Web Studio: Week 4 Turn-in #3

Name: Ivy Chen
UNI: ic2389
Due: Friday 9/28 at 2PM
COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. Allow users to accomplish tasks of browsing through movies on the home page, searching for movies/actors/directors/producers, viewing detailed information about a movie, and finding sequels of movies

Low Level Goals

This week's low level goals comprise the 10 user needs I identified from last week. I break down each of these low level goals into subgoals.

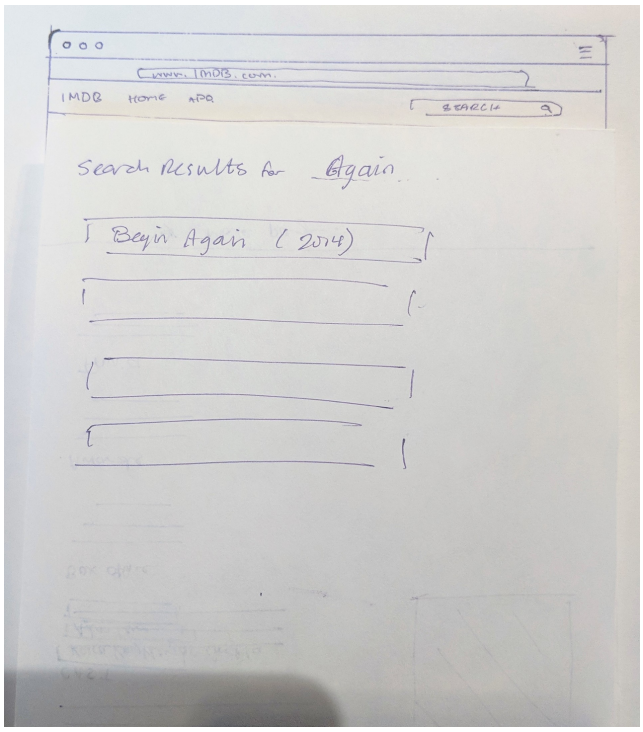
1. Allow users to look up actor filmography in chronological order
    a. On an actor's page, sort their filmography by release date of a movie they starred in using reverse chronological order
2. Users can browse movie posters, maps to the real world task (maybe outdated) of browsing shelves in a physical store looking for movies to watch
    a. Add a new field to the movie table to identify poster URL
3. Users can look up directors, producers of a movie
    a. Create separate/special roles for directors and producers
4. Look up awards a movie has won
    a. Create an Awards table with fields: Id, Event (eg. Academy), Award (eg. Oscar), Year (eg. Best Actor), Rank (eg. Win)
5. Find series of movies (eg. Three Flavours Cornetto trilogy, Spanish Apartment Trilogy, all the Hunger Games movies) and be able to identify the sequel/prequel from each movie's page
6. Allow users to view movie budget and box office earnings
    a. Add budget and box office earnings attributes to movie table
7. User can use a search bar and filters to find films with certain actors, genres, etc.
    a. Add a search bar

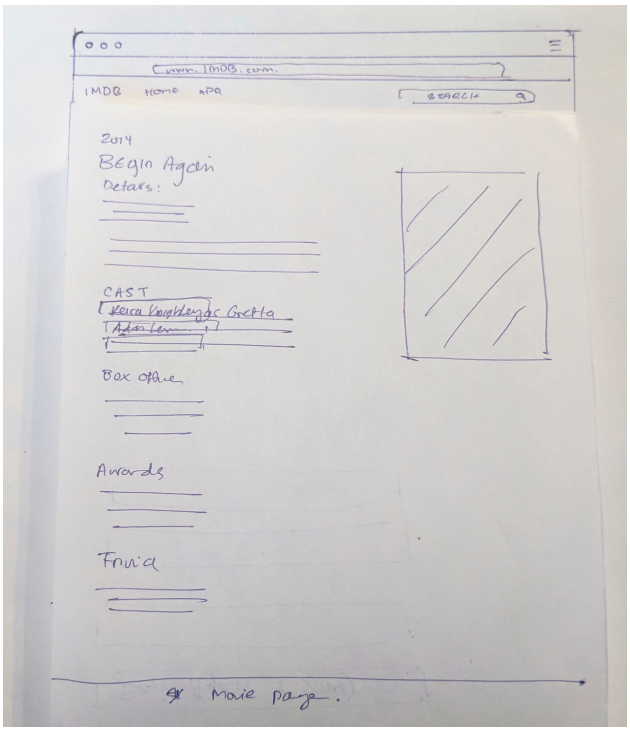
8. Users can look up the movie's primary language
    a. Add language attribute to movie table
9. Users can view if a movie is a book/play adaptation
    a. Add fields to movie table that are booleans for whether a movie has a book or play companion/is an adaptation
10. Look up an actor's most common co-stars/co-cast, ie. does an actor seem to work with a set of cast members/director/producer most frequently?

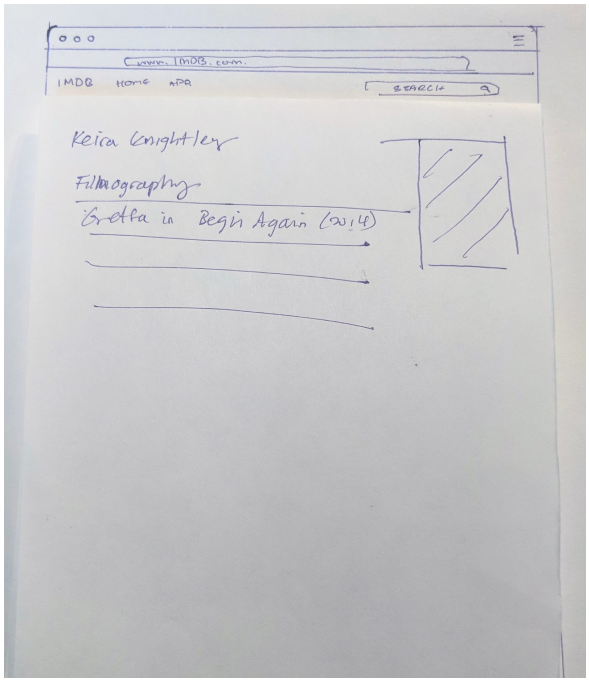
Lessons Learned Through Iteration

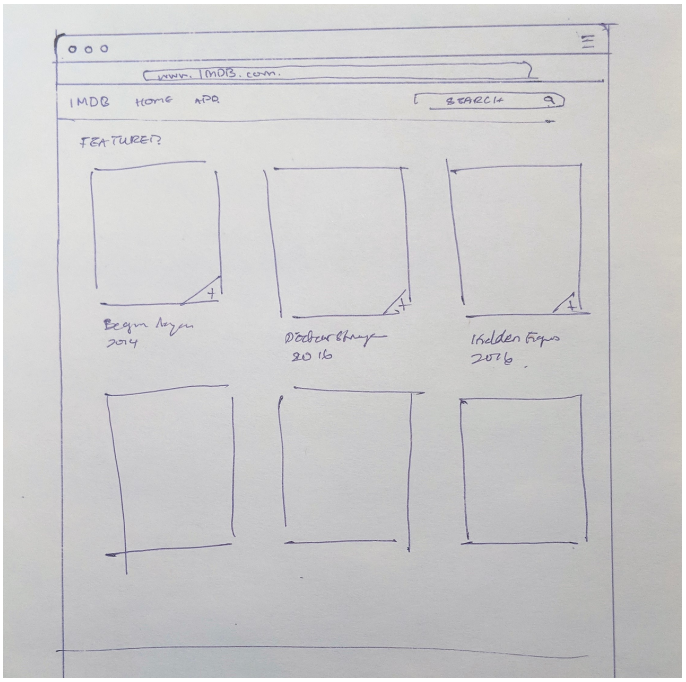
1. **Paper Prototype:** I drew sketches of my IMDB redesign and created a V1 paper prototype of how my site flow would work based on a cognitive walkthrough of my sketches. Then I asked one of my suite members to test the paper prototype. I defined a set of smaller user goals and asked her to “think out loud” while performing the tasks. I also took notes during her tasks to observe whether there were tasks that took longer than usual/were more difficult to accomplish.

   After she finished testing, I jotted down her feedback and realized there were deficiencies in my original design such as: overwhelming homepage that had too much information/text, movie pages should separate actors from directors/producers/writers, etc. I took her feedback into account when I revised my sketches and prototype.









**I had initial sketches and random other pieces of paper for modals, but I forgot to take photos of them before I left for Grace Hopper Conference :(

2. **Refactoring Database to Support Movie Sequels/Prequels:** My plan was to allow the database to support movie sequels and prequels by creating a `sequel` and `prequel` attribute for each movie in the `Movie` model. However, I realized that querying this would be difficult and it was difficult to view all the movies in a

particular franchise/series (eg. a user might want to look up all of The Lord of the Rings movies). I looked up a few different possible implementations, but wanted to flesh out a definitive set of tasks the user would need to accomplish eg. view movie franchises, click on sequel from the current movie, etc.)

I ended up deferring series of movies because I wanted to ask for feedback on implementation methods and behavior a user would expect before investing a lot of time on refactoring my database models to support prequels/sequels. I want to consider what the flow of a user looking for prequels/sequels is, and what pages they would expect to find what information. Then I can design my implementation based on real user feedback.

Some search results I found while looking for implementations:

- https://stackoverflow.com/questions/22634311/movie-relation-database-design
- https://stackoverflow.com/questions/4911007/storing-relationship-in-mysql-database

3. **Finding Most Common Co-Stars:** I initially thought it would be cool to see who are the most common co-stars of an actor, because it would allow you to see if an actor is in the same movie with the same few directors/producers/co-stars. However, I wasn't as certain over how to define "most common". My intuition was that I'd find all the movies of an actor and then take a frequency count of all the actors/directors/etc. that appear in all those movies, but I wasn't sure how to handle defining a threshold for "most common" and questions such as: what if an actor only starred in one movie, then is everyone in that movie a "most common co-star"? I also ran into the problem of whether finding most common co-stars is too specific a task/could be generalized to a better/broader feature.

   I solved the issue by removing the subgoal, because it seems to be difficult to define "most common" and perhaps I could reason that this task is actually part of a larger user need (such as comparing actors/movies). While I was trying out different queries, it seemed that "most common" was subjective depending on what I wanted to find in the data. I think a possible future feature could be a small tool in the app that lets you explore actor/movie, movie/movie, and actor/actor relationships.

Goal Progress

For each of the goals in part 1. Which items you completed? Show images to document each item (either of the UI or code).

Status Summary

✓ = DONE
➡ SOON = PARTIAL/DEFERRED
✖ = NOT DONE

High Level Goal	**Status**
Allow users to accomplish tasks of browsing through movies on the home page, searching for movies/actors/directors/producers, viewing detailed information about a movie, and finding sequels of movies	✓
Low Level Goals	**Status**
1. Allow users to look up actor filmography in chronological order a. On an actor's page, sort their filmography by release date of a movie they starred in using reverse chronological order	✓
2. Users can browse movie posters, maps to the real world task (maybe outdated) of browsing shelves in a physical store looking for movies to watch a. Add a new field to the movie table to identify poster URL	✓
3. Users can look up directors, producers of a movie a. Create role entries for directors and producers in the `stars` table	✓
4. Look up awards a movie has won a. Create an Awards table with fields: Id, Event (eg. Academy), Award (eg. Oscar), Year (eg. Best Actor), Rank (eg. Win) b. Create two association tables, one for awards that movies win, and one for awards that actors/directors/producers/cast members win	✓
5. Find series of movies (eg. Three Flavours Cornetto trilogy, Spanish Apartment Trilogy, all the Hunger Games movies) and be able to identify the sequel/prequel from each movie's page	➡ SOON

6. Allow users to view movie budget and box office earnings a. Add budget and box office earnings (gross and opening) attributes to movie table	✓
7. User can use a search bar and filters to find films with certain actors, genres, etc. a. Add a search bar	✓
8. Users can look up the movie's primary language a. Add language attribute to movie table	✓
9. Users can view if a movie is a book/play adaptation a. Add fields to movie table that are booleans for whether a movie has a book or play companion/is an adaptation	✓
10. Look up an actor's most common co-stars/co-cast, ie. does an actor seem to work with a set of cast members/director/producer most frequently?	✖

High Level Goal

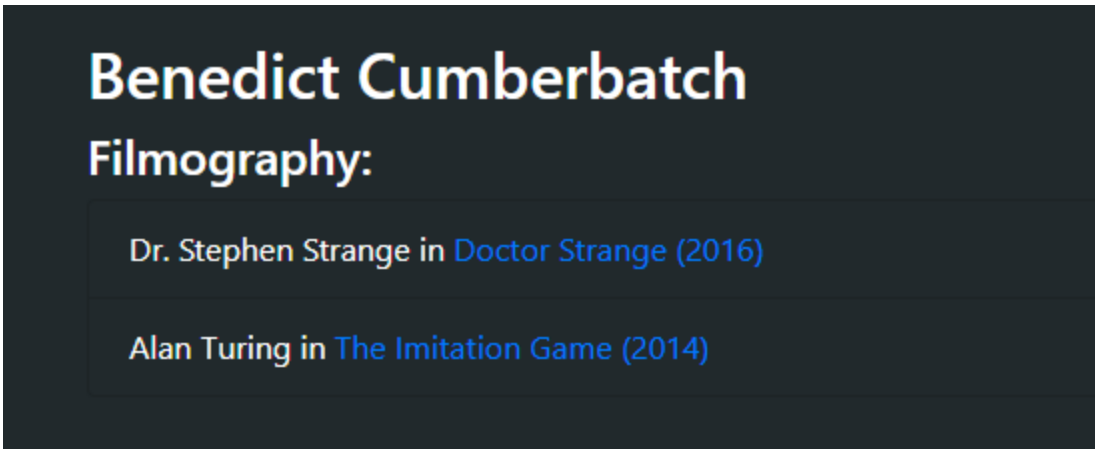
1. Allow users to accomplish tasks of browsing through movies on the home page, searching for movies/actors/directors/producers, viewing detailed information about a movie, and finding sequels of movies

Low Level Goals

I documented each of the low level goals that were completed.

1. Allow users to look up actor filmography in chronological order — DONE
   a. On an actor's page, sort their filmography by release date of a movie they starred in using reverse chronological order

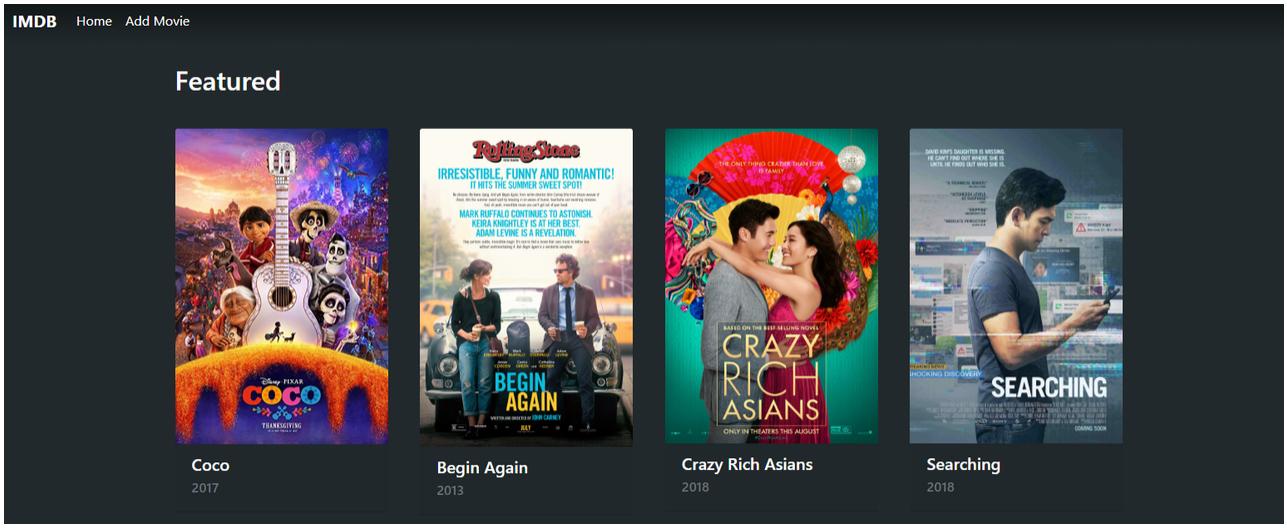
   I was able to figure out the SQL clause in SQLAlchemy necessary to order the movies by reverse chronological order of their release year.



Doctor Strange was a more recent movie, so it appears first

2. Users can browse movie posters, maps to the real world task (maybe outdated) of browsing shelves in a physical store looking for movies to watch — DONE
   a. Add a new field to the movie table to identify poster URL

   I used poster URLs from IMDB movie posters and revamped my main page to feature movie posters along with some basic movie information such as title. Users can click on a movie poster to go to the detailed movie page.



Revamped IMDB landing page feature movie posters

3. Users can look up directors, producers of a movie — DONE
   a. Create separate/special roles for directors and producers

   I realized this task didn't require any new tables. I re-used my table that maps talent (actors, cast, etc.) to roles in movies, and "director" or "producer" is a

possible role in this table. I also had to change the primary key to be a tuple of (movieId, talentId, role) to allow a single person to play multiple roles in a movie.

```python
from app import db

# Association Tables
stars = db.Table(
    'stars',
    db.Column('movieId', db.Integer, db.ForeignKey('movie.id', ondelete="CASCADE"), primary_key=True),
    db.Column('talentId', db.Integer, db.ForeignKey('talent.tid', ondelete="CASCADE"), primary_key=True),
    db.Column('role', db.String(140), primary_key=True)
    )

```

Updated primary key for stars table (maps movies to talent with roles)

4. Look up awards a movie has won
    a. Create an Awards table with fields: Id, Event (eg. Academy), Award (eg. Oscar), Year (eg. Best Actor), Rank (eg. Win)

    I created tables for awards, then association tables for movies that win awards and talent that wins awards.

```python
movieWinsAward = db.Table(
    'movieWinsAward',
    db.Column('movieId', db.Integer, db.ForeignKey('movie.id', ondelete="CASCADE"), primary_key=True),
    db.Column('awardId', db.Integer, db.ForeignKey('award.id', ondelete="CASCADE"), primary_key=True)
    )

talentWinsAward = db.Table(
    'talentWinsAward',
    db.Column('talentId', db.Integer, db.ForeignKey('talent.tid', ondelete="CASCADE"), primary_key=True),
    db.Column('awardId', db.Integer, db.ForeignKey('award.id', ondelete="CASCADE"), primary_key=True)
    )
```

```python
class Award(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    event = db.Column(db.Text, nullable=False)
    award = db.Column(db.Text, nullable=False)
    year = db.Column(db.Integer, nullable=False)
    rank = db.Column(db.Text)

    def __repr__(self):
        return "<Award {} {} {}".format(self.event, self.award, self.year)
```

5. Find series of movies (eg. Three Flavours Cornetto trilogy, Spanish Apartment Trilogy, all the Hunger Games movies) and be able to identify the sequel/prequel from each movie's page — DEFERRED

I ended up deferring series of movies because I wanted to ask for feedback on implementation methods and behavior a user would expect before investing a lot of time on refactoring my database models to support prequels/sequels. I want to consider what the flow of a user looking for prequels/sequels is, and what pages they would expect to find what information. Then I can design my implementation based on real user feedback.

6. Allow users to view movie budget and box office earnings — DONE
   a. Add budget and box office earnings attributes to movie table

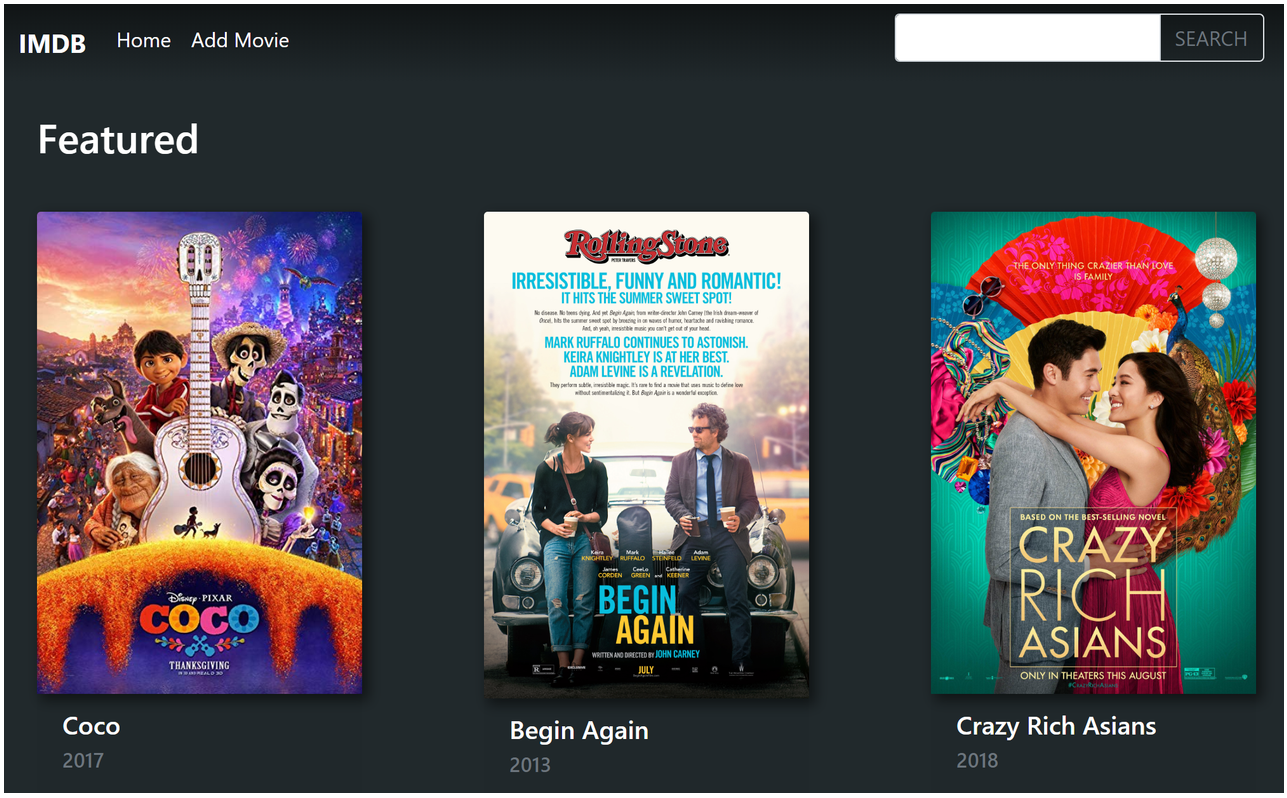
   I created additional attributes in the movie database to account for budget, box office opening weekend earnings, and box office gross domestic (USA).

Box office information displayed in its own section

7. User can use a search bar and filters to find films with certain actors, genres, etc. — DONE

a. Add a search bar

I simplified the types of queries to only search film names for now. I also want to add additional filters on the search results page.



Search bar in nav

8. Users can look up the movie's primary language — DONE
   a. Add language attribute to movie table

I added a new field, and also displayed the movie's primary language on its detailed page (see screenshot of movie page for low level goal #6).

```python
# Models
class Movie(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(140), nullable=False)
    year = db.Column(db.Integer, nullable=False)
    runtime = db.Column(db.Integer)
    overview = db.Column(db.Text)
    posterURL = db.Column(db.Text)
    releaseDate = db.Column(db.DateTime)
    countryOfOrigin = db.Column(db.Text)
    language = db.Column(db.String(70), default="English")
    budget = db.Column(db.Numeric)
    boxOfficeGross = db.Column(db.Numeric)
    boxOfficeOpeningWeekend = db.Column(db.Numeric)
    isPlay = db.Column(db.Boolean, default=False)
    isNovel = db.Column(db.Boolean, default=False)

    # Relationships
    talent = db.relationship('Talent', secondary=stars, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')
    genre = db.relationship('Genre', secondary=movieIsGenre, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')
    award = db.relationship('Award', secondary=movieWinsAward, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')

    def __repr__(self):
        return '<Movie {}, {}>'.format(self.title, self.year)
```

Added language field to Movie model

9. Users can view if a movie is a book/play adaptation — DONE
    a. Add fields to movie table that are booleans for whether a movie has a book or play companion/is an adaptation

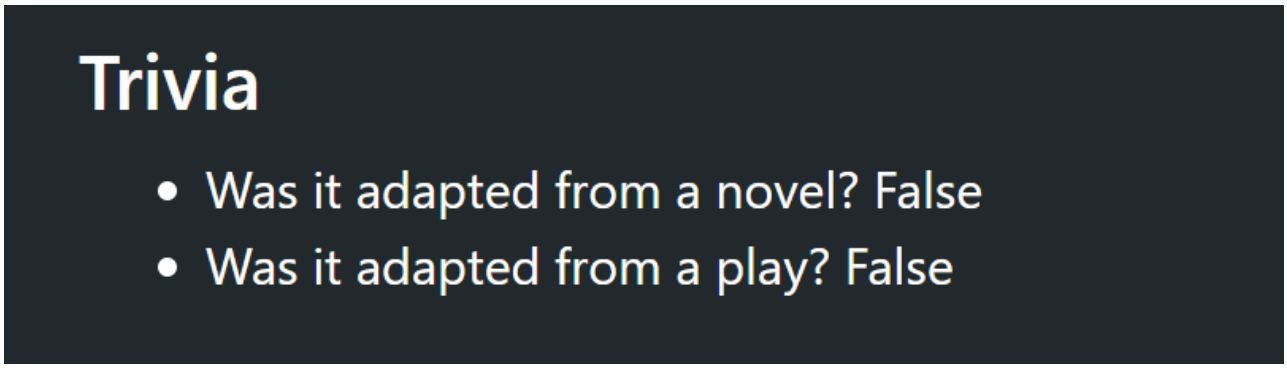
I added two boolean fields that designate if a movie was a play or book adaptation.

```python
# Models
class Movie(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(140), nullable=False)
    year = db.Column(db.Integer, nullable=False)
    runtime = db.Column(db.Integer)
    overview = db.Column(db.Text)
    posterURL = db.Column(db.Text)
    releaseDate = db.Column(db.DateTime)
    countryOfOrigin = db.Column(db.Text)
    language = db.Column(db.String(70), default="English")
    budget = db.Column(db.Numeric)
    boxOfficeGross = db.Column(db.Numeric)
    boxOfficeOpeningWeekend = db.Column(db.Numeric)
    isPlay = db.Column(db.Boolean, default=False)
    isNovel = db.Column(db.Boolean, default=False)

    # Relationships
    talent = db.relationship('Talent', secondary=stars, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')
    genre = db.relationship('Genre', secondary=movieIsGenre, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')
    award = db.relationship('Award', secondary=movieWinsAward, backref=db.backref('movie', cascade='all', lazy=True), lazy='subquery')

    def __repr__(self):
        return '<Movie {}, {}>'.format(self.title, self.year)
```

Added boolean fields to Movie model



Section shows if movie was adapted from novel/play