AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
ADDING VALUE TO ENGINEERING

Department of Computer

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.

This project aims at creating on ‘‘Movie Database System” which can be used by Admin and Users. The Admin to publish and Update the details of different different Movies and TV shows, where as User can use for finding details of movies and TV Shows.

While the IMDb movie database serves as a useful repository of movie information, it’s use as a source for aggregate movie reviews is limited. While sites like Rotten Tomatoes serve as a community portal for reviewers to come together as a community and collectively rate movies, they fall short in their ability to allow the user to quickly track the contributing artists that are part of the movie production (i.e. directors, actors, producers, etc.).

ACKNOWLEDGEMENT

We present the Database Management System Project report as part of the curriculum of the T.E. Computer Engineering. We wish to thank all the people who gave us unending support right from when the idea was conceived. We express sincere and profound thanks to our Database Management System professor and also guide **Prof Shilpa Pimpalkar**, and **HOD Mrs S. N. Zaware**, who is always ready to help with the most diverse problems that we have encountered along the way. We express sincere thanks to all staff and colleagues who have helped directly or indirectly in completing this seminar successfully.

AISSMS IOIT, Pune.

Index

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.

While the IMDb movie database serves as a useful repository of movie information, it’s use as a source for aggregate movie reviews is limited. While sites like Rotten Tomatoes serve as a community portal for reviewers to come together as a community and collectively rate movies, they fall short in their ability to allow the user to quickly track the contributing artists that are part of the movie production (i.e. directors, actors, producers, etc.). Additionally, box office receipts and weekly standings aren’t a component of either, but remain the focus of sites such as Hollywood Reporter. In order to create a more comprehensive site for Everything Movies, the database schema must be comprehensive enough to allow for multiple simultaneous queries (generated from HTML user forms through a JSP tag library architecture) through a “round-robin” JDBC connection pool, while still allowing for real time updates and contributions by the user community. Also, the schema must be designed to allow for table abstraction across a hardware topology with an index that exists upon its own network server (again for ease of scalability across a server topology as the connection pool grows to accommodate the anticipated user community.

“Movie Database System ” has been designed to computerized the following functions that are performed by the system:

1. View
   1.1 Details of All Person related to Movies or TV Shows
   1.2 Details of Movies and TV Shows
   1.3 Details of Reviews

2. Search
   2.1 Search Actors or directors or rest of cast of particular Movies or TV Shows.
   2.2 Search Movies or TV Shows of a particular Actor or Director.
   2.3 Rating of Movies
   2.4 Grossing of Movie or TV Shows worldwide
   2.5 Search Movies on based on Genres.
3. Add
   3.1 Add Actor and its Details
   3.2 Add Director and its Details
   3.3 Add Review
   2.1 Search Actors or directors or rest of cast of particular Movies or TV Shows.
4. Update
   4.1 Update by Admin.
   4.2 Update Suggestion by Users.

Requirement

Hardware Requirement Specification

Processor – AMD Ryzen 5 4600H with Radeon Graphics, 3000 Mhz, 6 Core(s), 12 Logical Processor(s)

RAM – 8GB D DR4 RAM

ROM – 512GB SSD and 1TB HDD

Operating System - Windows 10 Home

System Type - 64-bit Operating System,x64-based processor

Software Requirement Specification

Software Used:

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

Front-end:

- HTML, PHP
- CSS
- JS

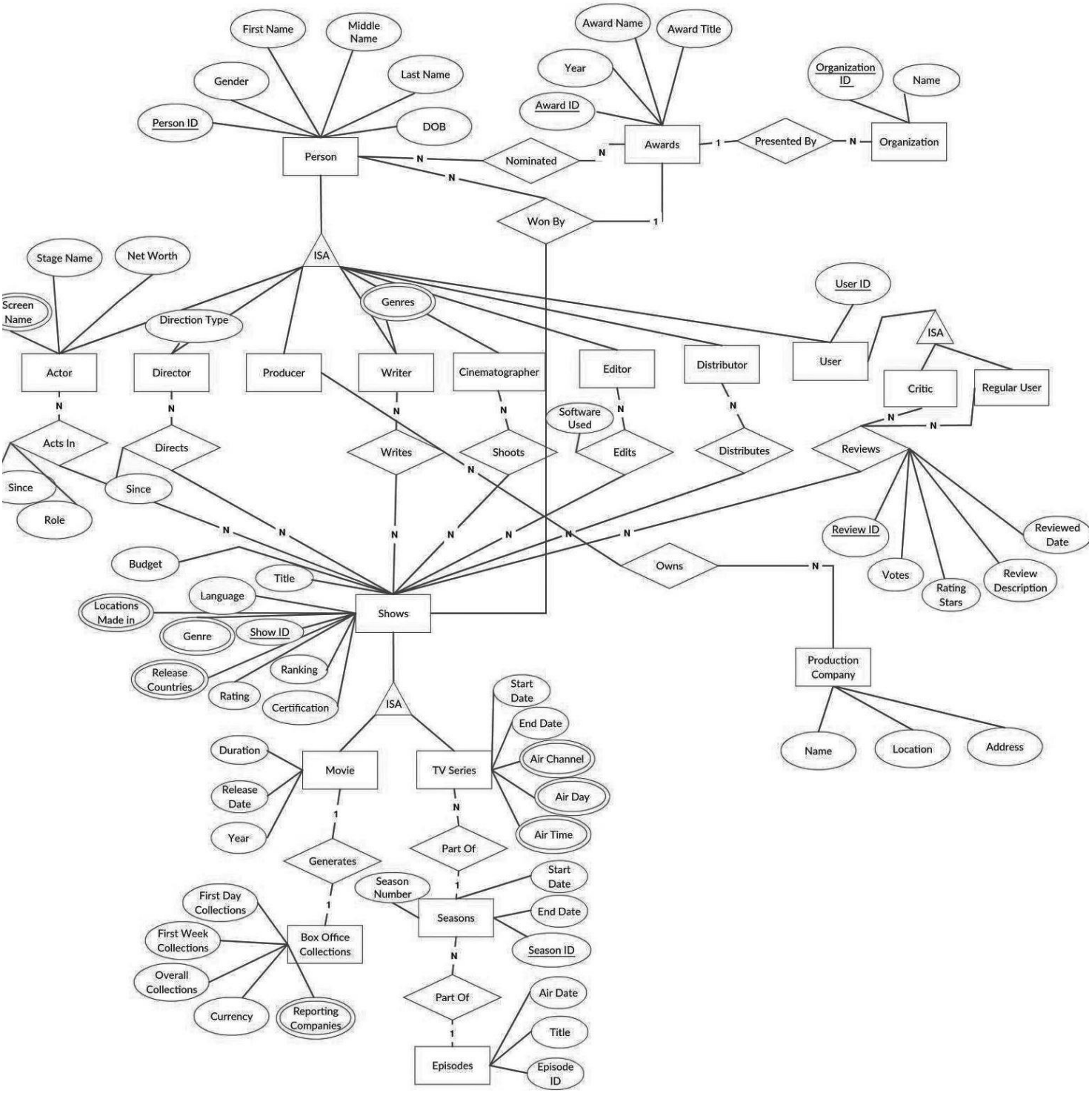
Back-end:

- PHP
- Shell -For shell scripting

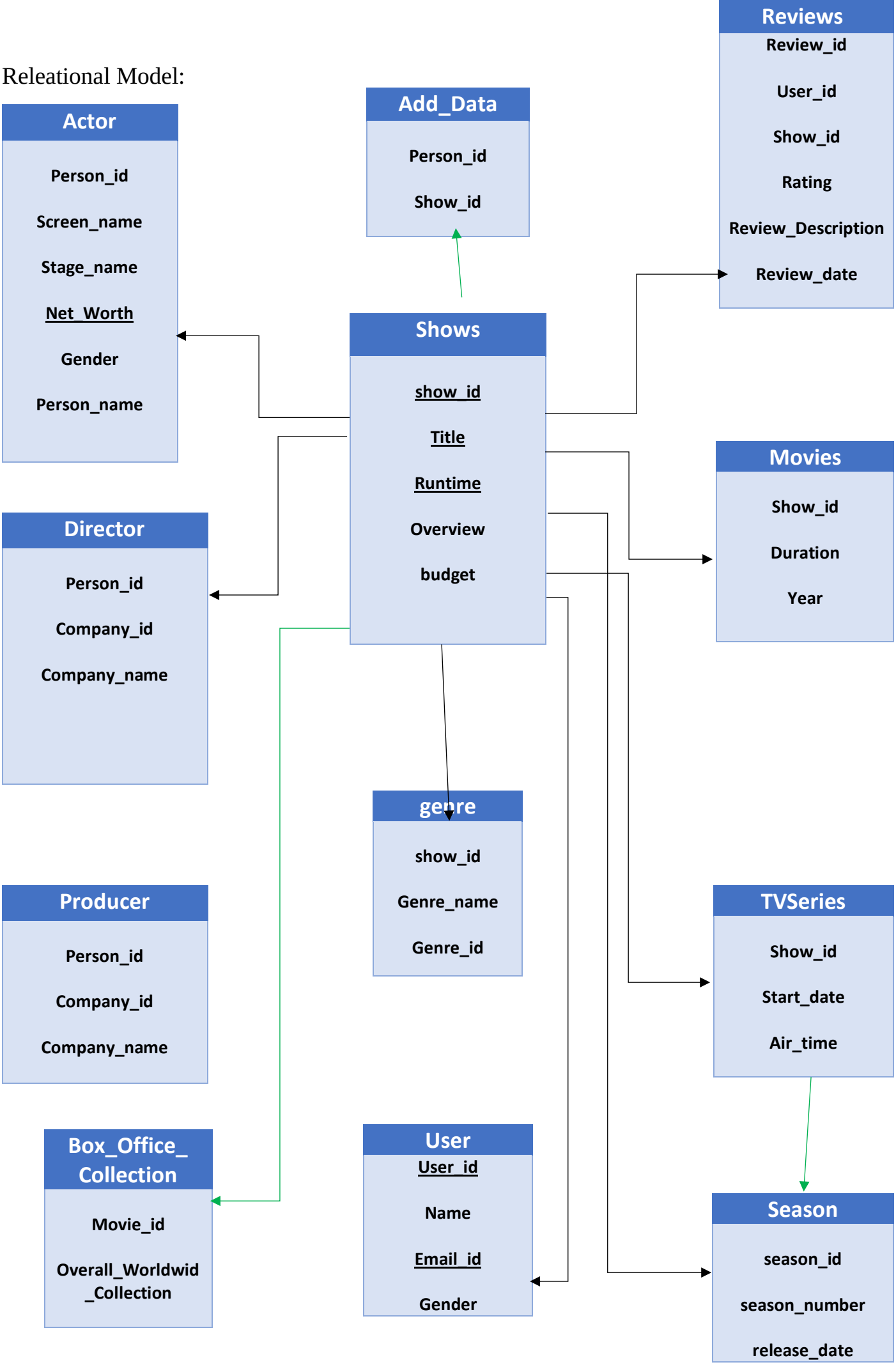
Database & Server:

- MySQL
- Apache Xampp

ER DIAGRAM


First Name
Middle Name
Last Name
Gender
Person ID
DOB
Person
Award Name
Award Title
Year
Award ID
Awards
Organization ID
Name
Organization
Presented By
Nominated
Won By
ISA
Stage Name
Net Worth
Screen Name
Direction Type
Genres
User ID
Actor
Director
Producer
Writer
Cinematographer
Editor
Distributor
User
ISA
Critic
Regular User
Acts In
Directs
Writes
Shoots
Software Used
Edits
Distributes
Reviews
Since
Role
Since
Budget
Title
Language
Locations Made in
Genre
Show ID
Release Countries
Ranking
Rating
Certification
Shows
Owns
Review ID
Votes
Rating Stars
Review Description
Reviewed Date
Production Company
Name
Location
Address
ISA
Duration
Release Date
Year
Movie
Start Date
End Date
Air Channel
Air Day
Air Time
TV Series
Generates
Part Of
Season Number
Seasons
Start Date
End Date
Season ID
First Day Collections
First Week Collections
Overall Collections
Currency
Box Office Collections
Reporting Companies
Part Of
Air Date
Title
Episodes
Episode ID

Releational Model:



Graphical User Interface

UI and Possible interactions with the Database:

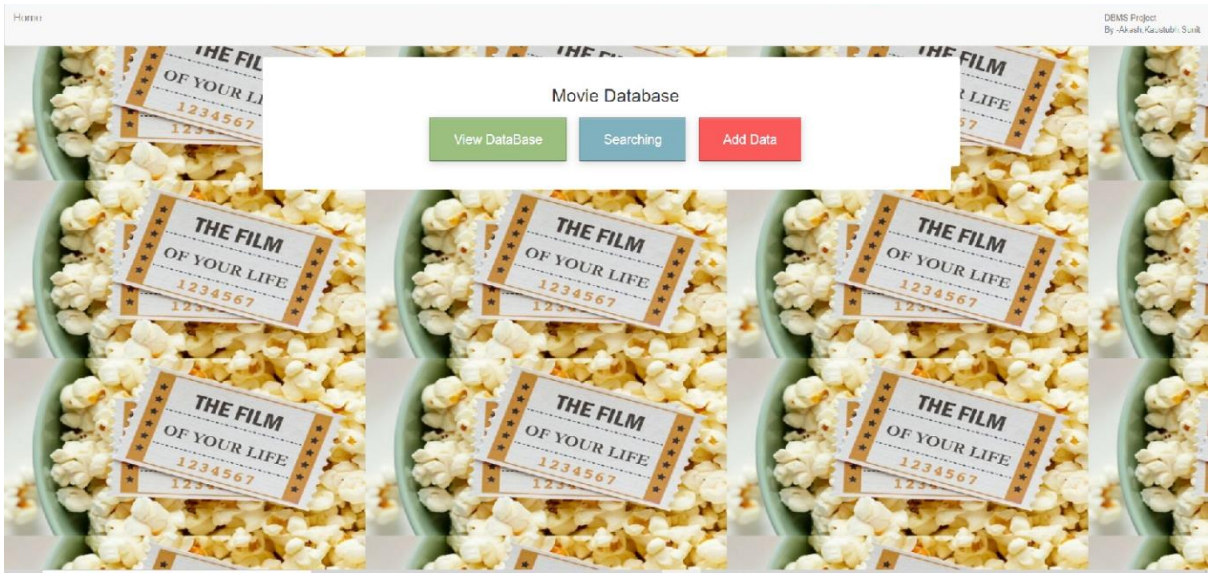
With the developed UI, on a broader level a user can perform three kinds of actions. View contents in database, Search for something and Add data.

View Contents:

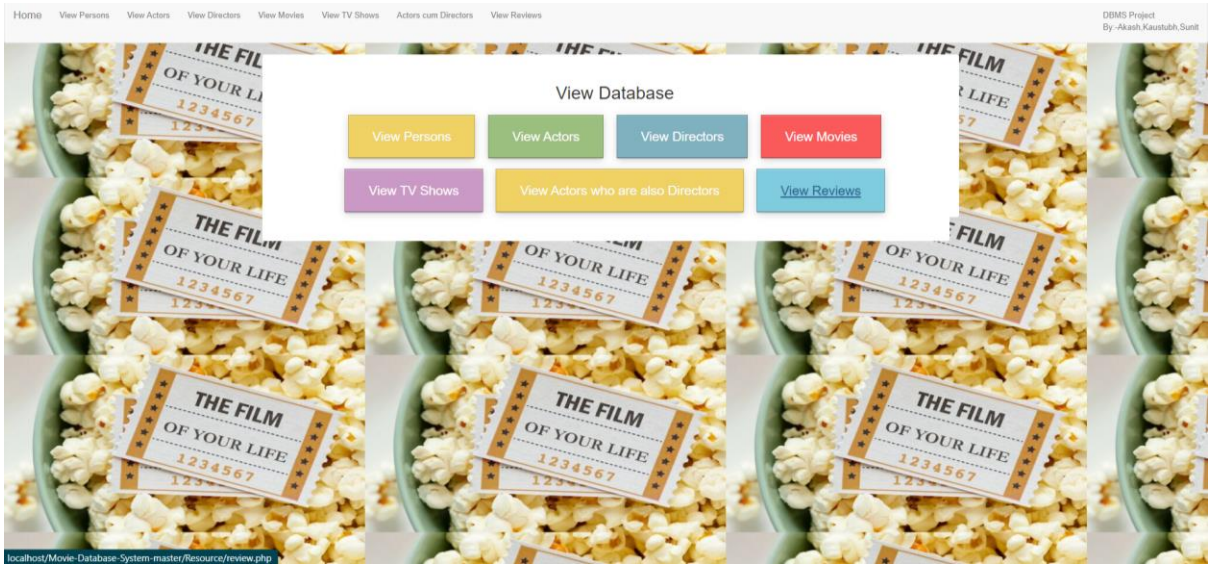
Using the developed system, a user can view things like,

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

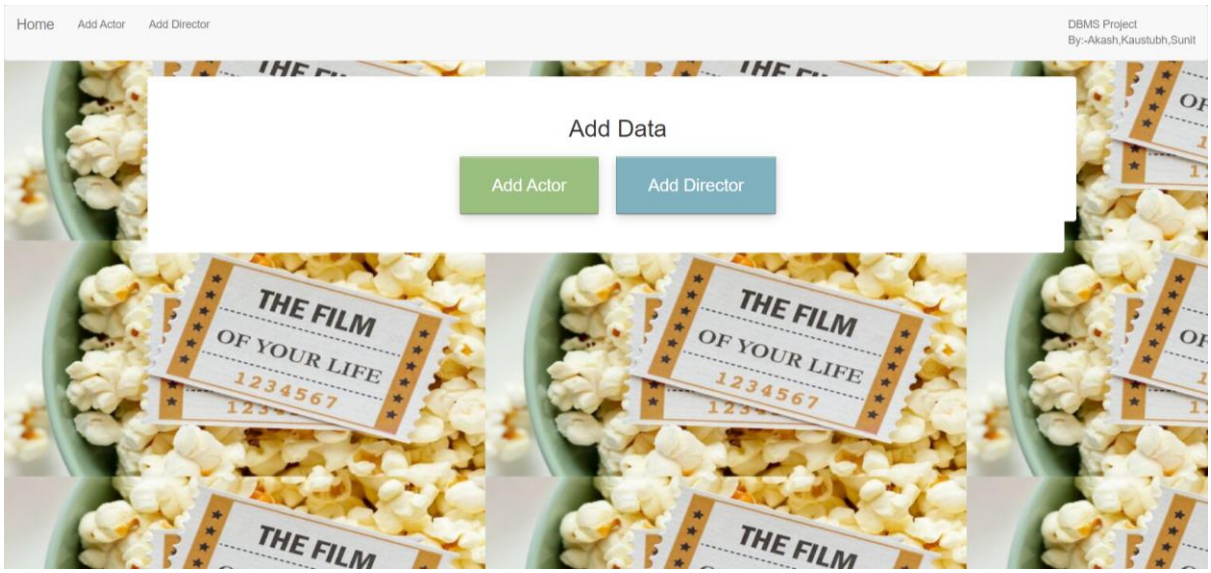
Home Page UI:



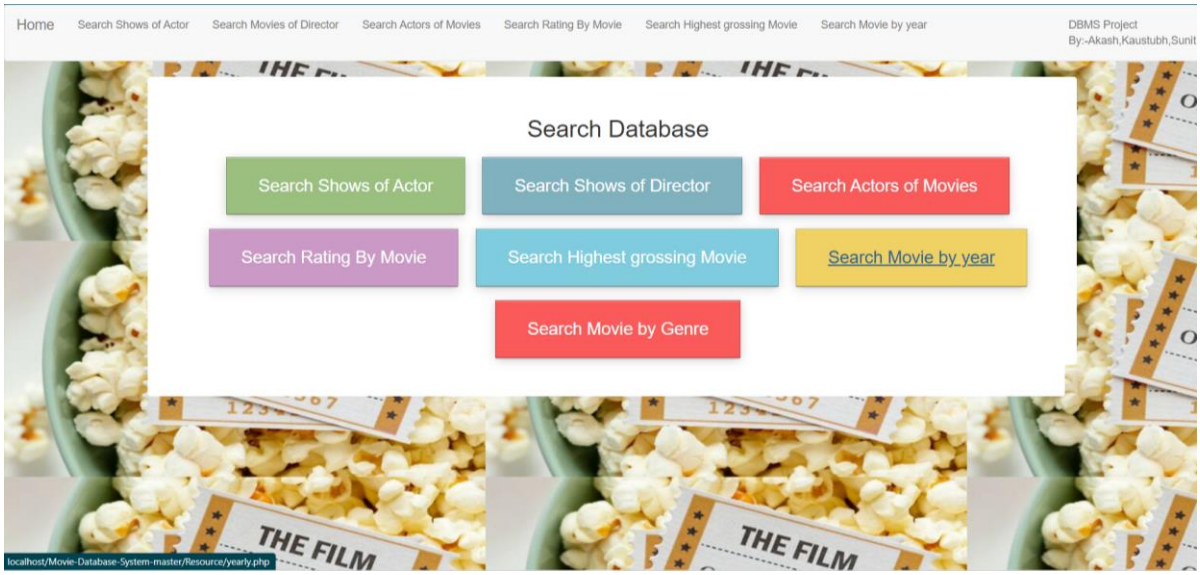
Movie Database View UI:



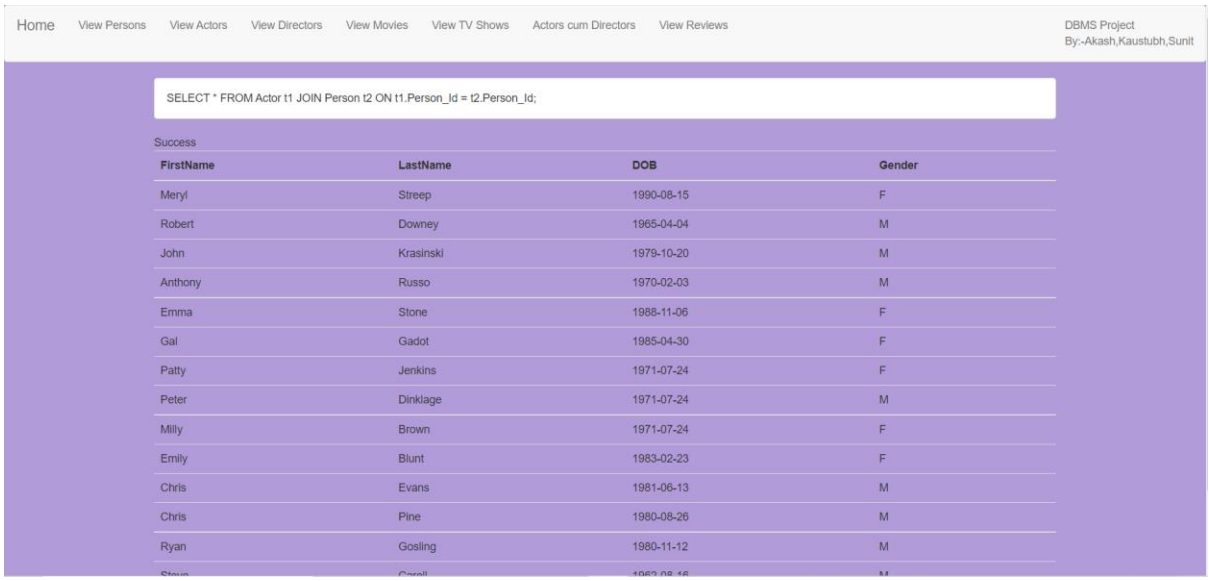
Add Actor Page UI:



Search Database UI:

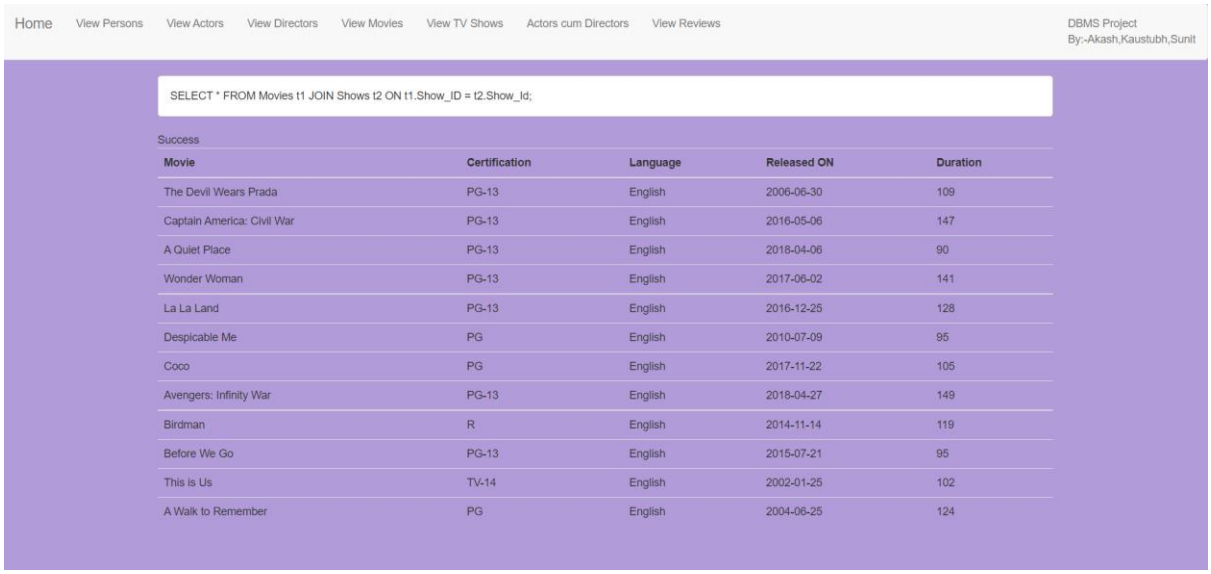


Actor UI:

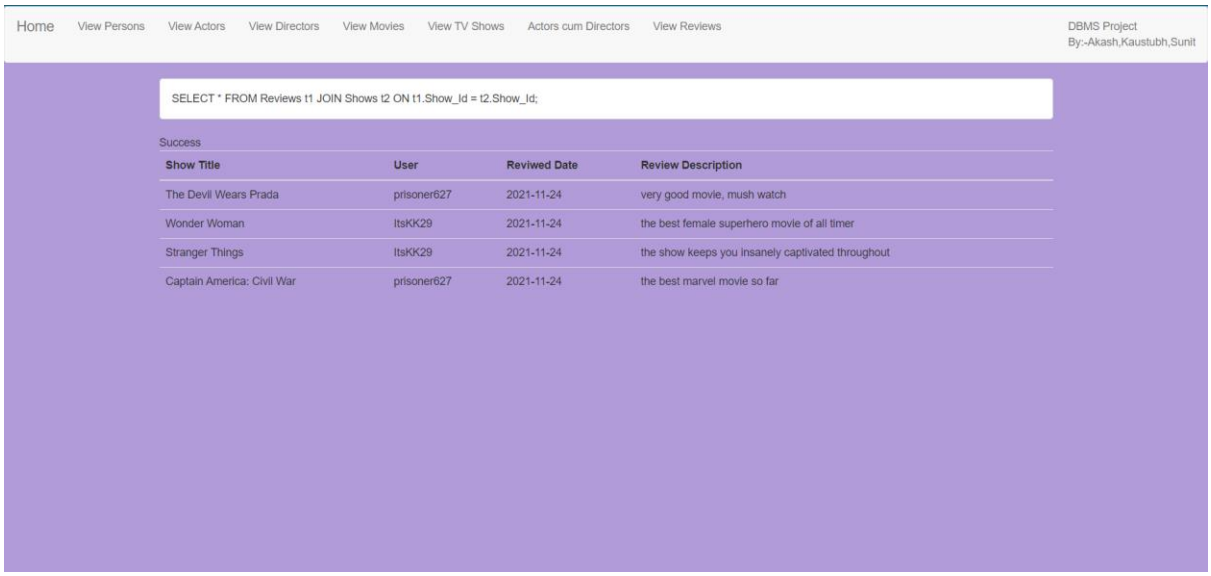


FirstName	LastName	DOB	Gender
Meryl	Streep	1990-08-15	F
Robert	Downey	1965-04-04	M
John	Krasinski	1979-10-20	M
Anthony	Russo	1970-02-03	M
Emma	Stone	1988-11-06	F
Gal	Gadot	1985-04-30	F
Patty	Jenkins	1971-07-24	F
Peter	Dinklage	1971-07-24	M
Milly	Brown	1971-07-24	F
Emily	Blunt	1983-02-23	F
Chris	Evans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

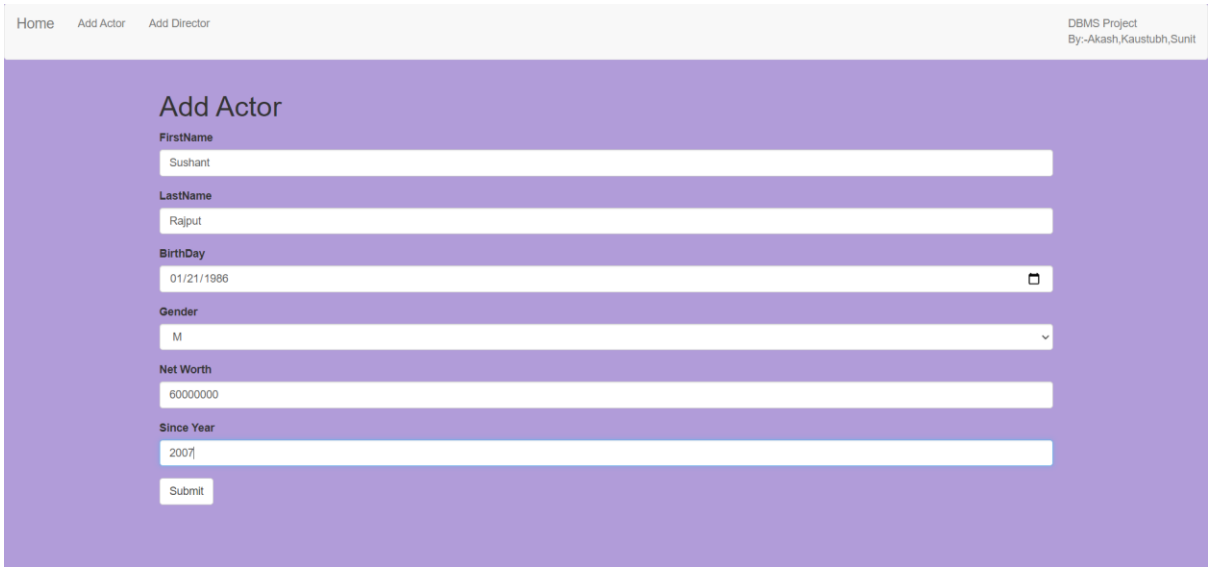
Movies UI:



Reviews UI:



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);

Testing Document

Sr. No.	User Case	Expected I/P	Expected O/P	Alternative	Actual O/P
1.	Home Page	Choice to find out details which is needed.	Display Movie Database Choices (View, Search and Add)	Not Found	Display Movie Database Choices (View,Search and Add)
2.	View Database Page	Choice to find out details of Movies, TV Shows, Actors, and Cast.	Display View Movie Database Choices (Person, Actor, Director, Movies, TV Shows, Reviews)	Not Found	Display View Database Choices (Person, Actor, Director, Movies, TV Shows, Reviews)
3.	Search Database Page	Choice to find out details of particular Movies, TV Shows, Actors, and Cast.	Display Search Movie Database Choices (Shows of Actor, Shows of Director, Actors of Movies, Rating by Movie, Highest Grossing Movie, Movie by Year, Movie by Genre)	Not Found	Display Search Movie Database Choices (Shows of Actor, Shows of Director, Actors of Movies, Rating by Movie, Highest Grossing Movie, Movie by Year, Movie by Genre)
4.	Add Data Page	Choice To add Data in Database (Actor, Director and Review)	Display Add Data Choices (Actor and Director)	Not Found	Display Add Data Choices (Actor and Director)
5.	Person Page	---	Details of all person related to movie casts	Not Found	Details of all person related to movie casts
6.	Actors Page	---	Details of all actors	Not Found	Details of all actors
7.	Directors Page	---	Details of Directors	Not Found	Details of Directors

8.	Movies Page	---	Details of Movies	Not Found	Details of Movies
9.	TV Shows Page	---	Details of TV Shows	Not Found	Details of TV Shows
10.	Reviews Page	---	Reviews by Users	Not Found	Reviews by Users
11.	Shows of Actor Page	Name of Actor	Details of Movies and TV Shows of Actor	Not Found	Details of Movies and TV Shows of Actor
12.	Shows of Director Page	Name of Director	Details of Movies and TV Shows directed by Director	Not Found	Details of Movies and TV Shows directed by Director
13.	Actors of Movies Page	Name of Movies	Details of Actor Played in Movie	Not Found	Details of Actor Played in Movie
14.	Rating of Movie Page	Name of Movies	Details And Rating of Movie	Not Found	Details And Rating of Movie
15.	Highest Grossing Movie Page	Year	Release Date, Budgets and Revenue of all movies of Year	Not Found	Release Date, Budgets and Revenue of all movies of Year
16.	Search by Year Page	Year	Details of Movies and TV Shows of Year	Not Found	Details of Movies and TV Shows of Year
17.	Movies By Genre Page	Genre	Rating, Certification and Movie name of Genre Type	Not Found	Rating, Certification and Movie name of Genre Type
18.	Add Actor Page	Details of Actor (First Name, Last Name, Birth Date, Gender, Net Worth, Since Year)	Display Query and Success Message	Not Found	Display Query and Success Message
19.	Add Director Page	Details of Director (First Name, Last Name, Birth Date, Gender, Net Worth, Since Year)	Display Query and Success Message	Not Found	Display Query and Success Message
20.	Add Review Page	Movie Name, User Id, Date, Review Description	Display Query and Success Message	Not Found	Display Query and Success Message

Conclusion

The entire project has been developed and deployed as per the need and requirement of User, its is found to be bog free as per the Testing standard that are Implemented.

This project provides an Easy and Effective Mechanism about Movies and TV Shows Details.

Thus we have performed and implemented mysql database application named "Movie Database System" using PHP Xampp and MySQL which shows complete movie based schema and data on a web based application.