

# BIL372 GRUP 15 PROJESİ

ALPER KAAAN YILDIZ, EGE TAN, SALİH SERT<sup>1</sup>

## 1. Genel Bakış

MovieForum bütün film sevdalılarına hitap eden, insanların film seçerken geniş bir topluluğun düşüncelerini görebildiği, filmler hakkındaki fikirlerini başka insanlarla paylaşabildiği, yayınlanmış veya henüz yayınlanmamış filmler ile ilgili önemli bilgiler öğrenebildiği bir web sitesidir.

Fikir covid-19 pandemi döneminde evlerinde uzun zaman geçirmiş ve geçirecek olan bir grup TOBB ETU öğrencisi tarafından ortaya atılmış ve bu dönemde evlerine kapanan olabildiğince çok kişiye ulaşabilmesi açısından bir web sitesi olarak uygulanmıştır. Bu sayede geniş bir izleyici topluluğuna hitap etmek, bu topluluğa yeni bireyler katmak ve insanların evlerinde daha uzun süre geçirmesi amaçlanmıştır.

Projenin sağladıkları sayesinde siteye kayıtlı olan her kullanıcı için kendi bölgesindeki filmleri ve sanatçıları tanıtmaya şansı olacaktır. Bu sayede izleyici topluluğu sadece popüler filmler ve ünlü sanatçıları değil de dünyanın dört bir yanından gelen bilgiler doğrultusunda bölgesel filmlerin ve sanatçıların varlığını keşfedeceklerdir.

## 2. Veritabanı Yapısı

Veritabanı Tabloları	Açıklama
<b>Actor</b>	Aktör bilgilerini tutan tablodur. Bu bilgiler “Create Actor” formunda doldurulur. Aktör sayfalarının görüntülenmesinde kullanılır.
<b>Comment</b>	Yapılan yorumları, bu yorumları kimlerin yaptığını, hangi filmlere yapıldığını ve tarihini tutan tablodur. Film sayfalarında yorumların görüntülenmesinde kullanılır.
<b>Country</b>	Ülke isimlerini tutan tablodur. Aktör, yapımcı ve yönetmen oluşturulurken kullanılır.
<b>Director</b>	<p>Yönetmene ait kişisel bilgileri ve sosyal medya hesaplarını tutan tablodur.</p> <p>Yönetmen profilinin görüntülenmesinde kullanılır. “Create Director” formu ile doldurulur.</p>

<b>Favorite</b>	<p>Kişinin favori filmlerini tutan tablodur.</p> <p>Profilden favori filmler listesine erişilirken kullanılır.</p>
<b>Genre</b>	<p>Film türlerini tutan tablodur. “Create Movie” formunda kullanılır.</p>
<b>Many_movie_has_many_actor</b>	<p>Filmler ve aktörler arasındaki “many-to-many” ilişkisini tutan tablodur.</p>
<b>Many_movie_has_many_director</b>	<p>Filmler ve yönetmenler arasındaki “many-to-many” ilişkisini tutan tablodur.</p>
<b>Many_movie_has_many_genre</b>	<p>Filmler ve türler arasındaki “many-to-many” ilişkisini tutan tablodur.</p>
<b>Many_movie_has_many_producer</b>	<p>Filmler ve yapımcılar arasındaki “many-to-many” ilişkisini tutan tablodur.</p>
<b>Movie</b>	<p>Film bilgilerini( çıkış tarihi, isim, bütçe, gelir, trailer, süre vb.) tutan tablodur. Film sayfalarının görüntülenmesinde kullanılır. “Create Movie” formu ile doldurulur.</p>

<b>MovieForumApp_user</b>	Kaydolan olan kullanıcıların bilgilerini tutar. Kullanıcı oturum açarken kullanılır.
<b>Producer</b>	Yapımcıya ait kişisel bilgileri ve sosyal medya hesaplarını tutan tablodur. "Create Producer" formu ile doldurulur.
<b>Rating</b>	Kullanıcıların değerlendirmelerini ve değerlendirme yaptıkları filmleri tutan tablodur. Kullanıcı profilinden bu filmlere erişilmesinde ve filmlerin ortalama değerlendirme puanlarının gösterilmesinde kullanılır.
<b>Reply</b>	Yorumlara yazılan cevapları ve bu cevapları kimlerin yazdığını tutan tablodur. Yorum cevaplarının görüntülenmesinde kullanılır.
<b>Django_session</b>	Giriş yapmış olan kullanıcıların oturum bilgilerini çerez halinde tutar. Kullanıcılar siteden ayrılırlar bile oturumdan çıkış yapmadıkları sürece oturumları açık kalır. Tekrar siteye girdiklerinde kolaylık sağlar.

### 3. İşlevsellik

Siteye giren herkes aktör, yapımcı, yönetmen ve film sayfalarına ulaşabilmektedir. Fakat sadece kayıtlı kullanıcılar bu sayfaları oluşturma, filmlere yorum yapma ve puan verme, profillerini görüntüleme, düzenleme ve favoriler listesine erişim sağlamaktadır. Kullanıcılar yorum yaptıkları veya puan verdikleri filmlere profillerinden erişebilmektedir. Ayrıca filmleri favoriler listesine ekleyebilir ve bu listeye de profilden ulaşabilmektedir.

### 4. Veritabanı Tasarım Süreci

Veritabanı tasarımında öncelikle miniworld assumption oluşturulmuş; varlıklar, bu varlıkların özellikleri ve diğer varlıklarla arasındaki ilişkiler tanımlanmıştır. Kullanıcılar, filmler, sanatçılar ve yorumlar tasarımdaki temel varlıklardır. Yorum yanıtları ve puanlandırmalar ise yardımcı ve tamamlayıcı görev görmektedir.

Miniworld assumption tamamlandıktan sonra EER diagramı oluşturulmuştur. Bu diyagramda varlıklar ve özellikleri, varlıklar arasındaki ilişkiler ve ilişki türleri açıkça belirtilmiş ve veritabanı oluşturulmasına ortam hazırlanmıştır. Projede web framework olarak Django kullanılmış ve veritabanı PostgreSQL ortamında oluşturulmuştur.

“Actor”, “director” ve “producer” tabloları ilgili sanatçıların bilgilerini saklamaktadır. Auto-increment özelliğine sahip “Actor\_id”, “director\_id” ve “producer\_id” bu tablolardaki primary keylerdir ve “country\_name” foreign keyi country tablosundaki country\_name’i referans göstermektedir. “Age” sütunu türetilmiş özellik olarak tanımlanmış, doğum tarihi bilgisinden kişinin şu anki yaşını göstermektedir. “Age”i hesaplamak için trigger kullanılmıştır.

Comment tablosu kullanıcıların yaptıkları yorumu tarihiyle birlikte saklamakla beraber, aynı zamanda yorumun yapıldığı film ve kullanıcı id’sini de içinde barındırır. “Comment\_id” bu tablonun primary keyidir ve auto-increment özelliğine sahiptir.. “Movie\_id\_movie” foreign keyi movie tablosundaki “movie\_id”yi, “user\_id\_user” ise “MovieForumApp\_user” tablosundaki “id”yi işaret etmektedir. Bu sayede yapılan yorumlar, filmler ve kullanıcılar arasındaki ilişki izlenebilmektedir.

“Favorite” tablosu kullanıcıların favori filmlerini içerisinde barındırır. Primary keyi “favorite\_id”; foreign keylerinden “movie\_id\_movie” movie tablosundaki “movie\_id”yi, “user\_id\_user” ise “MovieForumApp\_user” tablosundaki “id”yi işaret etmektedir.

*Many\_movie\_has\_many\_actor, Many\_movie\_has\_many\_director,* *Many\_movie\_has\_many\_genre,*  
*Many\_movie\_has\_many\_producer* tabloları varlıklar arasındaki “many-to-many” ilişkilerini bağladığı varlıkların primary keylerini foreign key şeklinde alarak göstermektedir. Bu tabloların primary keyleri “id” şeklinde tanımlanmıştır ve auto-increment özelliğine sahiptir.

“Movie” tablosu film bilgilerini içerisinde barındırır ve primary keyi auto-increment özelliğine sahip olan “movie\_id”dir. “Revenue” sütunu türetilmiş özelliktir ve “income - budget” şeklinde hesaplanır.

MovieForumApp\_user tablosu kullanıcı bilgilerini kaydolma tarihiyle birlikte saklar ve primary keyi auto-increment özelliğine sahip “id”dir.

Rating tablosu kullanıcı puanlandırmalarını içerisinde barındırır, ilgili film ve puanlandırma yapan kullanıcıya ait id’leri saklar. Primary keyi auto-increment özelliğine sahip “rating\_id”dir. Foreign keylerinden biri olan “movie\_id\_movie” movie tablosundaki “movie\_id”yi referans gösterirken “user\_id\_user” ise “MovieForumApp\_user” tablosundaki “id”yi işaret etmektedir.

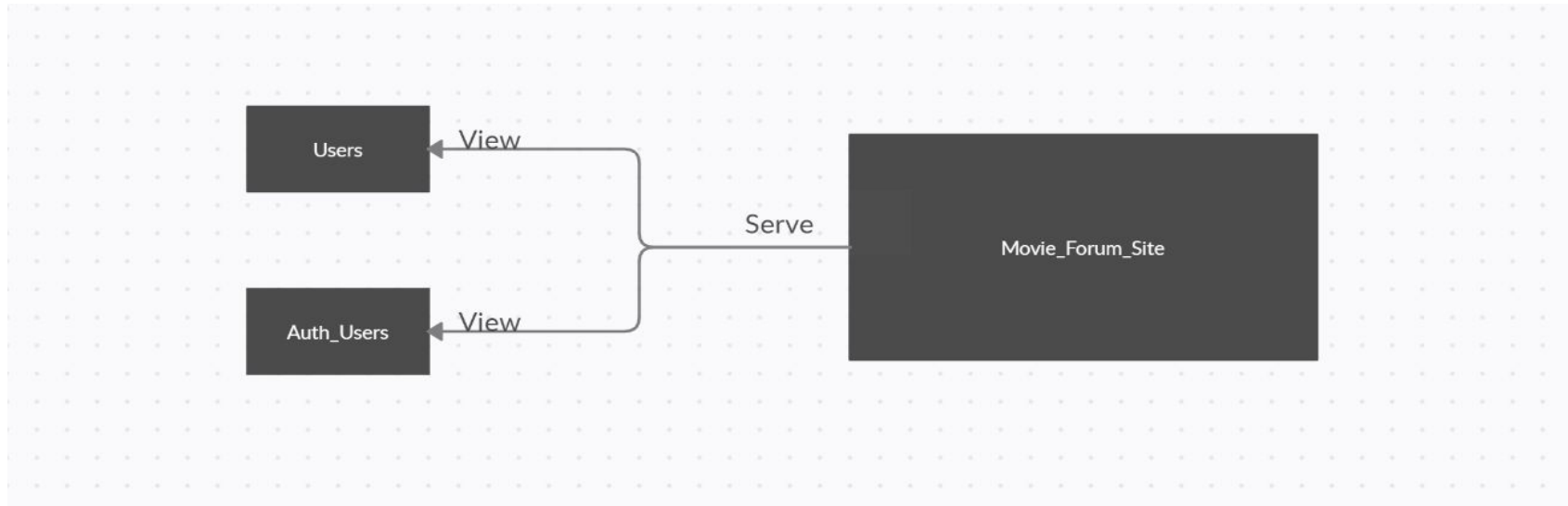
Reply tablosu yapılan yorumlara yazılan cevapları tarihiyle birlikte içerisinde barındırır. Primary keyi auto-increment özelliğine sahip “reply\_id”dir. Foreign keylerinden biri olan “comment\_id\_comment” comment tablosundaki “comment\_id”yi referans göstermekteyken “user\_id\_user” ise “MovieForumApp\_user” tablosundaki “id”yi işaret eder.

## 5. Özet

Proje bütün grup için güzel bir öğrenme deneyimiydi. SQL, HTML ve Python-Django becerilerimizi geliştirmemize izin verdi ve bir takım halinde uyum içinde çalışmayı gösterdi. MovieForum uygulamasını insanlara dünyanın farklı bölgelerindeki güzel ve kaliteli filmleri göstererek evlerinde geçirdikleri vakti daha eğlenceli kılmak için geliştirdik ve çabalarımızın sonuçlarından gayet memnunuz.

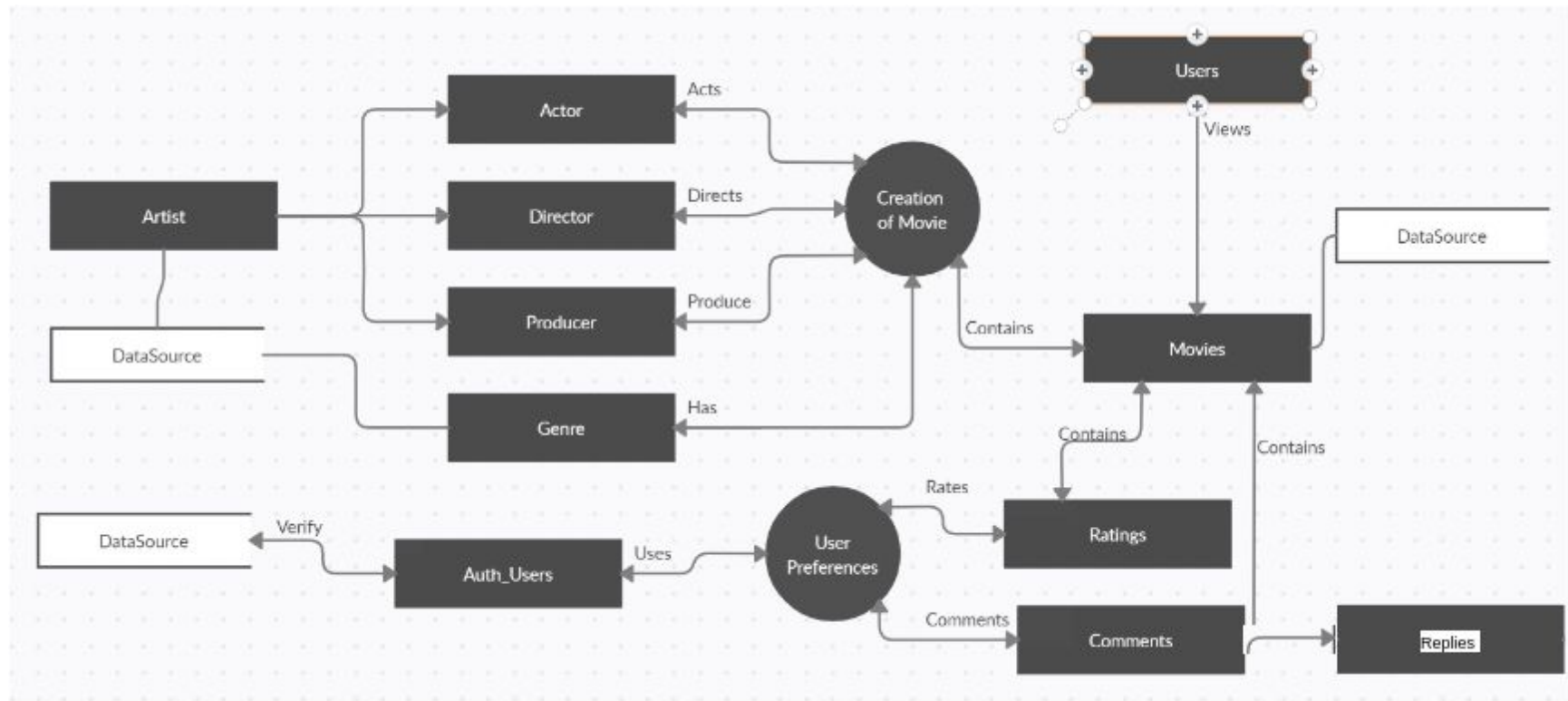
## Data Flow Diagram

Level 0:

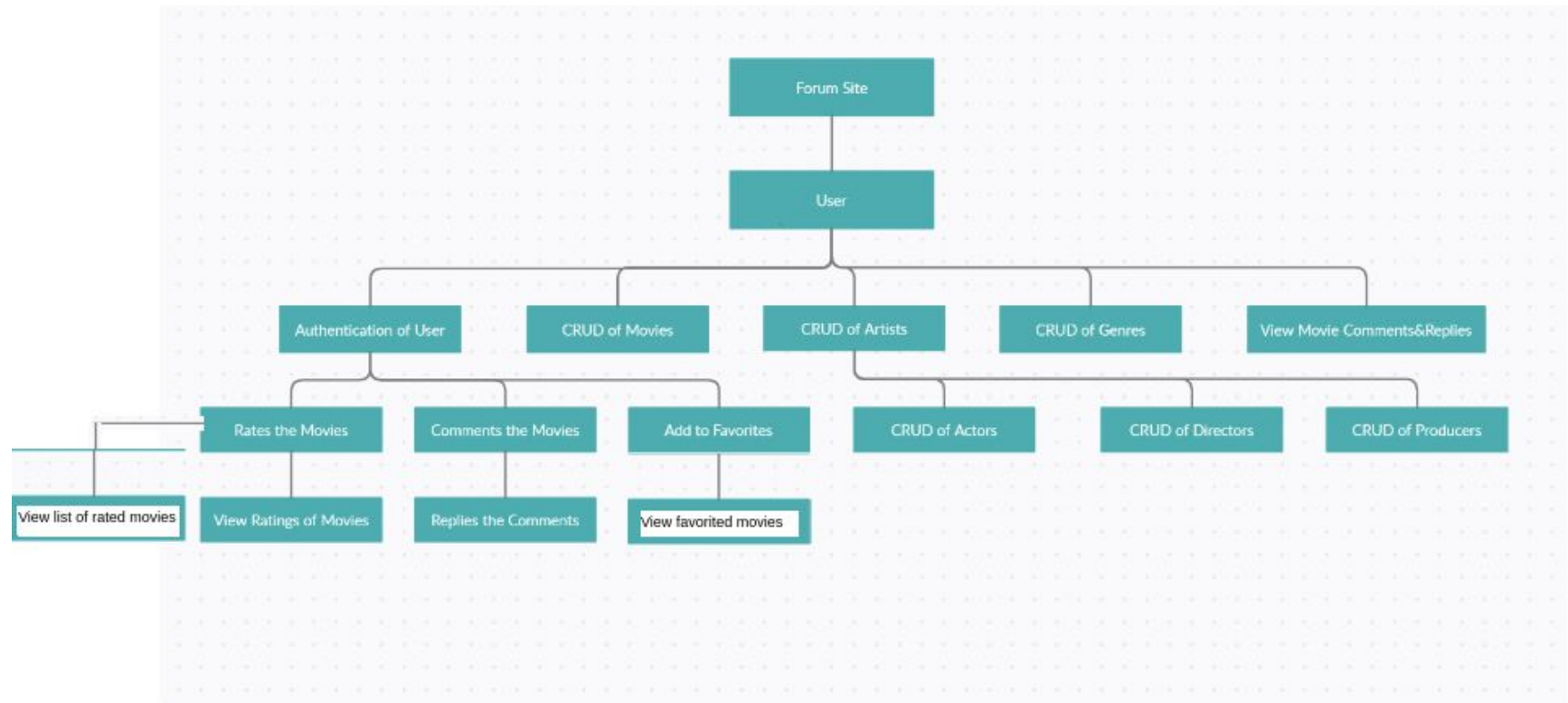




## Level 1:

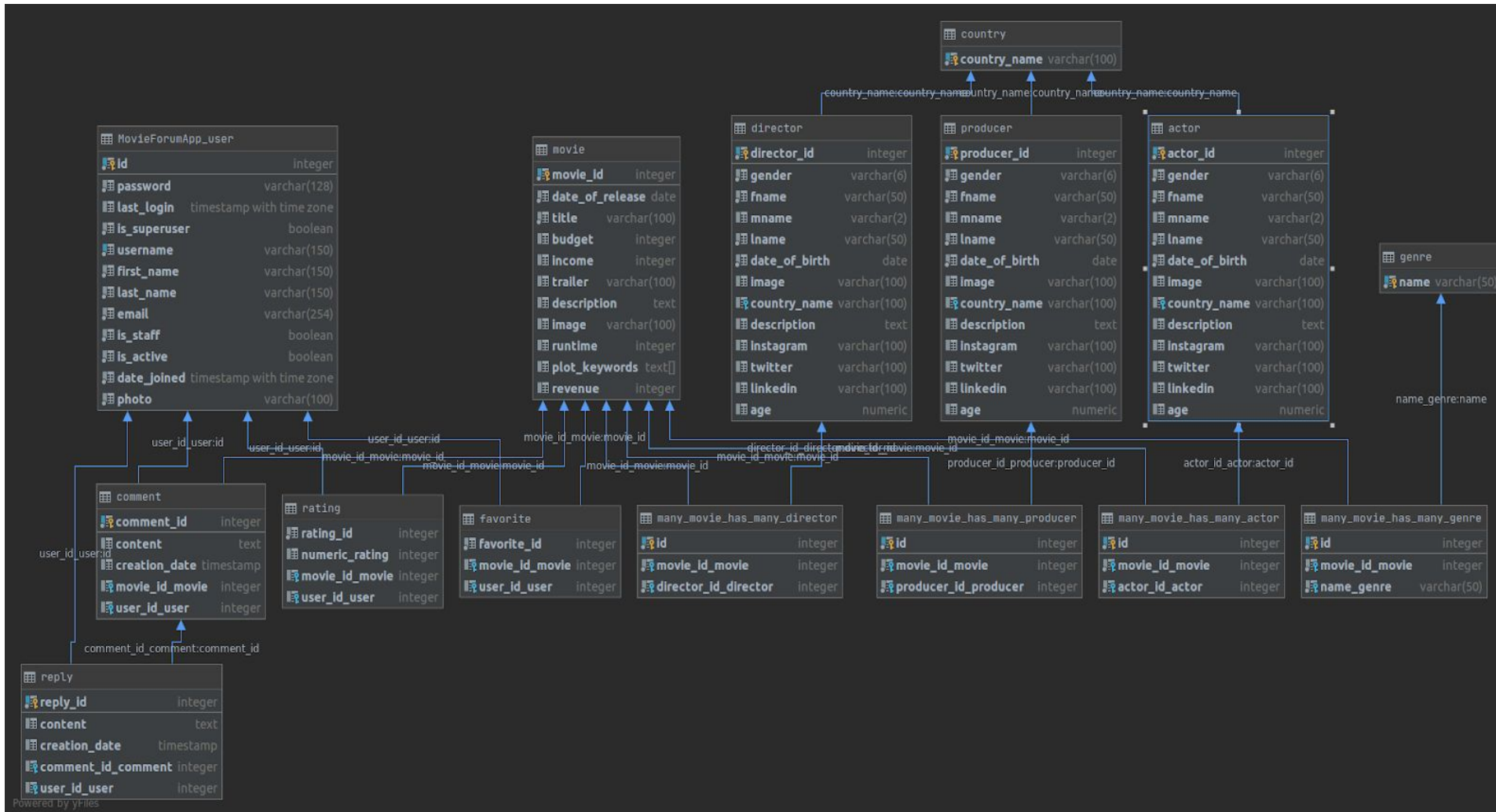


## HIPO

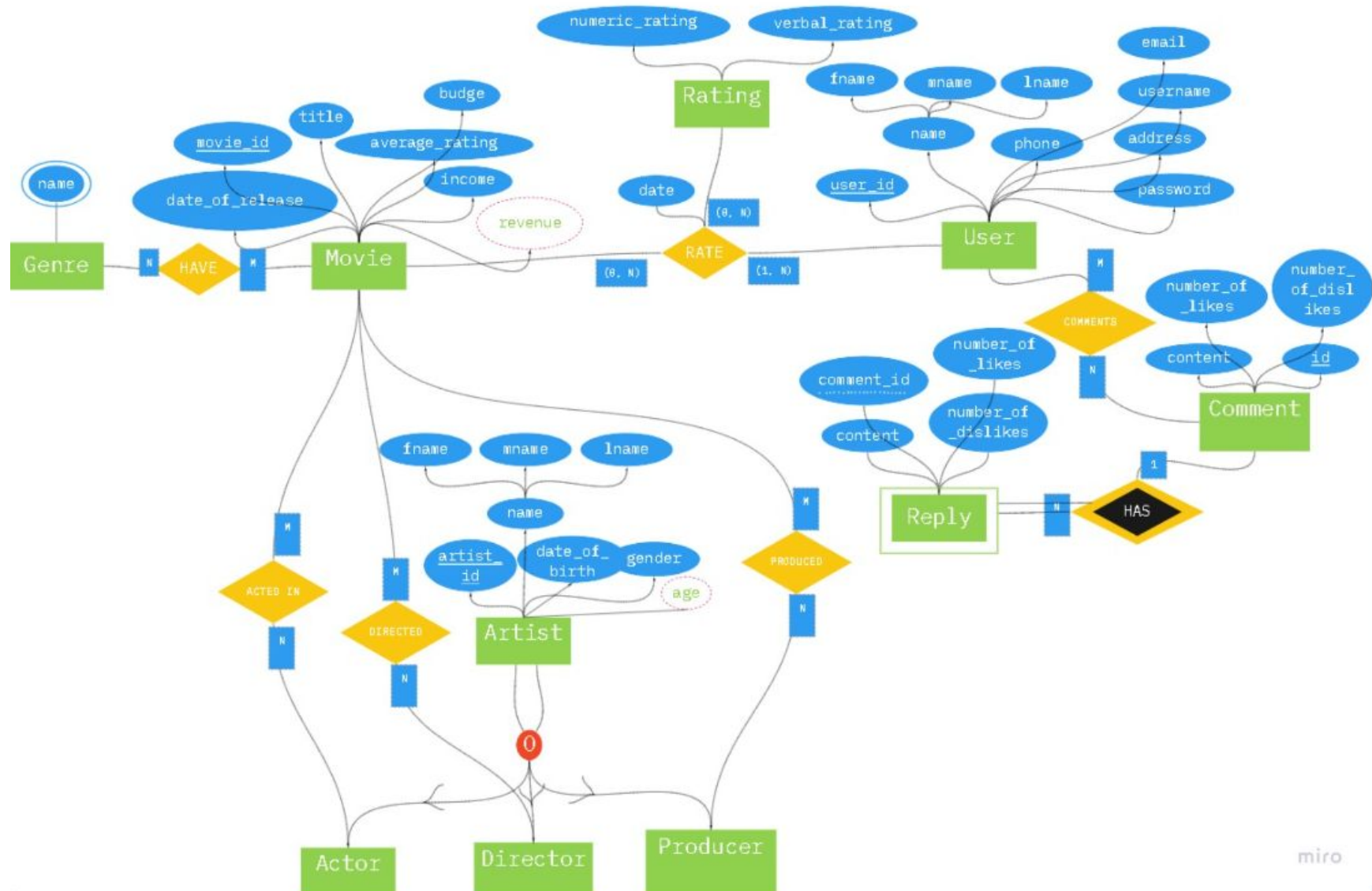


## APPENDIXES

### Appendix A. Database schema ( includes only tables relevant to this project, does not contain Django related tables.)



## Appendix B. ER/EER schema



## Appendix C. GUI SQL's & Explanation of each related GUI form.

### C.1 GUI Form of Login functionality.

This GUI provides functionality for a registered user to login.

The main SQL here is authenticating the user which has a structure like this:

```
SELECT * FROM "MovieForumApp_user" U WHERE U.username = provided_username AND U.password = provided_password
```

By executing this query, we will get a row or not. Depending on the row count returned, we will verify the login attempt.

The GUI form.

HOME MOVIES~ ACTORS~ PRODUCERS~ DIRECTORS~

LOGIN REGISTER

### LOGIN

USERNAME:  
Alper\_kaan\_yildiz

PASSWORD:  
\*

LOGIN

## C.2 GUI Form of Register functionality

This GUI provides functionality for a visitor to register.

The main SQL here is registering the user which has a structure like this:

```
INSERT INTO "MovieForumApp_user" (id, password, last_login, is_superuser, username, first_name, last_name, email, is_staff, is_active, date_joined, photo) VALUES (2, 'pbkdf2_sha256$216000$QKBV9o2LvUGn$HLLNU8uGE1x4ULadJOcy7RpCuPRiA9dWb0ktU+a+r4o=', '2020-12-07 03:15:06.463468', false, 'Demo', 'Demo', 'User', 'demouser@demo.com', false, true, '2020-12-07 03:15:06.085812', 'image/20/download.png');
```

By executing the query, we will first validate the register form and if valid, we will create an instance using the insert SQL above.

The GUI form.

## REGISTER

USERNAME:

FIRST NAME: LAST NAME:

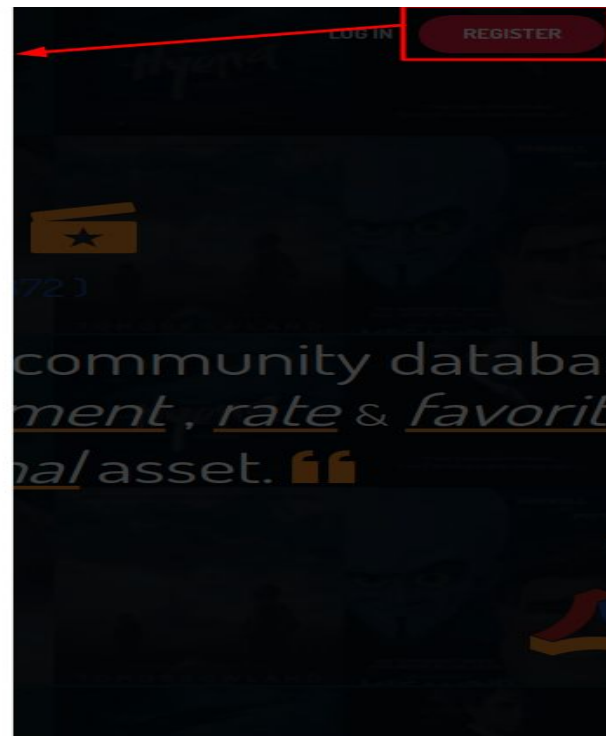
PHOTO:

No File Chosen

YOUR EMAIL:

PASSWORD:

RE-TYPE PASSWORD:



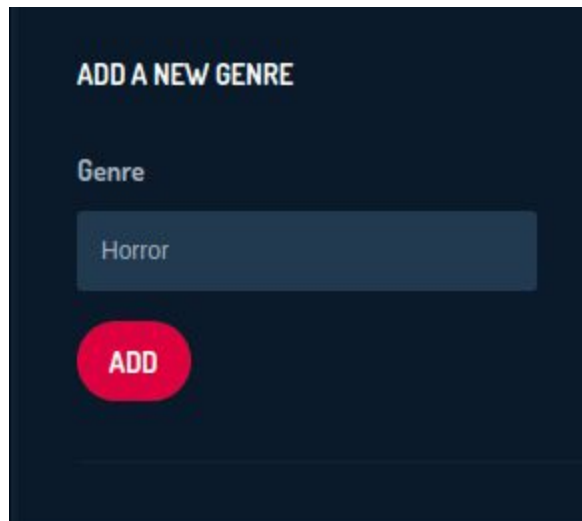
### C.3 GUI Form for adding a new genre functionality

This GUI provides functionality for adding a new genre.

The main SQL here is inserting a new genre which has a structure like this:

```
INSERT INTO genre (name) VALUES ('Horror');
```

By executing the query, we will insert a new genre to genres table.



ADD A NEW GENRE

Genre

Horror

ADD



#### C.4 GUI Form for adding a new movie functionality

This GUI provides functionality for adding a new movie.

The main SQL here is inserting a new movie which has a structure like this:

By executing the query, we will insert a new movie to movies table.

This form has 4 select fields, on selection each select field will send the data required to insert to ManyToMany tables.

The SQL below will insert a new movie.

```
INSERT INTO movie (date_of_release, title, budget, income, trailer, description, image, runtime,
plot_keywords, revenue) VALUES ('2020-11-30', 'Die Hard', 200, 100,
'https://www.youtube.com/watch?v=DLX62G4lc44', 'Description of the "Die Hard" movie.',
'image/20/die_hard_TWKH1zU.jpg', 120, '{perfect,actionfull,costly,worth-watching}', -100);
```

The SQL below will make two insertions to ManyToMany relationship table between actor and movie.

```
INSERT INTO public.many_movie_has_many_actor (movie_id_movie, actor_id_actor) VALUES (1, 1);
INSERT INTO public.many_movie_has_many_actor (movie_id_movie, actor_id_actor) VALUES (1, 2);
```

Director and Producer insertions are not shown for brevity.

The values (1, 1) and (1, 2) come from the select fields, which are all shown on the form image below.



## ADD A NEW MOVIE

Title

Release Date

Income

Budget

Runtime

Actors

Directors

Producers

Trailer

Image

Keywords

Genres

Description

ADD

Movie form has a date field.

income & budget fields will be used and the generated column "revenue" will be filled in.

Listing actors, directors, producers and genres are done by executing (SELECT \*) and sending the resulting rows to select field.

Image is an image field. We will store the image locally and save its path to table.

Keywords is a Postgres Array field. Each comma seperated keyword will be an element of the resulting array.

**C.5 GUI Form (Actually a view) for listing all movies.**

**This GUI provides functionality for viewing movies.**

**The main SQL here is selecting movies & their ratings which has a structure like this:**

**By executing the query,**

```
SELECT * FROM MOVIES
```

Not showing Python code here,

We will do a **for loop to iterate over** the resulting movies.

**And then get** the related **average\_rating** **by**:

```
SELECT AVG(NUMERIC_RATING) FROM RATINGS  
WHERE movie_id = movie.movie_id
```

40 STORIES OF SHEER ADVENTURE!



**DIE HARD**

★ 9.0 /10



**LORD OF THE RINGS**

★ 7.0 /10

This is actually a view.

When user clicks on Movie List, we will show all movies in the movies table.

For each resulting movie, we will also get the average rating from ratings table.

**C.5 GUI Form (Actually a view) for single movie detail.**

**This GUI provides functionality for viewing a single movie.**

**The main SQL here is filtering out the clicked movie and retrieve itself, many-to-many relationships (actors, directors etc.) which has a structure like this:**

```
SELECT * FROM MOVIE (We will get all attributes, then use the movie_id in below queries.  
SELECT * FROM MANY_MOVIE_HAS_MANY_ACTOR WHERE movie_id = movie_id  
SELECT * FROM MANY_MOVIE_HAS_MANY_DIRECTOR WHERE movie_id = movie_id  
SELECT * FROM MANY_MOVIE_HAS_MANY_PRODUCER WHERE movie_id = movie_id
```

40 STORIES OF SHEER ADVENTURE!



▶ WATCH TRAILER

## Die Hard

NOV. 30, 2020



ADD TO FAVORITES

If user clicks this button, we will create a new row in the favorite table with attributes user\_id and movie\_id

Single movie detail page.

We will perform several queries here.



9.0/10

1 Ratings

Your Rating:



If user already rated, which we understand by querying the ratings table with attributes user\_id and movie\_id, we show amount of stars equal to rating.

OVERVIEW

REVIEWS

CAST & CREW

Description of the "Die Hard" movie.

### CAST

Bruce Willis

Morgan J. Freeman

Director:

Stars:

Bruce Willis, Morgan J. Freeman,

Get each actor from ManyToMany table of Actor-M relationship.

Genres:

Fantasy, Romance, Political,

Get each genre...

### USER REVIEWS

[See All Reviews >](#)

Release Date:

Nov. 30, 2020

Run Time:

141 min

Plot Keywords:

perfect actionfull costly worth- watching

## C.6 GUI Form for filtering - searching.

This GUI provides functionality for filtering out the wanted movies.

The main SQL here is:

```
SELECT * FROM MOVIE WHERE GENRE = GENRE1 OR GENRE = GENRE 2 ... AND RELEASE_YEAR >
input_release_year_begin AND RELEASE_YEAR < input_release_year_end
```



DIE HARD

★ 9.0/10



LORD OF THE RINGS

★ 7.0/10

We also implemented filter feature. You can select the genre(s) and year range of released movies, we will execute the filtering SQL query and return the resulting movies.

### FILTER MOVIES

Genres

Horror x

Comedy x



Adventure x

Sci-Fi x

Release Year

From year ...

To year ...

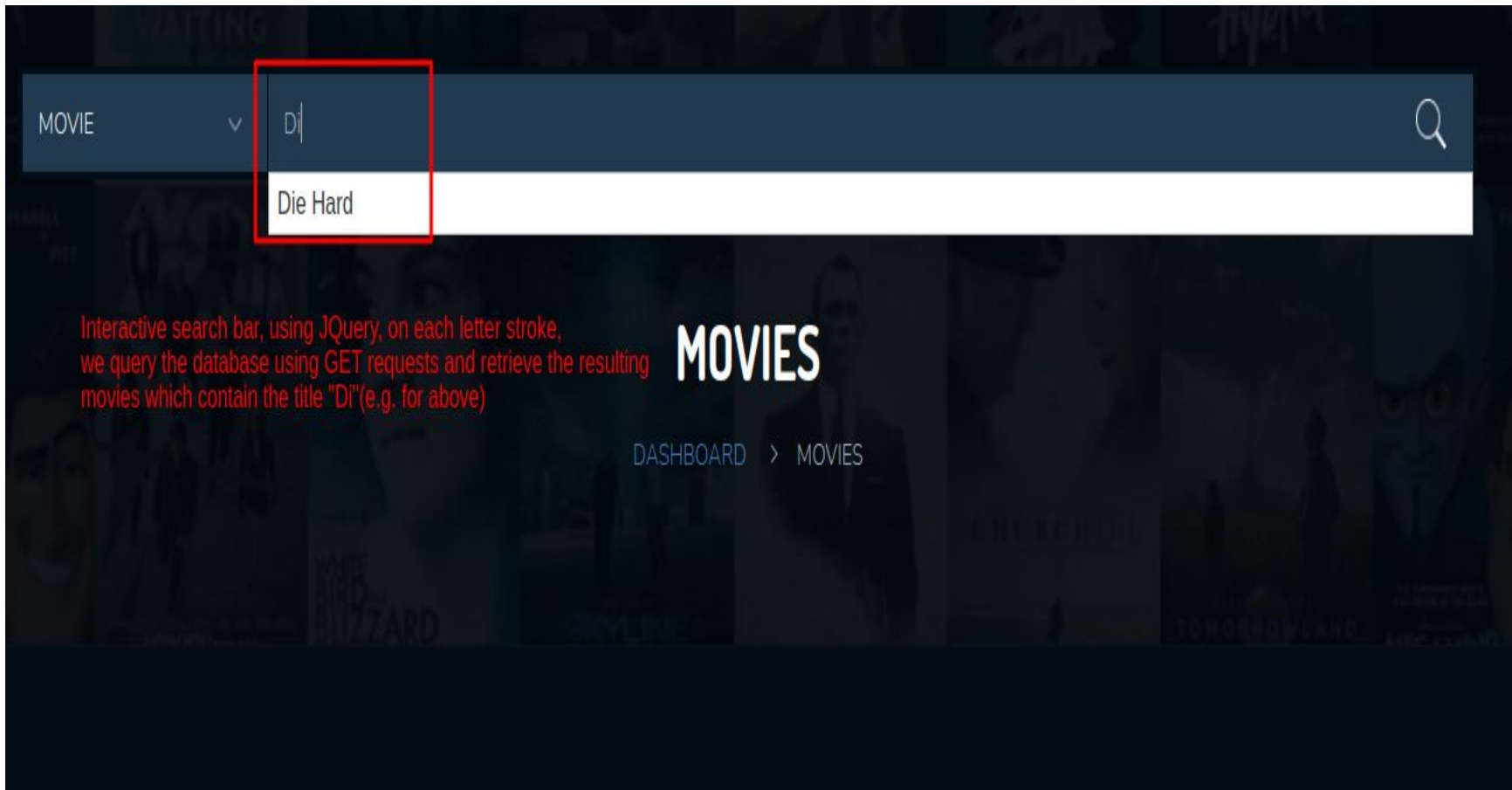
SUBMIT

### C.7 GUI Form for searching movies.

**This GUI provides functionality for searching the wanted movie.**

**The main SQL here is:**

```
SELECT * FROM MOVIE WHERE title = users_input_on_title_field
```



### C.7 GUI Form for comments & replies.

This GUI provides functionality for viewing comments & replies. And making new comments & replies.

The main SQL here is:

```
SELECT *, COUNT(*) FROM COMMENTS WHERE movie_id = movie_id_of_the_movie_we_are_browsing
```

A user can have many comments of many movies. A comment may have many replies.





Reviews written to

## Die Hard

Found 2 reviews in total.

alper\_kaan\_yildiz

Dec. 7, 2020, 7:08 p.m.

This is a demo review for project report. Hi!

demouser

Dec. 7, 2020, 7:08 p.m.

Hey, that is a very nice review. Here is a reply.

alper\_kaan\_yildiz

Dec. 7, 2020, 7:08 p.m.

Thank you! I appreciate it.

Make a reply.

REPLY

demouser

Dec. 7, 2020, 7:08 p.m.

Wow, this movie is really great.

Make a reply.

REPLY

Share Your Thoughts

COMMENT

Query the comments table and retrieve the count of rows where their movie id is "Die Hard"s movie id.

ForeignKey to comment owner user.

Comment datetimefield.

ForeignKey to reply owner user.

A movie has many comments.

A comment has many replies.

C.8 GUI Form for listing all actors.

The main SQL here is:

```
SELECT * FROM ACTORS  
SELECT * FROM ACTORS WHERE country_name = wanted_country AND year...
```

# ACTOR LIST

DASHBOARD > ACTORS

A filter similar to movies, but now we filter by actors country & birth year.

Found 4 actors in total

Total actors.



**Bruce Willis**

ACTOR, USA

Actor and musician Bruce Willis is well known for ...

Actor image path is queried then rendered on HTML.



**Morgan J. Freeman**

ACTOR, USA

With an authoritative voice and calm demeanor, thi...

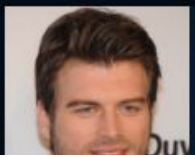
A ForeignKey to Country table.



**Leonardo DiCaprio**

ACTOR, USA

Few actors in the world have had a career quite as...



**Kivanc Tatlitug**

ACTOR, TURKEY

Kivanc Tatlitug (Turkish pronunciation: [kw'vantf] ...

## SEARCH ACTOR

Actor Country

Turkey

Birth Year

From ye

To year

SUBMIT

## FEATURED ACTOR



**Bruce Willis**

ACTOR

A random actor chosen from actors table.

### C.9 GUI Form for creating an actor.

The main SQL here is:

```
INSERT INTO public.actor (gender, fname, mname, lname, date_of_birth, image, country_name, description,
instagram, twitter, linkedin) VALUES ('Male', 'Morgan', 'J.', 'Freeman', '1937-06-01',
'image/20/morgan_freeman.jpg', 'USA', 'With an authoritative voice and calm demeanor...Rest of the
desription is cut for brevity', 'https://www.instagram.com/morganfreeman/?hl=en',
'https://twitter.com/mjfree', 'https://www.linkedin.com/in/morgan-freeman-4ab95a81');
```

We are not inserting to age here, because age will be computed using `now() - year_part(date_of_birth)` provided by PostgreSQL.

## ADD A NEW ACTOR

First Name

Middle Name

Last Name

Date of Birth

Gender

Country

Instagram profile

Twitter profile


LinkedIn profile

Description

Movies acted in

Die Hard

Lord of the rings



Picture

Choose File

No file chosen

ADD

Rest of the form is not explained for brevity.

Adding a new director or producer has also similar forms.

Again, many-to-many between movie and an actor.

**C.10** This appendix is added for explanation. We are not including director and producer related form because they are quite similar to actor forms and views.

**C.11** SQL script for creating tables. ([GitHub repository, .sql file link](#))  
This .sql also has small examples for inserting users, genres, countries.