Progress Report

1. A brief description of your application.

   We are planning to make a web interface which allow users to see movie recommendations based on the current popular movies from multiple sources and user's own preference. The movie site would allow user to login and fill out a form to indicate his preference, including his favorite genre, actor, and director. The website would also store the watch list and want to watch list of the user. On the back end, the website will use API of the popular movie sites, such as Imdb, rotten tomatoes, and yahoo movies, etc to get information of the movie as tags. Additionally, the application will also get the number of comments that are relevant to a movie from reddit and twitter which will influence the popularity index on our site. And the application will calculate its own popularity score based those multiple sources. The site will also allow features such as writing movie reviews, get url to watch or purchase the movie, etc.

2. A plan for getting the data to populate your database, as well as some sample data.

   To get reddit data: Use a python script with PRAW (Python Reddit API Wrapper). Create an instance using an API key, and go through the weekly top posts of /r/movies post by post, adding any posts that mention the movie and their score. Similarly, the script goes through each of the comments, and records the date, time, and score. It iterates through the comments with a queue, starting with the top level ones and moving down.

   To get Twitter data: Use Python tweepy api to obtain an authentication token. Obtain a cursor from the API. Track the usage of hashtags related to movies, and match the ocurrence of hashtag names of movies with those on our list.

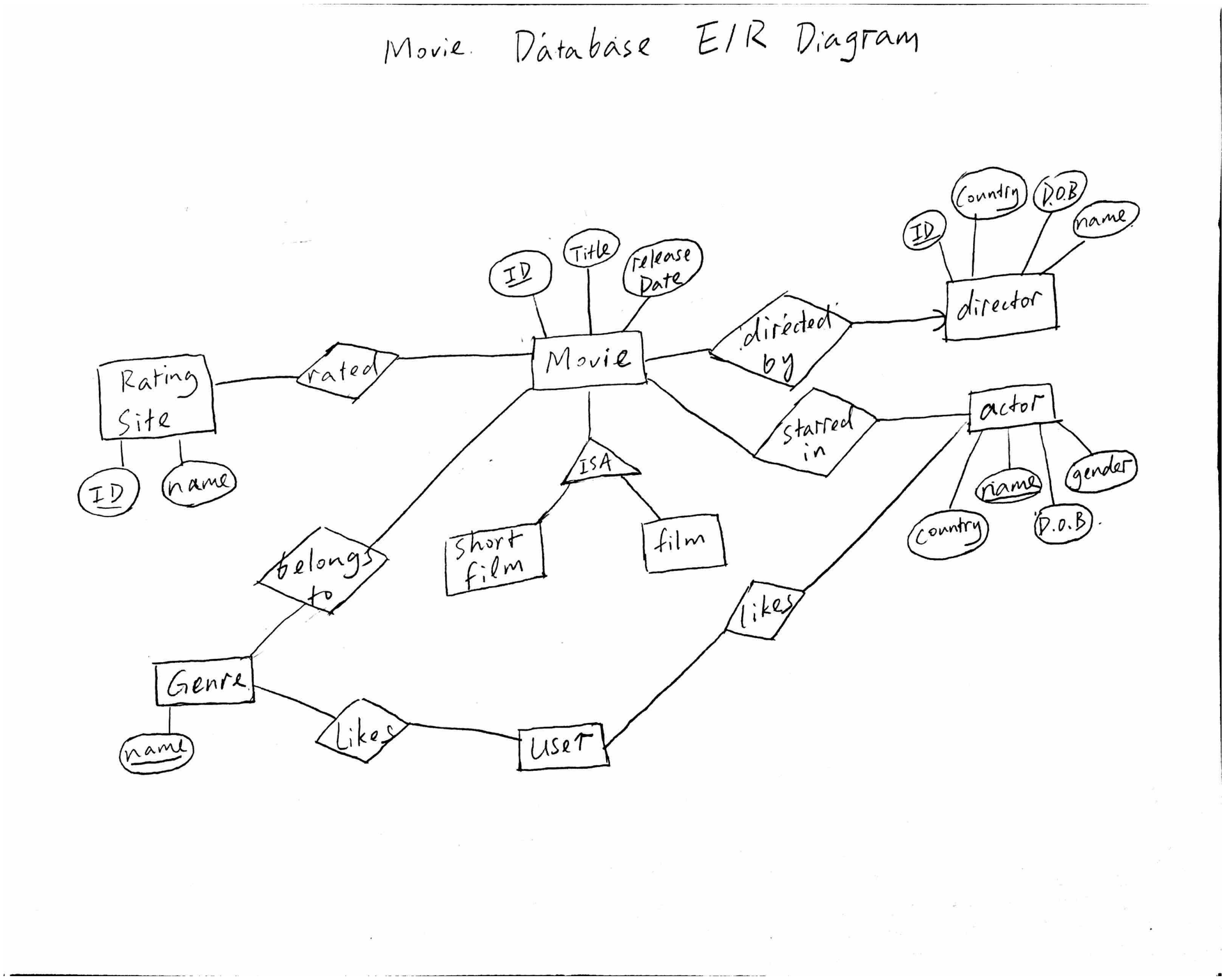
```
import praw
import re
import MySQLdb

db = MySQLdb.connect(host="db.host.url", user="me", passwd="123", db="cs316_project")
reddit = praw.Reddit(user_agent='Comment Extraction', client_id='xLVOBTSVWhVo0A',
client_secret='jGLMgJ25D8r2EgALZV6Gitw_UG4');

r_movies = reddit.subreddit('movies')
posts = r_movies.top(time_filter='month', limit=None)

cursor = db.cursor()
```

```
for post in posts:
   searchTitle = re.search("Ghost in the Shell|GITS",post.title,re.I)
   if searchTitle:
       cur.execute("INSERT INTO Votes VALUES
(Ghost_In_The_Shell,"+post.score+","+"Submission,"+post.created+","+post);
       print (post.title)
       print (post.score)
       post.comments.replace_more(limit=0)
       comment_queue = post.comments[:]
       while comment_queue:
           comment = comment_queue.pop(0)
           searchComments = re.search("Ghost in the Shell|GITS",comment.body,re.I)
           if searchComments:
               cur.execute("INSERT INTO Votes VALUES
(Ghost_In_The_Shell,"+comment.score+","+"Comment,"+comment.created+","+comment);
           comment_queue.extend(comment.replies)
```

3. A list of assumptions that you are making about the data being modeled.

- Due to the organic nature of the language, the movie might be referenced with different names on twitter or reddit comments. Thus we are assuming we can get a relatively accurate count of the number of mentioning on reddit and twitter.
- We also assume that the movie info from the other sites are accurate, for example the name of director, year, genre, etc. We can double check those info by getting the data from multiple sites and choose the overlapping one.
- Since we are creating our own popularity index for the movie, we need to make sure our formula of calculating the index is good so that our index is meaningful.

4. An E/R diagram for your database design.



5. A list of database tables with keys declared.

Movie (id, title, release date)

Director (id, country, D.O.B.)

DirectedBy (movie, director)

Actor (name, country, gender, D.O.B.)

StarredIn (actor, movie)

RatingSite(id, name)

Rates(ratingSite, movie, rating)

BelongsToGenre(movie, genre)

User (id, name, active)

UserLikesMovie(user, movie)

UserLikesGenre(user, genre)

UserLikesActor(user, actor)

6. A description of the Web interface. You can write a brief English description of how users interact with the interface (e.g., “the user selects a car model from a pull-down menu, clicks on the ‘go’ button, and a new page will display all cars of this model that are available for sale”). Or, instead, you can submit a canned demo version of the website.

The web interface would have user login system, so the database stores user individual information. It will prompt user to fill out a preference form, which could just be a screen asking user to click the types of movie they like, and what they have seen recently. Those data will be saved in the database that would be associated with the user information. User can browse the popular movie lists they are created for them based on their preference. They can click on those movies to get the link to stream or purchase the movie to add the movie to want to watch list. Furthermore, user can go back to change their preference information or add movie to their want to watch list.

Another major feature related to the database on the web interface is user can do queries search on the movie. User can define something like get all the certain genre movie with a popularity above a certain number.