

Anime Recommendation System

Dhairya Ajitkumar Patel
Meng ECE, University of Waterloo
Student ID: 20906076
da32pate@uwaterloo.ca

Rumna Samanta
Meng ECE, University of Waterloo
Student ID: 20883387
rsamanta@uwaterloo.ca

Abstract—The rapid growth of Japan’s animation industry has resulted in a plethora of anime films that have piqued the interest of diverse audiences. Each anime film has its own distinct personality that corresponds to a certain user’s preferences. The recommendation system based on search results can help the client to find appropriate anime recommendation based on genre, anime score, etc. In our study we have made use of Kaggle dataset of anime which contains information of around 17,562 anime which are rated by different 325,772 users, our system will help client to fetch results based on querying the data and getting the results based on scores, genre, age rating etc. In addition to that, we have also made a provision for Admin User to add, modify and delete any record in case of any additions or changes are required in the database.

Index Terms—Anime, Registered User, Non-Registered User, Admin User, Client, Server, Command Line Interface (CLI)

I. INTRODUCTION

Numerous types of activities can now be done online thanks to technological advancements. With the advent of the internet in real-time, which captures every piece of information received, some information is increasingly being stored in cyberspace. As a result, a lot of information is speculative. Because the information may be unrelated to them, it makes it difficult for consumers or users to find their preferences. The goal of the recommendation system is to solve this issue. A recommendation system allows users to easily find relevant content without having to search manually.

Users can use the recommendation system to find products that might be useful. In terms of anime and manga, more and more versions with engaging stories and series quality are appearing. Users will have a hard time choose which movie to view first due to this issues.

To overcome the above mentioned issues, in this project we will create a client based CLI which will help client to search the anime based on several search criteria like name, genre, ratings, popularity etc. Furthermore, we are also helping the database user who maintains the database record to modify and add new anime information without loading the data repetitively. The implementation of project is done using the dataset from Kaggle. [1]

II. THE OVERVIEW AND IMPLEMENTATION

A. Client - Server Architecture

As shown in Fig.1, our design is based on fundamental concept of client and server side architecture. The client here is the local computer which in our scope is CLI application. The

Server may be local or remote. The Server is MySQL server which has our anime database. The Client/or CLI sends in the request to the server to fetch some information from the server. The MySQL server in response sends in the required result to the clients CLI.

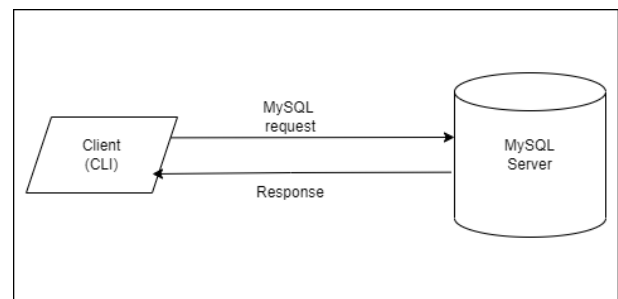


Fig. 1. Client - Server architecture

B. Implementation

The Project is based on three types of users:

- 1.) Non-Registered User
- 2.) Registered User
- 3.) Admin User

The Non-Registered User has the provision to only search anime based on name, anime age rating and date of release. The Registered User has an expanded provision to search results based on name, score, genre, anime popularity, age rating, synopsis. Besides this the Registered User can also modify his/her rating which he/she previously rated. The Admin User is the database(DB) admin who has the provision to add new anime also modify or update a particular anime record. In our database we have two admins who have the privilege to perform these operations.

In the following section we will describe our server side implementation i.e. the ER model in section C and ER model to relational schema in section D

C. ER model

In this section the Fig.2 describes the ER model of our Anime recommendation system. It describes the entities, their relationships and cardinality constraints. From the Fig. 2 Entity AnimeRatingComplete is in total participation with AnimeUserRatingWatchStatus. AnimeRatingComplete is inherited to AnimeUserWatchStatus For this project we have


```
#Code to search Animes based on score- Registered User
def SearchByAnimeRating():
    os.system('cls')
    print("Press 1 see the animes with Score between 9-10")
    print("Press 2 see the animes with Score between 8-9")
    print("Press 3 see the animes with Score between 7-8")
    print("Press 4 see the animes with Score between 6-7")
    print("Press 5 see the animes with Score between 5-6")
    print("Press 0 to exit")
    UserChoice = int(input("Please enter your choice: "))
    if UserChoice==1:
        SearchByAnimeRatingQuery="Select Name,Score from AnimeInfo where Score>=9 and Score<=10;"
    elif (UserChoice == 2):
        SearchByAnimeRatingQuery = "Select Name,Score from AnimeInfo where Score>=8 and Score<=9;"
    elif (UserChoice == 3):
        SearchByAnimeRatingQuery = "Select Name,Score from AnimeInfo where Score>=7 and Score<=8;"
    elif (UserChoice == 4):
        SearchByAnimeRatingQuery = "Select Name,Score from AnimeInfo where Score>=6 and Score<=7;"
    elif (UserChoice == 5):
        SearchByAnimeRatingQuery = "Select Name,Score from AnimeInfo where Score>=5 and Score<=6;"
    try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to fetch values...!!",ie)
```

Fig. 10. Search by anime scores - Registered User

```
#Code to search Animes based on Genres- Registered User
def SearchByAnimeByGenre():
    #Code to fetch different Genres available in the DB
    GenreList="Select distinct Genre from Genre;"
    try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to fetch values...!!",ie)
    ch = int(input("Press 0 to exit: "))
    if (ch == 0):
        return 0
    #Code to prepare a Genre Name list seperated by ,
    GenreInput=input("Enter the genres seperated by , to search anime:")
    G_list=GenreInput.split(",")
    temp6=""
    if(len(G_list)>1):...
    else:
        temp6=str(G_list[0])
    #Code to fetch animes based on user Genre input
    if(len(G_list)==1):
        GenreQuery="SELECT Distinct Name from AnimeInfo INNER join Genre on AnimeInfo.MALID=Genre.MALID\"
        \" where Genre =\"'+str(temp6)+'\";"
    else:
        GenreQuery = "SELECT Distinct Name from AnimeInfo INNER join Genre on AnimeInfo.MALID=Genre.MALID\"
        \" where Genre =\"'+str(temp6)+'\";"
```

Fig. 11. Search by Anime Genre - Registered User

many top trending anime he/she wants for eg. top 3 or top 5 trending anime. This functionality is shown in Fig 12 Figure

```
#Code to search Anime based on popularity - Registered User
def SearchForTrendingAnime():
    os.system('cls')
    limit=input("Please enter how many trending Anime you want to view: ")
    TrendingAnimeQuery="Select DISTINCT Name from AnimeInfo order by popularity DESC limit "+str(limit);"
    try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to fetch values...!!",ie)
    ch = int(input("Press 0 to exit: "))
    if (ch == 0):
        return 0
```

Fig. 12. Search top trending Anime - Registered User

14 shows the code for how the registered user can modify his/her previously rated anime. The user can modify the rating and search again to view the updated results.

Admin user is a DB admin who can is able to insert/update an anime record, insert/delete a user from the database. Fig 15 and 16 shows the code for insertion into AnimeInfo and UserInfo table. Fig 17 shows the code for modifying anime record and Fig 18 shows the code for deleting a user record from UserInfo table.

```
#Code to fetch synopsis of an Anime - Registered User
def FetchSynopsis():
    os.system('cls')
    AnimeName = input("Please type anime name to search: ")
    UserAnimeQuery = "Select Name,Synopsis from AnimeInfo where Name Like \"'\" + AnimeName + \"'\"";
    try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to fetch values...!!", ie)
    ch = int(input("Press 0 to exit"))
    if (ch == 0):
        return 0
```

Fig. 13. Get anime synopsis - Registered User

```
#Code to modify the rating of an Anime that the user had watched and rated previously - Registered User
def RateAnime(UserID):
    os.system('cls')
    print("List of Animes that you have watched and rated..")
    AnimeRatingQuery="Select AnimeInfo.Name, AnimeRatingComplete.Rating,AnimeRatingComplete.MALID from \"
    \"AnimeInfo INNER JOIN AnimeRatingComplete\"
    \" on AnimeInfo.MALID=AnimeRatingComplete.MALID where UserID="+str(UserID)+";"
    try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to fetch values...!!", ie)
    ch = int(input("Press 0 to exit: "))
    if (ch == 0):
        return 0
    AnimeName = input("Enter the anime name to modify the rating: ")
    UserRating = int(input("Enter a score between 0-10: "))
    for(Name,Rating,MALID) in animes:
        if(AnimeName==Name):
            UpdateRatingQuery="Update AnimeRatingComplete set Rating="+str(UserRating)+" where\"
            \" MALID="+str(MALID)+" and UserID="+str(UserID)+";"
            try:...
except mysql.connector.IntegrityError as ie:
    print("Failed to update record...!!",ie)
    ch = int(input("Press 0 to exit: "))
    if (ch == 0):
        return 0
```

Fig. 14. Modifying user's anime score - Registered User

```
#Code for Admin to insert values into sql tables - Admin User
def InsertRecords():
    os.system('cls')
    print("Press 1 to insert into the table AnimeInfo")
    print("Press 2 to insert into the table UserInfo")
    print("Press 0 to exit")
    UserChoice=int(input("Please enter your choice: "))
    if(UserChoice==0):
        return 0
    elif(UserChoice==1):
        MALID = input("Input MALID: ")
        while (1):...
        Name=input("Input Anime Name: ")
        while(1):...
        EnglishName = input("Input Anime English Name: ")
        Type = input("Input Anime Type: ")
        Episodes = input("Input Anime Episodes: ")
        while (1):...
        format = "%Y-%m-%d"
        AiredStr=input("Input Anime Aired_start: ")
        while(1):...
        AiredEndStr = input("Input Anime Aired_end: ")
        while (1):...
        Duration_in_min_per_episode = input("Input Duration_in_min_per_episode: ")
        while (1):...
        Popularity = input("Input Popularity: ")
```

Fig. 15. Inserting a record - Admin User

```

elif(UserChoice==2): #Insert into userinfo table - Admin user
    flag=0
    os.system('cls')
    while(1):...
    if (Flag==0):
        lastrow="SELECT * FROM userinfo WHERE userid=(SELECT max(userid) FROM userinfo);"
        mycursor = mydb.cursor()
        mycursor.execute(lastrow)
        lastentry=mycursor.fetchall()
        for l in lastentry:
            lastUid=int(l[0])
            UserInfoInsertQuery="Insert into UserInfo " \
                                "values (" +str(lastUid+1)+", "+str(lastUid+1)+", \'"+role+" \');"
        try:...
        except mysql.connector.IntegrityError as ie:...
        except mysql.connector.errors.DatabaseError as er:...

```

Fig. 16. Inserting a record - Admin User

```

#Code for Admin to update records in sql table - Admin User
def UpdateRecords():
    os.system('cls')
    while(1):
        MALID=int(input("Enter MALID to update record in AnimeInfo table: "))
        if (isinstance(MALID, int)):
            break
        else:
            MALID = input("Wrong input! Please enter a valid MALID: ")
        AttributeName=input("Enter the Name of the attribute to be updated: ")
        AttributeValue=input("Enter the value of the attribute to be updated: ")
        SearchQuery="Select * from animeinfo where MALID="+str(MALID)+";"
        try:
            mycursor=mydb.cursor(buffered=True)
            mycursor.execute(SearchQuery)
            if(mycursor.rowcount==0):...
        else:
            UpdateQuery="Update AnimeInfo set "+str(AttributeName)+"='"+str(AttributeValue)+"' " \
                        "where MALID="+str(MALID)+";"
            try:...
            except mysql.connector.IntegrityError as ie:...
            except mysql.connector.errors.DatabaseError as er:...
        except mysql.connector.IntegrityError as ie:...0
        except mysql.connector.errors.DatabaseError as er:...

```

Fig. 17. Updating anime record - Admin User

```

#Code for Admin to delete a record from sql table- Admin User
def DeleteRecord():
    os.system('cls')
    uid = int(input("Enter UserID to delete a record from the UserInfo table: "))
    UserDeleteQuery = "Delete from UserInfo where UserID=" + str(uid) + ";";
    try:
        mycursor = mydb.cursor()
        mycursor.execute(UserDeleteQuery)
        mydb.commit()
        print("Record deleted successfully..")
        mycursor.close()
        ch = int(input("Press 0 to exit: "))
        if (ch == 0):
            return 0
    except mysql.connector.IntegrityError as ie:
        print("Failed to insert values...!!",ie)
        ch = int(input("Press 0 to exit: "))
        if (ch == 0):
            return 0
    except mysql.connector.errors.DatabaseError as er:
        print("Error: {}".format(er))
        ch = int(input("Press 0 to exit: "))
        if (ch == 0):
            return 0

```

Fig. 18. Delete user record - Admin User

III. TESTING THE CASES IN CLI

Now that we have seen the codes which will be used for testing our recommendation system, we will now show and run some test cases and validate the results on the expected output. We have run command prompt with administrator rights and changed the location to the directory where all the .py files are present (here its in *C:\users\dhair*). The main file is named as *AnimeMain.py*

A. Testing for Non-Registered Users

Fig 19 shows the main screen where users are allowed to select from the available options. Non-Registered Users are able to choose option 2.

```

C:\ Administrator: Command Prompt - python3 AnimeMain.py
Press 1 to login
Press 2 to search anime
Press 0 to exit
Please select one option:

```

Fig. 19. Home Screen

On selecting option 2 the user is now able to get the anime results based on Name, Anime Age Rating and Release Date (Fig 20).

```

C:\ Administrator: Command Prompt - python3 AnimeMain.py
Press 1 to search anime by Name
Press 2 to search anime by Anime Age Rating
Press 3 to search anime by Release Date
Press 0 to exit
Please select one option:

```

Fig. 20. Options for Non Registered Users

The non-registered user searches for "Cowboy Bebop". The results are shown in Fig.21

Now if the user chooses option '0' and chooses to search the anime based on release dates he will be prompted to enter two dates. The result displayed will list the anime who were premiered between the two dates. The results are shown in fig.22

Presence of index on *Aired_start* optimises searching for results. Figure 23 shows the explain command that is run on


```

Administrator: Command Prompt - python3 AnimeMain.py
Please type anime name to search: Cowboy Bebop
+-----+-----+-----+-----+
| Name | Score | Episodes | Released Date |
+-----+-----+-----+-----+
| Cowboy Bebop | 8.78 | 26 | 1998-04-03 |
| Cowboy Bebop:The Movie | 8.39 | 5 | 2001-09-01 |
| Cowboy Bebop: Yose Atsume Blues | 7.44 | 1 | 1998-06-26 |
| Cowboy Bebop: Ein no Natsuyasumi | 6.25 | 1 | 2012-12-21 |
+-----+-----+-----+-----+
Press 0 to exit:

```

Fig. 21. Search results based on Name

```

Administrator: Command Prompt - python3 AnimeMain.py
+-----+-----+
| Name | Aired |
+-----+-----+
| Mobile Fighter G Gundam | 1994-04-01 |
| Future GPX Cyber Formula Zero | 1994-04-01 |
| Boku no Kachi | 1994-04-01 |
| Montana Jones | 1994-04-02 |
| Gene Diver | 1994-04-04 |
| Karaoke Senshi Mike-tarou | 1994-04-04 |
| Goal FH: Field Hunter | 1994-04-04 |
| Pacusi | 1994-04-04 |
| Soccer Fever | 1994-04-04 |
| Daisuki! Nendomama | 1994-04-04 |
+-----+-----+
Press 1 for next page
Press 2 to exit
Chose one option:

```

Fig. 22. Search results based on Release Date of Anime

MySQL Workbench which confirms the use of index named `idx_duration` as initialized in DDL statements (Fig 3).

```

28 * explain select * from Anime_View_NonRegUser where Aired_start between '1990-04-01' and '1994-05-01';
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | animeinfo | 1000 | range | idx_duration | idx_duration | 4 | 750 | 100.00 | 100.00 | Using index condition; Using MRR |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
Result 8 x:

```

Fig. 23. 'EXPLAIN' command on search query based on release date

B. Testing for Registered Users

If the user is Registered user then the user will be asked to enter the his/her ID and password after choosing option 1 of the main screen. The home screen for Registered User after logging in is as shown in Fig. 24. The user has option to choose the Anime based on Name, Scores, Genre, Trending anime, search for synopsis of a particular anime and modify his/her anime score which the user previously rated.

For searching the results based on Genre, we have index `idx_Genre`. The search results for genre 'Action' is shown in

```

Administrator: Command Prompt - python3 AnimeMain.py
Welcome Registered User
Press 1 to search Anime
Press 2 to search Animes based on Anime Score
Press 3 to search Animes based on Genres
Press 4 to search for Trending Anime
Press 5 to read synopsis of Anime
Press 6 to modify your Anime Rating
Press 0 to exit
Please select one option:

```

Fig. 24. Options for Registered User

Fig. 25. The explain command is shown in Fig.26 confirms the use of created index.

```

Administrator: Command Prompt - python3 AnimeMain.py
+-----+-----+
| Anime Name |
+-----+-----+
| Cowboy Bebop |
| Cowboy Bebop:The Movie |
| Trigun |
| Witch Hunter Robin |
| Eyeshield 21 |
| Initial D Fourth Stage |
| Naruto |
| One Piece |
| Tennis no Ouji-sama |
| Ring ni Kakero 1 |
+-----+-----+
Press 1 for next page
Press 2 to exit
Chose one option:

```

Fig. 25. Results based on Genre

Recommendation of Anime can be helpful to user based on popularity. User might want to get a list most popular anime which are trending. We have made this possible for registered user where user enters the number of top trending anime and the result based on popularity is fetched. The results (top 3 trending anime) is shown in the fig.27

Registered Users are able to modify the ratings of anime which they have previously rated. Here Registered User with ID 5 has rated the anime as shown in Fig 28. Now, the user wants to modify the anime named "Shuffle!" the user is asked

30 • explain SELECT Distinct Name from AnimeInfo INNER join Genre on AnimeInfo.MALID=Genre.MALID where Genre = 'Action';

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Genre		ref	PRIMARY,db_Genre	db_Genre	120	const	3888	100.00	Using where; Using index; Using temporary
1	SIMPLE	AnimeInfo		eq_ref	PRIMARY	PRIMARY	4	animeInfo.Genre.MALID	1	100.00	

Fig. 26. 'EXPLAIN' command on search query based on Genre

```
Administrator: Command Prompt - python3 AnimeMain.py
Please enter how many trending Anime you want to view: 3
+-----+
| Name |
+-----+
| Yama no Susume: Next Summit |
| Tsuki to Laika to Nosferatu |
| Scarlet Nexus |
+-----+
Press 0 to exit:
```

Fig. 27. search results of top 3 trending anime

to enter the new value as shown in fig. 29 and the updated result is shown in fig 30

```
Administrator: Command Prompt - python3 AnimeMain.py
List of Animes that you have watched and rated..
+-----+
| Name | Rating |
+-----+
| School Rumble | 10 |
| Azumanga Daioh | 9 |
| Shuffle! | 2 |
| Fruits Basket | 10 |
| Gunslinger Girl | 9 |
| Sen to Chihiro no Kamikakushi | 9 |
| Elfen Lied | 10 |
| Girls Bravo: First Season | 9 |
| Hoshi no Koe | 8 |
| Peach Girl | 8 |
+-----+
Press 1 for next page
Press 2 to exit
Chose one option:
```

Fig. 28. List of Anime rated by user 5

C. Testing for Admin Users

In our project we have two DB admin with UserID 0 and UserID 1. Their role is 'Admin User'. On logging in with admin user id and password. Admin user is able to insert a new anime record, update current anime record. The DB admin can delete a particular record. In our case we are deleting a record from User database.

Fig.31 shows the list of available choices the admin wishes to alter the record in the database. We have added a check

```
Administrator: Command Prompt - python3 AnimeMain.py
Enter the anime name to modify the rating: Shuffle!
Enter a score between 0-10: 10
Rating for Shuffle! updated successfully!!
Press 0 to exit: 0
```

Fig. 29. Anime rating to be modified with new rating

```
Administrator: Command Prompt - python3 AnimeMain.py
List of Animes that you have watched and rated..
+-----+
| Name | Rating |
+-----+
| School Rumble | 10 |
| Azumanga Daioh | 9 |
| Shuffle! | 10 |
| Fruits Basket | 10 |
| Gunslinger Girl | 9 |
| Sen to Chihiro no Kamikakushi | 9 |
| Elfen Lied | 10 |
| Girls Bravo: First Season | 9 |
| Hoshi no Koe | 8 |
| Peach Girl | 8 |
+-----+
Press 1 for next page
Press 2 to exit
Chose one option:
```

Fig. 30. Modified rating from 2 to 10 for anime 'Shuffle!'

```
Administrator: Command Prompt - python3 AnimeMain.py
Welcome Admin
Press 1 to insert record
Press 2 to update record
Press 3 to delete record
Print 0 to exit
Please select one option:
```

Fig. 31. Choices for Admin User'

constraint named 'CHK_MALID' which checks that MALID should always be greater than zero. The DB admin is trying to enter the value of MALID say '0' as shown in fig. 32. The check constraint is violated and so an error from MySQL is fetched and printed on the CMD stating the admin to enter correct value.

```
Administrator: Command Prompt - python3 AnimeMain.py
Press 1 to insert into the table AnimeInfo
Press 2 to insert into the table UserInfo
Press 0 to exit
Please enter your choice: 1
Input MALID: 0
Input Anime Name: a
Input Anime score: 2
Input Anime English Name: a
Input Anime Type: tv
Input Anime Episodes: 3
Input Anime Aired_start: 1994-03-01
Input Anime Aired_end: 1994-05-01
Input Duration_in_min_per_episode: 12
Input Popularity: 123
Error: 3819 (HY000): Check constraint 'CHK_MALID' is violated.
Press 0 to exit:
```

Fig. 32. Inserting incorrect anime record

```
Administrator: Command Prompt - python3 AnimeMain.py
Press 1 to insert into the table AnimeInfo
Press 2 to insert into the table UserInfo
Press 0 to exit
Please enter your choice: 1
Input MALID: 50000
Input Anime Name: Dark
Input Anime score: 7
Input Anime English Name: Dark
Input Anime Type: TV
Input Anime Episodes: 3
Input Anime Aired_start: 1998-01-02
Input Anime Aired_end: 1999-08-01
Input Duration_in_min_per_episode: 23
Input Popularity: 133
1 row inserted successfully
Press 0 to exit: _
```

Fig. 33. Inserting a new anime record

Fig.33 shows insertion of a new anime record in the database. Fig 34 shows updating the anime from 'Dark' to 'Dark Knight'

Fig. 35 shows deleting the user from user database.

IV. FUTURE SCOPE

In this section we will mention some future directions that could be worked on. The following features will be added to

```
Administrator: Command Prompt - python3 AnimeMain.py
Enter MALID to update record in AnimeInfo table: 50000
Enter the Name of the attribute to be updated: Name
Enter the value of the attribute to be updated: Dark Knight
Successfully updated 1 records
Press 0 to exit: _
```

Fig. 34. Modifying Anime record

```
Administrator: Command Prompt - python3 AnimeMain.py
Enter UserID to delete a record from the UserInfo table: 16
Record deleted successfully..
Press 0 to exit: _
```

Fig. 35. Deleting user data from user database

our existing project:

- 1) Data entry operator: These users will add information about a new anime into the DB system and will only have add privilege to the DB tables.
- 2) Auditor: These users will do auditing on the anime recommendation system. These users will decide whether to keep an anime in the system or to remove it from the system based on some parameters (reviews, popularity). The users will have remove access to some DB tables.
- 3) Developer: These users will be able to modify code in the system and will have add/ remove/ modify/alter access to the DB.
- 4) Data-Mining: Data mining implementation in our project will help our recommendation system to find hidden relationships (for example between watching status and Scores etc) and help to any client user to get better recommendations.

V. CONCLUSION

In this work we have successfully designed a database that can be used to search the anime data based on various search criteria like name, popularity, genre etc. This will help users to get better recommendation. The database created can be altered using DB admin to keep data updated.

REFERENCES

- [1] Valdivieso, H. (2021, July 13). Anime recommendation database 2020. Kaggle. Retrieved December 23, 2021, from https://www.kaggle.com/hernan4444/anime-recommendation-database-2020?select=watching_status.csv