CINEMA-A_Z

CS251 course project

Description

In this project, we made a web application which will help the user by giving all the ratings, cast, category/genre, user reviews, platform and so on given the title of a Movie/TV show.

Features

- The platform had a user login. The user will be able to have 3 lists
  - To watch movies
  - Watched movies
  - favourites
- Features of movie to be displayed to user are ratings, plot, languages and other useful information regarding a film

WorkFlows

1.Register App

User Registration

- Fill all details on Registration page and click on signup button.
- If valid user
  - YES - add it to database and redirect to login page
  - NO - No access
- **Models**
  - used django default user model.
- **Views**
  - register view
- **URL** connected to a view `path('register/',v.register,name="register"),`

User Login

- Fill Details and click on login button.
- Check if user is registered
  - if YES Redirect to Home page
  - else Invalid login
- **Models**
  - used django default user model.
- **Views**
  - used django default login view
- URL connected to view `path('',include("django.contrib.auth.urls")),`

2.Main App

Searching For Movie

- Whenever a movie name is searched first checks whether it is previously searched or not
  - YES - Display Stored information
  - NO - create new Search model and scrap and save search model
- **Models**
  - SearchClass

        - Fields
        - name (movie name which is searched)
        - movieobjects (obtained movie objects after scraping for moviename)
        - tvshowobjects (obtained movie objects after scraping for moviename)
- **Views**
    - searchresults
        - Parameters
        - response (POST request from search button)
- **URL** connected to View `path("searchresults/",views.searchresults,name="searchresults"),`

Scraping

- The scraping code will get called just when the user types the movie name 1st time.
- Scraps movie titles of all search results obtained by searching for movie name in rotten tomatoes.
- If movie title already exists in database
    - YES - display it directly.
    - NO - scrap all features of movie and display it.
- Scrapped data like plot, language, cast... from rotten tomatoes and stored them as a seperate lists, each element corresponds to the feauture of respective search result.
- If any particular feature is not found or not added to the websites, then the elements in our list stores the string 'N/A'
- Similarily For TV shows.

Adding movies to Database

- After scrapping movie titles check if it exists in database
    - if YES - By movie id display it.
    - Else - Create model Movie Object with features from scrapped data and Save (Add to Database)
- **Models**
    - Movie
        - Fields - As mentioned in features
- **Views**
    - results
        - parameters - moviename as parameter to start scraping.
            - **URL** connected to View `path("results/<str:moviename>",views.results,name="results"),`

Adding TVShows to Database

- After scrapping tvshow titles check if it exists in database
    - if YES - By tvshow id display it.
    - Else - Create model tvshow Object with features from scrapped data and Save (Add to Database)
- **Models**
    - tvshow
        - Fields - As mentioned in features
- **Views**
    - results
        - parameters - moviename as parameter to start scraping.
            - **URL** connected to View `path("results/<str:moviename>",views.results,name="results"),`

Creating watchlist,watchedlist and favorites

- Click on Add to wishlist button
- Check if item already added to wishlist - YES - Remove from wishlist. - if NO - check if item is in watched list - if YES - Wont add to wishlist - else - Add to wishlist
    - Similarily for other lists.
    - **Model**
    - for Movie model - Connects using ManytoMany Field `users_wishlist = models.ManyToManyField( User, related_name="users_wishlist", blank=True) users_favlist = models.ManyToManyField( User, related_name="users_fav", blank=True) users_watchedlist = models.ManyToManyField( User, related_name="users_watchedlist", blank=True)`
    - for tvshow model - Connects using ManytoMany Field `users_wishlist = models.ManyToManyField( User,`

        `related_name="users_tvwishlist", blank=True) users_favlist = models.ManyToManyField( User, related_name="users_tvfav", blank=True) users_watchedlist = models.ManyToManyField( User, related_name="users_tvwatchedlist", blank=True)`
    - **View**
    - add_to_wishlist
        - parameters - movie id(by id movie added to wishlist of user)
    - add_to_tvwishlist
        - parameters - tvshow id(by id tvshow added to wishlist of user)
    - **URL** connected to View `path("wishlist/add_to_wishlist/<int:id>", views.add_to_wishlist, name="user_wishlist"), path("wishlist/add_to_tvwishlist/<int:id>", views.add_to_tvwishlist, name="user_tvwishlist"), path("favlist/add_to_favlist/<int:id>", views.add_to_favlist, name="user_favlist"), path("favlist/add_to_tvfavlist/<int:id>", views.add_to_tvfavlist, name="user_tvfavlist"), path("watchedlist/add_to_watchedlist/<int:id>", views.add_to_watchedlist, name="user_watchedlist"), path("watchedlist/add_to_tvwatchedlist/<int:id>", views.add_to_tvwatchedlist, name="user_tvwatchedlist"),`

Templates

1.Main App

- home.html
    - whenever user logins home page is diaplayed
    - rendered by view - home `return render(response, "main/home.html", {})`
- results.html
    - After searched for movie it shows results here.
    - rendered by view - searchresults and results `return render(response, "main/results.html", send)` `return render(response, "main/results.html", stuff)`
- particular.html
    - Full info of movie shown here
    - rendered by view - cinema `return render(response, "main/particular.html", {"movie": movie})`
- showparticular.html
    - Full info of tvshow shown here
    - rendered by view - show `return render(response, "main/showparticular.html", {"show": tvsho})`
- wishlist.html
    - Wishlist of respective user is shown in respective user wishlist pages both movies and tvshows
    - rendered by view - wishlist `return render(request, "main/watchlist.html", {"mwishlist": movies,"twishlist":tvshws})`
- watchedlist.html
    - Watchedlist of respective user is shown in respective user watchedlist pages both movies and tvshows
    - rendered by view - watchedlist `return render(request, "main/watchedlist.html", {"mwatchedlist": movies,"twatchedlist":tvshws})`
- favlist.html
    - Favoritelist of respective user is shown in respective user favlist pages both movies and tvshows
    - rendered by view - favlist `return render(request, "main/favlist.html", {"mfavlist": movies,"tfavlist":tvshws})`

2.Register App

- register.html
    - user signup page is shown here
    - rendered by view - register `return render(response,"register/register.html",{"form" :form})`
- login.html
    - user login page is shown here
    - rendered by view - login (django default)