

Web Studio: Week 6 Turn-in #5

Name: Ivy Chen

UNI: ic2389

Due: Friday 10/12 at 2PM

COMS6998 Adv. Web Studio

Original Goals

High Level Goal

1. Find/search for a movie series by name of series/name of movie/genre. For each character in a series, find all the actors who've played that role.

Low Level Goal

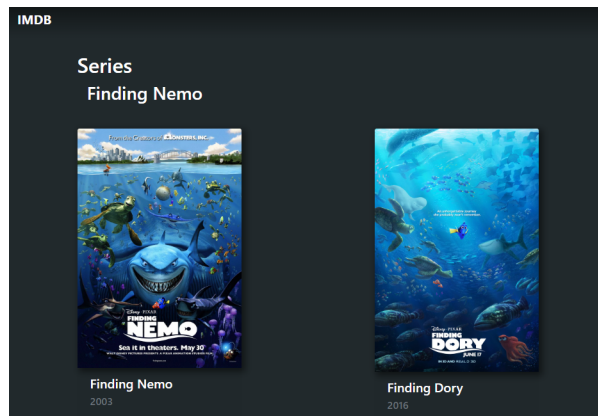
1. Remove "Home" button and redesign navigation bar to include sign in and user info.
2. Simplify home page and add instructional text that guides my user to accomplish their goal.
3. Add text on Series page that differentiates it from movie pages.
4. Create links that flow from Series -> Movies and Movies -> Series so a user can identify movies in a series and series a movie belongs to.
5. Enhance multi-search bar to search for movie series/name of movie/genre.
6. Simplify layout on Series page to highlight the characters in a movie.
7. One feature to add for user account: Users can mark which movies in a series they have already watched.

Lessons Learned Through Iteration

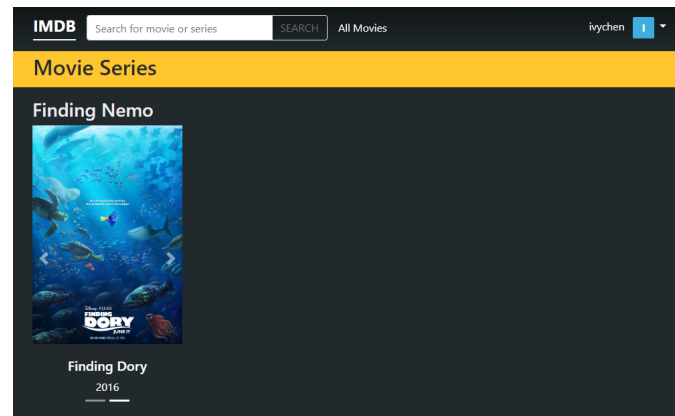
1. My plan was to simplify the site to focus on finding/searching for series and providing overview information on movie series. Based on feedback from studio last week, my series pages contained movie information that I already displayed on individual movie pages, so it was redundant and probably overwhelming. In addition, the home page was cluttered because I displayed both movie series and all movies. I ran into the problem of figuring out how to better organize my

content in terms of information hierarchy and making sure that there was neither too much nor too little content on the page.

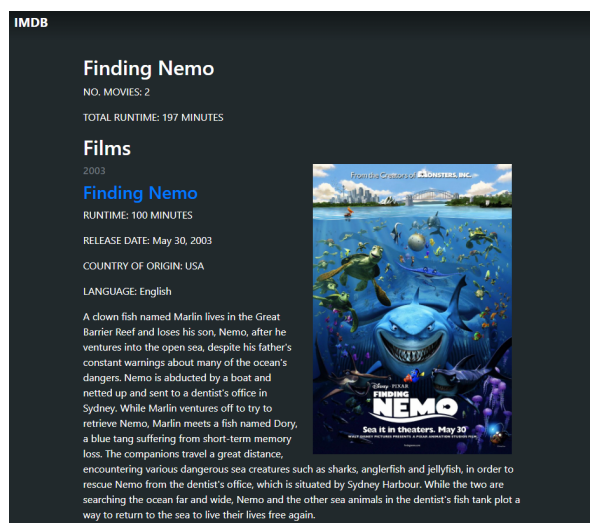
I solved the problem by reorganizing content on my site. I took inspiration from studio session last week. For example, one person created a page that displayed all movies in their database, so I moved “All Movies” from my home page to a separate page that lists all movies by alphabetical order. Thus, the homepage would only feature series. On the home page, I used carousels to group movies together into series instead of showing individual movies within a series subheading — this gives me more space to display movie series. In addition, on each movie series page, I removed detailed information about each film in a series and instead provided links to the individual movie pages.



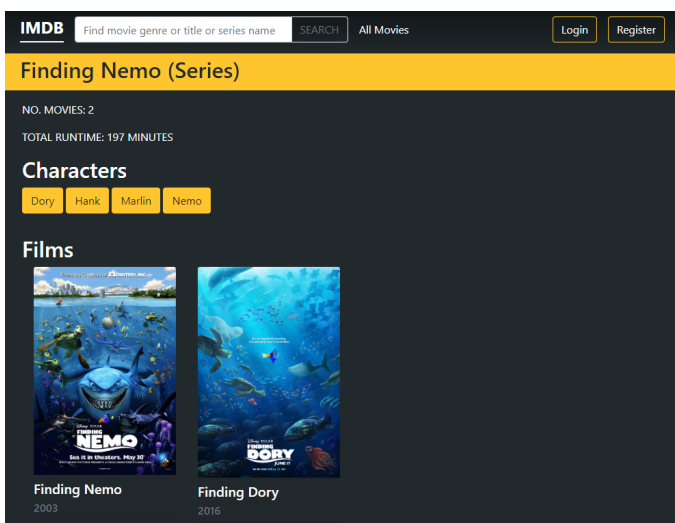
Before: home page



After: home page



Before: series page



After: series page

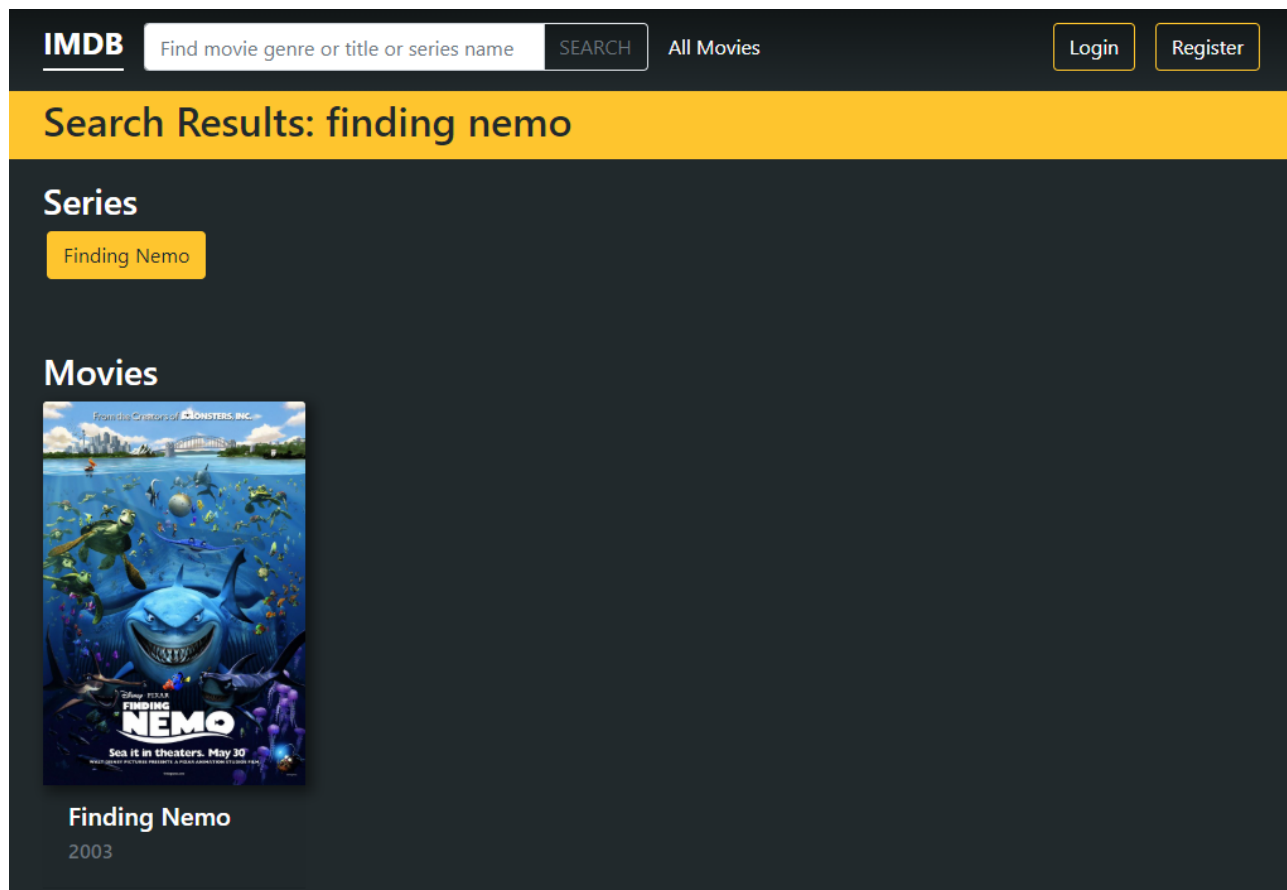
2. My plan was to create a better search experience by introducing multi-search for movies and series based on movie/series name and genre of movies. Initially, I imagined the semantics of the search would be:

- User enters search query `q`
- `q` can match a movie title, genre, or series name
- `q` returns *all movies* such that title = `q`, genre = `q` or series name = `q`

But I ran into the problem of displaying information redundantly, and realizing that my search didn't improve movie series discovery because I was returning *movies* and not *series*. Thus, I solved my problem by changing the search semantics to be:

- User enters search query `q`
- `q` can match a movie title, genre, or series name
- `q` return all movies such that title = `q`, genre = `q` **and** all series (by name) such that series name = `q`

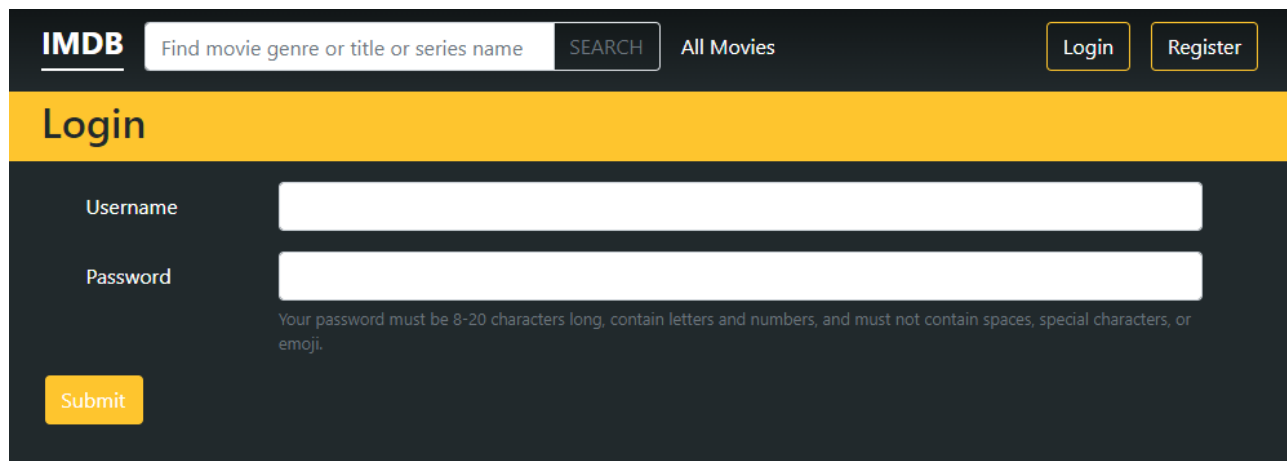
The search results page displays two sections: Series and Movies. The results for series are links to the matching series page, and the results for movies included all movies whose title or genre match the query. In movie results, I also considered adding all movies that belong to a series with matching series name, but the information seemed redundant. I reasoned that a user who wants to see all movies belonging to a series can search by series name, then navigate to the series page.



User-uploaded image: image.png

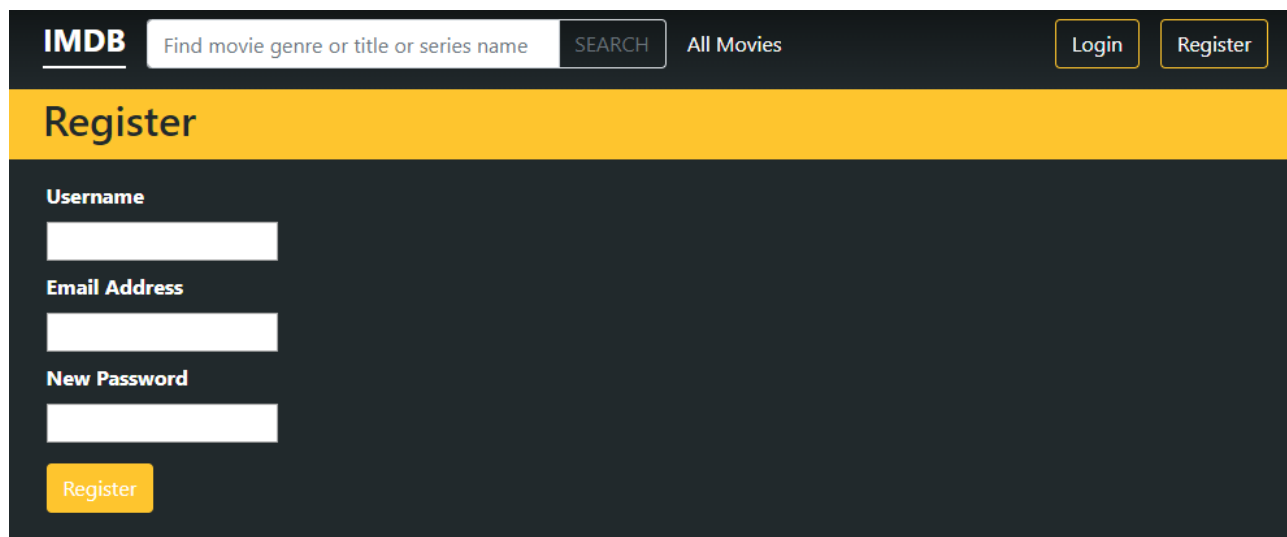
3. My plan was to add login and registration, along with a feature for password recovery. I initially created forms for registration that included email address to allow for username/password recovery, but I wasn't sure how to send emails/manage password recovery in Flask. I don't think it's possible while running my app locally. I think I need to host my application, set up a service that sends emails from the app, etc.

I solved my problem by opting not to add the password recovery email. I created the login and registration forms. While signing up, users still need to register with their email address while signing up.



The image shows the IMDb login interface. At the top, there is a dark header with the IMDb logo, a search bar with the placeholder text "Find movie genre or title or series name", a "SEARCH" button, and links for "All Movies", "Login", and "Register". Below the header is a yellow banner with the word "Login". The main form area has a dark background. It contains labels for "Username" and "Password" next to white input fields. Below the password field is a note: "Your password must be 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji." At the bottom left of the form is a yellow "Submit" button.

Login form



The image shows the IMDb registration interface. It has the same dark header as the login page. Below the header is a yellow banner with the word "Register". The main form area has a dark background. It contains labels for "Username", "Email Address", and "New Password" next to white input fields. At the bottom left of the form is a yellow "Register" button.

Registration form

Goal Progress

Status Summary

✓ = DONE

→
SOON = PARTIAL/DEFERRED

✗ = NOT DONE (REMOVED)

High Level Goal	Status
1. Find/search for a movie series by name of series/name of movie/genre. For each character in a series, find all the actors who've played that role.	✓

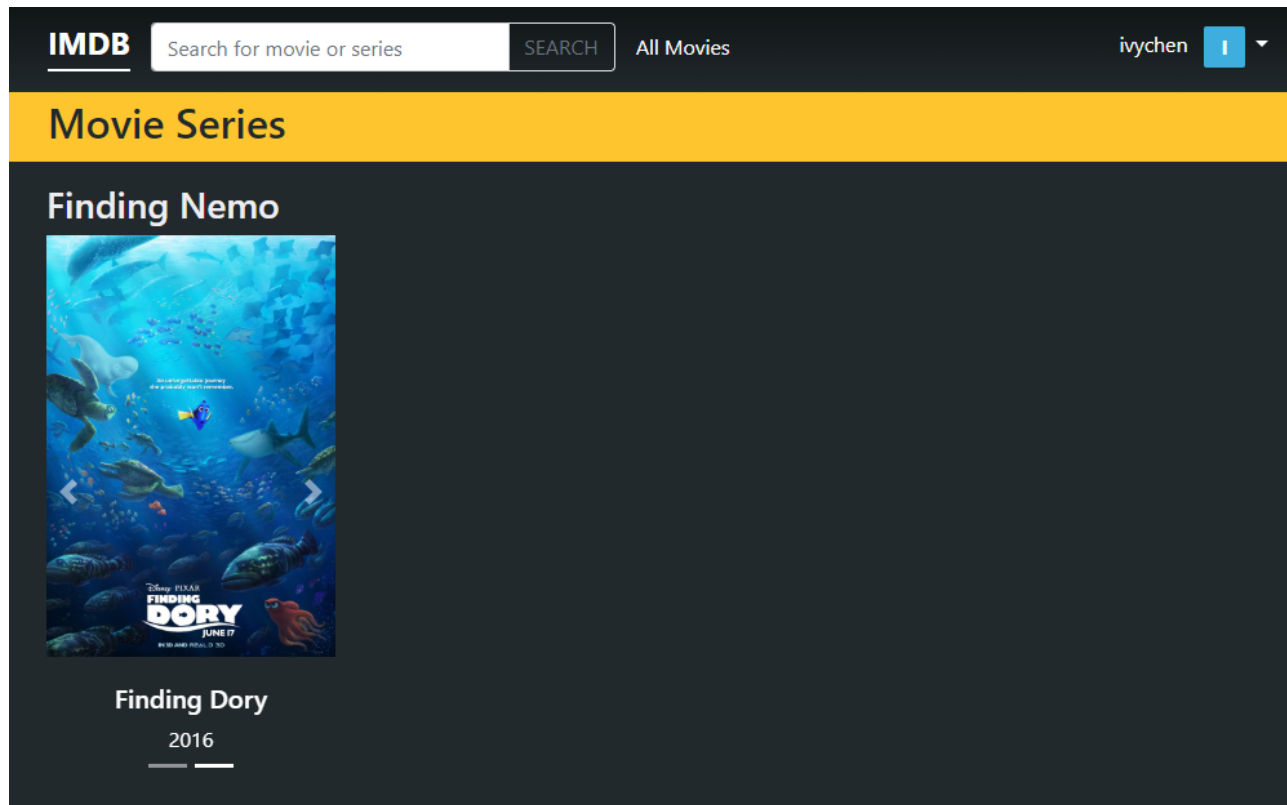
Low Level Goals	Status
1. Remove "Home" button and redesign navigation bar to include sign in and user info.	✓
2. Simplify home page and add instructional text that guides my user to accomplish their goal.	✓
3. Add text on Series page that differentiates it from movie pages.	✓
4. Create links that flow from Series -> Movies and Movies -> Series so a user can identify movies in a series and series a movie belongs to.	✓
5. Enhance multi-search bar to search for movie series/name of movie/genre.	✓
6. Simplify layout on Series page to highlight the characters in a movie.	✓
7. One feature to add for user account: Users can mark which movies in a series they have already watched.	✓

High Level Goal

1. Find/search for a movie series by name of series/name of movie/genre. For each character in a series, find all the actors who've played that role. This week, I want to narrow down the scope of the app to focus on series specifically.

Low Level Goal

1. Remove "Home" button and redesign navigation bar to include sign in and user info.
 - a. I removed the "Home" button because it was redundant with the brand icon. I also moved around elements on the nav bar to include login/register/logout.
 - b. Also, in order to enable login/registration and user accounts, I created login and registration pages, as well as a user table in the database that records (username, password, email).

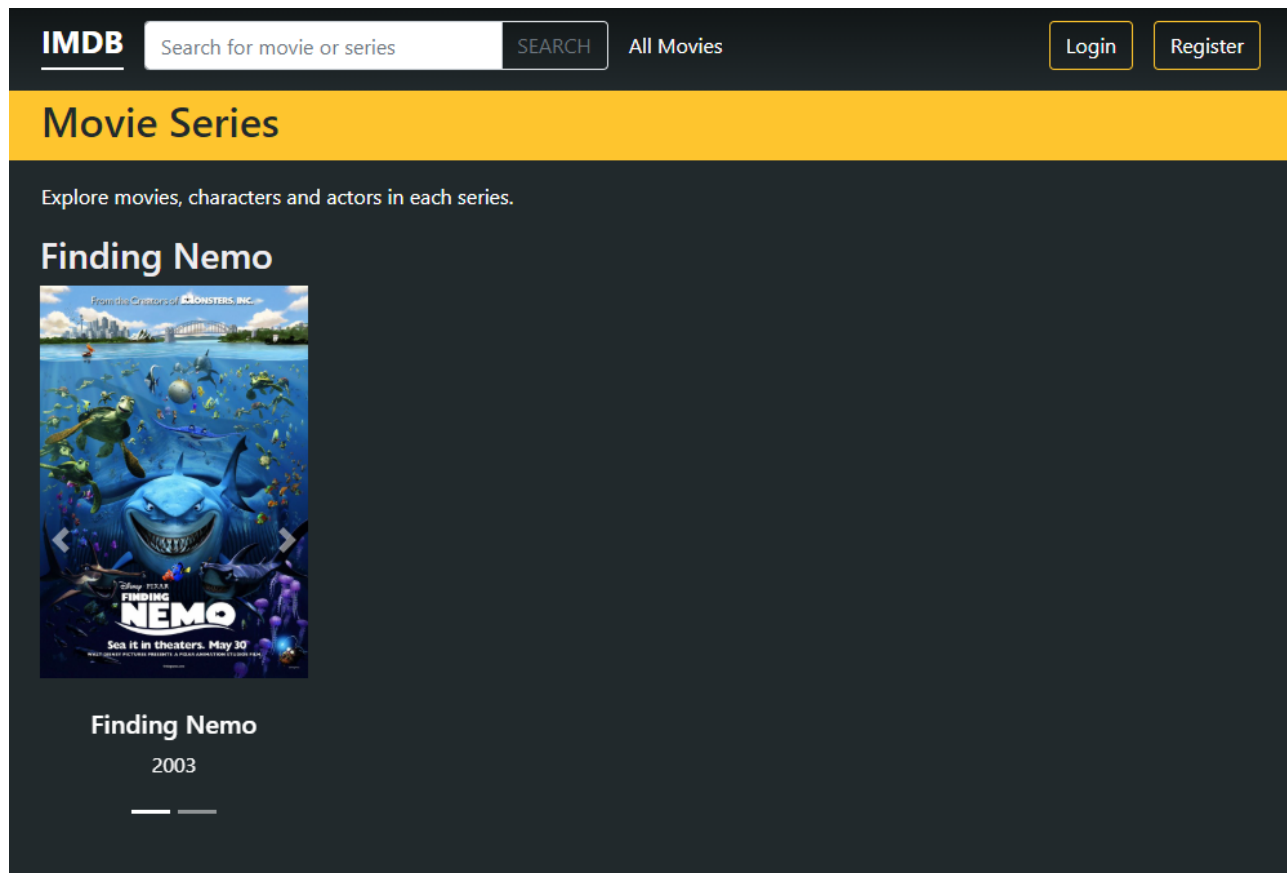


Home page with modified nav bar

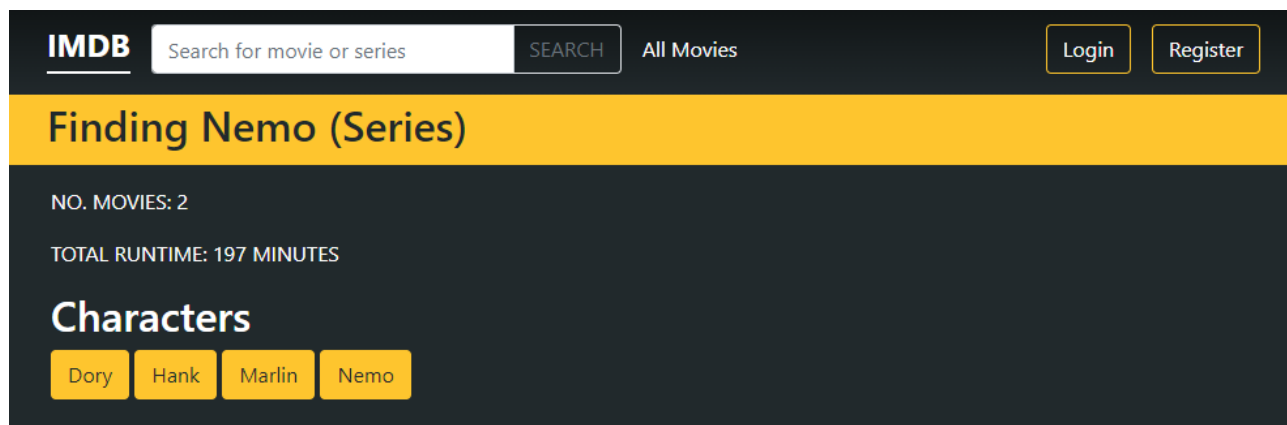


Nav bar with Login/Register buttons

2. Simplify home page and add instructional text that guides my user to accomplish their goal.
 - a. I simplified the home page, and added a short instruction text for users.



3. Add text on Series page that differentiates it from movie pages.
 - a. Added a label on the title of series pages to differentiate them from regular movie pages.



Label after series title

4. Create links that flow from Series -> Movies and Movies -> Series so a user can identify movies in a series and series a movie belongs to.
 - a. On the series page, user can click on each movie to be taken to the detailed movie page. On each individual movie page, I created a section of series that the movie is included in.

IMDB

SEARCH

All Movies

2003 | ANIMATION

Finding Nemo

RUNTIME: 100 MINUTES

RELEASE DATE: 2003-05-30 00:00:00

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.

Series

Finding Nemo

Series section on movie page

5. Enhance multi-search bar to search for movie series/name of movie/genre.
 - a. I enhanced the search feature on the site to search for movies by title or genre, and search movie series by name. The search results page contains both series and movies that match the query. I eliminated duplicates in Python before passing them to the template.


IMDBAll Movies

Search Results: finding nemo

Series

Finding Nemo

Movies



Finding Nemo
2003

Found 1 series and 1 movie with this query

6. Simplify layout on Series page to highlight the characters in a movie.
 - a. I simplified the layout on the Series page to only contain subheaders for Characters and Films (that are in the series). This makes the page less cluttered and a user can click on the movie poster/title to find more detailed information about each movie in the series.

IMDBAll Movies

Finding Nemo (Series)

NO. MOVIES: 2
TOTAL RUNTIME: 197 MINUTES

Characters


Dory

Hank


Marlin

Nemo

Films

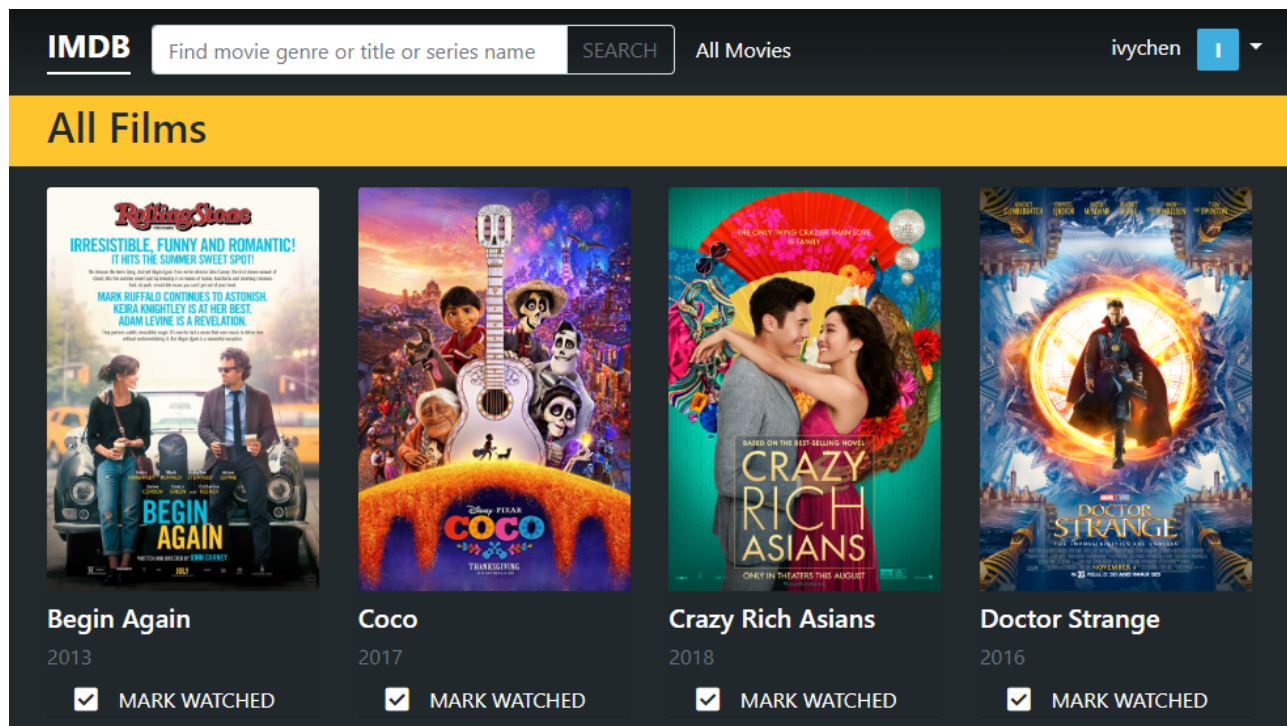


Finding Nemo
2003



Finding Dory
2016

7. One feature to add for user account: Users can mark which movies in a series they have already watched.
 - a. I first created a new database table to record watched movies for each user by storing rows of (username, movieId) pairs. A user must be signed in before they can view the movies they have watched. The “Mark Watched” only appears on movies on the “All Movies” page, and movies on a series page.



Users can mark movies as watched

```

299 @app.route('/watched', methods=['POST', 'GET'])
300 def watched():
301     if request.method == "POST":
302         movieId = request.form['id']
303         watch = db.session.query(models.watchedMovies)\
304             .filter_by(username=current_user.username, movieId=movieId).one()
305
306         if watch:
307             statement = models.watchedMovies.delete()\
308                 .where(db.and_(models.watchedMovies.c.username ==
309                     • current_user.username, models.watchedMovies.c.movieId ==
310                     • movieId))
311         else:
312             statement =
313             • models.watchedMovies.insert().values(username=current_user.username,
314             • movieId=movieId)
315
316         db.session.execute(statement)
317         db.session.commit()
318
319         return redirect(request.referrer or url_for('main'))
320
321     return redirect('/')

```

Code that executes when "Mark Watched is clicked"

