॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS INSTITUTE OF TECHNOLOGY

DBMS Project Phase-1 Report on

Movie and T.V. Series Recommendation System

SUBMITTED BY

DIVILI SATHVIK	**MAYANK ARORA**
18070124025	**18070124040**
ARIPIRALA SAI SRIVATHSAV	**PRANAV SHARMA**
18070124014	**18070124051**

Under the Guidance of

Prof. Shruti Patil

DEPARTMENT OF INFORMATION TECHNOLOGY

SYMBIOSIS INSTITUTE OF TECHNOLOGY, PUNE

PUNE-412115

2020-21

INDEX

- **INTRODUCTION**

  This is a phase-1 report for the project “**Movie and T.V. series recommendation system**” which consists of problem we are trying to solve along with the solution, functional requirements and all other initially essential details of the project. This project is an implementation model of movie recommendation system’s database in MySQL.

- **PROBLEM STATEMENT**

  With the rapid development and advancement of digital world especially in the field of media and entertainment, a lot of content to watch online is being accumulated on various streaming platforms like OTTs etc. This availability of huge amount of content, now turned out into a boon and bane situation because though it provides users with various choices and options to get entertained on the other hand it also results in few practical problems like:

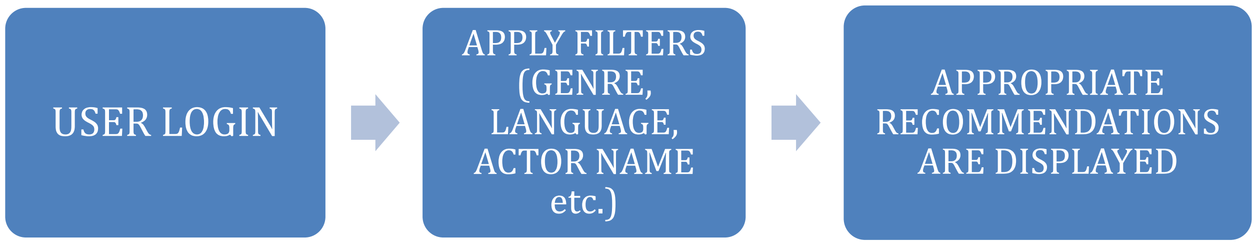
  - User gets perplexed with more than enough choices in-hand.
  - Finding the right OTT platform for a movie/T.V. series he wishes to watch becomes difficult.
  - Lacks the right suggestion which suits his/her taste.

- **OBJECTIVES**

  So, the project “Movie and T.V. series recommendation system” is a solution that helps the user to get recommended with a proper movie or T.V. series based upon various parameters like genre, actor/actress, rating, languages etc. and also helps the user to know on which OTT platform the respective movie/ T.V. series is available. This helps the user to know the list of movies that perfectly matches his choices thus reducing the chance of getting confused and also intimating where to watch it.

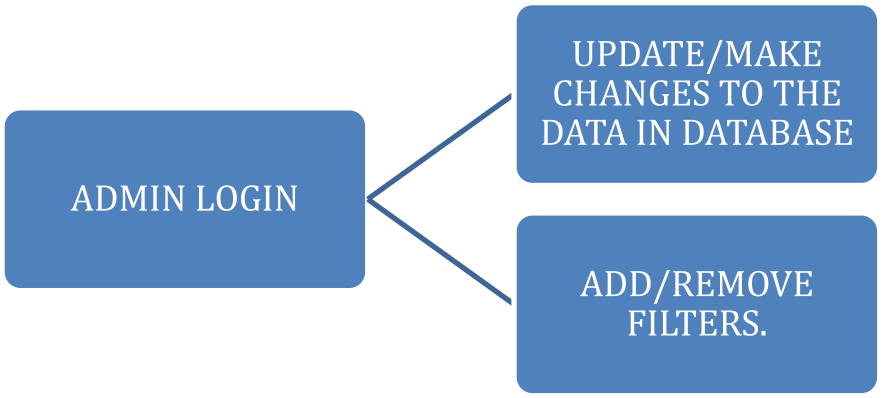
  The system’s architecture/modules would be as follows:

  USER MODULE:


USER LOGIN
APPLY FILTERS (GENRE, LANGUAGE, ACTOR NAME etc.)
APPROPRIATE RECOMMENDATIONS ARE DISPLAYED

- User need to login with his unique id and password.
- User can make use of various filters that are available, based upon his choices and can get appropriate recommendations.

ADMIN MODULE:



- Admin need to login with his unique id and password
- Admin can make changes to the database and also can make changes to the filters (add new ones or delete few) that are available for users to use.

- **<u>FUNCTIONALITIES:</u>**

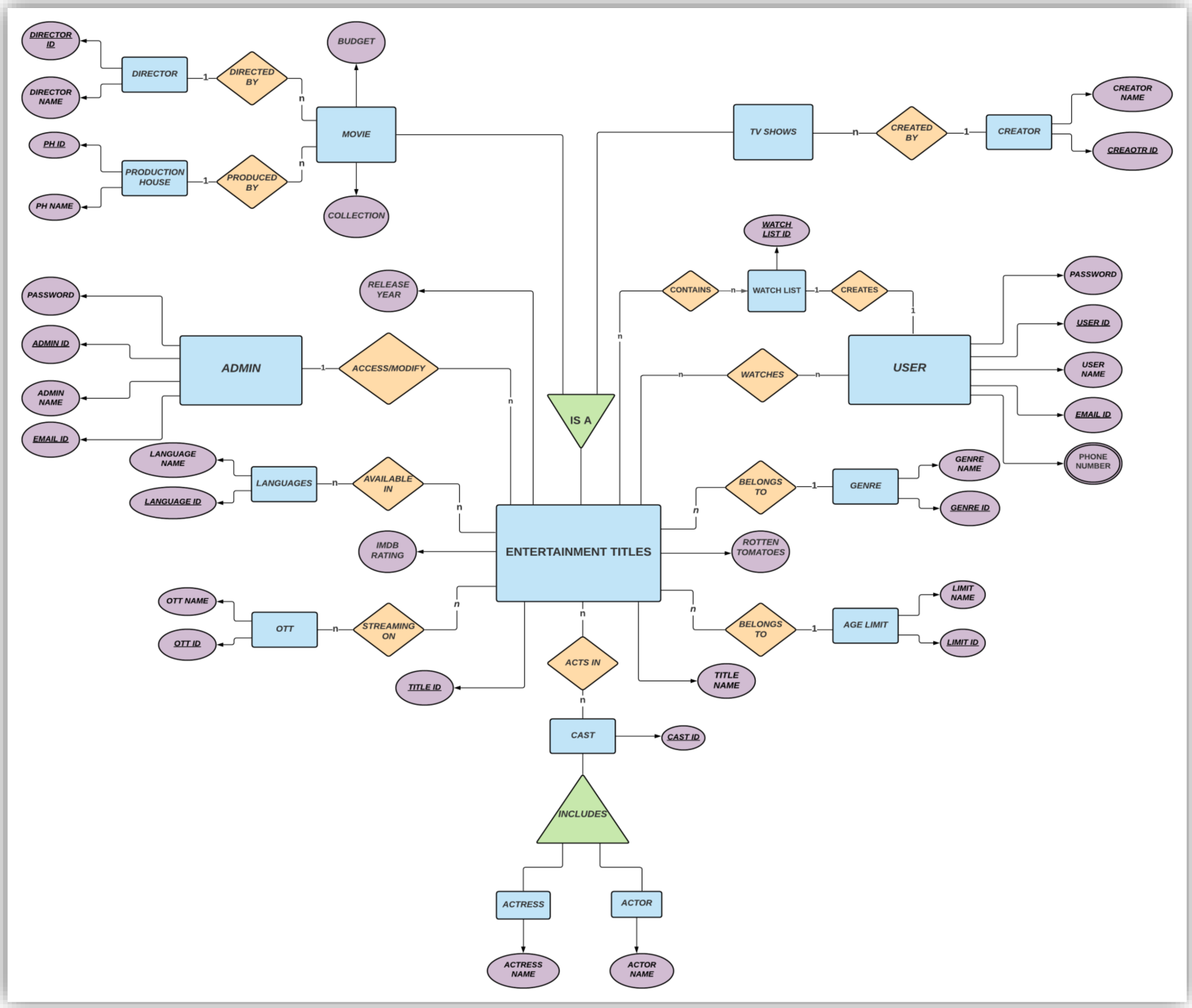
  - ➢ Every user should sign up using his/her unique id and password.
  - ➢ Initially user is provided with 2 options to select between movie and T.V. series which are under entertainment title.
  - ➢ After selecting one of the options, respective filters which are available under movies and T.V. series are displayed to choose between them.
  - ➢ Available filters for movies are name of the actor, actress, director, genre, age limit, ratings, release year, language, budget, box-office collection, streaming platform and production house.
  - ➢ Available filters for T.V. shows are name of the actor, actress, genre, age limit, ratings, release year, language, streaming platform, creator of the show.
  - ➢ User can also maintain a common list of all the movies and T.V. shows which he had already watched.
  - ➢ Admin should also sign up using his/her unique id and password.
  - ➢ Admin has the access to both movies and T.V. series to modify them

- **ENTITIES AND THEIR RELATIONSHIPS**

ENTITES	**ATTRIBUTES**
Entertainment Titles (ET)	titleName, titleID, imdbRating, rottenTomatoesRating, Release year,
Movie	budget, collection
TVshows	-
Cast	castID
Actor	actorName
Actress	actressName
OTT	ottName, ottID
AgeLimit	limitName, limitID
Genre	genreName, genreID
Language	languageName, languageID
Admin	adminName, adminID, emailID, password
User	userName, userID, emailID, password, PhoneNumber
Watchlist	WatchID
Creator	creatorName, creatorID
Director	directorName, directorID
ProductionHouse	phName, phID

RELATIONSHIP	**CARDINALITY**
ET ***is a*** Movie.	-
ET ***is a*** TV show.	-
Cast ***acts in*** ET.	Many to many
Cast ***includes*** Actor.	-
Cast ***includes*** Actress.	-
ET ***has*** Age Limit.	Many to one
ET ***Streaming on*** OTT.	Many to many
ET ***belongs to a*** Genre.	Many to one
ET ***Available in*** Languages.	Many to many
Admin ***modifies*** ET.	One to many
User ***watches*** ET.	Many to many
User ***Creates*** Watchlist.	One to one
Watchlist ***Contains*** ET.	Many to many
TV Shows ***created by*** Creator.	Many to one
Movie ***directed by*** Director.	Many to one
Movie ***produced by*** Production House.	Many to one

- **E-R /EER DIAGRAM**



- **RELATIONAL SCHEMA**

a. Movie

titleID	titleName	rottenTomatoesRating	imdbRating	releaseYear	budget
collection	*genreID*	*limitID*	*directorID*	*phID*	*adminID*

b. TVshows

titleID	titleName	rottenTomatoesRating	imdbRating	releaseYear	*genreID*
limitID	*creatorID*	*adminID*			

c. Genre

genreID	genreName

d. AgeLimit

limitID	limitName

e. OTT

ottID	ottName

f. Languages

languageID	languageName

g. Actor

castID	actorName

h. Actress

castID	actressName

i. Director

directorID	directorName

j. ProductionHouse

phID	phName

k. Creator

creatorID	creatorName

l. DbUser

<u>userID</u>	userName	password	emailID

m. PhoneNumber

<u>phoneID</u>	*userID*	phone

n. DbAdmin

<u>adminID</u>	adminName	password	emailID

o. ActsIn

<u>actingID</u>	*titleID*	*castID*

p. StreamingOn

<u>streamID</u>	*titleID*	*ottID*

q. AvailableIn

<u>availID</u>	*titleID*	*languageID*

r. Watches

<u>watchID</u>	*titleID*	*userID*

s. Watchlist

<u>listID</u>	*userID*

t. Contain

<u>containID</u>	*titleID*	*listID*

- **<u>KEYS IN EACH RELATION</u>**

a. Movie
   i. Candidate Key: titleID, titleName
   ii. Primary Key: titleID
   iii. Foreign Key: genreID, limitID, directorID, phID, adminID
   iv. Alternate Key: titleName

b. TV Show
   i. Candidate Key: titleID, titleName
   ii. Primary Key: titleID
   iii. Foreign Key: genreID, limitID, creatorID, adminID

        iv. Alternate Key: titleName

c. Genre
    i. Candidate Key: genreID, genreName
    ii. Primary Key: genreID
    iii. Foreign Key: NULL
    iv. Alternate Key: genreName
d. AgeLimit
    i. Candidate Key: limitID, limitName
    ii. Primary Key: limitID
    iii. Foreign Key: NULL
    iv. Alternate Key: limitName
e. OTT
    i. Candidate Key: ottID, ottName
    ii. Primary Key: ottID
    iii. Foreign Key: NULL
    iv. Alternate Key: ottName
f. Language
    i. Candidate Key: languageID, languageName
    ii. Primary Key: languageID
    iii. Foreign Key: NULL
    iv. Alternate Key: languageName
g. Actor
    i. Candidate Key: castID
    ii. Primary Key: castID
    iii. Foreign Key: NULL
    iv. Alternate Key: NULL
h. Actress
    i. Candidate Key: castID
    ii. Primary Key: castID
    iii. Foreign Key: NULL
    iv. Alternate Key: NULL
i. Director
    i. Candidate Key: directorID
    ii. Primary Key: directorID
    iii. Foreign Key: NULL
    iv. Alternate Key: NULL
j. ProductionHouse
    i. Candidate Key: phID, phName

ii. Primary Key: phID

iii. Foreign Key: NULL
iv. Alternate Key: phName

k. Creator
    i. Candidate Key: creatorID
    ii. Primary Key: creatorID
    iii. Foreign Key: NULL
    iv. Alternate Key: NULL
l. User
    i. Candidate Key: userID, emailID
    ii. Primary Key: userID
    iii. Foreign Key: NULL
    iv. Alternate Key: emailID
m. PhoneNumber
    i. Candidate Key: phoneID, phone
    ii. Primary Key: phoneID
    iii. Foreign Key: userID
    iv. Alternate Key: phone
n. Admin
    i. Candidate Key: adminID, emailID
    ii. Primary Key: adminID
    iii. Foreign Key: NULL
    iv. Alternate Key: emailID
o. ActsIn
    i. Candidate Key: actingID
    ii. Primary Key: actingID
    iii. Foreign Key: titleID, castID
    iv. Alternate Key: NULL
p. StreamingOn
    i. Candidate Key: streamID
    ii. Primary Key: streamID
    iii. Foreign Key: titleID, ottID
    iv. Alternate Key: NULL
q. AvailableIn
    i. Candidate Key: availID
    ii. Primary Key: availID
    iii. Foreign Key: titleID, languageID

iv. Alternate Key: NULL

r. Watches
   i. Candidate Key: watchID
   ii. Primary Key: watchID
   iii. Foreign Key: titleID, userID
   iv. Alternate Key: NULL

s. Watchlist
   i. Candidate Key: listID
   ii. Primary Key: listID
   iii. Foreign Key: userID
   iv. Alternate Key: NULL

t. Contains
   i. Candidate Key: containID
   ii. Primary Key: containID
   iii. Foreign Key: titleID, listID
   iv. Alternate Key: NULL

- **CODD'S RULE**

Rule 0:

Definition: This rule states that for a system to qualify as an **RDBMS**, it must be able to manage database entirely through the relational capabilities.

Justification: Yes, this the primary rule for all databases to satisfy and this database is also able to manage entirely through its relational capabilities. Hence this rule is applied.

Rule 1: Information rule:

Definition: All information (including metadata) is to be represented as stored data in cells of tables. The rows and columns have to be strictly unordered.

Justification: This database has all the data in the form of rows and columns, also the data in each row is unique, every table has its own and unique value called primary key, each cell contains only one value and the order of rows and columns doesn't affect the meaning of the tables. Hence this rule is applied.

Rule 2: Guaranteed Access:

Definition: Each unique piece of data (atomic value) should be accessible by: **Table Name + Primary Key (Row) + Attribute(column)**.

NOTE: Ability to directly access via POINTER is a violation of this rule.

Justification: Since each table in this database has its own primary key it is possible to access every atomic value logically without using any pointer or physical address of the data. Hence this rule is applied.

Rule 3: Systematic treatment of NULL:

Definition: Null has several meanings, it can mean missing data, not applicable or no value. It should be handled consistently. Also, Primary key must not be null, ever. Expression on NULL must give null.

Justification: This database will not have any null value (missing value, unknown value or unacceptable value) and most importantly any primary key will not be null. Hence this rule is applied.

Rule 4: Active Online Catalog:

Definition: Database dictionary(catalogue) is the structure description of the complete **Database** and it must be stored online. The Catalogue must be governed by same rules as rest of the database. The same query language should be used on catalogue as used to query database.

Justification: As this database do not have any database dictionary this rule is not applied.

Rule 5: Powerful and Well-Structured Language:

Definition: One well-structured language must be there to provide all manners of access to the data stored in the database. Example: **SQL**, **QBE** etc. If the database allows access to the data without the use of this language, then that is a violation.

Justification: Since our database accepts SQL (which is one of SQL/QBE) to access, modify and manipulate this rule is also applied.

Rule 6: View Updation Rule:

Definition: All the view that are theoretically updatable should be updatable by the system as well.

Justification: The view that is given in the database to the user is also in the form of tables only and not any other alternative virtual way. Hence this rule is not applied.

Rule 7: Relational Level Operation:

Definition: There must be Insert, Delete, Update operations at each level of relations. Set operation like Union, Intersection and minus should also be supported.

Justification: This database supports all the set operations and relational algebra which include join, union, intersection, minus, delete, insert etc. Hence this rule is also applied.

<u>Rule 8: Physical Data Independence:</u>

<u>Definition</u>: The physical storage of data should not matter to the system. If say, some file supporting table is renamed or moved from one disk to another, it should not affect the application.

<u>Justification</u>: This database and its data access methods will not be affected with the change in physical storage of data hence it satisfies the rule of physical data independence.

<u>Rule 9: Logical Data Independence:</u>

<u>Definition</u>: If there is change in the logical structure (table structures) of the database the user view of data should not change. Say, if a table is split into two tables, a new view should give result as the join of the two tables. This rule is most difficult to satisfy.

<u>Justification:</u> This rule is not applied by this database.

<u>Rule 10: Integrity Independence:</u>

<u>Definition</u>: The database should be able to enforce its own integrity rather than using other programs. Key and Check constraints, trigger etc, should be stored in Data Dictionary. This also make **RDBMS** independent of front-end.

<u>Justification:</u> Irrespective of the front-end, database has its own integrity and key values are stored in data dictionary. Hence this rule is applied.

<u>Rule 11: Distribution Independence:</u>

<u>Definition</u>: A database should work properly regardless of its distribution across a network. Even if a database is geographically distributed, with data stored in pieces, the end user should get an impression that it is stored at the same place. This lays the foundation of **distributed database**.

<u>Justification</u>: As we are not sure about the distribution of the database and how it could affect the data manipulation, we are not sure this rule could be applied or not. Hence considering this rule is not applied

<u>Rule 12: Nonsubversion Rule:</u>

<u>Definition</u>: If low level access is allowed to a system it should not be able to subvert or bypass integrity rules to change the data. This can be achieved by some sort of looking or encryption.

<u>Justification:</u> Since this database is made in MySQL and it forms an instant of the tables while we view data there will be no changes applied to the actual tables and data which will not allow to circumvent data integrity and security at low level access. Hence this rule is applied.