Project Submission : Mock Popcorn

CSN – 351 DBMS 2017-18

Part I

Submitted By:

Ayush Chauhan 14115027

Kunal Bansal 14116035

Shobhit Mittal 14116061

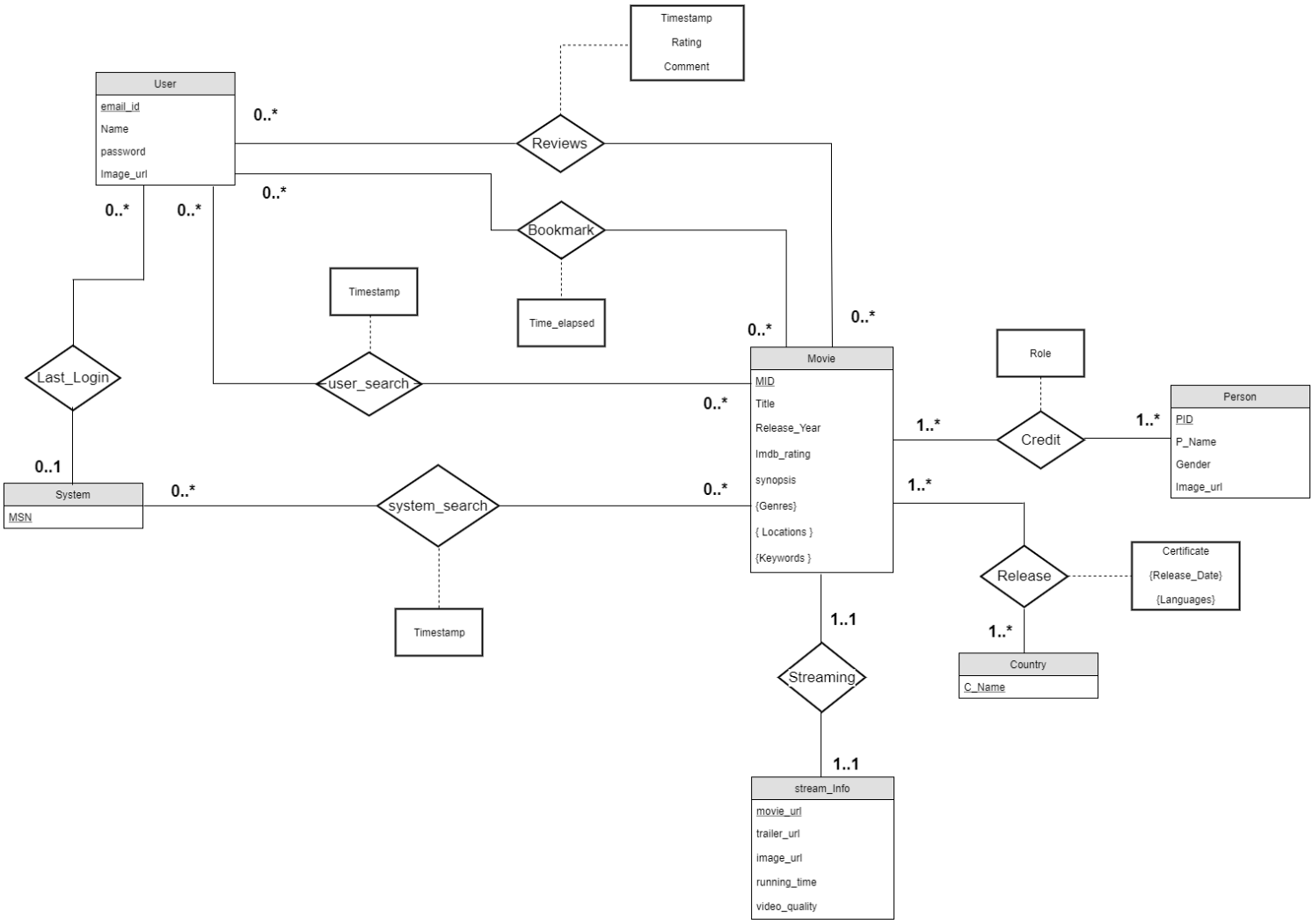
Joydeep Das 14116032

Shubham Agarwal 14116063

I. Assumptions

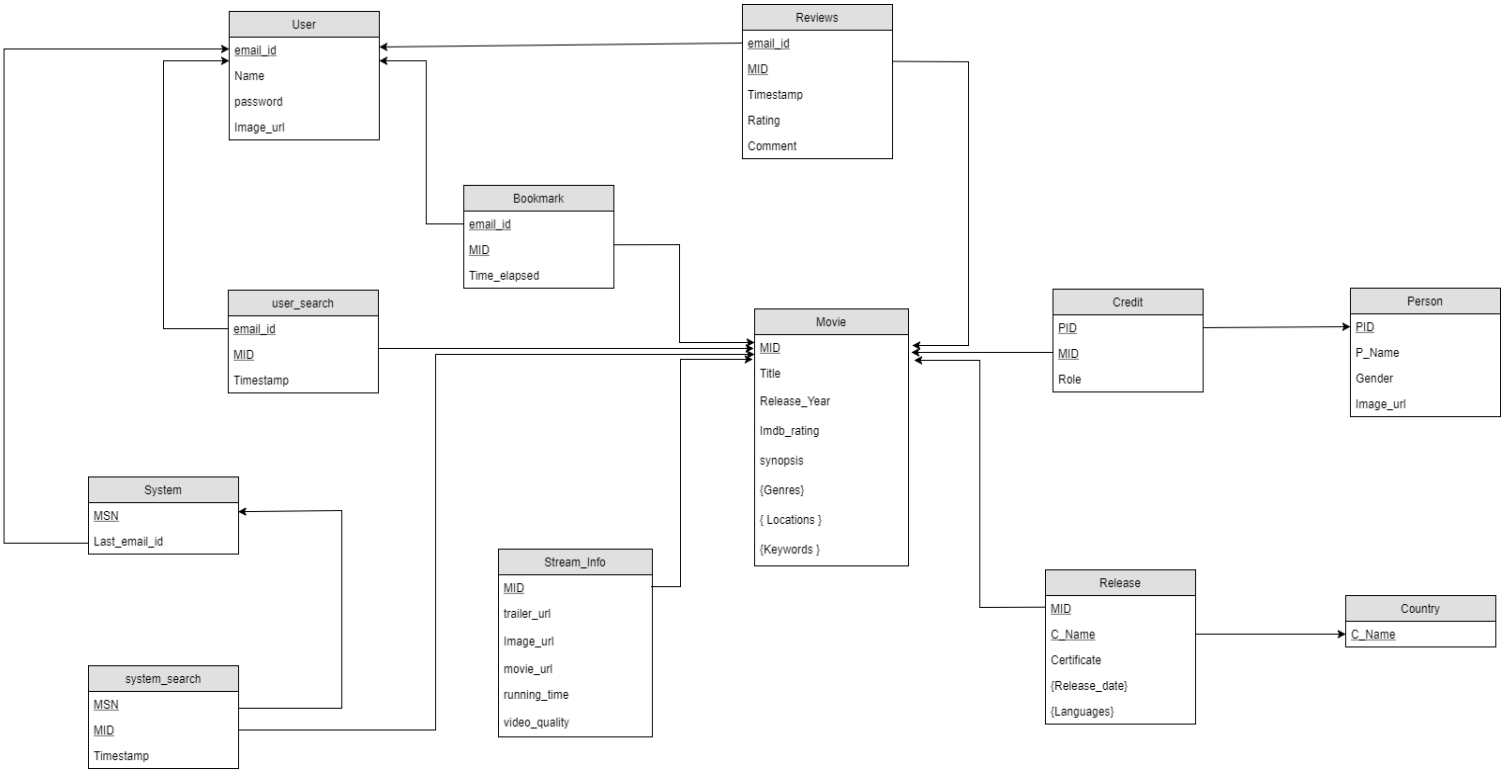
- Video quality is considered to be a single valued attribute.
- No two movies with same title are released in the same year.
- Synopsis is unique for every movie. Movies with same title or movie sequels don't have the same synopsis.
- Each user can access his account only through one system at a time for the purpose of submitting reviews. Although, an account may be logged in through multiple systems at a time.

II. E-R Diagram



III. Schema Diagram

Keeping in mind all the cardinality and integrity constraints, the above entities and relations are represented as tables in the following Schema diagram:



IV. Functional Dependencies

Movie :

MID ->Title, Release_Year, Imdb_rating, Synopsis, Genres, Locations, Keywords

(Title, Release_Year) ->MID, Imdb_rating, Synopsis,Genres,Locations, Keywords

Synopsis -> MID, Title, Release_Year, Imdb_rating, Genres, Locations, Keywords

Person :

PID -> P_Name, Gender, Image_url

Image_url -> PID, P_Name, Gender

Credit :

(MID,PID) -> Role

Country :

No non-trivial FD's.

Release :

(MID, C_Name) -> Certificate, Release_Date, Languages

Stream_Info:

MID -> Movie_url, Image_url, Running_time, Video_quality

Movie_url -> MID, Trailer_url, Image_url, Running_time, Video_quality

Trailer_url -> MID, Movie_url, Image_url, Running_time, Video_url

Image_url -> MID, Movie_url, Trailer_url, Running_time, Video_quality

User:

Email_Id -> Name, Password, Image_url

Reviews:

(Email_id, MID) -> Timestamp, Rating, Comment

(Email_id, timestamp) -> MID, Rating, Comment

Bookmarks:

(Email_id, MID) -> time_elapsed

System:

MSN -> Last_email_id

User_Search:

(Email_id, MID) -> Timestamp

(Email_id, Timestamp) -> MID

System_search:

(MID, MSN) -> timestamp

(MSN, Timestamp) -> MID

V. Minimal Cover

Movie:

Step 1.

MID -> Title

MID -> Release_Year

MID -> Imdb_rating

MID -> Synopsis

MID -> Genres

MID -> Locations

MID -> Keywords

(Title, Release_Year) -> MID

(Title, Release_Year) -> Imdb_rating

(Title, Release_Year) -> Synopsis

(Title, Release_Year) -> Genres

(Title, Release_Year) -> Locations

(Title, Release_Year) -> Keywords

Synopsis -> MID

Synopsis ->Title

Synopsis -> Release_Year

Synopsis -> Imdb_rating

Synopsis -> Genres

Synopsis -> Locations

-> Keywords

<u>Step 2:</u>

$\{Release\_Year\}^+ = \{ Release\_Year \}$

$\{Title\}^+ = \{Title \}$

Hence, no reduction possible.

<u>Step 3:</u>

MID -> Title

MID -> Release_Year

MID -> Imdb_rating

MID -> Synopsis

MID -> Genres

MID -> Locations

MID -> Keywords

(Title, Release_Year) -> MID

Synopsis -> MID

Person:

<u>Step 1:</u>

PID -> P_Name

PID -> Gender

PID -> Image_url

Image_url -> PID

Image_url -> P_Name

Image_url -> Gender

<u>Step 2 :</u>

No reduction possible.

<u>Step 3:</u>

PID -> P_Name

PID -> Gender

PID -> Image_url

Image_url -> PID

Credit:

(MID,PID) -> Role

No reductions possible.

Country:

No non-trivial FD's

Release:

(MID, C_Name) -> Certificate

(MID, C_Name) -> Release_Date

(MID, C_Name) -> Languages

No reductions possible.

Stream_Info:

<u>Step 1:</u>

MID -> Movie_url

MID ->Trailer_url

MID -> Image_url

MID -> Running_time

MID -> Video_quality

Movie_url -> MID

Movie_url -> Trailer_url

Movie_url -> Image_url

Movie_url -> Running_time

Movie_url -> Video_quality

Trailer_url -> MID

Trailer_url -> Movie_url

Trailer_url -> Image_url

Trailer_url -> Running_time

Trailer_url -> Video_quality

Image_url -> MID

Image_url -> Trailer_url

Image_url -> Movie_url

Image_url -> Running_time

Image_url -> Video_quality

Step 2:

No reduction possible.

Step 3:

MID -> Movie_url

MID ->Trailer_url

MID -> Image_url

MID -> Running_time

MID -> Video_quality

Movie_url -> MID

Trailer_url -> Movie_url

Image_url -> Trailer_url

User:

Email_Id -> Name

Email_Id -> Password

Email_Id -> Image_url

No reductions possible.

Reviews:

Step 1:

(Email_id, MID) -> Timestamp

(Email_id, MID) -> Rating

(Email_id, MID) -> Comment

(Email_id, timestamp) -> MID

(Email_id, timestamp) -> Rating

(Email_id, timestamp) -> Comment

Step 2:

No reductions possible.

Step 3:

(Email_id, MID) -> Timestamp

(Email_id, MID) -> Rating

(Email_id, MID) -> Comment

(Email_id, timestamp) -> MID

Bookmarks:

(Email_id, MID) -> time_elapsed

No reductions possible.

System:

MSN -> Last_email_id

No reductions possible.

User_Search:

(Email_id, MID) -> Timestamp

(Email_id, Timestamp) -> MID

No reductions possible.

System_search:

(MID, MSN) -> timestamp

(MSN, Timestamp) -> MID

No reductions possible.

VI. Normalization

The keys, prime and nonprime attributes for each table are written before decomposition. The decomposition up to BCNF is such that all the dependencies are preserved and the decomposed tables form a lossless join.

Movie:

MID -> Title

MID -> Release_Year

MID -> Imdb_rating

MID -> Synopsis

MID -> Genres

MID -> Locations

MID -> Keywords

(Title, Release_Year) -> MID

Synopsis -> MID

Candidate keys: { (MID), (Title, Release_Year), (Synopsis) }

Prime Attributes: { MID, Title, Release_Year, Synopsis }

Non-Prime Attributes: { Imdb_rating, Genres, Locations, Keywords }

<u>1NF:</u>

Multi-valued attributes: {Genres, Locations, Keywords}

Therefore, making all attributes atomic for 1NF, the decomposed tables are:

Movie: {MID, Title, Release_Year, Imdb_rating, Synopsis}

Keys: { (MID), (Title, Release_Year), (Synopsis) }

Movie_Genres: {MID, Genres}

Keys: { (MID, Genres) }

Movie_Locations: {MID, Locations}

Keys: { (MID, Locations) }

Movie_keywords: {MID, Keywords}

Keys: { (MID, Keywords) }

2NF:

Movie:

The only Non-prime attribute (*Imdb_rating*) is fully functionally dependent on all the candidate keys of Movie. Therefore, the table is already in 2NF.

Movie_Genres:

No non-prime attribute.

Movie_Locations:

No non-prime attribute.

Movie_keywords:

No non-prime attribute.

3NF:

Movie:

No transitive dependency.

Movie_Genres:

No non-prime attribute.

Movie_Locations:

No non-prime attribute.

Movie_keywords:

No non-prime attribute.

BCNF:

Movie:

For every FD: X->Y, X is a superkey, hence already in BCNF.

Movie_Genres:

Already in BCNF.

Movie_Locations:

Already in BCNF.

Movie_keywords:

Already in BCNF.

4NF:

Movie:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

Movie_Genres, Movie_Locations, Movie_keywords:

No non-trivial MVD, hence already in 4NF.

5NF:

Movie:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

Movie_Genres, Movie_Locations, Movie_keywords:

No non-trivial JD, hence already in 5NF.

Person:

PID -> P_Name

PID -> Gender

PID -> Image_url

Image_url -> PID

Candidate keys: { (PID), (Image_url) }

Prime Attributes: { PID, Image_url }

Non-Prime Attributes: { P_Name, Gender }

1NF:

No multivalued attribute, already in 1NF.

2NF:

All the candidate keys are single attribute keys; therefore the table is in 2NF.

3NF:

None of the non-prime attributes have a transitive dependency on any of the keys.

BCNF:

For every FD: X->Y, X is a superkey, hence already in BCNF.

4NF:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

5NF:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

Credit :

(MID,PID) -> Role

Candidate keys: { (MID,PID) }

Prime Attributes: { MID, PID}

Non-Prime Attributes: { Role }

1NF:

No multivalued attribute, already in 1NF.

2NF:

The non-prime attribute Role is fully functionally dependent on the key (*MID, PID*).

3NF:

No transitive dependency from the key to any non-prime attribute.

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

No MVD from a non-superkey tuple; therefore, already in 4NF.

5NF:

No non-trivial JD exists in this table; hence, already in 5NF.

Country:

This table contains only one attribute, *C_Name*. Thus, this table satisfies conditions for all the normalised forms, i.e. 1NF, 2NF, 3NF, BCNF, 4NF, 5NF.

Release:

(MID, C_Name) -> Certificate

(MID, C_Name) -> Release_Date

(MID, C_Name) -> Languages

Candidate keys: { (MID, C_Name) }

Prime Attributes: { MID, C_Name}

Non-Prime Attributes: { Certificate, Release_Date, Languages}

1NF:

Multi-valued attributes: { Release_Date, Languages}

Therefore, making all attributes atomic for 1NF, the decomposed tables are:

Movie_Certificates: { MID, C_Name, Certificate}

Keys : { (MID, C_Name) }

Movie_dates: { MID, C_Name, Release_Date}

Keys: { (MID, C_Name, Release_Date) }

Movie_Languages: { MID, C_Name, Languages}

Keys: { (MID, C_Name, Languages) }

2NF:

Movie_Certificates:

Non-prime attribute (*Certificate*) is fully functionally dependent on the key (*MID, C_Name*).

Movie_dates:

No non-prime attribute, already in 2NF.

Movie_Languages:

No non-prime attribute, already in 2NF.

3NF:

Movie_Certificates:

No transitive dependency from the key to non-prime attribute, hence already in 3NF.

Movie_dates:

No non-prime attribute, already in 3NF.

Movie_Languages:

No non-prime attribute, already in 3NF.

BCNF:

Movie_Certificates:

The LHS of the only FD is a superkey, hence already in BCNF.

Movie_dates:

No non-trivial FD, hence already in BCNF.

Movie_Languages:

No non-trivial FD, hence already in BCNF.

4NF:

No non-trivial MVD in any of the decomposed tables, hence all of them are in 4NF.

5NF:

No non-trivial JD in any of the decomposed tables, hence all of them are in 5NF.

Stream_Info:

MID -> Movie_url

MID ->Trailer_url

MID -> Image_url

MID -> Running_time

MID -> Video_quality

Movie_url -> MID

Trailer_url -> Movie_url

Image_url -> Trailer_url

Candidate keys: { (MID), (Movie_url), (Trailer_url), (Image_url) }

Prime Attributes: { MID, Movie_url, Trailer_url, Image_url }

Non-Prime Attributes: { Running_time, Video_quality}

<u>1NF:</u>

No multivalued attribute, already in 1NF.

<u>2NF:</u>

All the candidate keys are single attribute keys, therefore the table is in 2NF.

<u>3NF:</u>

There is no transitive dependency from the key to any non-prime attribute.

<u>BCNF:</u>

For every FD: X->Y, X is a superkey, already in BCNF.

<u>4NF:</u>

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

<u>5NF:</u>

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

User:

Email_Id -> Name

Email_Id -> Password

Email_Id -> Image_url

Candidate keys: { Email_id }

Prime Attributes: { Email_id }

Non_prime Attributes: { Name, Password, Image_url }

1NF:

No multivalued attribute, already in 1NF.

2NF:

The only candidate key (*Email_id*) is a single attribute key; therefore the table is in 2NF.

3NF:

No transitive dependency exists in the table.

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

No non-trivial MVD, hence already in 4NF.

5NF:

No non-trivial JD, hence already in 5NF.

Reviews:

(Email_id, MID) -> Timestamp

(Email_id, MID) -> Rating

(Email_id, MID) -> Comment

(Email_id, timestamp) -> MID

Candidate keys: { (MID, Email_id), (Email_id, Timestamp) }

Prime Attributes: { MID, Email_id, Timestamp }

Non_prime Attributes: { Rating, Comment }

1NF:

No multivalued attribute, already in 1NF.

2NF:

All non-prime attributes are fully functionally dependent on both the keys of the table.

3NF:

No transitive dependency.

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

5NF:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

Bookmarks:

(Email_id, MID) -> time_elapsed

Candidate keys: { (MID, Email_id) }

Prime Attributes: { MID, Email_id }

Non_prime Attributes: { time_elapsed }

1NF:

No multivalued attribute, already in 1NF.

2NF:

The only non-prime attribute (*time_elapsed*) are fully functionally dependent on the key.

3NF:

No transitive dependency. .

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

5NF:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

System:

MSN -> Last_email_id

Candidate keys: { MSN }

Prime Attributes: { MSN }

Non_prime Attributes: { Last_email_id }

1NF:

No multivalued attribute, already in 1NF.

2NF:

The only key of the table, *MSN* is a single attribute key. Thus, the table is in 2NF.

3NF:

The table has only 1 FD, therefore it is in 3NF.

BCNF:

For the only FD, *MSN -> Last_email_id*, *MSN* is a superkey; hence, already in BCNF.

4NF:

No non-trivial MVD exists in the table; hence, already in 4NF.

5NF:

No non-trivial JD exists in the table; hence, already in 5NF.

User_Search:

(Email_id, MID) -> Timestamp

(Email_id, Timestamp) -> MID

Candidate keys: { (MID, Email_id), (Email_id, Timestamp) }

Prime Attributes: { MID, Email_id, Timestamp }

Non_prime Attributes: None

1NF:

No multivalued attribute, already in 1NF.

2NF:

No non-prime attribute, already in 2NF.

3NF:

No non-prime attribute, already in 3NF.

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

5NF:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

System_search:

(MID, MSN) -> timestamp

(MSN, Timestamp) -> MID.

Candidate keys: { (MID, MSN), (MSN, Timestamp) }

Prime Attributes: { MID, MSN, Timestamp }

Non_prime Attributes: None

1NF:

No multivalued attribute, already in 1NF.

2NF:

No non-prime attribute, already in 2NF.

3NF:

No non-prime attribute, already in 3NF.

BCNF:

For every FD: X->Y, X is a superkey, already in BCNF.

4NF:

There is no non-trivial MVD such that the LHS is not a superkey. Hence, the table is already in 4NF.

5NF:

Amongst all the non-trivial JDs that can be formed, every decomposed table turns out to be a superkey of the original table. Hence, the table is already in 5NF.

VII. Final Normalized Tables

Movie: {MID, Title, Release_Year, Imdb_rating, Synopsis}

Movie_Genres: {MID, Genres}

Movie_Locations: {MID, Locations}

Movie_keywords: {MID, Keywords}

Person: {PID, P_Name, Gender, Image_url}

Credit: {MID, PID, Role}

Country: {C_Name}

Movie_Certificates: { MID, C_Name, Certificate}

Movie_dates: { MID, C_Name, Release_Date}

Movie_Languages: { MID, C_Name, Languages}

Stream_Info: {MID, Movie_url, Trailer_url, Image_url, Running_time, Video_quality}

User: {Email_Id, Name, Password, Image_url}

Reviews: {Email_id, MID, Timestamp, Rating, Comment}

Bookmarks: {Email_id, MID, time_elapsed}

System: {MSN, Last_email_id}

User_search: {Email_id, MID, Timestamp}

System_search: {MSN, MID, Timestamp}

Part II

SQL Commands & Features Implemented

Commands to create the tables:

```
create database Mock_Popcorn;
use Mock_Popcorn;
CREATE TABLE Movie
(
MID varchar(10) not null ,
Title varchar(50) not null ,
Release_Year year not null,
Imdb_rating float(1) not null,
Synopsis varchar(500) not null,
primary  key (MID)
);

CREATE TABLE Movie_Genres
(
MID varchar(10) not null ,
Genres varchar(50) not null,
primary key (MID,Genres),
Foreign key (MID) references Movie(MID)
);

CREATE TABLE Movie_Locations
(
MID varchar(10) not null ,
Locations varchar(50) not null,
primary key (MID,Locations),
Foreign key (MID) references Movie(MID)
```

```
);

CREATE TABLE Movie_keywords
(
MID varchar(10) not null ,
Keywords varchar(50) not null,
primary key (MID,Keywords),
Foreign key (MID) references Movie(MID)
);

CREATE TABLE Person
(
PID varchar(10) not null ,
P_Name varchar(50) not null unique,
Gender varchar(10) not null,
Image_url varchar(500),
primary key (PID)
);

CREATE TABLE Credit
(
MID varchar(10) not null ,
PID varchar(10) not null ,
Role varchar(50) not null,
primary key (MID,PID),
Foreign key (MID) references Movie(MID),
Foreign key (PID) references Person(PID)
);


CREATE TABLE Country
```

```
(
C_Name varchar(50) not null,
primary key (C_Name)
);

CREATE TABLE Movie_Certificates
(
MID varchar(10) not null ,
C_Name varchar(50) not null,
Certificate varchar(50) not null,
primary key (MID,C_Name),
Foreign key (MID) references Movie(MID),
Foreign key (C_Name) references Country(C_Name)
);

CREATE  TABLE Movie_dates
(
MID varchar(10) not null ,
C_Name varchar(50) not null,
Release_Date date not null,
primary key (MID,C_Name,Release_Date),
Foreign key (MID) references Movie(MID),
Foreign key (C_Name) references Country(C_Name)
);

CREATE  TABLE Movie_Languages
(
MID varchar(10) not null ,
C_Name varchar(50) not null,
Languages varchar(50) not null,
primary key (MID,C_Name,Languages),
```

```
Foreign key (MID) references Movie(MID),
Foreign key (C_Name) references Country(C_Name)
);

CREATE  TABLE Stream_Info
(
MID varchar(10) not null ,
Movie_url varchar(500) not null,
Trailer_url varchar(500),
Image_url varchar(500) not null,
Running_time int(10) not null,
Video_quality int(10) not null,
primary key (MID),
FOREIGN KEY (MID) references Movie(MID)
);

CREATE  TABLE User
(
Email_Id varchar(50) not null ,
Name varchar(50) not null,
Password varchar(50) not null,
Image_url varchar(500),
primary key (Email_Id)
);

CREATE  TABLE Reviews
(
Email_Id varchar(50) not null ,
MID varchar(10) not null,
Time_stamp TIMESTAMP not null,
Rating float(1) not null,
```

```
Comment varchar(500) not null,
primary key (Email_Id,MID) ,
FOREIGN KEY (MID) references Movie(MID) ,
FOREIGN KEY (Email_Id) references User(Email_Id)
);

CREATE  TABLE Bookmarks
(
Email_Id varchar(50) not null ,
MID varchar(10) not null,
time_elapsed int(10) not null,
primary key (Email_Id,MID) ,
FOREIGN KEY (MID) references Movie(MID),
FOREIGN KEY (Email_Id) references User (Email_Id)
);

CREATE TABLE System
(
MSN varchar(500) not null,
Last_email_Id varchar(50),
primary key (MSN) ,
FOREIGN KEY (Last_email_Id) references User(Email_Id)
);

CREATE  TABLE User_search
(
Email_Id varchar(50) not null ,
MID varchar(10) not null,
time_stamp timestamp not null,
primary key (Email_Id,MID) ,
FOREIGN KEY (MID) references Movie(MID),
```

```
FOREIGN KEY (Email_Id) references User(Email_Id)
);

CREATE  TABLE System_search
(
MSN varchar(500) not null ,
MID varchar(10) not null,
time_stamp timestamp not null,
primary key (MSN,MID),
FOREIGN KEY (MSN) references System (MSN),
FOREIGN KEY (MID) references Movie(MID)
);
```

Dumping the tables with low volume actual data:

```
1 •  SELECT * FROM classroompopcorn.movie_certificates;
```

Result Grid | Filter Rows: | Edit: | Export

MID	C_Name	Certificate
mm0001	Australia	U/A
mm0001	India	U/A
mm0001	Srilanka	A
mm0002	New Zealand	A
mm0002	Pakistan	A
mm0003	India	U
mm0003	Nepal	U/A
mm0003	West Indies	U
mm0004	India	U/A
mm0004	Pakistan	A
mm0005	Bangladesh	U
mm0005	Maldives	U
mm0006	France	U/A
mm0006	Germany	U/A
mm0006	USA	U/A
mm0007	India	U
mm0008	Zimbabwe	U/A
mm0009	Bhutan	A

```
1 •  SELECT * FROM classroompopcorn.movie_certificates;
```

Result Grid | Filter Rows: | Edit: | Export/

MID	C_Name	Certificate
mm0001	Australia	U/A
mm0001	India	U/A
mm0001	Srilanka	A
mm0002	New Zealand	A
mm0002	Pakistan	A
mm0003	India	U
mm0003	Nepal	U/A
mm0003	West Indies	U
mm0004	India	U/A
mm0004	Pakistan	A
mm0005	Bangladesh	U
mm0005	Maldives	U
mm0006	France	U/A
mm0006	Germany	U/A
mm0006	USA	U/A
mm0007	India	U
mm0008	Zimbabwe	U/A
mm0009	Bhutan	A

```
1 •  SELECT * FROM classroompopcorn.movie;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Cont

MID	Title	Release_Year	Imdb_rating	Synopsis
mm0001	Queen	2014	9.2	Rani, a 24-year-old homely girl, decides to go o...
mm0002	Anand	1971	8.8	Anand, a cancer patient, lives his life to the full...
mm0003	Andaz Apna Apna	1994	8.7	Amar and Prem belong to middle-class families b...
mm0004	Gangs Of Wassepour	2011	8.6	A gangster (Manoj Bajpayee) clashes with the r...
mm0005	Taare Zameen Par	2007	8.6	Ishaan cannot seem to get anything right at his...
mm0006	Rang De Basanti	2006	8.6	When Sue selects a few students to portray va...
mm0007	Dil Chahta Hai	2001	8.6	Three friends who share a deep bond are separ...
mm0008	Sholay	1975	8.6	Jai and Veeru, two small-time crooks, are hired ...
mm0009	The Bastard Child	2013	8.5	Children of War, also known as The Bastard Chil...
mm0010	Bhaag Milkha Bhaag	2013	8.5	Milkha Singh or the 'Flying Sikh' overcomes man...
mm0011	A Wednesday	2008	8.5	A retired police commissioner recounts the most...
mm0012	3 Idiots	2009	8.5	In college, Farhan and Raju form a great bond ...
mm0013	Black Friday	2004	8.4	Black Friday is the day following Thanksgiving D...
mm0014	Barfi!	2012	8.3	Shruti loves Barfi, a hearing and speech-impaire...
mm0015	Black	2005	8.2	Black is the darkest color, the result of the abse...
mm0016	Dev.D	2009	8.1	After breaking up with his childhood love Paro, ...
mm0017	Dabba	2013	8	School boy Stanley does not carry lunch, which ...
mm0018	Highway	2014	7.9	Veera, a young bride-to-be, is abducted by a c...

```
1 •  SELECT * FROM classroompopcorn.credit;
```

Result Grid | Filter Rows: | Edit:

MID	PID	Role
mm0001	pp0001	Actor
mm0001	pp0002	Director
mm0001	pp0003	Producer
mm0002	pp0004	Actor
mm0002	pp0005	Actor
mm0002	pp0006	Director
mm0002	pp0008	Producer
mm0003	pp0007	Actor
mm0003	pp0008	Actor
mm0003	pp0009	Director
mm0003	pp0010	Producer
mm0004	pp0011	Actor
mm0004	pp0012	Director
mm0004	pp0013	Producer
mm0005	pp0006	Director
mm0005	pp0007	Actor
mm0005	pp0009	Producer
mm0005	pp0014	Actor
mm0006	pp0007	Actor

```
1 •  SELECT * FROM classroompopcorn.country;
```

Result Grid | Filter Rows: | Edit:

C_Name
Australia
Bangladesh
Bhutan
France
Germany
India
Maldives
Nepal
New Zealand
Pakistan
Srilanka
USA
West Indies
Zimbabwe
NULL

```
1 •  SELECT * FROM classroompopcorn.movie_certificates;
```

Result Grid | Filter Rows: | Edit: | Export

MID	C_Name	Certificate
mm0001	Australia	U/A
mm0001	India	U/A
mm0001	Srilanka	A
mm0002	New Zealand	A
mm0002	Pakistan	A
mm0003	India	U
mm0003	Nepal	U/A
mm0003	West Indies	U
mm0004	India	U/A
mm0004	Pakistan	A
mm0005	Bangladesh	U
mm0005	Maldives	U
mm0006	France	U/A
mm0006	Germany	U/A
mm0006	USA	U/A
mm0007	India	U
mm0008	Zimbabwe	U/A
mm0009	Bhutan	A

```
1 •  SELECT * FROM classroompopcorn.bookmarks;
```

Result Grid | Filter Rows: | Edit:

Email_Id	MID	time_elapsed
a@a.com	mm0001	0
a@a.com	mm0007	0
a@a.com	mm0015	0
b@b.com	mm0001	0
heerdeepdas994@gmail.com	mm0014	114
heerdeepdas994@gmail.com	mm0021	75
HeerSnBrown@gmail.com	mm0003	91
HeerSnBrown@gmail.com	mm0010	97
HeerSnoBrown@gmail.com	mm0002	69
HeerSnoBrown@gmail.com	mm0007	123
HeerSnow@gmail.com	mm0005	83
HeerSnow@gmail.com	mm0007	72
HeerSnow@gmail.com	mm0014	117
iovdeepdas994@gmail.com	mm0001	47
shobhitmittal96@gmail.com	mm0010	0
shobhitmittal96@gmail.com	mm0014	0
shobhitmittal96@gmail.com	mm0018	0
shobhitmittal96@gmail.com	mm0021	0
NULL	NULL	NULL

```
1 •  SELECT * FROM classroompopcorn.stream_info;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

MID	Movie_url	Trailer_url	Image_url	Running_time	Video_quality
mm0001	https://www.youtube.com/embed/aKHbxoCJI8I	https://www.youtube.com/embed/eLUHD-DUonk	http://t2.gstatic.com/images?q=tbn:ANd9GcRrt...	146	360
mm0002	https://www.youtube.com/embed/9RShJwvaWKA	https://www.youtube.com/embed/tfGX2AEaMUU	http://1.bp.blogspot.com/-mqKF6RTd4g/Vqmls...	123	1080
mm0003	https://www.youtube.com/embed/7M_iHUawccI	https://www.youtube.com/embed/fd_w7Ow8biU	https://upload.wikimedia.org/wikipedia/en/1/15...	161	360
mm0004	https://www.youtube.com/embed/dz3uCM-MrGY	https://www.youtube.com/embed/i-AkWDkXcMY	http://t0.gstatic.com/images?q=tbn:ANd9GcRI...	160	1080
mm0005	https://www.youtube.com/embed/XJYm6JYMetA	https://www.youtube.com/embed/tn_2Ie_itX8	http://www.gstatic.com/tv/thumb/movieposter...	165	360
mm0006	https://www.youtube.com/embed/c769V25oX08	https://www.youtube.com/embed/l-BTOTtcGmk	http://www.gstatic.com/tv/thumb/movieposter...	171	720
mm0007	https://www.youtube.com/embed/1RC7v4R2w7s	https://www.youtube.com/embed/m13b25V0B10	http://www.gstatic.com/tv/thumb/movieposter...	185	360
mm0008	https://www.youtube.com/embed/TPMVr2XNH6w	https://www.youtube.com/embed/EwT6RdS-Nac	http://www.gstatic.com/tv/thumb/dvdboxart/7...	204	720
mm0009	https://www.youtube.com/embed/P3xWa-oBtdO	https://www.youtube.com/embed/P3xWa-oBtdO	https://images-na.ssl-images-amazon.com/imag...	160	480
mm0010	https://www.youtube.com/embed/fEKruX_roY	https://www.youtube.com/embed/3uICXnnW86U	http://t2.gstatic.com/images?q=tbn:ANd9GcSo...	189	720
mm0011	https://www.youtube.com/embed/7Jza0dIJGda	https://www.youtube.com/embed/u7RmUrMo770	https://images-na.ssl-images-amazon.com/imag...	104	720
mm0012	https://www.youtube.com/embed/aUXOMdShivw	https://www.youtube.com/embed/xvszmNXdM4w	http://www.gstatic.com/tv/thumb/movieposter...	171	1080
mm0013	https://www.youtube.com/embed/UhB6gup1ifc	https://www.youtube.com/embed/2csia1kZTZ4	http://www.gstatic.com/tv/thumb/movieposter...	167	480
mm0014	https://www.youtube.com/embed/X_a9IXvt3ro	https://www.youtube.com/embed/nO3FYUaSiC8	http://www.gstatic.com/tv/thumb/movieposter...	151	720
mm0015	https://www.youtube.com/embed/O5WPRW1b...	https://www.youtube.com/embed/Smd_xZHCCzI	https://images-na.ssl-images-amazon.com/imag...	124	360
mm0016	https://www.youtube.com/embed/EmxaK0R1YUM	https://www.youtube.com/embed/9UVZorrlvtw	http://www.gstatic.com/tv/thumb/dvdboxart/1...	171	480
mm0017	https://www.youtube.com/embed/D64hK8OoiZY	https://www.youtube.com/embed/IJIGJtb_N7E	http://t0.gstatic.com/images?q=tbn:ANd9GcRP...	105	720
mm0018	https://www.youtube.com/embed/RmdI3ANHF-c	https://www.youtube.com/embed/LSrDD52bx4A	http://t2.gstatic.com/images?q=tbn:ANd9GcTH...	133	1080

```
1 •  SELECT * FROM classroompopcorn.movie_languages;
```

Result Grid | Filter Rows: | Edit: | Exp

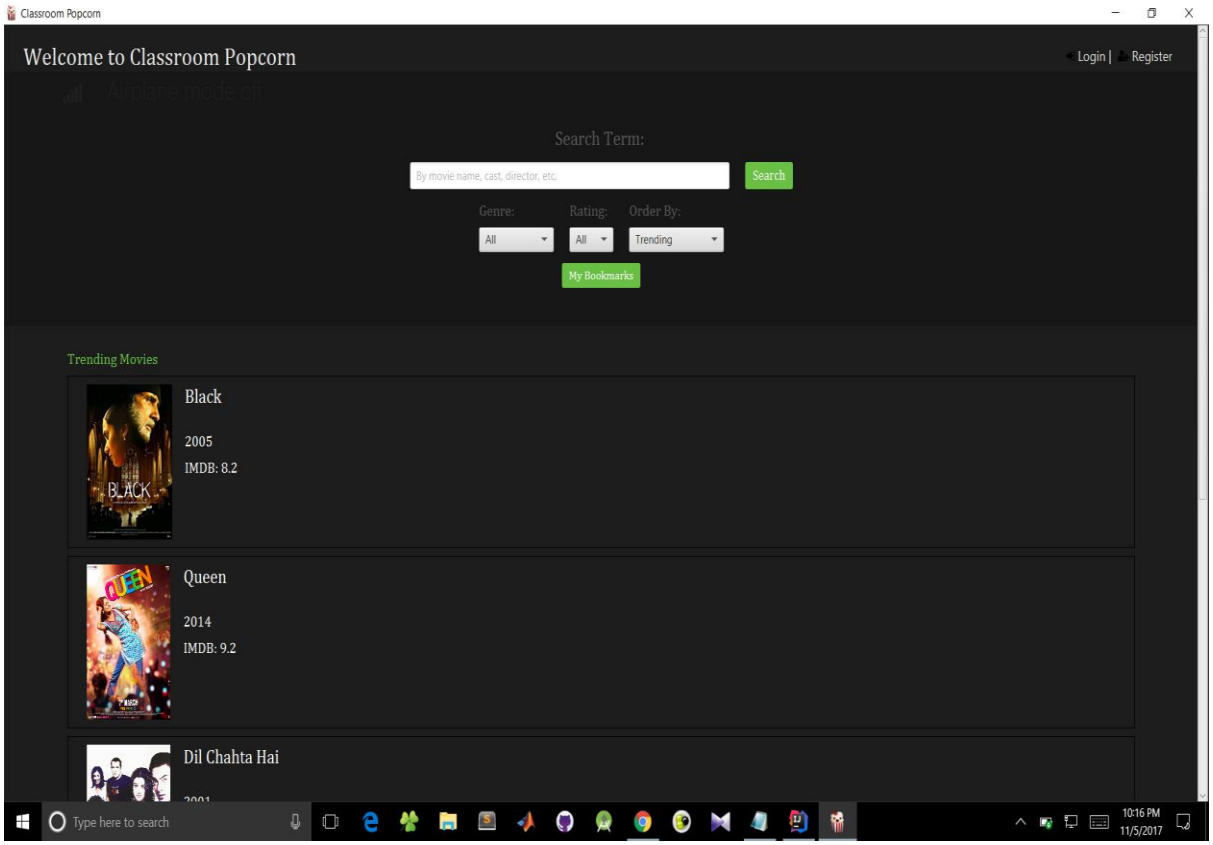
MID	C_Name	Languages
mm0001	Australia	English
mm0001	Australia	Spanish
mm0010	Australia	English
mm0010	Australia	Spanish
mm0014	Australia	English
mm0014	Australia	Portuguese
mm0016	Australia	English
mm0016	Australia	Spanish
mm0025	Australia	English
mm0025	Australia	Spanish
mm0029	Australia	English
mm0029	Australia	Portuguese
mm0005	Banglad...	Bengali
mm0020	Banglad...	Bengali
mm0009	Bhutan	Chinese
mm0024	Bhutan	Chinese
mm0006	France	French

```
1 •  SELECT * FROM classroompopcorn.movie_locations;
```

Result Grid | Filter Rows: | Edit: | Export/

MID	Locations
mm0001	oxford
mm0002	hawaii
mm0002	paris
mm0003	london
mm0004	newyork
mm0005	olenco
mm0005	vatican
mm0006	oxford
mm0006	paris
mm0006	winevard
mm0007	amityville
mm0008	istanbul
mm0009	olenco
mm0009	london
mm0010	winevard

HOME PAGE :

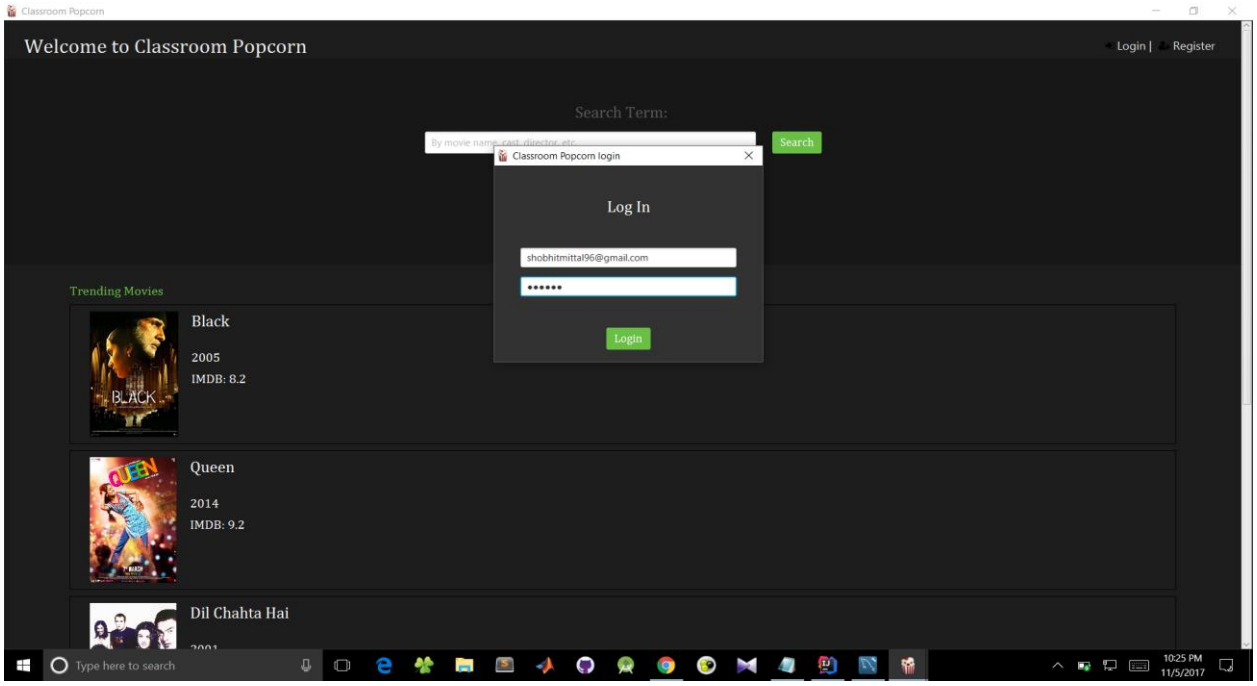


The First page which includes Movie search box, Combo Box of Genres, Imdb_Rating, Order by and My bookmarks. User Login and Sign up Buttons for Login details at the top right and bottom half of the page consists the movies fetched based on queries given in the search and Combo Box .

1. Login/ Sign up Buttons :

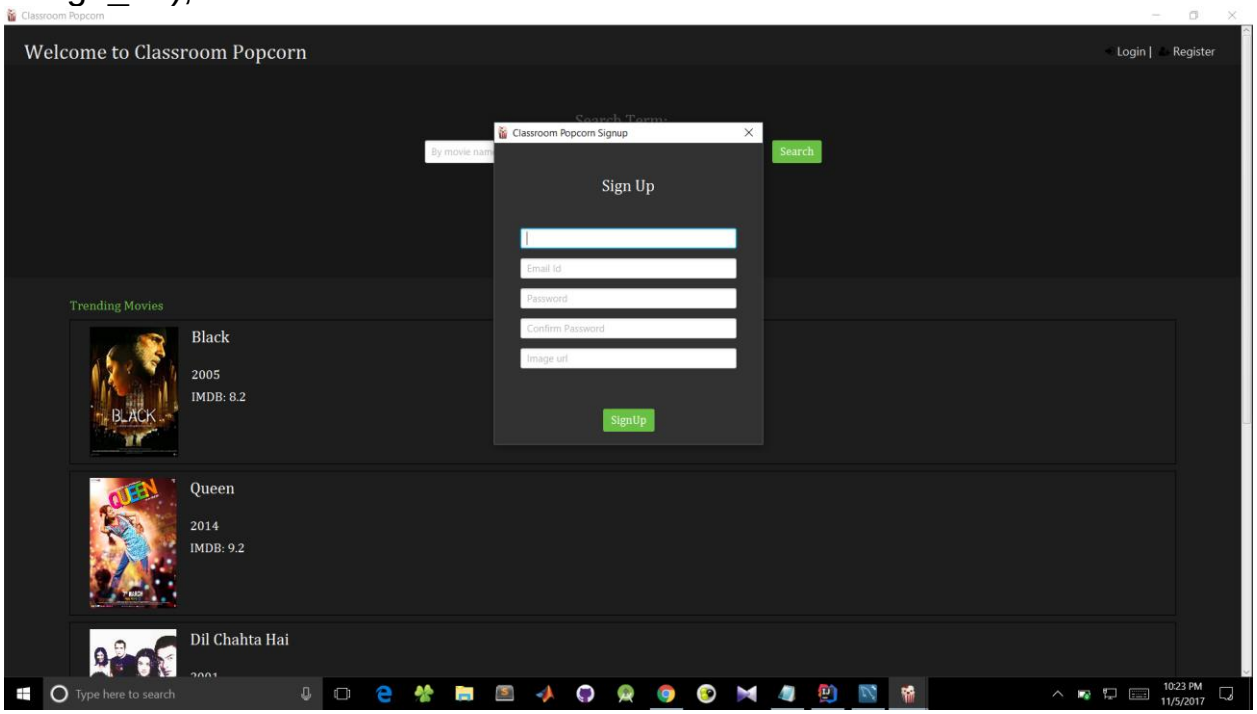
Login – The button selects the user from User table with the input credentials as emailId and password.

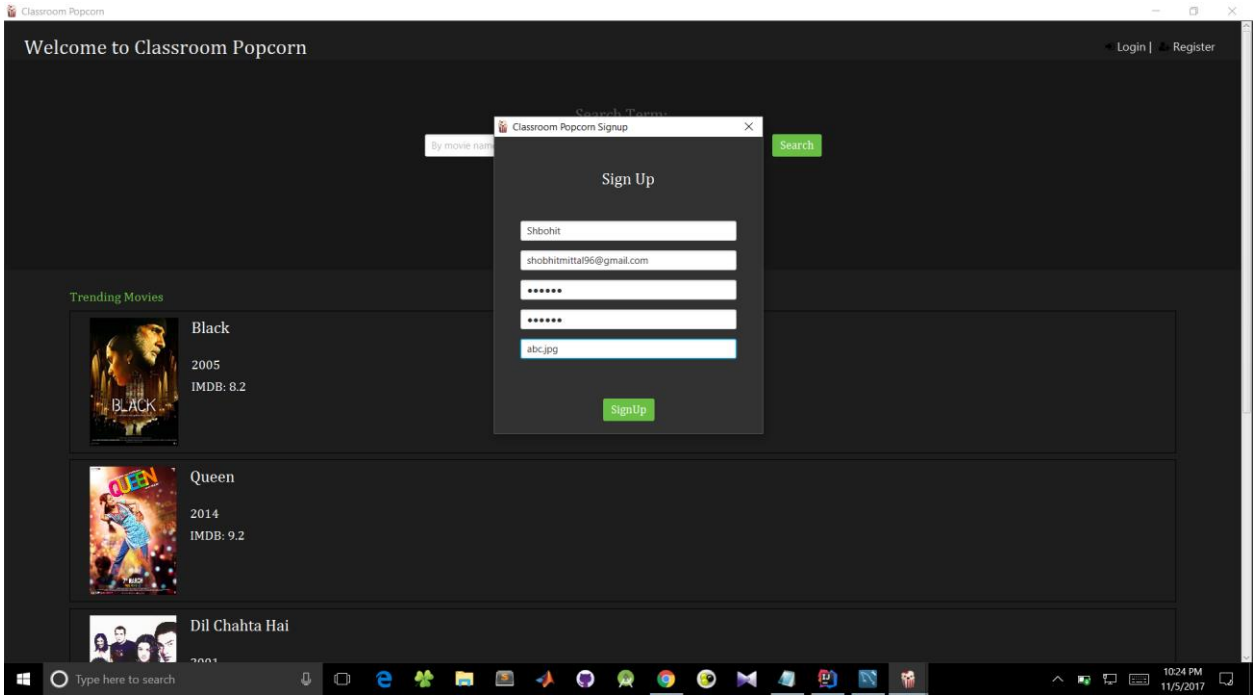
SQL Query – select * from classroompopcorn.User where Email_Id=emailId and Password=password;



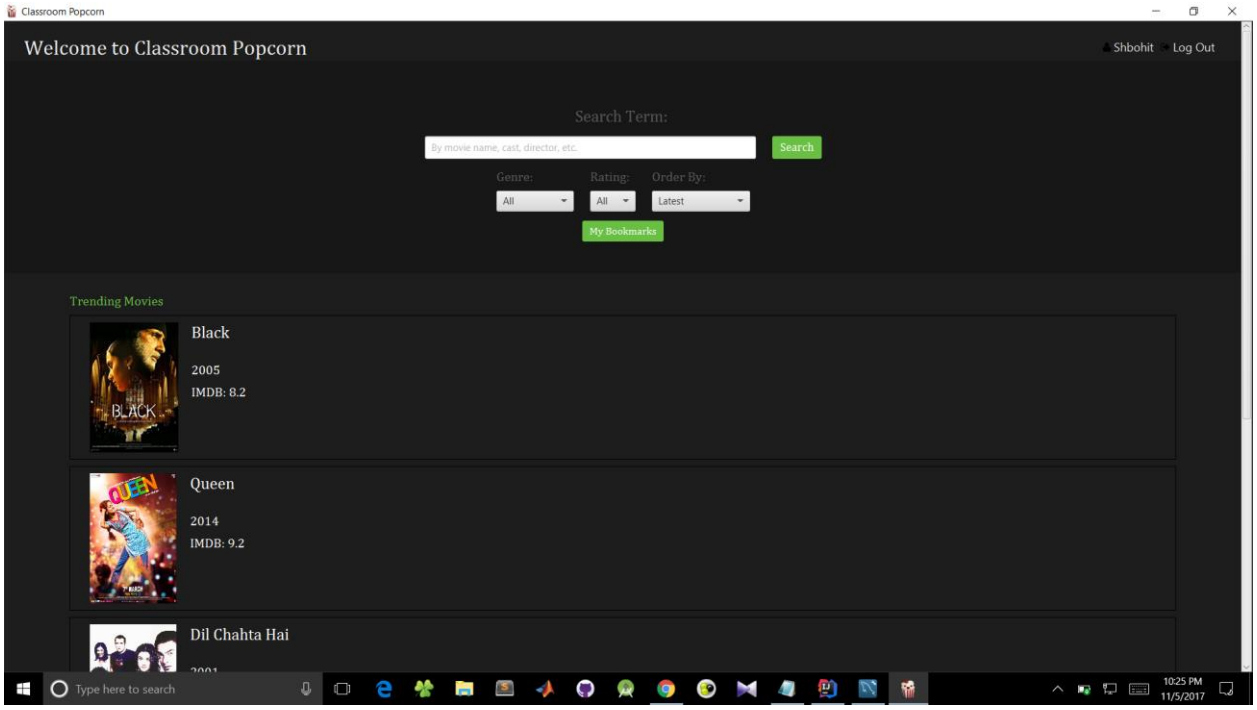
Register - The button inserts the data into User table with full name, emailId, password and photo. Appropriate error message in case of invalid entry.

SQL Query – insert into User values(emailId, Name, password, Image_url);





The full name of user is displayed on the homepage when logged in. All the activity henceforth is stored in the database corresponding to the emailId if user is logged in. Otherwise it will be stored corresponding to the MSN fetched automatically from the system.



Upon selecting any movie on the homepage, the moviepage corresponding to that movie opens up and an entry is stored in the database for this movie search in ‘User_search’ table if user is logged in and ‘System_search’ if no user is logged in.

SQL Query –

insert into User_search values(emailId, movieid, timestamp);

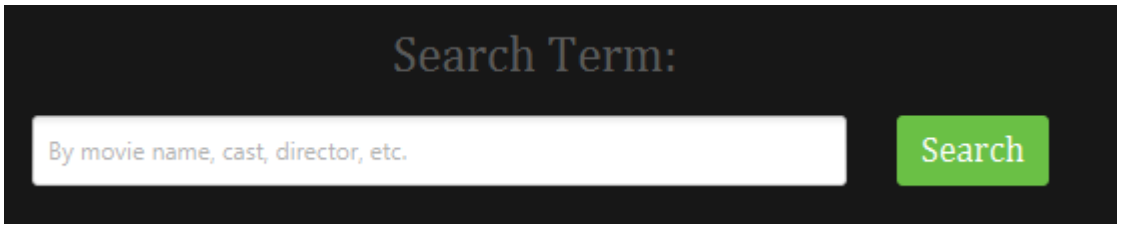
insert into System_search values(MSN, movieid, timestamp);

2. Search Box and Search Button:

2.a Code :

```
TextField searchBox = new TextField();
searchBox.setPromptText("By movie name, cast, director, etc.");
searchBox.setStyle("-fx-focus-color: transparent;");
searchBox.setPrefColumnCount(35);
searchBox.setPrefHeight(35);
searchBox.focusedProperty().addListener((observable,  oldValue,  newValue) -> {
    if(newValue && firstTime.get()){
        searchLayout.requestFocus(); // Delegate the focus to container
        firstTime.setValue(false); // Variable value changed for future references
    }
});

Button searchButton = new Button( text: "Search");
searchButton.setStyle("-fx-focus-color: transparent;");
searchButton.setFont(new Font( name: "Cambria",  size: 18));
searchButton.setStyle("-fx-background-color: #6ac045;");
searchButton.setTextFill(Color.web("#fff"));
searchButton.setCursor(Cursor.HAND);
```

2.a. Screenshot :

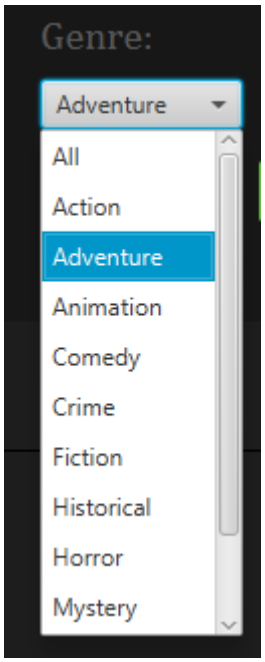


3 . Genres :

3.a. Code :

```
VBox genreCollection = new VBox( spacing: 10);
Label genreLabel = new Label( text: "Genre: ");
genreLabel.setFont(new Font( name: "Cambria",  size: 20));
genreLabel.setTextFill(Color.web("#5a5a5a"));
ComboBox genreComboBox = new ComboBox();
genreComboBox.getItems().addAll(
        ...elements: "All",
        "Action",
        "Adventure",
        "Animation",
        "Comedy",
        "Crime",
        "Fiction",
        "Historical",
        "Horror",
        "Mystery",
        "Romantic",
        "Thriller"
);
genreComboBox.getSelectionModel().selectFirst();
genreCollection.getChildren().addAll(genreLabel,genreComboBox);
```

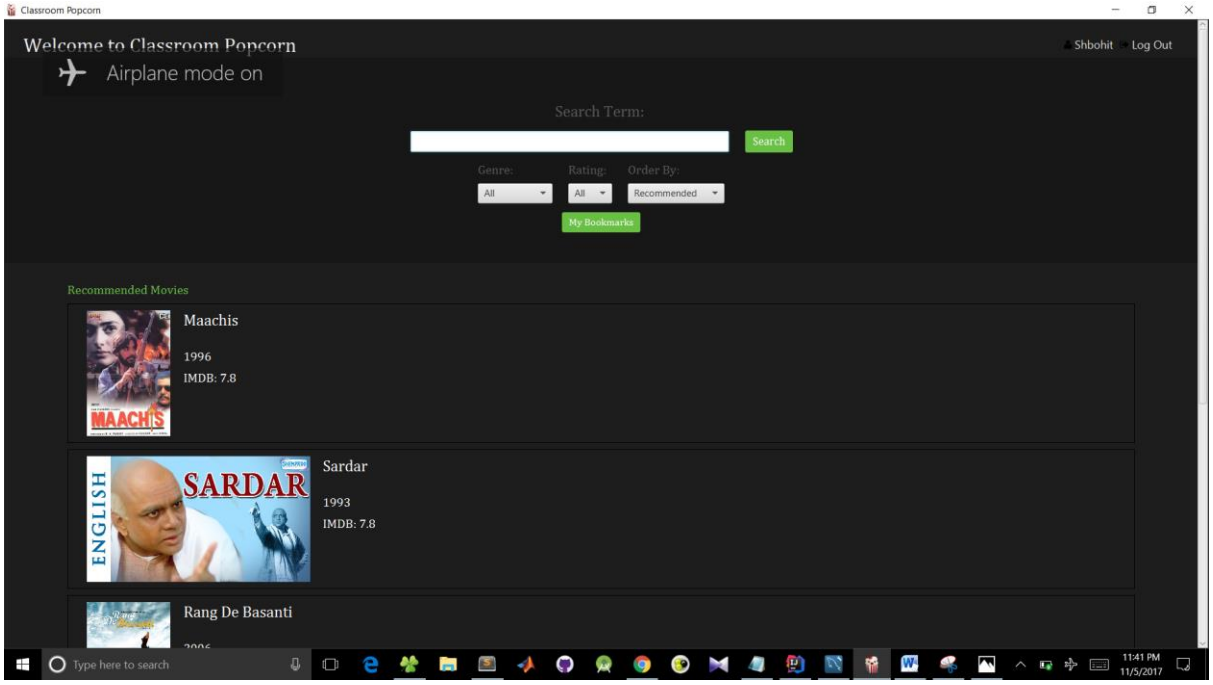
3.b : Screenshot:



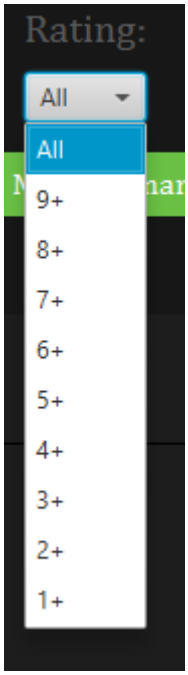
4. Imdb-Rating :

4.a Codes :

```java
VBox ratingCollection = new VBox( spacing: 10);
Label ratingLabel = new Label( text: "Rating: ");
ratingLabel.setFont(new Font( name: "Cambria",  size: 20));
ratingLabel.setTextFill(Color.web("#5a5a5a"));
ComboBox ratingComboBox = new ComboBox();
ratingComboBox.getItems().addAll(
        ...elements: "All",
        "9+",
        "8+",
        "7+",
        "6+",
        "5+",
        "4+",
        "3+",
        "2+",
        "1+"
);
ratingComboBox.getSelectionModel().selectFirst();
ratingCollection.getChildren().addAll(ratingLabel,ratingComboBox);
```



4. b : Screenshot :

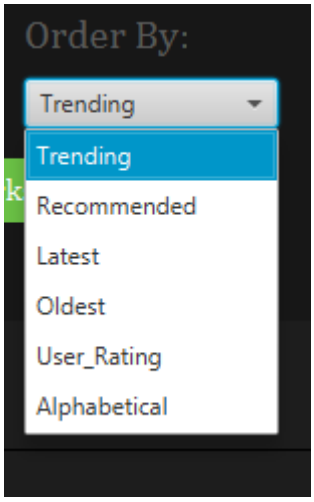


5. Order by :

5.a. Code:

```
String orderBy = orderComboBox.getValue().toString();
if (orderBy.equals("Latest"))
    condition = condition + " ORDER BY Release_Year desc";
else if (orderBy.equals("Oldest"))
    condition = condition + " ORDER BY Release_Year asc";
else if (orderBy.equals("Alphabetical"))
    condition = condition + " ORDER BY Title asc";

else if (orderBy.equals("Trending") && condition.isEmpty())
    condition = condition + "Z";
else if (orderBy.equals("Recommended") && condition.isEmpty())
    condition = condition + "Y";-
else if (orderBy.equals("User_Rating") && condition.isEmpty())
    condition = condition + "X";
```

5.b Screenshot :

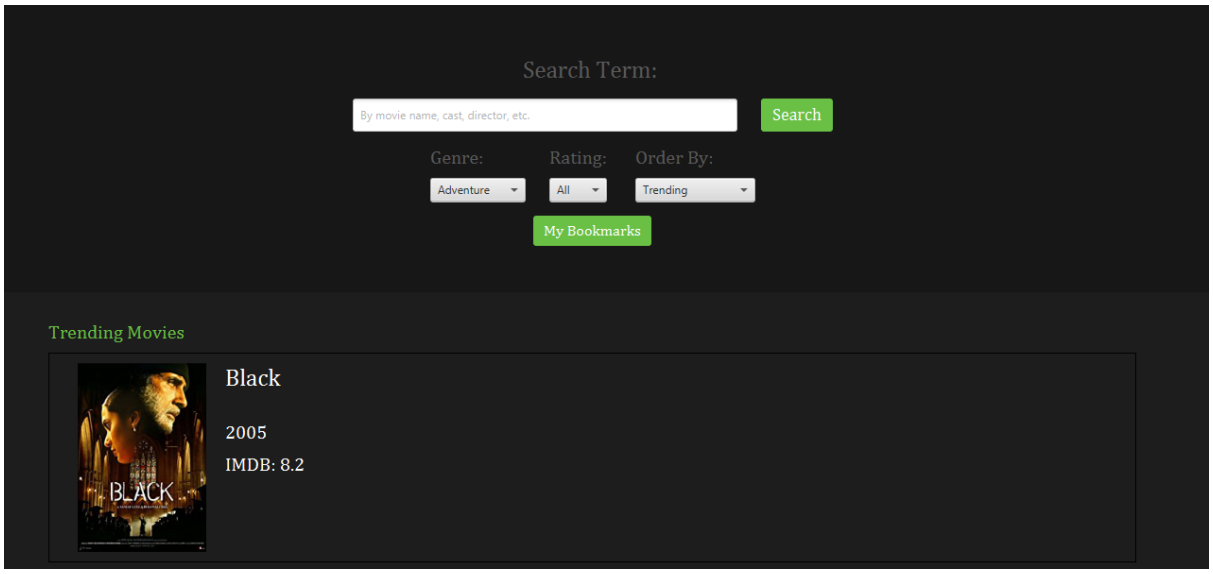


6. Segmenting the search field and search Result :

6.a. Code :

```
searchLayout.setTop(searchVB);
searchLayout.setBottom(searchResult);
```

6.b Screenshot :



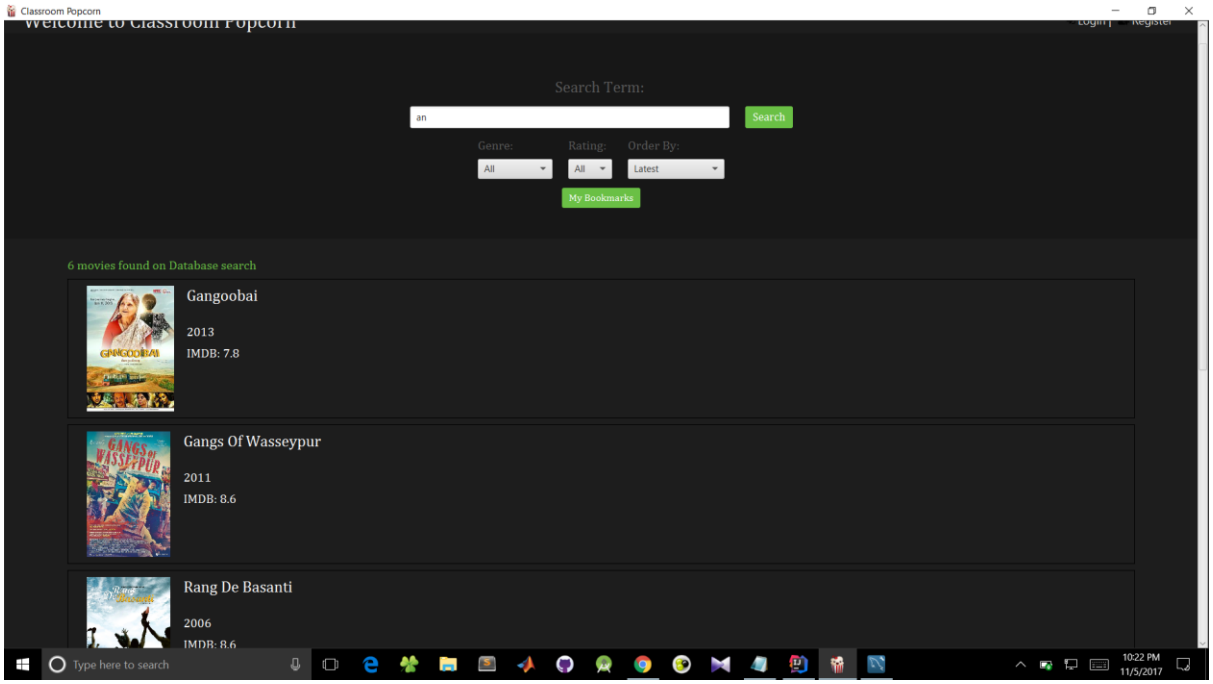
Search Queries and Result :

1.Search movies on the basis of partial title of the movie:

1.a: Query

```
if (!searchBox.getText().isEmpty())
    condition = condition+" AND Title LIKE '%"+searchBox.getText()+"%'";
```

1.b: Screenshot :

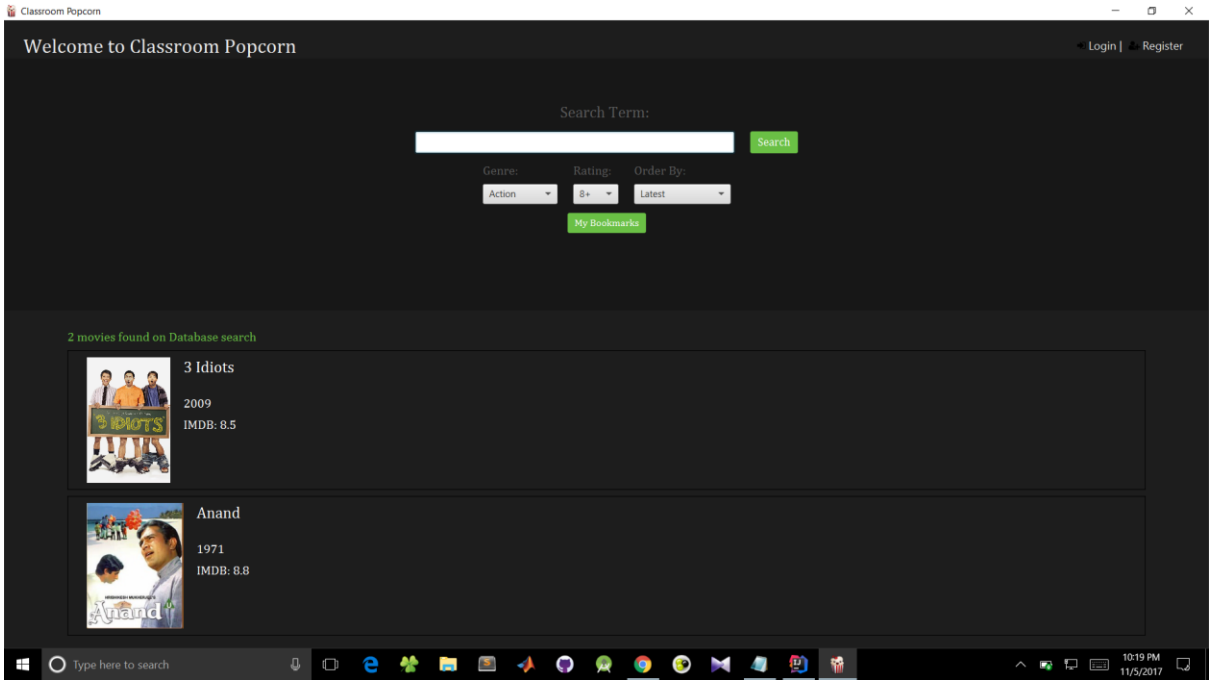


2. filter movies on the basis of genre and Imdb rating:

2.a: Query

```
if (genre.equals(""))
    condition="WHERE Imdb_rating>"+rate;
else
    condition="WHERE Imdb_rating>"+rate+" AND Genres LIKE '%"+genre+"%'";
```

2.b: Screenshot



3. Order the movies by trending (based on recency of all user/system searches):

3.a: query :

```
query = "select distinct A.MID as MID, Title, Release_Year, Imdb_rating, Synopsis, Image_url \n" +

    "from ((((select distinct MID ,sum((time_stamp)) as ST1\n" +

    "\t  from user_search as U\n" +

    "\t  group by U.MID) as A join\n" +

    "\t  (select distinct MID ,sum((time_stamp)) as ST2\n" +

    "\t  from system_search as S\n" +

    "\t  group by S.MID) as B on (A.MID=B.MID)) join movie  on (A.MID=movie.MID) ) join stream_info on (A.MID=stream_info.MID)) \n" +

    "\t  \n" +

    "order by A.ST1+B.ST2 desc limit 5;\n";
```

3.b: Screenshot :

4.Order the movies by recommended (based on genre of movies searched by same user/system)

4.a: On the basis of User

Query:

```
query = " select distinct A.MID, Title, Release_Year, Imdb_rating, Synopsis, Image_url \n" +

    "from (((select distinct M01.MID,count(distinct U.MID) as CM1\n" +

    "\t  from user_search as U, movie_genres as G1,movie_genres as G2,movie as M01\n" +

    "        where U.email_id= '"+login_details+"' and U.MID= G1.MID and G1.genres = G2.genres and \n" +

    "        G2.MID = M01.MID\n" +

    "\t  group by M01.MID) as A join movie  on (A.MID=movie.MID) ) join stream_info on (A.MID=stream_info.MID)) \n" +

    "order by A.CM1 desc limit 5; " ;
```

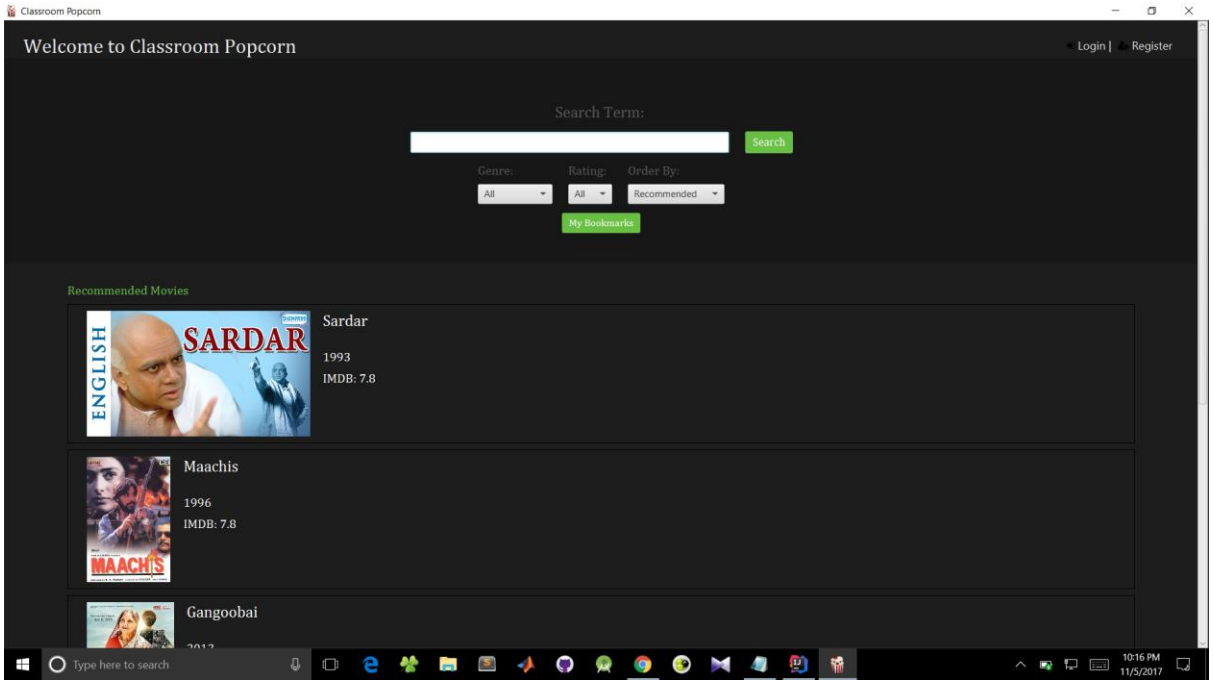
Screenshot:

4.b: On the basis of System:

Query :

```
query = " select distinct A.MID, Title, Release_Year, Imdb_rating, Synopsis, Image_url \n" +

    "from (((select distinct M01.MID,count(distinct S.MID) as CM1\n" +

    "\t  from system_search as S, movie_genres as G1,movie_genres as G2,movie as M01\n" +

    "        where S.MSN='"+login_details+"' and S.MID= G1.MID and G1.genres = G2.genres and \n" +

    "        G2.MID = M01.MID\n" +

    "\t  group by M01.MID) as A join movie  on (A.MID=movie.MID) ) join stream_info on (A.MID=stream_info.MID))   \n" +

    "order by A.CM1 desc limit 5;";
```

4.a.Screenshot:

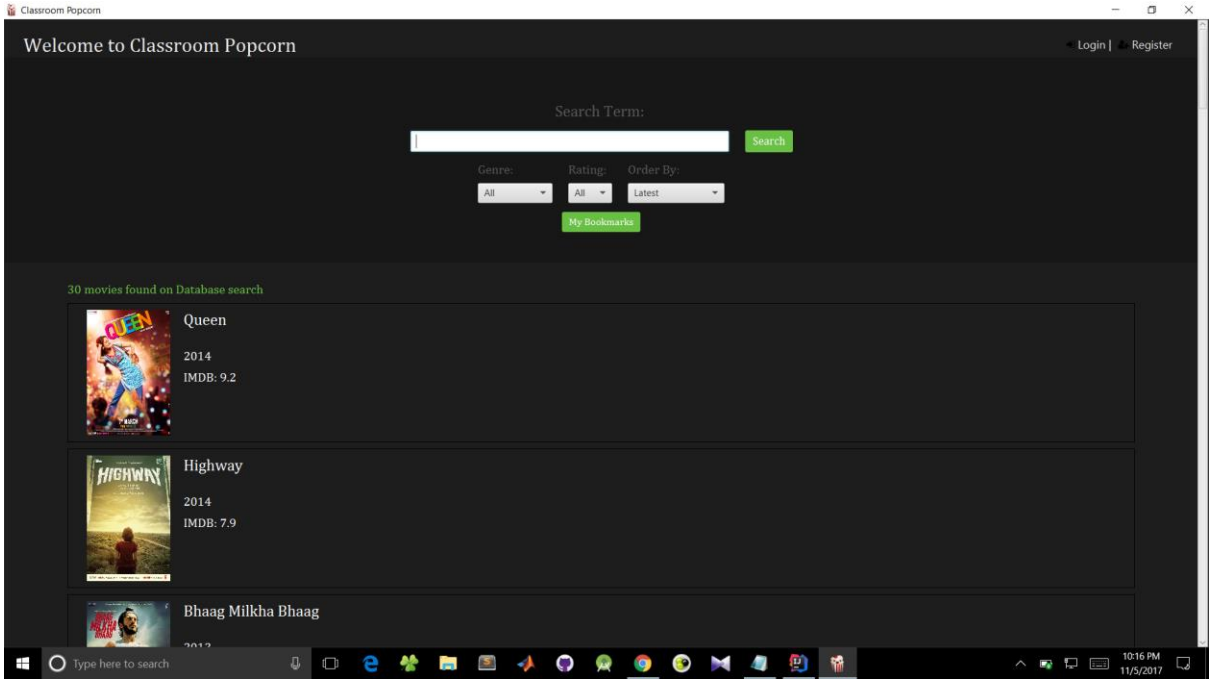


5. Order the movies by latest release :

5.a. Query :

```
if (orderBy.equals("Latest"))
  condition = condition + " ORDER BY Release_Year desc";
```

5.b. Screenshot :

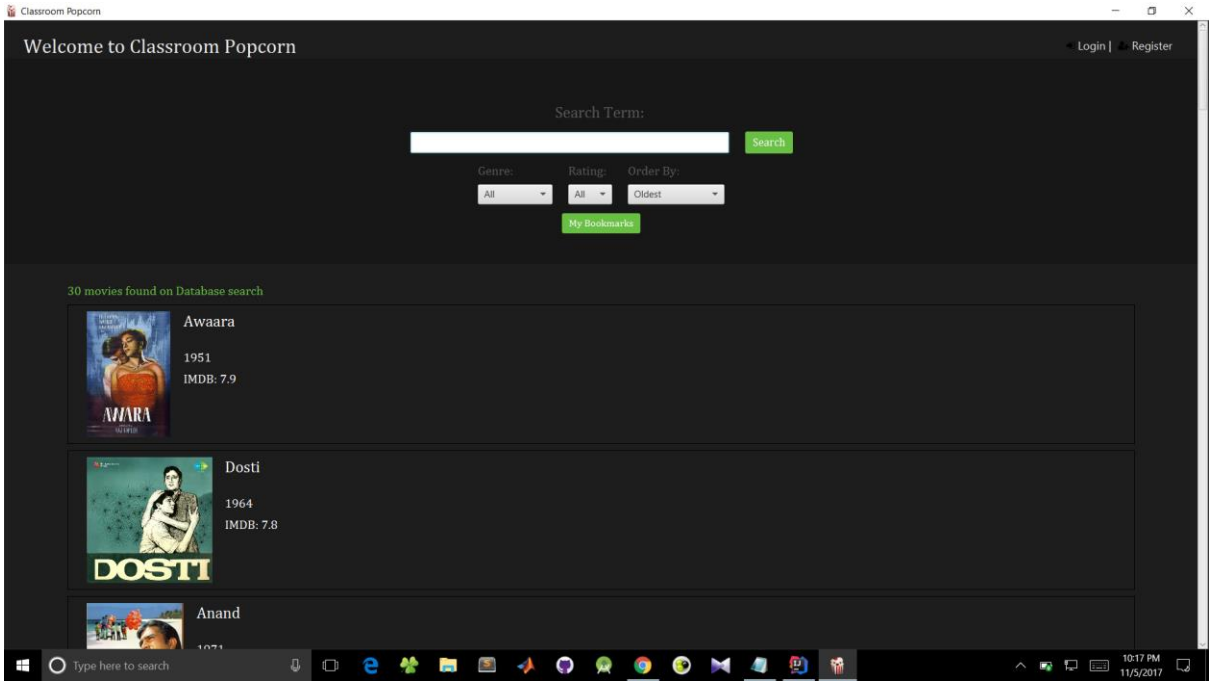


6.Order the movies by oldest release :

6.a Query:

```
else if (orderBy.equals("Oldest"))
   condition = condition + " ORDER BY Release_Year asc";
```

6.b. Screenshot :



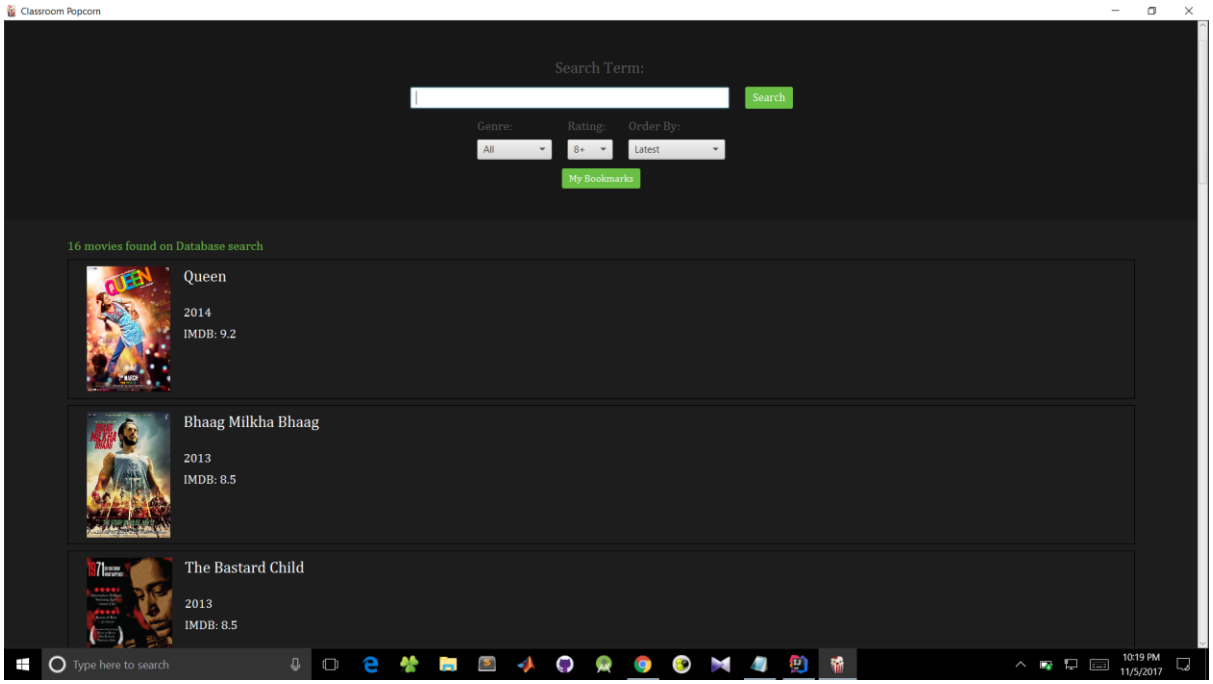
-

7.order the movies by user rating:

7.a Query :

```
query = " select distinct A.MID, title, release_year, imdb_rating, Synopsis, image_url\n" +
      "from (select distinct R.MID, sum(rating) as SR1\n" +
      "      from reviews as R\n" +
      "      group by R.MID) as A natural join movie natural join stream_info\n" +
      "order by A.SR1 desc limit 5;";
```

7.b Screenshot :

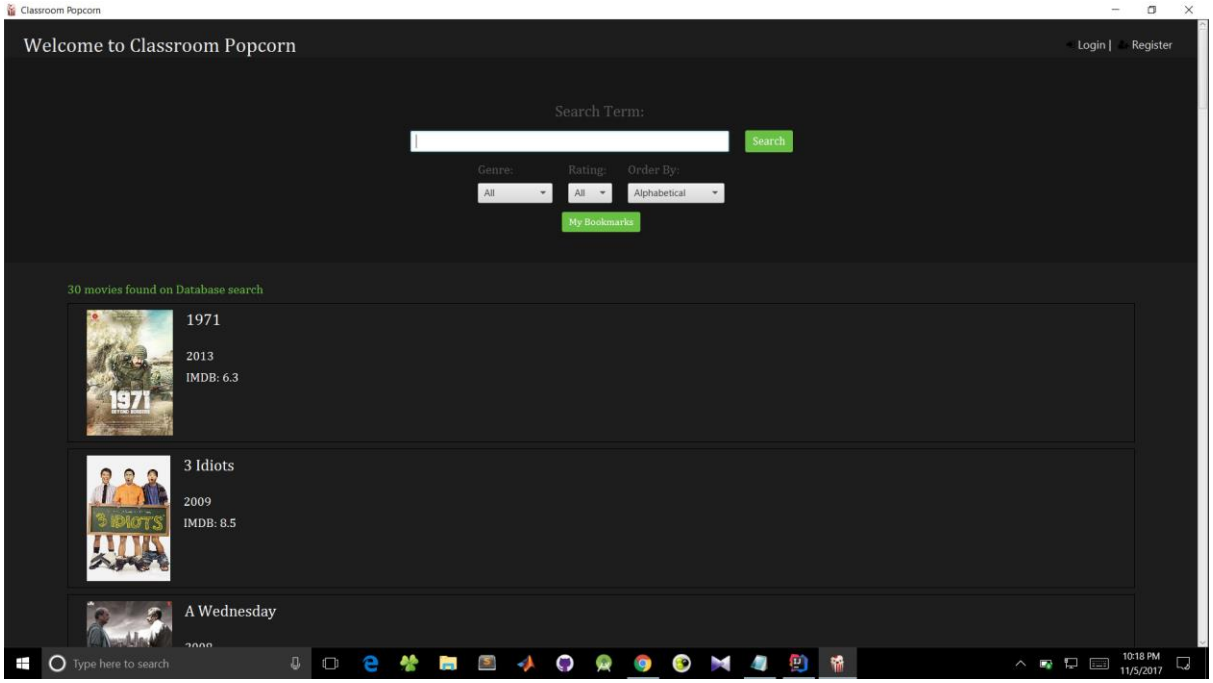


8.order the movies by Alphabetic:

8.a Query :

```
else if (orderBy.equals("Alphabetical"))
  condition = condition + " ORDER BY Title asc";
```

8.b Screenshoot :

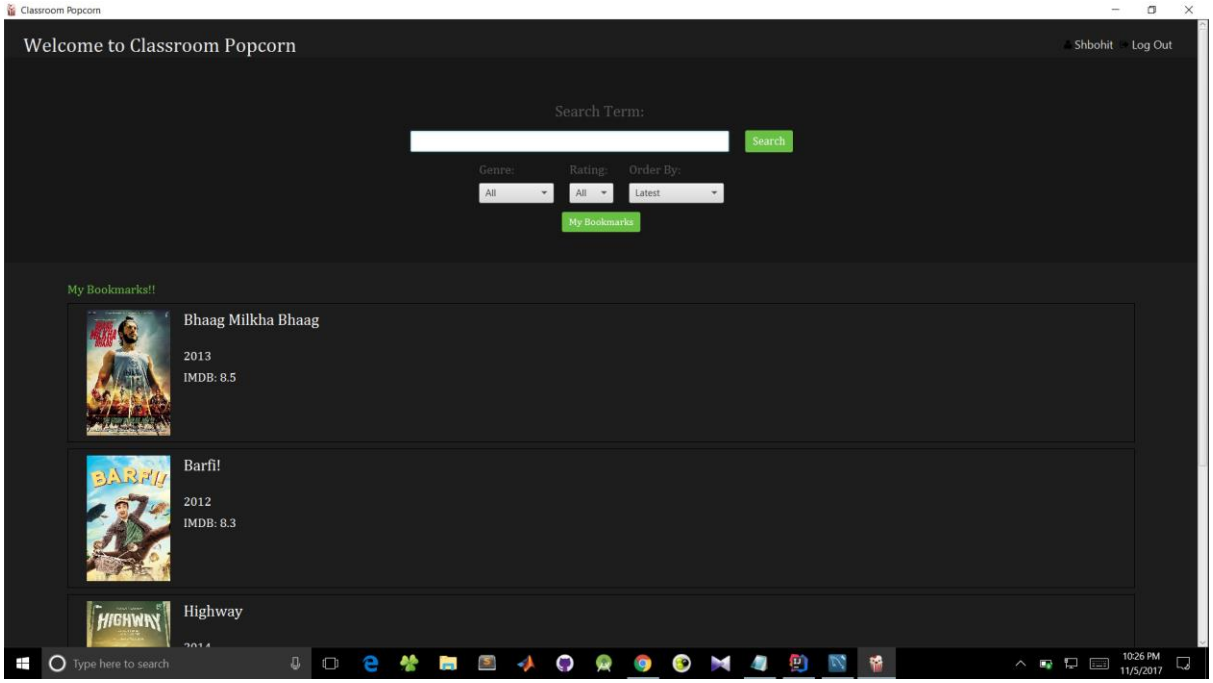


9.List the movies bookmarked by the same user :

9.a Query :

query = "select MID,Title, Synopsis,Release_Year, Imdb_rating, Image_url from movie natural join stream_info where MID in (select distinct MID from bookmarks where Email_Id= '" + login_details + "');";

9.b Screenshot :

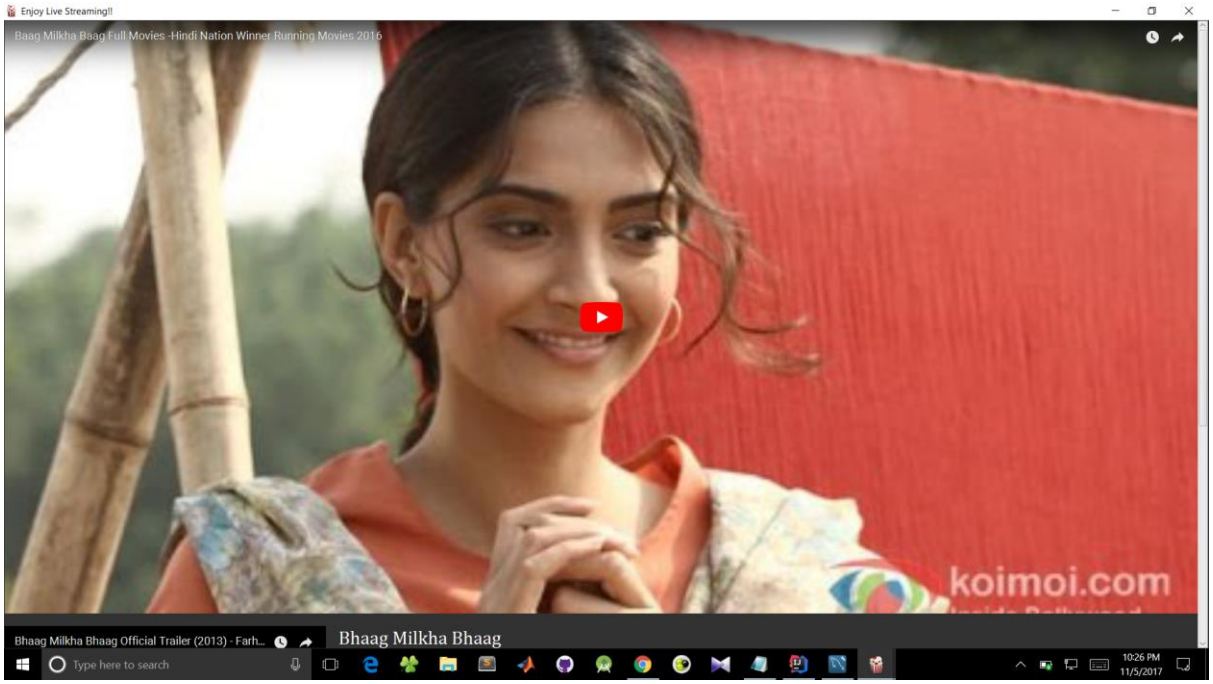


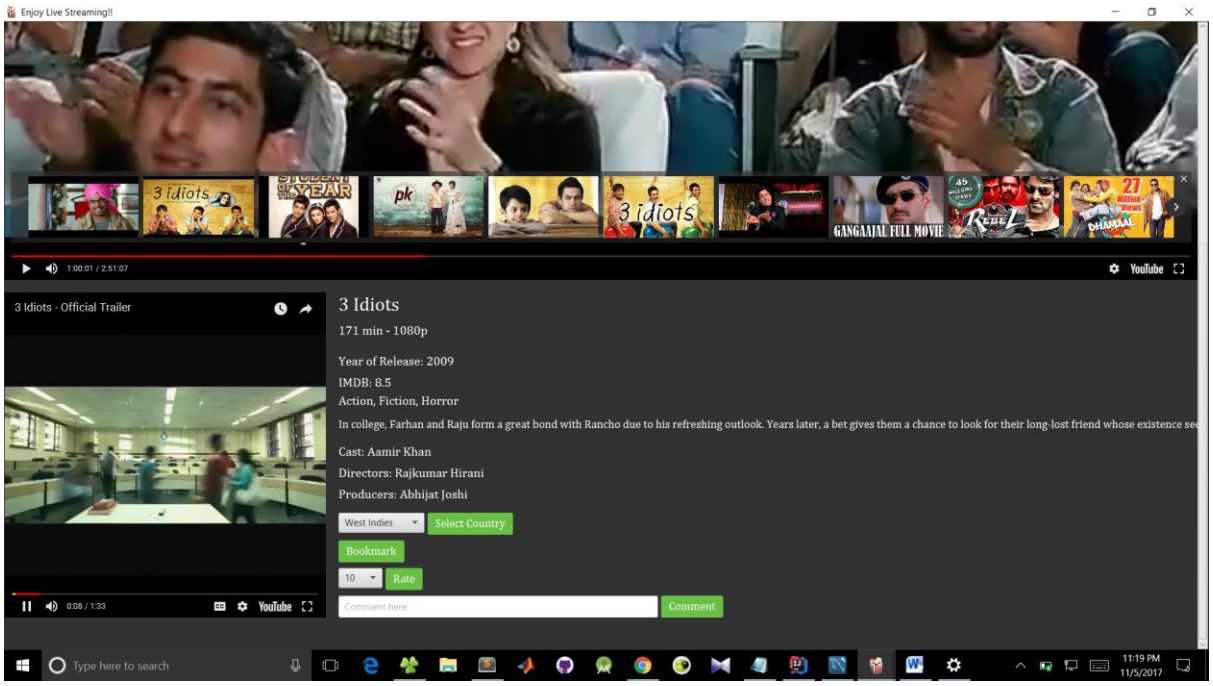
Movie Page

Live streaming

Live streaming of the full length movie and the trailer from youtube.

*query = "select * from classroompopcorn.Stream_Info where MID = '"+movieid+"';"*





Movie Details

Display of title, runtime, video_quality, year of release, imdb rating, genres and synopsis of the movie.

```
query_genre = "select * from classroompopcorn.Movie_Genres where MID = '"+movieid+"';"

query = "select * from classroompopcorn.movie where MID = '"+movieid+"';"
```

Black

124 min - 360p

Year of Release: 2005

IMDB: 8.2

Comedy, Romance

Black is the darkest color, the result of the absence or complete absorption of visible light. It is an achromatic color, literally a color without hue, like white and gray.

Cast, director and producer

Display the list of all the persons who have some credit in the movie. The role they play in the movie is also written.

```
query1 = "select * from classroompopcorn.Person natural join classroompopcorn.Credit where MID = '"+movieid+"' and Role = 'Director';"

query2 = "select * from classroompopcorn.Person natural join classroompopcorn.Credit where MID = '"+movieid+"' and Role = 'Actor';"

query3 = "select * from classroompopcorn.Person natural join classroompopcorn.Credit where MID = '"+movieid+"' and Role = 'Producer';"
```

Cast: Amitabh Bachchan, Rani Mukerji

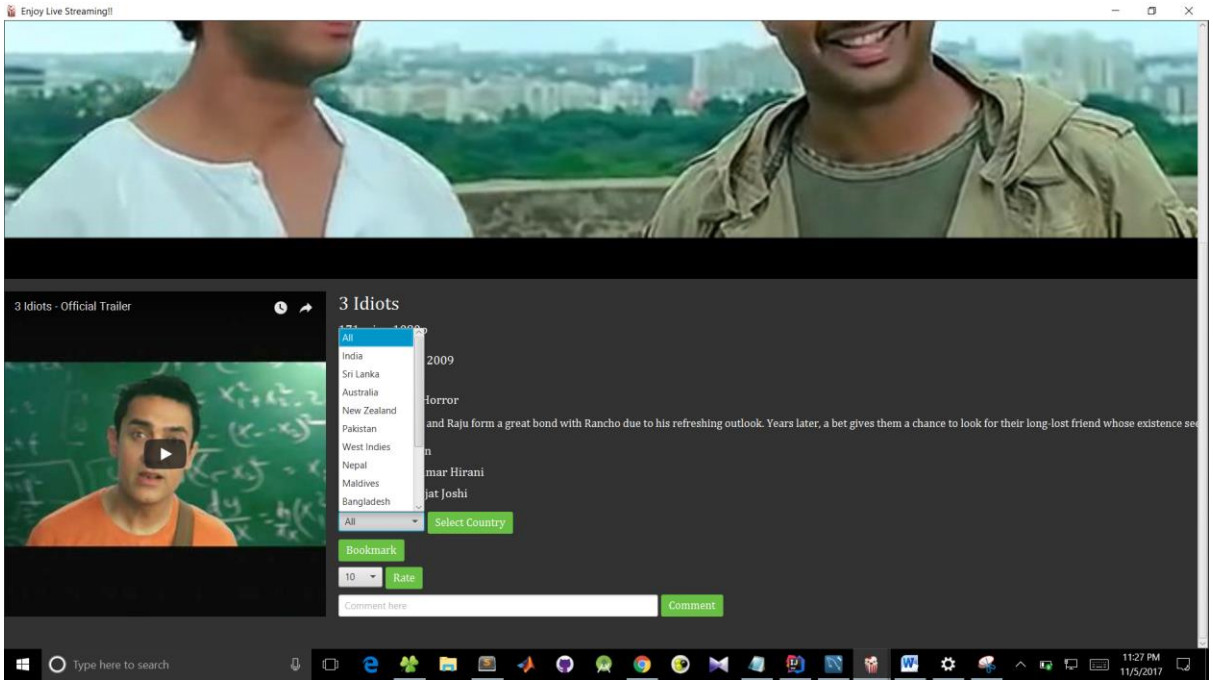
Directors: Bhavani Iyer

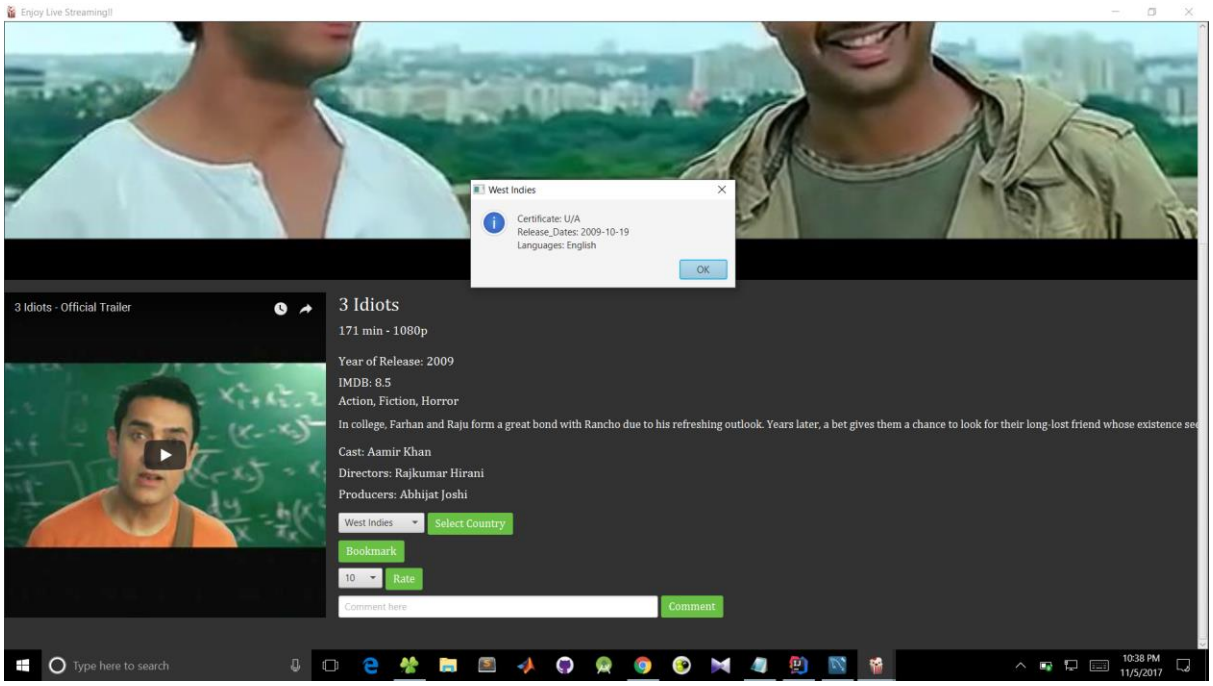
Producers: Sanjay Leela Bhansali

Country wise Releases

A drop down menu of all the countries in the database - Selecting a country displays the release details of the movie specific to the country. These include the release dates, languages released in and the movie certificate.

```
query_certificate = "select Certificate from classroompopcorn.Movie_Certificates where MID ='"+movieid+"' and C_Name='"+c_select+"';"

query_release_date = "select Release_Date from classroompopcorn.Movie_dates where MID ='"+movieid+"' and C_Name='"+c_select+"';"

query_languages = "select Languages from classroompopcorn.Movie_Languages where MID ='"+movieid+"' and C_Name='"+c_select+"';"
```





Add to Bookmark

Bookmark button to bookmark a particular movie. This inserts a row in the database for that user and that movie along with the stream time elapsed.

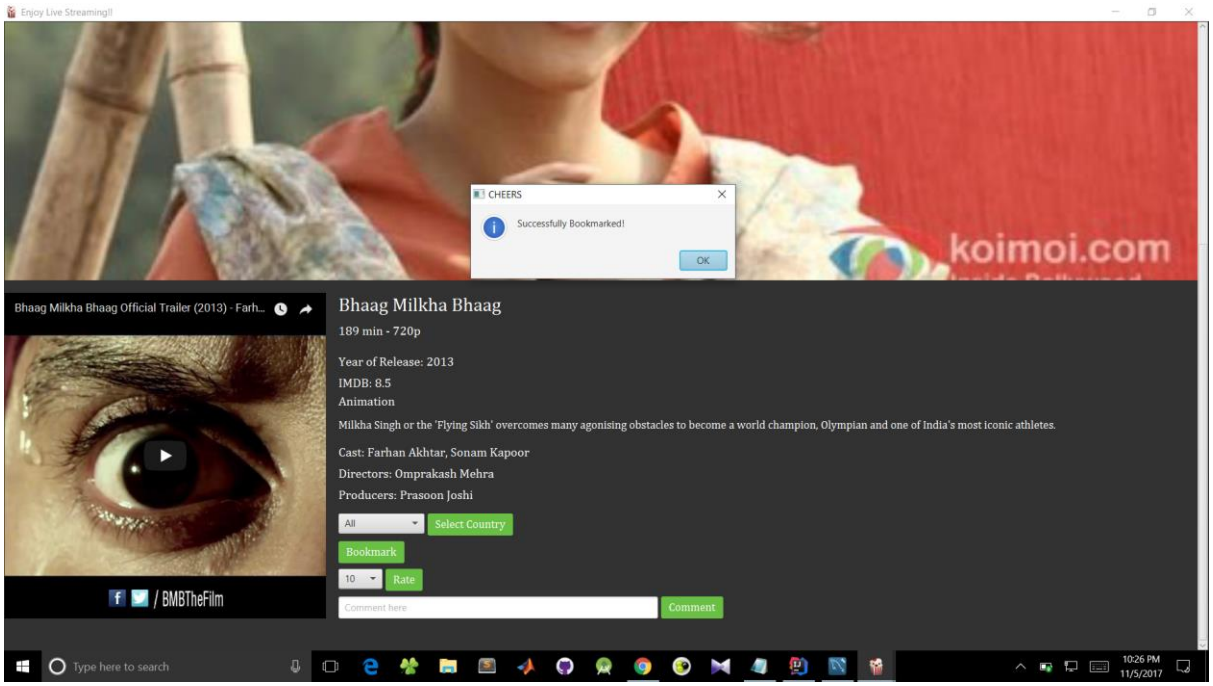
In case no user is logged in on the system upon bookmark, a warning pops up asking to login first .

Also, a notification window confirms a successful bookmark.

query_add_bookmark = ”insert into classroompopcorn.bookmarks values (Email_Id, MID, time_elapsed);”

query_check_login = “select Last_email_id from classroompopcorn.system where MSN=fetched_msn;”

Warning_PopUP = ClassNameHere.infoBox("Please Login First!", "WARNING");



If the last email Id obtained is not null then the user is logged in.

User Rating

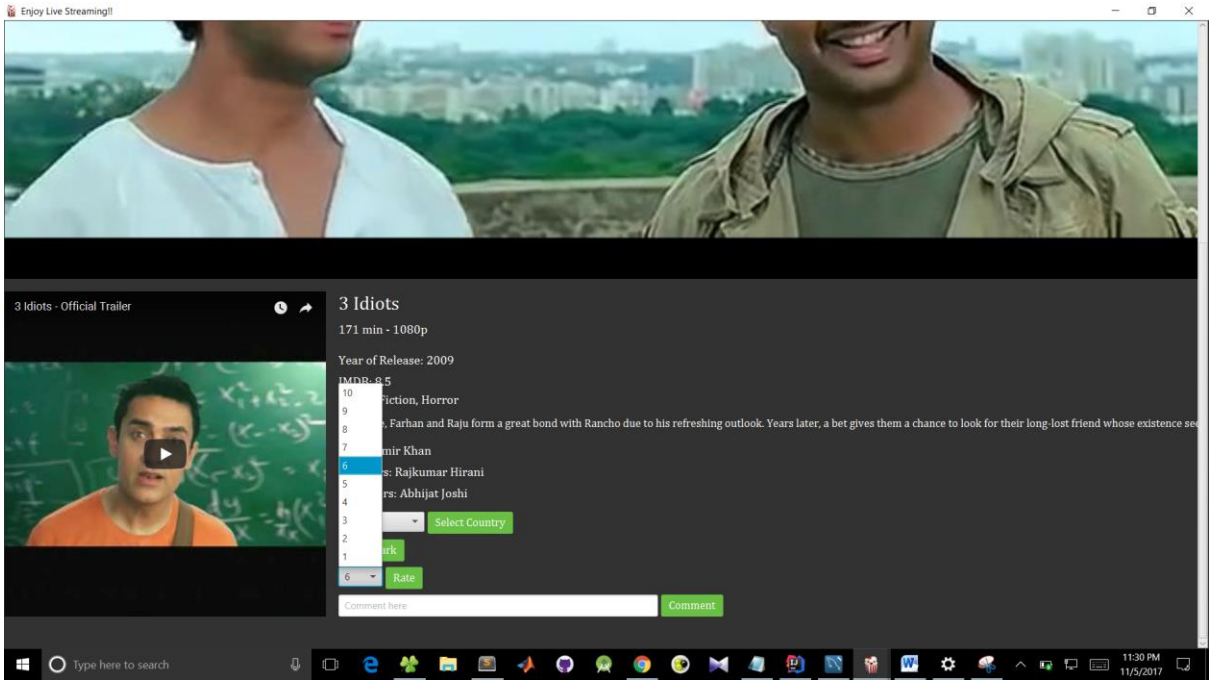
A drop down list for the user to rate the movie from 1 to 10. The rating given is stored in the database in real time

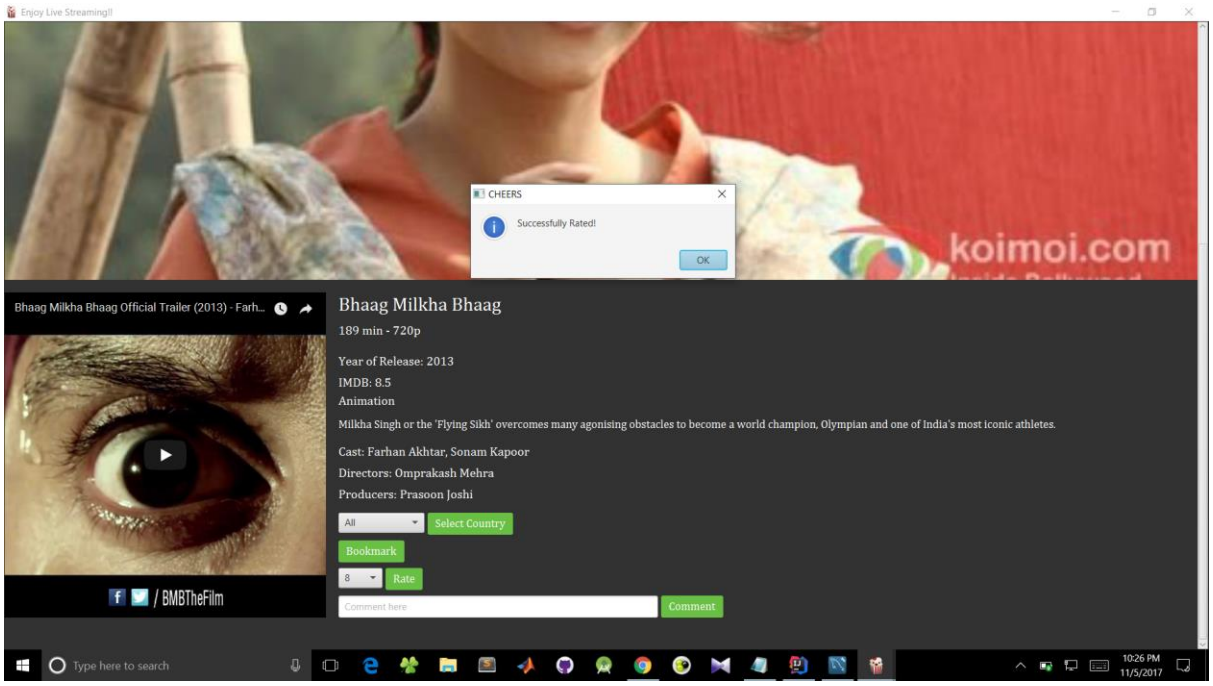
Notification windows similar to bookmark are employed for successful rating and requirement of log-in.

query_rating_insert = "insert into classroompopcorn.reviews values(Email_Id, MID, Time_stamp, Rating, Comment)”

query_rating_update = “Update classroompopcorn.reviews set Time_stamp ='"+fetched_timestamp+"' , Rating ='"+rating+"' where Email_Id = '"+email_id+"' and MID = '"+movieid+"';”

Successful_Rating_PopUp =ClassNameHere.infoBox("Successfully Rated!", "CHEERS");





User Comment

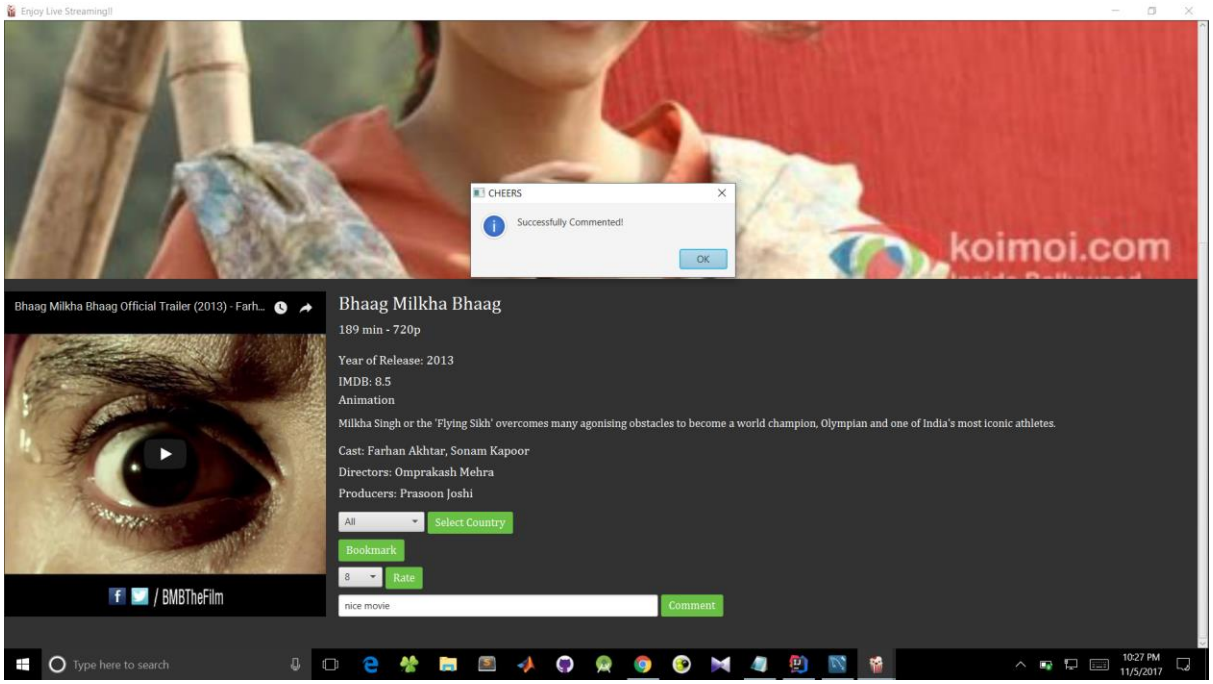
A text box for the user to type in a comment for the movie and push it to the database.

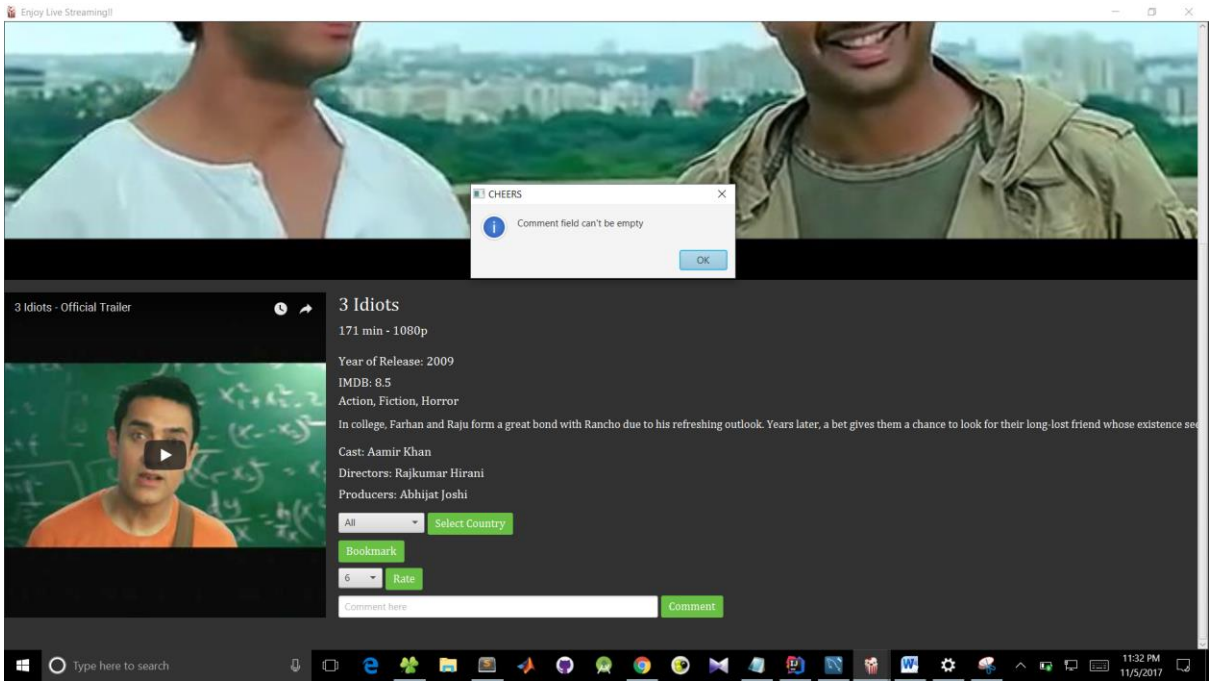
Here, the notification window appears upon the submission of comment in 3 specific cases

(i) Empty comment box

(ii) User logged-out

(iii) Comment successfully stored in database



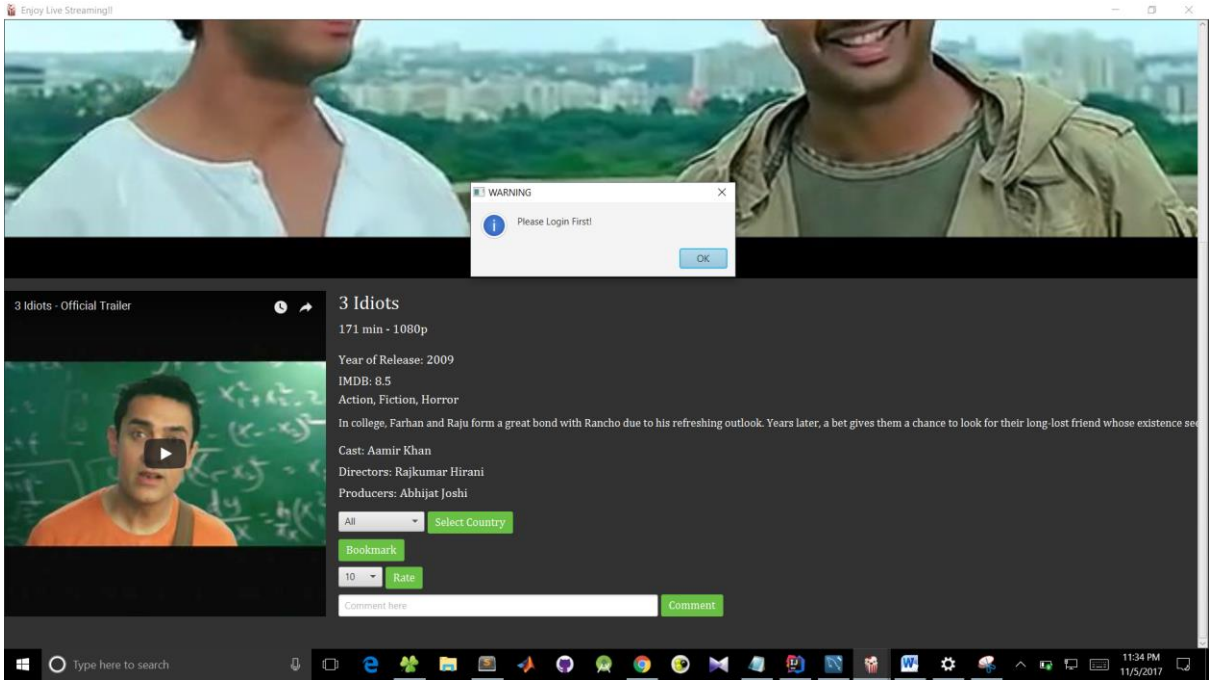


query_Comment_insert = "insert into classroompopcorn.reviews values(Email_Id, MID, Time_stamp, Rating, Comment)"

query_Comment_update = "Update classroompopcorn.reviews set Time_stamp ='"+fetched_timestamp+"' , Rating ='"+rating+"' where Email_Id = '"+email_id+"' and MID = '"+movieid+"';"

Successful_Comment_PopUp = ClassNameHere.infoBox("Successfully Commented!", "CHEERS");

Empty_Comment_Box = searchBox.setText("") ;



Dynamic Updation of Database

In the last 3 operations mentioned above, i.e., bookmark, rating and comment, a row is inserted for each new pair of (emailId, Movie_id).

If that particular operation is being repeated for the above pair, an update command is executed to update the already present row in the database.

query_Comment_update = "Update classroompopcorn.reviews set Time_stamp ='"+fetched_timestamp+"' , Rating ='"+rating+"' where Email_Id = '"+email_id+"' and MID = '"+movieid+"';"

query_rating_update = "Update classroompopcorn.reviews set Time_stamp ='"+fetched_timestamp+"' , Rating ='"+rating+"' where Email_Id = '"+email_id+"' and MID = '"+movieid+"';"