Capstone #1 Proposal

Sivadej Kitchpanich

March 17, 2020

What 2 Watch: A Bilingual Netflix Search Tool

GitHub Repo: https://github.com/sivadej/capstone-project-1

Overview

This website will provide users the ability to search the Netflix movie catalog with an emphasis on language options. Users will select two languages, then the website will display information from movie titles which offer language options that match both selections in a combination of audio and subtitles. For example, a selection of English and Spanish languages should return a list of movies that are available in English audio with Spanish subtitles as well as Spanish audio movies with English subtitles. Other common search filters (year, rating, etc.) will be a part of the search function. Users will be able to create an account to save movie lists.

User Demographics

Anyone who is technically savvy enough to use a search function of any website or app should be able to make use of this site. Emphasis will be placed on easy usability. Given that the first version of this site will be restricted to the US Netflix catalog, users will be expected to have at least a basic understanding of the English language.

Data/API

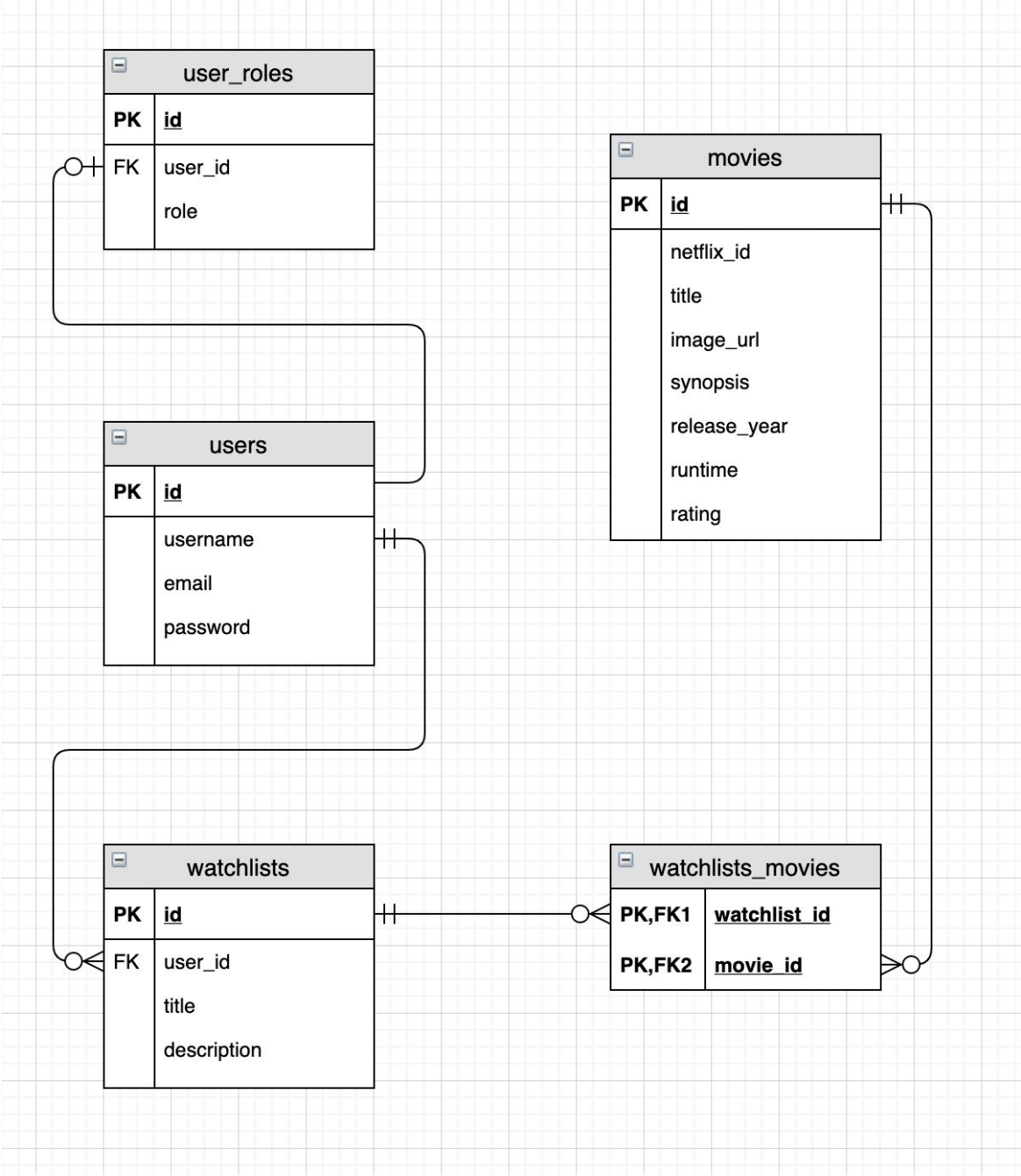
The movie data will be gathered using “uNoGS: Unofficial Netflix API”. Documentation can be found at https://rapidapi.com/unogs/api/unogs. uNoGS was selected because:

- It is one of the highest user-rated APIs for Netflix data.
- There is no official Netflix API.
- It contains the necessary movie language data.
- It is free to use (although somewhat limited)

Design Approach

Become familiar with the API. This app’s primary functionality will be dependent on successful results from the API. Making sure that the app is sending the proper search queries will be very important.

Design the database. The database will store user accounts and the user's saved watch lists. Movie data could potentially be stored in the database, but it hasn't yet been determined how the app will handle it. Alternatively, `watchlists_movies.movie_id` could be used to obtain movie data from the API instead of the movies table.



First draft of app database design.

Determine use cases and user flows:

- *Home*
    - Search form

        - Links to login, logout, user registration, account profile, admin (based on login status)
    - *Search results*
        - List of movies with title, image, summary, and link to movie details.
        - Add a movie to a watchlist (if logged in)
    - Create user account
    - Log in to account
    - Movie details
    - User account home
        - 'My watchlists'
            - Create Watchlist
            - Watchlist Detail
        - 'Edit profile'
    - Admin page
        - 'Manage watchlists'
        - 'Manage users'

Determine project requirements. Upon approval of this proposal, a Project Requirements document will be written to specify:

- Technologies to be utlized (ex: Python backend with Flask, PostgreSQL database, deploy to Heroku, Bootstrap and vanilla Javascript frontend)
- Details of all page templates, functions, routes, etc.
- Front end styling and design
- Testing details
- Project timeline

Security Concerns

- *Database:* Search requests and responses generated between the app and the API should not contain any user profile information other than language preferences. No sensitive user data should be stored in the database other than a user-generated password, which should be stored securely as a hashed string using *bcrypt*.
- *Session:* Session data should only store user's login status and most recent search query.
- *User roles:* For this version, there should an Admin role that grants a user access to all data. It is not yet determined yet how the app will handle this authorization. Each user should only have access to their own profile and Watchlist pages.

Potential API Issues

- This app will be using a free version of the API, which has a limited amount of daily requests. Reaching the limit will cause the app's search function to become unusable. Request limits can be increased but at a cost. It is assumed that the limited version will be sufficient for this project.
- Getting detailed data for a stored watchlists may require multiple API requests to obtain, which could impact performance as well as the request limit. For example, if rendering a user's list of movies in a watchlist page, and the app obtains a movie title based on a foreign key id in a watchlists table, an external request would need to be made for each individual movie title. In this case, avoiding normalizing and instead storing movie titles alongside movie ids may be acceptable, especially considering movie titles never change.

Stretch Goals

- Render website in various languages based on user account preference
- Additional options to export or share Watchlists