Web Studio: Week 5 Turn-in #4

Name: Ivy Chen
UNI: ic2389
Due: Friday 10/5 at 2PM
COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. Users can find movie series with their prequels/sequels/remakes/spinoffs, and for each character in the series, view all the actors who have portrayed that character.

**After critique session with Prof. Chilton & peers, I tweaked my high-level goal to be more specific.

Low Level Goal

1. Refactor movie database to build support for movie series.
2. Add at least 2 movie series.
3. Update homepage to feature movie series.
4. Write SQL query to compute total runtime for all movies in series.
5. Write SQL query to query all actors who've played the same role in a movie series.
6. Create movie series page template to display all movies in series.
7. Write SQL query to find most common actors that have played in a series (actors that have played in the most movies in a series).
8. Enhance search to search by movie title or movie series.

User Persona

Name	Jane
Bio	Jane is a film enthusiast and loves movie series because she enjoys getting wrapped up in an entire movie universe. She

	enjoys the sense of being drawn into a movie's world by watching movies in the same series, and track character portrayal over time. She frequently hosts movie marathons and likes to find exciting film series to screen.
Goal	Jane has several specific goals she wants to accomplish: • Find a movie series by searching for a genre • Find out how long watching an entire movie series will take • Find the actors who've played a character in a series

Lessons Learned Through Iteration

1. One of the first things I did this week was to define a very specific user need based on the feedback/critique session. My original plan was to implement a way for users to view movie series (ie. prequels/sequels). I still wanted to keep it related to the task: "a user goes on the website to find movie series." But I ran into a problem because my task was too general.

   After thinking about it more and brainstorming scenarios of *why* users would want to view movie series, I came up with a couple of ideas: users want to marathon a movie series and need to find information on the genre and total runtime of a series, users want to find all the actors who've played a certain character in the series (eg. all the actors who've played Batman). I decided that the second reason was a case where I could see users being curious about all the actors who've played X in a film series, so I changed my high level goal to fit this user need.

2. My plan as a subgoal was to enhance the search bar to query by movie title, year, genre, etc, but I realized enhancing the search bar wasn't helping towards my high level goal of displaying movie series and allowing users to view all the actors who played a single character. The search bar currently only searches based on movie title (implemented last week) and my plan was to augment it this week, but since I revised my high-level and low-level goals around a specific user need, I realized it wasn't a top priority.

   Thus, I solved my issue by removing the search bar enhancement as a subgoal, which allowed me to concretely decide on low-level goals that make progress

towards my high-level goal. If I do have time at the end, I think additional search bar enhancements (ie. fuzzy searching) would be a cool feature to add.

3. One of my low-level goals was to write a query that returns the most common actors in a series ie. actors that have appeared in the most movies in a series. However, I don't think this low-level goal directly contributed to my high-level goal. My high-level goal is focused more on the actors for each *character*, but this sub-goal was focused on the actors themselves.

   Thus, I removed this subgoal for this week and focused more on incremental changes that directly contributed to helping the user hone in on the characters in a series, and the actors who played those characters.

4. Testing. I tested the latest version of my application with my roommate and asked her to complete the following task: find actors who play "Marlin" in the "Finding Nemo" series.

   Observations:
   - The task was straightforward since I had little sample data in my database.
   - She easily found the "Characters" heading on the movie series page.
   - Difficult to figure out what is a clickable link and what is not clickable.
   - Wish there was a way to click on the series from the movie's detailed page.

   What I learned is that there are a lot of details to consider when adding a new page/structure. For me, I added movie series, but need to better link them/make it discoverable through existing pages.

Goal Progress

Status Summary

✓ = DONE
➡ SOON = PARTIAL/DEFERRED
✖ = NOT DONE (REMOVED)

High Level Goal	Status
Users can find movie series with their prequels/sequels/remakes/spinoffs, and for each character in the	✓

series, view all the actors who have portrayed that character.	
Low Level Goals	**Status**
1. Refactor movie database to build support for movie series.	✓
2. Add at least 1 movie series.	✓
3. Update homepage to feature movie series.	✓
4. Write SQL query to compute total runtime for all movies in series and count of movies in series.	✓
5. Write SQL query to query all actors who've played the same role in a movie series.	✓
6. Create movie series page template to display all movies in series.	✓
7. Write SQL query to find most common actors that have played in a series (actors that have played in the most movies in a series).	✗
8. Enhance search to search by movie title or movie series.	✗

High Level Goal

1. Users can find movie series with their prequels/sequels/remakes/spinoffs, and for each character in the series, view all the actors who have portrayed that character.

   I was able to complete this goal. I can measure success of this high level goal by having a user test my application and see if they are able to find a movie series (by browsing home page) and seeing all actors who've portrayed that character in the series.

Low Level Goal

I documented progress on each of the low-level goals I completed below.

1. Refactor movie database to build support for movie series.

a. I refactored the movie database by adding two new tables: `series` and `movieInSeries` to represent that movies belong to larger series.

```
moviesInSeries = db.Table(
    'moviesInSeries',
    db.Column('seriesId', db.Integer, db.ForeignKey('series.seriesId', ondelete="CASCADE"), primary_key=True),
    db.Column('movieId', db.Integer, db.ForeignKey('movie.id', ondelete="CASCADE"), primary_key=True),
    db.Column('sequence', db.Integer, primary_key=True)
)
```

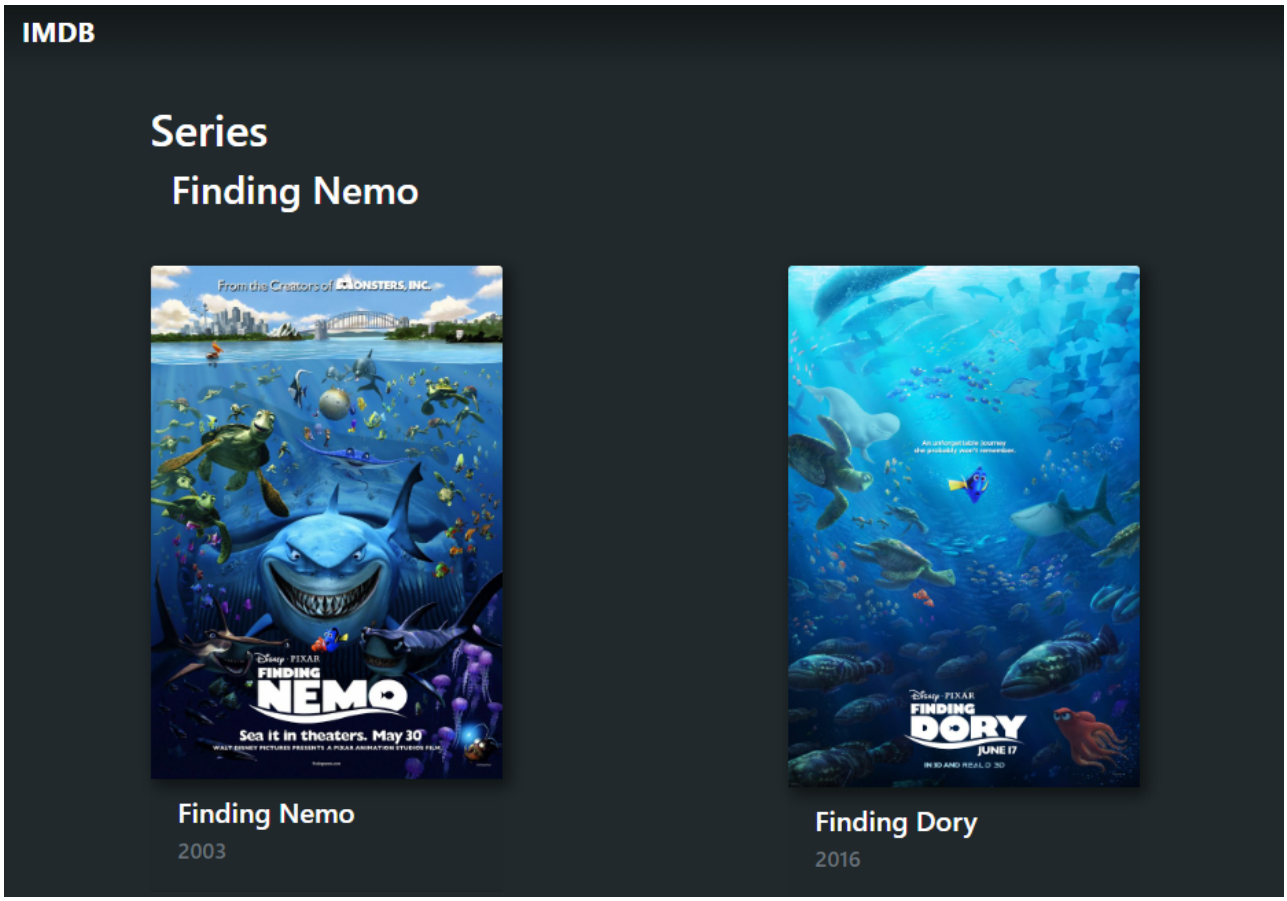
models.py

```
class Series(db.Model):
    seriesId = db.Column(db.Integer, primary_key=True)
    seriesName = db.Column(db.Text)

    # Relationships
    movie = db.relationship('Movie', secondary=moviesInSeries, backref=db.backref('series', cascade='all', lazy=True), lazy='subquery')
```

models.py

2. Add at least 1 movie series.
    a. I added the `Finding Nemo` series. If a series doesn't have a general name, then I named the series by the first movie's name.



Finding Nemo series on home page

3. Update homepage to feature movie series.

a. I updated the home page to feature movie series that contain at least 1 movie in the database. (See previous screenshot of Finding Nemo series).

4. Write SQL query to compute total runtime for all movies in series and count of movies in series.
   a. I realized that I didn't need SQL queries to compute these values. I could compute these values in Python and pass them to the template for each movie series page.

```python
@app.route('/series/<id>', methods=['GET'])
def series(id):
    series = models.Series.query.get(id)
    seriesMovies = db.session.query(models.Movie)\
        .join(models.movieInSeries)\
        .filter_by(seriesId=id, movieId=models.Movie.id)\
        .order_by(models.Movie.year).all()

    totalrt = sum([m.runtime for m in seriesMovies])

    return render_template('series.html', series=series, seriesMovies=seriesMovies, datetime=datetime, totalrt=totalrt, len=len)
```

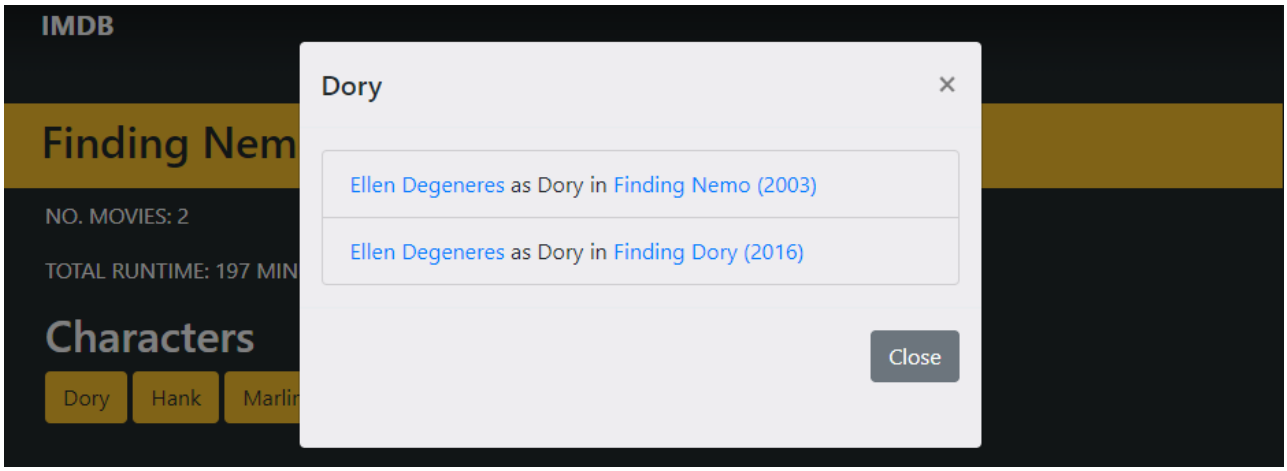
app.py

```
{% extends "base.html" %}
{% block title %} IMDB {% endblock %}

{% block body %}
  {% block content %}
  <h2>{{ series.seriesName }}</h2>
  <p>NO. MOVIES: {{ len(seriesMovies) }}</p>
  <p>TOTAL RUNTIME: {{ totalrt }} MINUTES</p>
  <h2>Films</h2>
    {% for movie_data in seriesMovies %}
    <img src={{ movie_data.posterURL }} class="movie-poster rounded float-right" alt="...">
    <div class="movie">
      <h6 class="mb-2 text-muted">{{ movie_data.year }}
      <h3><a href="/movie/{{movie_data.id}}">{{ movie_data.title }}</a></h3>
      <p>RUNTIME: {{ movie_data.runtime }} MINUTES</p>
      <p>RELEASE DATE: {{ datetime.strftime(movie_data.releaseDate, "%B %d, %Y") }}</p>
      <p>COUNTRY OF ORIGIN: {{ movie_data.countryOfOrigin }}</p>
      <p>LANGUAGE: {{ movie_data.language }}</p>
      <p>{{ movie_data.overview }}</p>
    </div>
    {% endfor %}
  {% endblock %}
{% endblock %}
```

series.html

5. Write SQL query to query all actors who've played the same role in a movie series.
   a. In order to return the actors who've played a certain character in a movie series, I ended up first querying by the distinct characters, then actors.



Actors who've played this character

6. Create movie series page template to display all movies in series.
    a. I created a movie series page that display a bit of information about each movie in the series, and has links to the full movie description page for each movie.

```
@app.route('/series/<id>', methods=['GET'])
def series(id):
    series = models.Series.query.get(id)
    seriesMovies = db.session.query(models.Movie)\
        .join(models.movieInSeries)\
        .filter_by(seriesId=id, movieId=models.Movie.id)\
        .order_by(models.Movie.year).all()

    totalrt = sum([m.runtime for m in seriesMovies])

    return render_template('series.html', series=series, seriesMovies=seriesMovies, datetime=datetime, totalrt=totalrt, len=len)
```

app.py

IMDB

Finding Nemo

NO. MOVIES: 2

TOTAL RUNTIME: 197 MINUTES

Films

2003

Finding Nemo

RUNTIME: 100 MINUTES

RELEASE DATE: May 30, 2003

COUNTRY OF ORIGIN: USA

LANGUAGE: English

A clown fish named Marlin lives in the Great Barrier Reef and loses his son, Nemo, after he ventures into the open sea, despite his father's constant warnings about many of the ocean's dangers. Nemo is abducted by a boat and netted up and sent to a dentist's office in Sydney. While Marlin ventures off to try to retrieve Nemo, Marlin meets a fish named Dory, a blue tang suffering from short-term memory loss. The companions travel a great distance, encountering various dangerous sea creatures such as sharks, anglerfish and jellyfish, in order to rescue Nemo from the dentist's office, which is situated by Sydney Harbour. While the two are searching the ocean far and wide, Nemo and the other sea animals in the dentist's fish tank plot a way to return to the sea to live their lives free again.

Finding Nemo series page