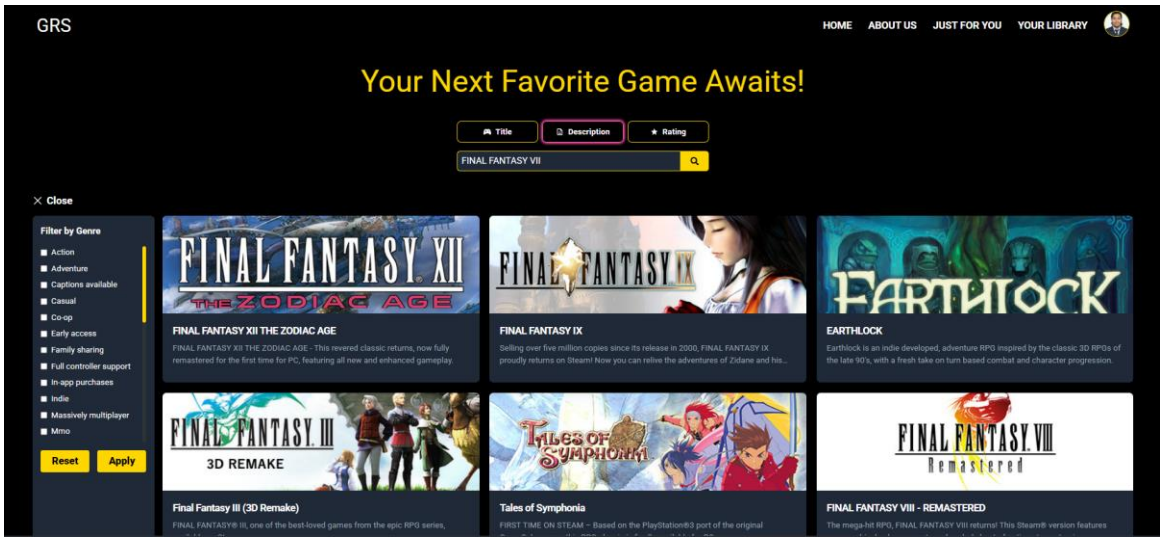
c. Registration Page



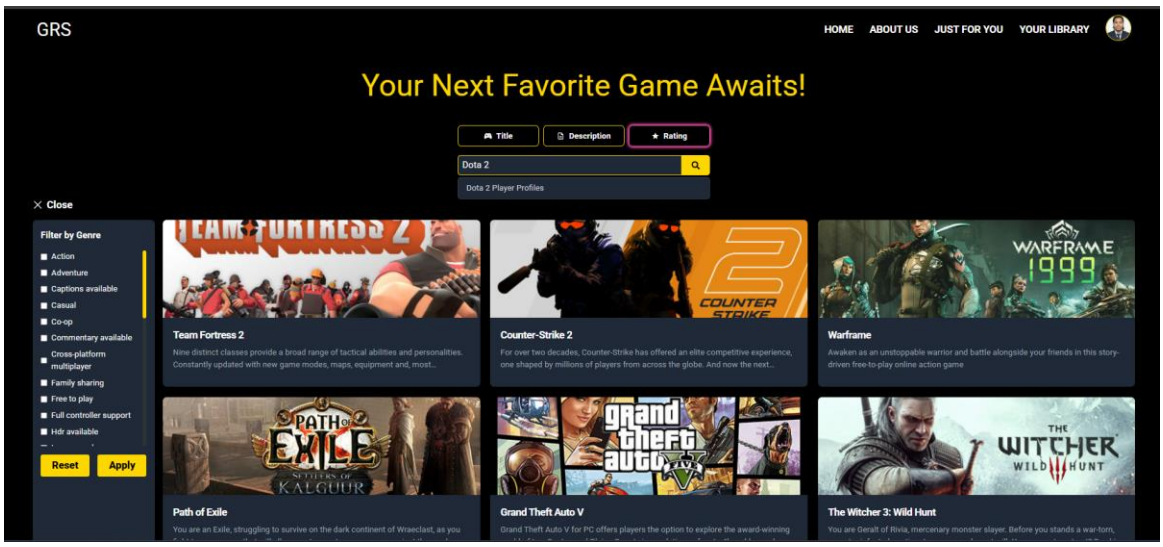
d. After Login



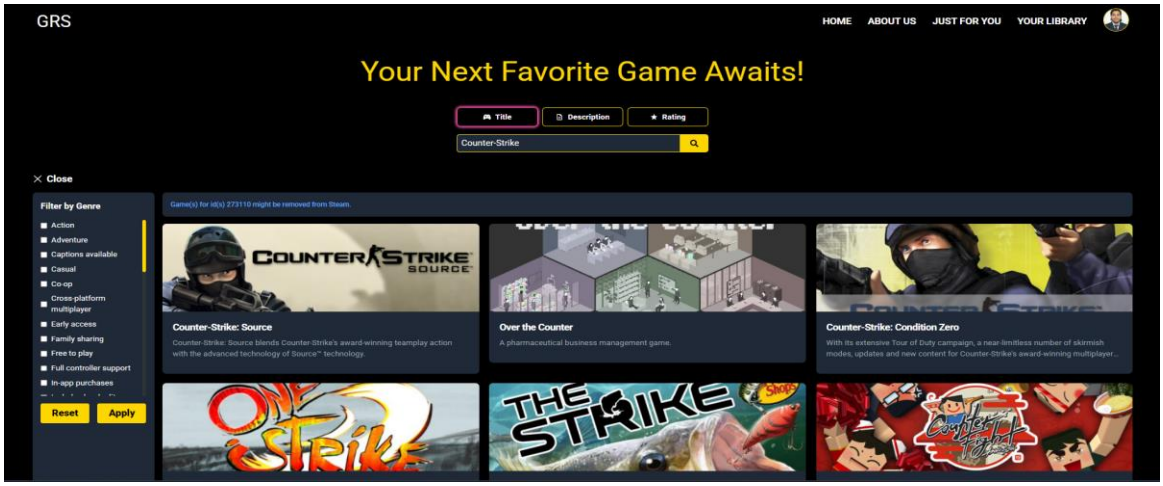
e. Description Based Recommendations



f. Rating Based Recommendations

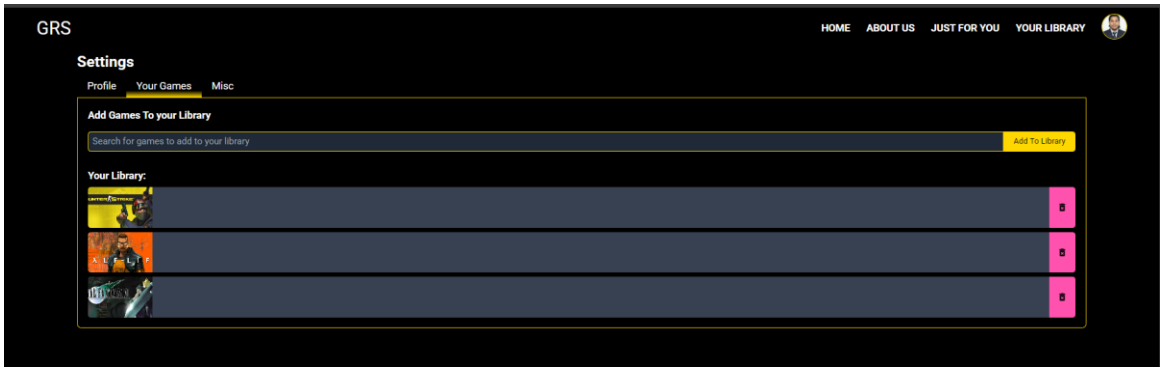


g. Title Based Recommendations

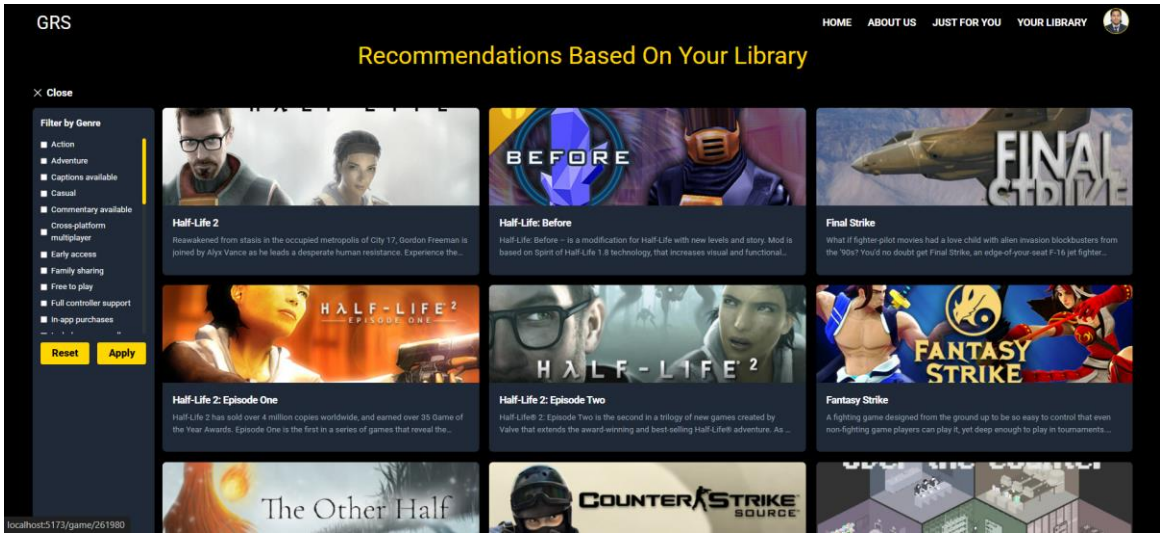


h. Adding Games to library

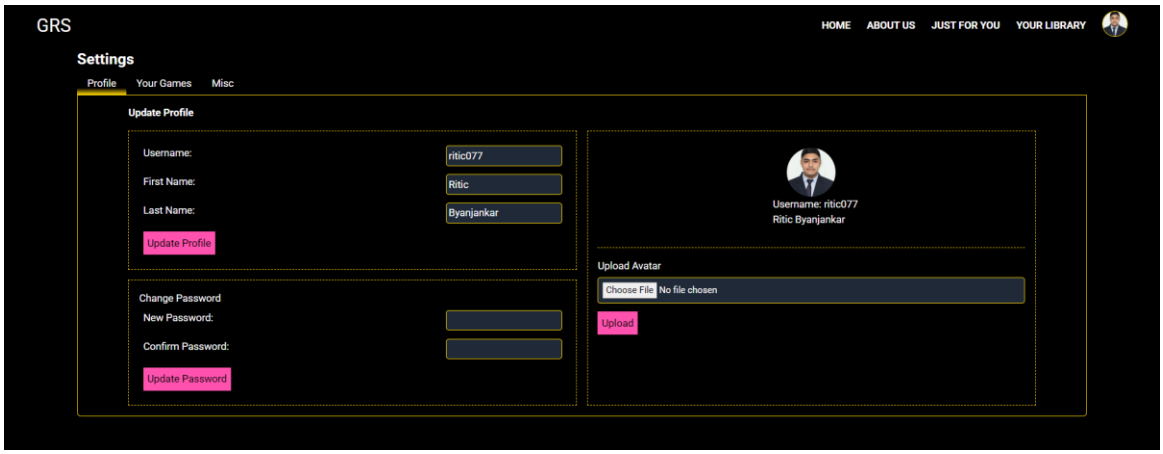
i. Games in Library



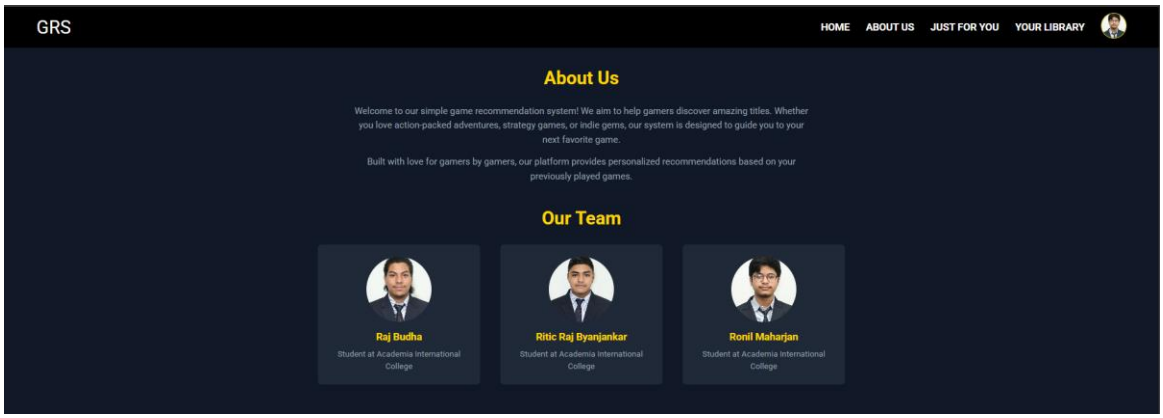
j. Personalized Recommendations



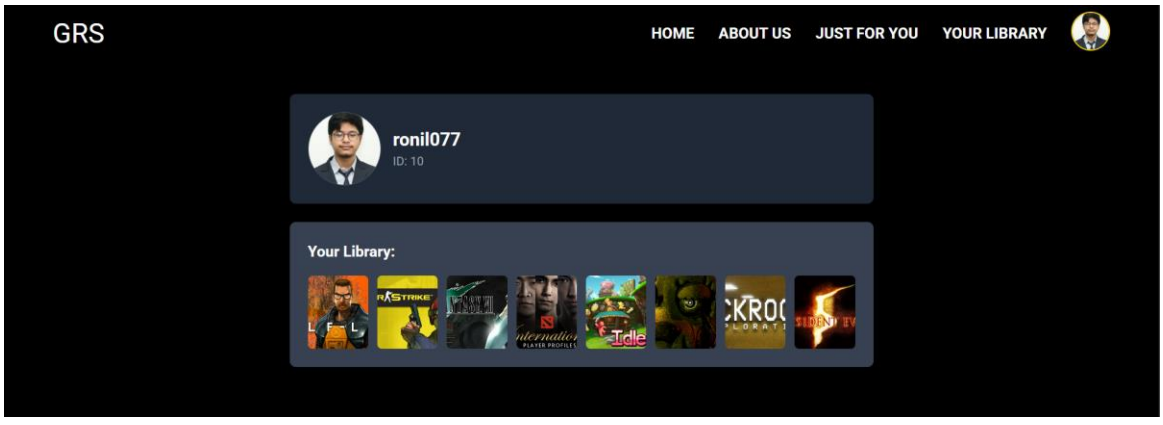
k. Profile Settings Page



l. About Us Page



m. Profile Page



Algorithm used

Algorithm:

//After the preprocessing of the data we perform TF-IDF Vectorization to convert the text into numerical feature vectors. It reflects how important a word is to a document in a collection.

Code:

```
cv=TfidfVectorizer(max_features=5000,stop_words='english')
vectors=cv.fit_transform(preprocessed_games_df['tags'])
with open('vectors_final.pkl', 'wb') as f:
    pickle.dump(vectors, f)
```

Algorithm for generating recommendations:

```
Function recommendation_by_description(game_title):
//load game
games_path = finders.find('src/final_dataset.csv')
          games = reduce_memory(pd.read_csv(games_path))
//get index of the input game
index = games[games['title'] == game].index[0]
//load the created vector pkl
vectors=pickle.load(open(" 'vectors_final.pkl",'rb'))
//get the vector for input game_title
item_vector = vectors[index]
//Compute the similarity score using cosine_similairity
similarities = cosine_similarity(item_vector, vectors).flatten()
//get the recommendation_index values after sorting similarity form high to low
recommended_indices = similarities.argsort()[::-1]
//make a list of recommended games
          game_lists=[]
for i in recommended_indices[1:n_recommendation]:
          game_lists.append(games.iloc[i].title)
Return game_lists
```

Code implemented in backend:

```
def get_vectors_from_cache():
   vectors = cache.get('vectors_final')
   if vectors is None:
      vectors_final_pickle_path = finders.find('src/vectors_final.pkl')
      vectors=pickle.load(open(vectors_final_pickle_path,'rb'))
      cache.set('vectors_final', vectors, timeout=3600)  # Cache for 1 hour
   return vectors


@api_view(['GET'])
def recommendation_by_description(request,game):
```

```
games_path = finders.find('src/final_dataset.csv')
games = reduce_memory(pd.read_csv(games_path))
n_recommendation = 20
# similarity = get_similarity_from_cache()
try:
    index = games[games['title'] == game].index[0]
    vectors = get_vectors_from_cache()
    item_vector = vectors[index]
    similarities = cosine_similarity(item_vector, vectors).flatten()
    recommended_indices = similarities.argsort()[::-1]
    game_lists=[]
    for i in recommended_indices[1:n_recommendation]:
        game_lists.append(games.iloc[i].app_id)
    # print(game_lists)

    return Response(game_lists)
except IndexError:
     return JsonResponse({'error': 'Cannot find the game specified'}, status=400)
```