AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department Of Computer Engineering

A DBMS PROJECT REPORT

ON

MOVIE DATABASE SYSTEM

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERINGAISSMS IOIT

TE Computer Engineering

SUBMITTED BY

STUDENT NAME	ERP No:
Kaustubh Kabra	**38**
Sunit Lohade	**48**
Akash Mete	**52**

2020 -2021

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report
"MOVIE DATABASE SYSTEM"
Submitted by

STUDENT NAME	ERP No:
Kaustubh Kabra	38
Sunit Lohade	48
Akash Mete	52

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Prof. Shilpa Pimpalkar** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(**Prof. Shilpa Pimpalkar**) (**Dr. S.N.Zaware**)

Guide Head of Computer Department,

Place: Pune Date: 20/12/2020

Abstract

We have developed a Movie database system, where the information regarding Actors, Directors, Movies, Reviews, ratings etc will be saved. Going through the project description and websites like “IMDB”, we have identified few entities, found the relationships between them, Constructed the database, scrapped the data from IMDB, Inserted the data into Database and Designed an UI in WINDOWS using MySql and PHP.

Introduction

The entire project of Movie Database has been developed in ‘ubuntu’. We have implemented the front end of the UI using ‘HTML’ and we made use of ‘MySQL’ to create, store and modify the Database and its data. The Front end i.e. HTML pages were connected to the DBMS using ‘PHP’. SQL tables can be accessed and modified using the internal library of PHP. The developed system can be hosted on any server, in our case we used Apache Xampp on Windows localhost server to host the same.

Software Requirement Specification

Software Used:

- Xampp
- PhpMyAdmin
- VS Code
- MySQL Shell
- MySQL Workbench
- Chrome or Any Web Browser

Front-end:

- HTML, PHP
- CSS
- JS

Back-end:

- PHP
- Shell -For shell scripting

Database & Server:

- MySQL
- Apache Xampp

Graphical User Interface

UI and Possible interactions with the Database:

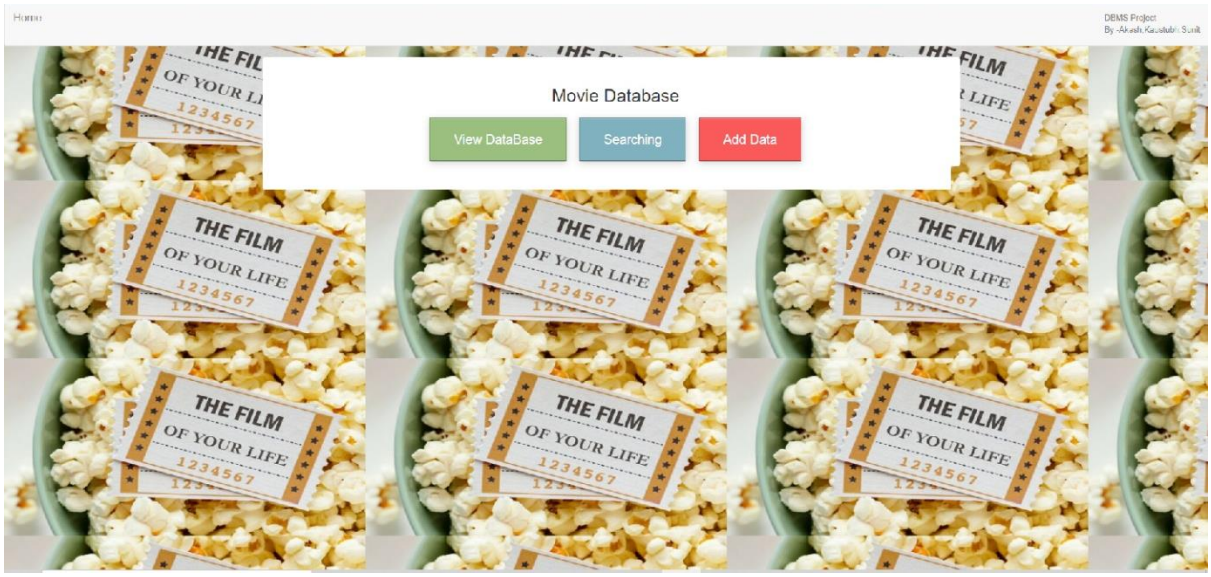
With the developed UI, on a broader level a user can perform three kinds of actions. View contents in database, Search for something and Add data.

View Contents:

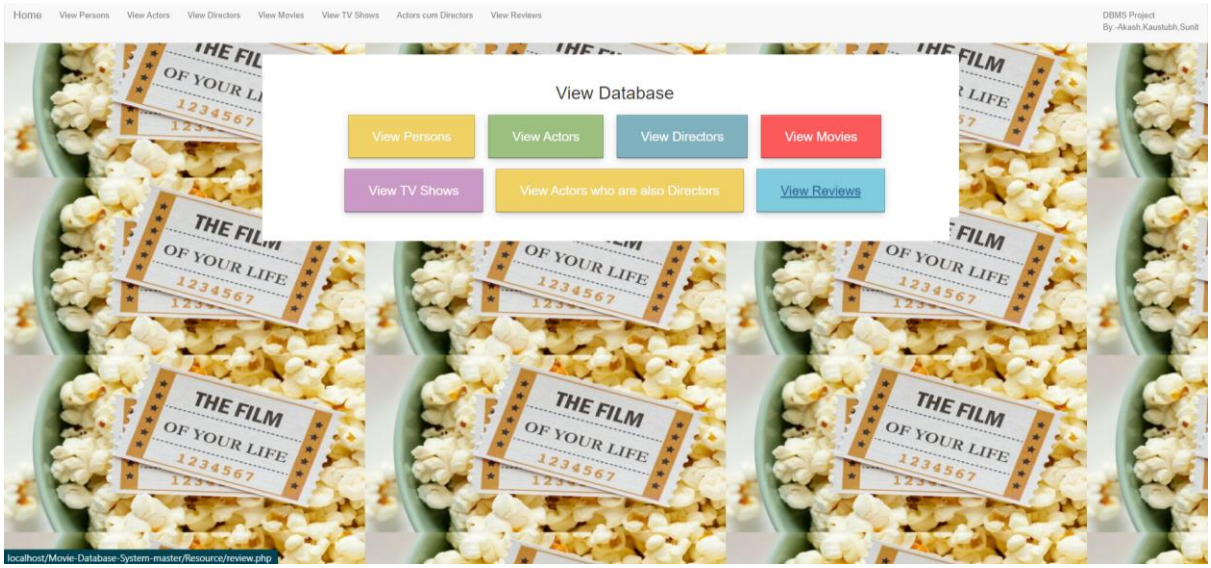
Using the developed system, a user can view things like,

1. Actors in the Database
2. Directors in the Database
3. Movies in the Database
4. TV Shows in the Database
5. Reviews in the Database

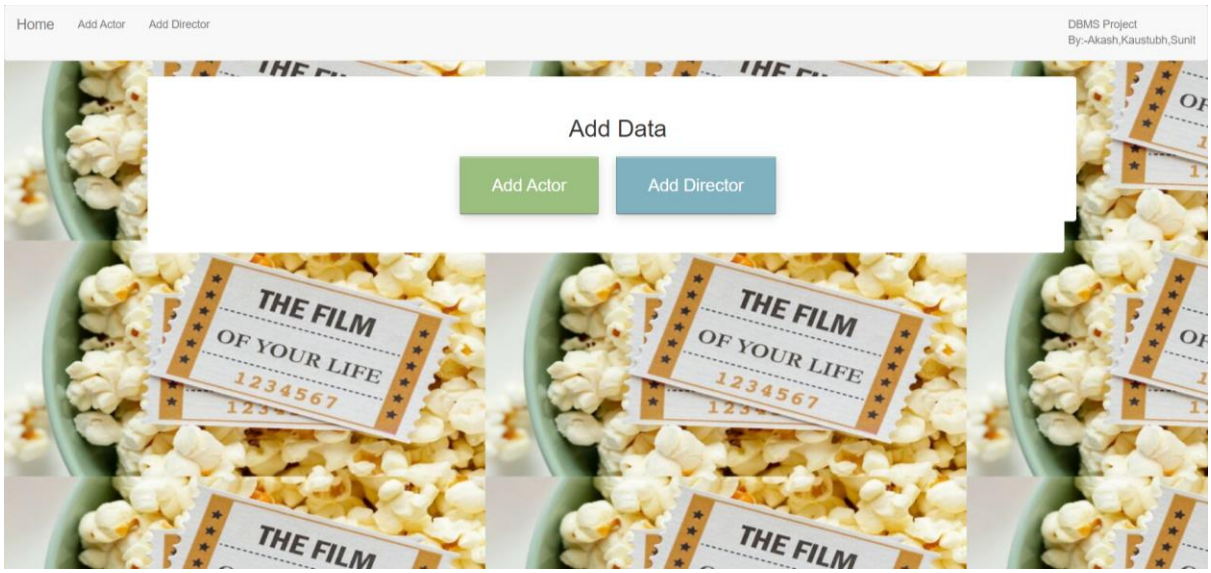
Home Page UI:



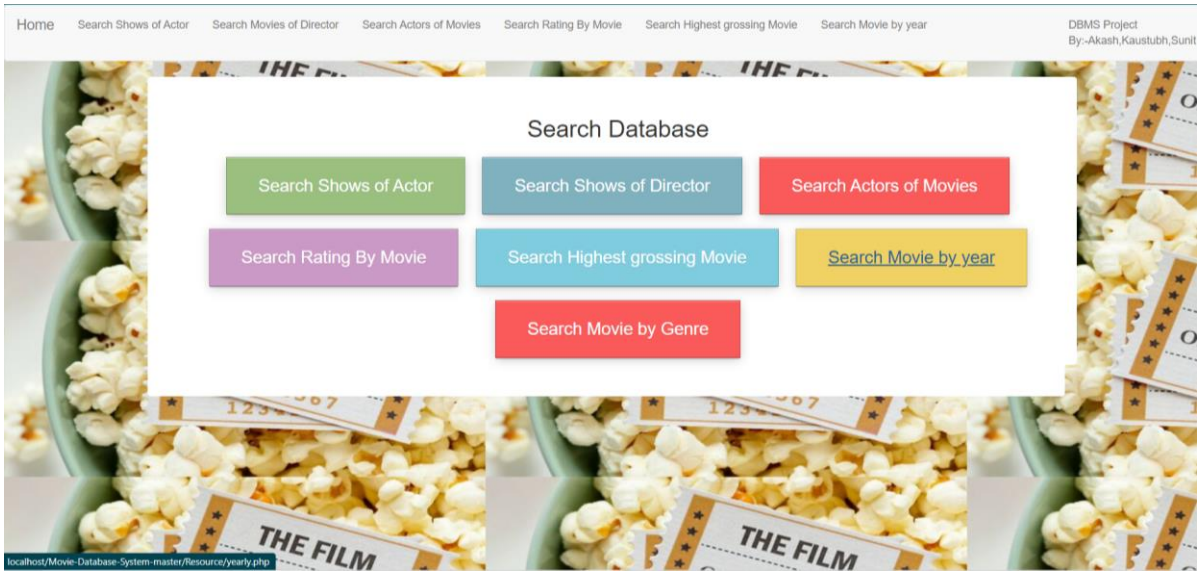
Movie Database View UI:



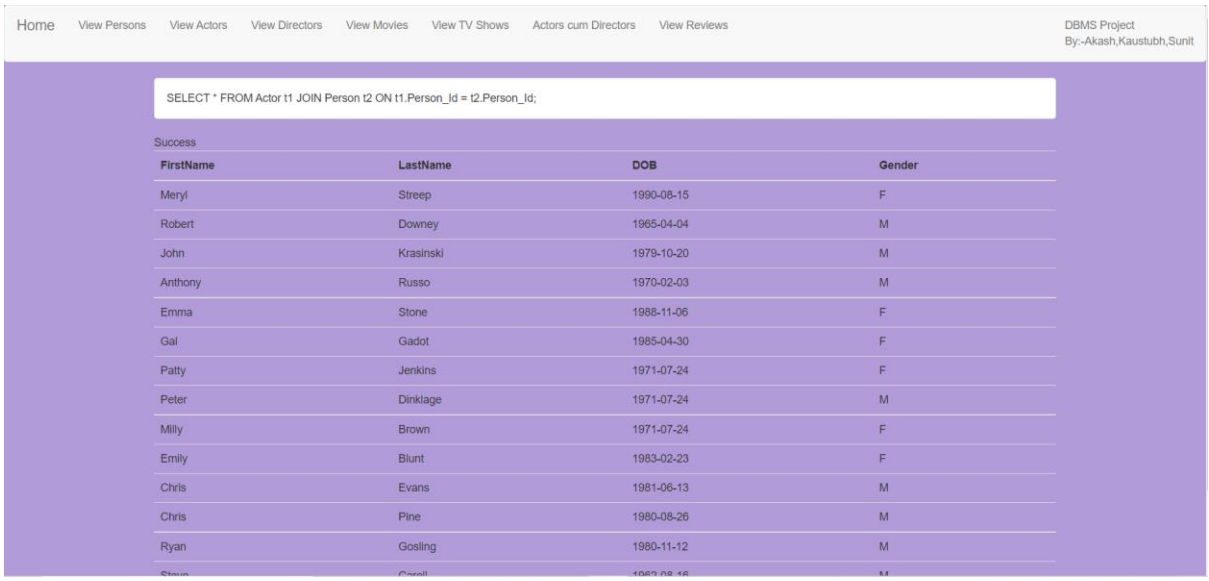
Add Actor Page UI:



Search Database UI:

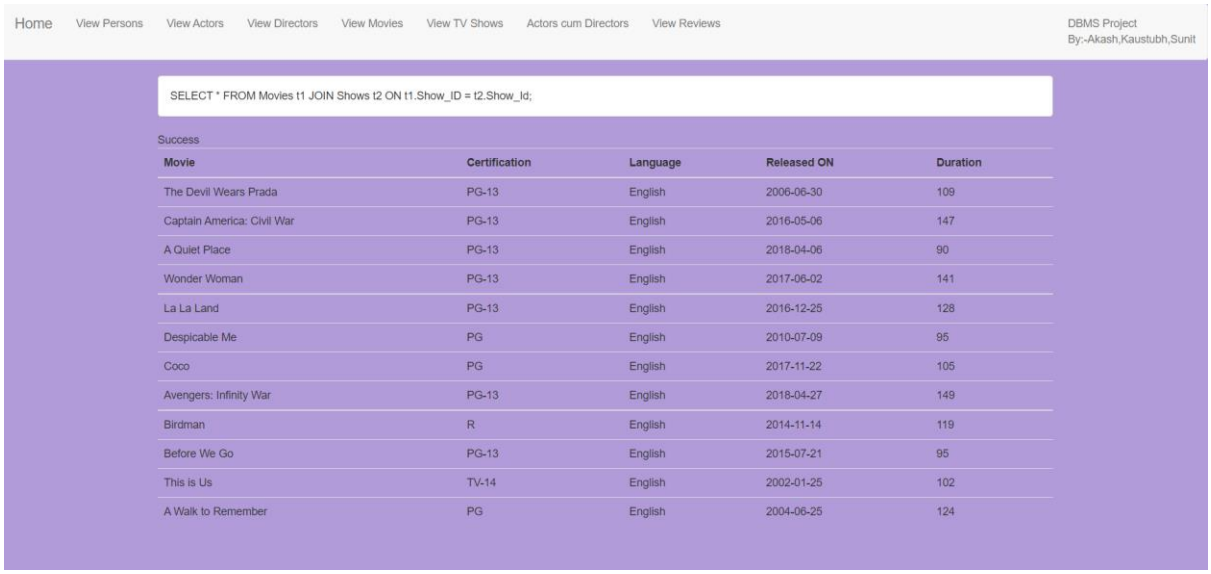


Actor UI:



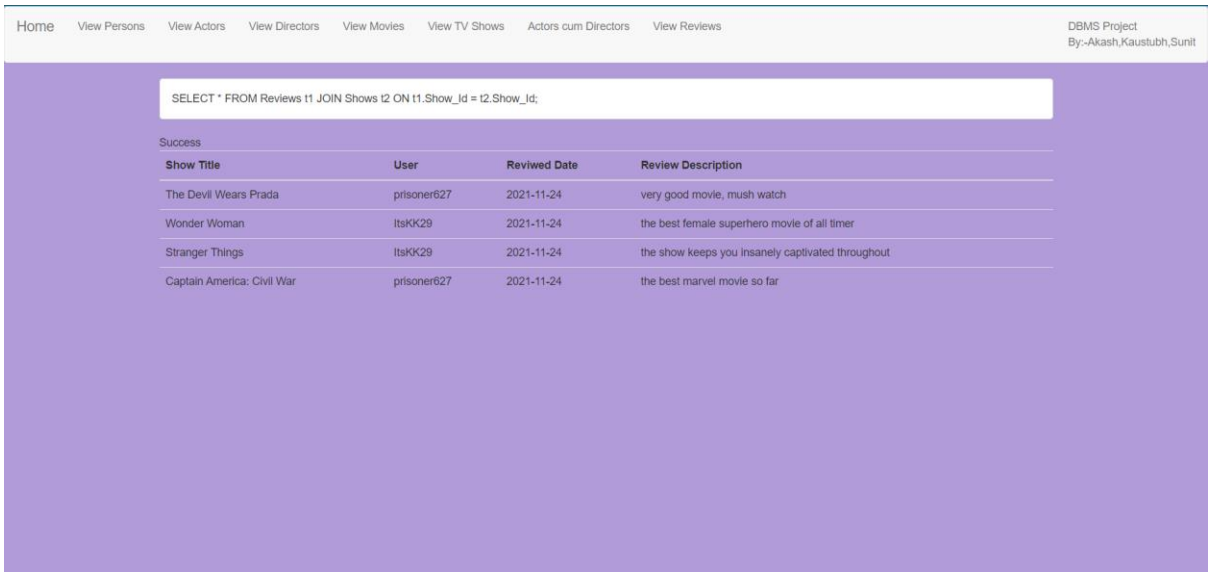
FirstName	LastName	DOB	Gender
Meryl	Streep	1990-08-15	F
Robert	Downey	1965-04-04	M
John	Krasinski	1979-10-20	M
Anthony	Russo	1970-02-03	M
Emma	Stone	1988-11-06	F
Gal	Gadot	1985-04-30	F
Patty	Jenkins	1971-07-24	F
Peter	Dinklage	1971-07-24	M
Milly	Brown	1971-07-24	F
Emily	Blunt	1983-02-23	F
Chris	Evans	1981-06-13	M
Chris	Pine	1980-08-26	M
Ryan	Gosling	1980-11-12	M

Movies UI:



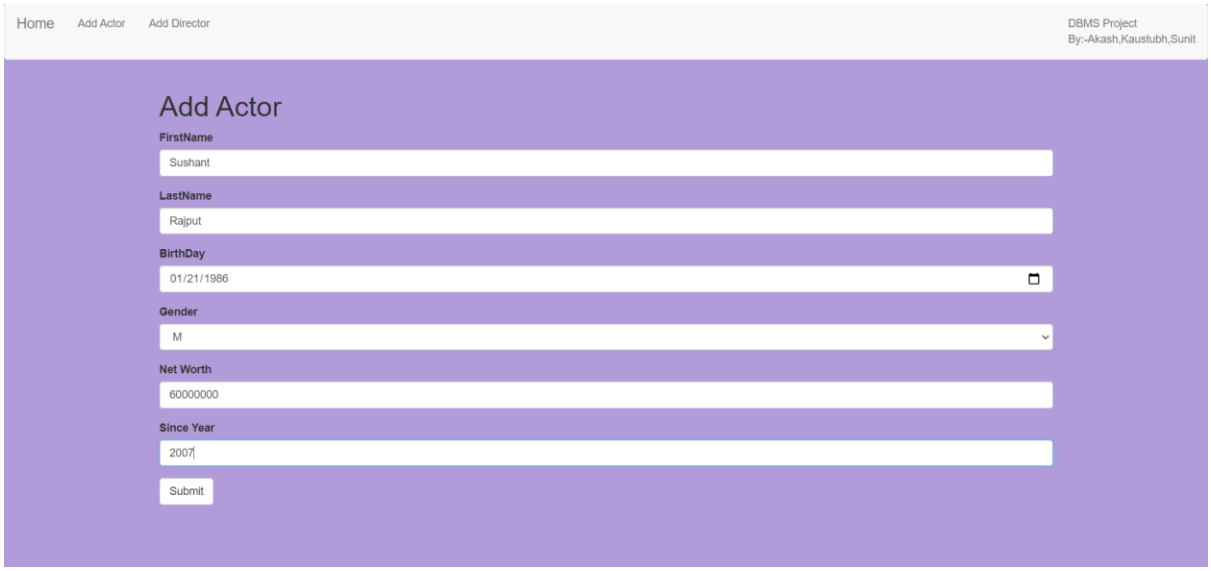
Movie	Certification	Language	Released ON	Duration
The Devil Wears Prada	PG-13	English	2006-06-30	109
Captain America: Civil War	PG-13	English	2016-05-06	147
A Quiet Place	PG-13	English	2018-04-06	90
Wonder Woman	PG-13	English	2017-06-02	141
La La Land	PG-13	English	2016-12-25	128
Despicable Me	PG	English	2010-07-09	95
Coco	PG	English	2017-11-22	105
Avengers: Infinity War	PG-13	English	2018-04-27	149
Birdman	R	English	2014-11-14	119
Before We Go	PG-13	English	2015-07-21	95
This is Us	TV-14	English	2002-01-25	102
A Walk to Remember	PG	English	2004-06-25	124

Reviews UI:



Show Title	User	Reviwed Date	Review Description
The Devil Wears Prada	prisoner627	2021-11-24	very good movie, mush watch
Wonder Woman	ItsKK29	2021-11-24	the best female superhero movie of all timer
Stranger Things	ItsKK29	2021-11-24	the show keeps you insanely captivated throughout
Captain America: Civil War	prisoner627	2021-11-24	the best marvel movie so far

Add Actor UI:



Source Code:

Transactions Description:

The transactions that we have implemented can broadly be divided into 3 parts.

1. Transactions to ‘View the existing Database’
2. Transactions to ‘Search the Database’
3. Transactions to allow User to ‘Add new entries into the Database’

Transactions to ‘View the existing Database’

1. View Persons

a. This allows us to view all the People present in the Database.

b. This is done by selecting all entries from

i. The ‘Person’ Table - which contains information about every single Person present in the database

c. SQL Code:

```
SELECT * FROM Person;
```

2. View Actors

a. This allows us to view all the Actors present in the Database.

b. This is done by performing a Join on

i. The ‘Actor’ table - which contains information about who among the

Persons are Actors,

ii. The ‘Person’ Table - which contains information about every single Person present in the database

c. SQL Code:

```
SELECT * FROM Actor t1 JOIN Person t2
ON t1.Person_Id = t2.Person_Id;
```

3. View Directors

a. This allows us to view all the Directors present in the Database.

b. This is done by performing a Join on

i. The ‘Director’ table - which contains information about who among the Persons are Directors,

ii. The ‘Person’ Table - which contains information about every single Person present in the database

c. SQL Code:

```
SELECT * FROM Director t1 JOIN Person t2
ON t1.Person_Id = t2.Person_Id;
```

4. View Movies

a. This allows us to view all the Movies present in the Database.

b. This is done by performing a Join on

i. The ‘Movie’ table - which contains information about which among the Shows are Movies,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series

c. SQL Code:

```
SELECT * FROM Movies t1 JOIN Shows t2
ON t1.Show_ID = t2.Show_Id;
```

5. View TV Shows

a. This allows us to view all the TV Shows present in the Database.

b. This is done by performing a Join on

i. The ‘TV Series’ table - which contains information about which among the Shows are TV Series,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series

c. SQL Code:

```
SELECT * FROM TVSeries t1 JOIN Shows t2
ON t1.Show_ID = t2.Show_Id;
```

6. Search People who are both Actors and Directors

a. This allows us to view all the people who are both Actors and Directors.

b. This is done by performing a Join on

i. The ‘Director’ table - which contains information about who among the Persons are Directors,

ii. The ‘Actor’ table - which contains information about who among the Persons are Actors,

iii. The ‘Person’ Table - which contains information about every single Person present in the database.

c. SQL Code:


```
SELECT * From Director t1 JOIN Actor t2
```

ON t1.Person_Id = t2.Person_Id JOIN Person t3

ON t1.Person_Id = t3.Person_Id;

7. View Reviews

a. This allows us to view all the Reviews provided by the Users for the Shows

existing in the Database.

b. This is done by performing a Join on

i. The ‘Reviews’ table - which contains information about the Reviews provided by the Users,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series

c. SQL Code:

SELECT * FROM Reviews t1 JOIN Shows t2

ON t1.Show_Id = t2.Show_Id;

Transactions to ‘Search the Database’

1. Search Shows of Actors

a. This allows us to Search the Database for all the Shows (Movies + TV Series) of

a particular actor. The input for the Actor is provided by the User.

b. This is done by performing a Join on

i. The ‘Acting’ Table - which contains the Shows and Actor pairs, i.e which actors acted in which Shows,

ii. The ‘Actor’ table - which contains information about who among the

Persons are Actors,

iii. The ‘Person’ Table - which contains information about every single Person present in the database,

iv. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.

c. SQL Code:

```
SELECT * FROM Acting t1 JOIN Actor t2
ON t1.Actor_Id = t2.Person_Id JOIN Person t3
ON t3.Person_Id = t2.Person_Id JOIN Shows t4
ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%Robert%' or t3.Last_Name LIKE '%Robert%';
```

2. Search Shows of Directors

a. Similar to the above case, this allows us to Search the Database for all the Shows of a particular Director. The input for the Director is provided by the User.

b. This is done by performing a Join on

i. The ‘Directing’ Table - which contains the Shows and Director pairs, i.e which Directors directed in which Shows,

ii. The ‘Director’ table - which contains information about who among the Persons are Directors,

iii. The ‘Person’ Table - which contains information about every single Person present in the database,

iv. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.

c. SQL Code:

```
SELECT * FROM Direction t1 JOIN Director t2
ON t1.Director_Id = t2.Person_Id JOIN Person t3
```

```
ON t3.Person_Id = t2.Person_Id JOIN Shows t4
ON t1.Show_Id = t4.Show_Id
WHERE t3.First_Name LIKE '%John%' or t3.Last_Name LIKE '%John%';
```

3. Search Actors of Shows

a. This allows us to Search the Database for all the Actors of a particular Show. The input for the Show is provided by the User.

b. This is done by performing a Join on

i. The ‘Actor’ table - which contains information about who among the Persons are Actors,

ii. The ‘Acting’ Table - which contains the Shows and Actor pairs, i.e which actors acted in which Shows,

iii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series,

iv. The ‘Person’ Table - which contains information about every single Person present in the database.

c. SQL Code:

```
SELECT * FROM Actor t1 JOIN Acting t2
ON t1.Person_Id = t2.Actor_Id JOIN Shows t3
ON t3.Show_Id = t2.Show_Id JOIN Person t4
ON t4.Person_Id = t1.Person_Id
WHERE t3.Title LIKE '%La La land%';
```

4. Search Rating Of Movie

a. This allows us to Search the Database for the Rating of a particular Movie. The input for the Movie is provided by the User.

b. This is done by performing a Join on

i. The ‘Movie’ table - which contains information about which among the Shows are Movies,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.

c. SQL Code:

```
SELECT * FROM Movies t1 JOIN Shows t2
ON t1.Show_Id = t2.Show_Id WHERE t2.Title LIKE '%Avengers%';
```

5. Search Movies by Year

a. This allows us to Search the Database for the Movies that were released in a particular year. The input for the Year is provided by the User.

b. This is done by performing a Join on

i. The ‘Movie’ table - which contains information about which among the Shows are Movies,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series.

c. SQL Code:

```
SELECT * FROM Movies t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id WHERE t1.Year = 2017;
```

6. Search Highest Grossing Movie by Year

a. This allows us to Search the Database for the Highest Grossing Movies of a particular year. The input for the Year is provided by the User.

b. This is done by performing a Join on

i. The ‘Box_Office_Collections’ table - which contains information about the Box Office Collections of a particular Movie,

ii. The ‘Shows’ Table - which contains information about every single Show

present in the database, Show includes both Movies + TV Series,

iii. The ‘Movie’ table - which contains information about which among the Shows are Movies.

c. The result provides all the Movies released in that year ordered in descending

order of their Box Office Collections, the first entry indicating the highest grossing

movie of that year.

d. SQL Code:

```
SELECT * FROM Box_Office_Collections t1 JOIN Shows t2

ON t1.Movie_Id = t2.Show_Id JOIN Movies t3 ON t1.Movie_Id = t3.Show_Id
WHERE t3.Year = '2017' ORDER BY Overall_Worldwide_Collections DESC;
```

7. Search Shows by Genre

a. This allows us to Search the Database for the Shows of a particular genre. The input for the Genre is provided by the User.

b. This is done by performing a Join on

i. The ‘In_Genre’ Table - which contains the Genre and Show pairs, i.e which Show belongs to which Genre,

ii. The ‘Shows’ Table - which contains information about every single Show present in the database, Show includes both Movies + TV Series,

iii. The ‘Genres’ table - which contains information about all the available Genres.

c. SQL Code:

```
SELECT * from In_Genre t1 JOIN Shows t2 ON t1.Show_Id = t2.Show_Id JOIN
Genres t3 ON t1.Genre_Id = t3.Genre_Id WHERE t3.Name = "Action"
```

1. Add Actor

a. This allows us to Add an Actor to the Database. The details of the Actor are provided by the User.

b. This is done by performing an Insert into both the Person as well as actor table.

i. The ‘Person’ Table - which contains information about every single Person present in the database. The Actor ID is obtained from the Person ID after inserting in to the Person Table.

ii. The ‘Actor’ table - which contains information about who among the Persons are Actors.

c. SQL Code:

```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB)

VALUES ('M','Mark','Ruffalo',null,'1967-09-22');

INSERT INTO Actor (Person_Id, Net_Worth, Since_Year)

VALUES ( (SELECT Person_Id FROM Person WHERE Gender = 'M' AND

First_Name = 'Mark' AND Last_Name = 'Ruffalo' AND DOB ='1967-09-
22'),30,1989);
```

2. Add Director

a. Similar to the above case, this allows us to Add a Director to the Database. The details of the Director are provided by the User.

b. This is done by performing an Insert into both the Person as well as Director table.

i. The ‘Person’ Table - which contains information about every single Person present in the database. The Director ID is obtained from the Person ID after inserting in to the Person Table.

ii. The ‘Director’ table - which contains information about who among the Persons are Directors.

c. SQL Code:

```
INSERT INTO Person (Gender, First_Name, Last_Name, Middle_Name, DOB)

VALUES ('M','Nick','Cassavetes',null,'1954-05-21');

INSERT INTO Director (Person_Id, Direction_Type, Since_Year) VALUES (
```

```
(SELECT Person_Id FROM Person WHERE Gender = 'M' AND First_Name ='Nick' AND

Last_Name = 'Cassavetes' AND DOB = '1954-05-21'),'Movie',1970);
```

Conclusion: By studying and applying the concepts of Database Management System from Group A and Group B, we implemented Movie Database System mini-project.