Question5：

MovieAdvertisingAndPromotionBot

i). Work bot can do

Basically, a MovieAdvertisingAndPromotionBot should be able to cluster existed movies, new movies and give direct recommendations for a specific movie or give recommendation based on the users' watching history. It will make advertisement or promotions to users so that users can know some movies they are interested in.
A MAAPBot is useful in several ways, it could make our entertainments better.
For viewers, they can see their own watching history, favorites and dislikes. They can easily update their preferences for movies and can get recommendations. For administrators, they can query on metadata of any movie and make modifications to it. For video service providers, they can push movie links on their service to the recommendations for viewers.

ii). The general system description

The main workflow of the MAAPBot can be described as follows. We give our user an UI on the bot where they can register, sign in and manage their own watching history or their favorite movies just as using the computer to access the website. After that, user can explicitly ask for recommendations for a specific movie or ask for movies simply based on what we have about this user's inclination. So, we must give the bot a screen and a Wifi.

iii). The real-world scenarios:

Every user or family should have a special ID. The bot can search on the internet as Google and have access to all the TV and computer of the user such that it can read the past watching and searching history of the user related to movies. So, it can generalize some movie lists based on the history and promote them to the user.
The bot can also use like a movie recommend website on the internet. Users can input some keywords on the monitor of the bot and input their ID, the bot will then show the movies that it recommends to the user on the screen. And the bot can also generalize a list every day and give it to users like an everyday advertisement on TV.

iv). Representation of data

Viewers are movie consumers who are willing to share their tastes on movies with us and wish to get recommendations back.
System Data Input for Viewer: Username, password, Movies Watched, Movies liked, Movies disliked
Input Data Types for Viewer: username: String; password: String; moviesWatched: [movieID]; moviesLiked: [movieID]; moviesDisliked: [movieID] movieID: String (The unique ID of a movie from IMDB)
System Data Output for Viewer: Movie recommendations, movie links(if any)
Output Data Types for Viewer: recommendations: [movieID]

Administrators are the ones who can update movies in our model and modify the metadata for movies.
System Data Input for administrator: Operations, modifications
Input Data Types for administrator: Operations: [addMovie, dropMovie] Modifications: [updateTitle, updateCast, updateDateOfRelease…]
System Data Output for administrator: Operation or modification status
Output Data Types for administrator: status: Boolean
Video service provider scenario description: Video service provider are those who wish to put links in our system so that viewers can choose to watch movies that we recommend on their sites.
System Data Input for Video service provider: Movie links
Input Data Types for Video service provider: movieURL: String
System Data Output for Video service provider: Status
Output Data Types for Video service provider: status: Boolean

And to build our model, we use these features to build the model for clustering:

Genres: Action, Adventure, Animation……
Keywords: violence, culture……
Directors
Cast
Crew
Vote average

The data structure for a movie's metadata is an array of real values which are normalized to characterize a specific aspect of the movie.

v). Algorithm and model building:

Basically, the core of the bot is to use K-means algorithm to cluster movies. This is the data preparation and model building step. Once we have trained our model with data, we can start providing recommendation service. For this part, we use K-nn algorithm to find the nearest neighbors of movies that users watch. These two steps constitute the main idea of the MAAPBot. Among many K-means algorithms, we choose Lloyd's Algorithm which is an efficient heuristic algorithm for clustering.

```
Algorithm 1: Use Lloyd's algorithm to cluster
  Input: Dataset of movie metadata
  Output: Trained model
  randomly assign all data items to a cluster;
  loop
      compute centroids for each cluster;
      reassign each data item to cluster of closest
        centroid;
  until no change in cluster assignments;
```

Algorithm 2: Use K-nn algorithm to find nearest neighbors

```
Input: $x$, a point for which label is not known
Output: label classes
Calculate $d(x, x_i) i = 1, 2, ..., n$; where $d$ denotes the
  Euclidean distance between the points.;
Arrange the calculated $n$ Euclidean distances in
  non-decreasing order.;
Let $k$ be $a + ve$ integer, take the first $k$ distances from
  this sorted list.;
Find those k-points corresponding to these k-distances.;
Let $k_i$ denotes the number of points belonging to the
  i-th class among k points i.e. $k \geq 0$;
if $ki > kj \, \forall i \neq j$ then
  | put $x$ in class $i$.;
```

vi). Constraints that may put into consider:

There are some constraints about the data that may cause error of the bot, I list them below.
Primary Key Constraint: Every movie in our dataset should have a unique ID which is given by IMDB.
Not Null Constraint: Every movie in our dataset should have a value in each field. No null value is accepted.
Normalization Constraint: The values in every field should be normalized before they participate in the computation. This is to make sure every field is measured on the same scale.
Quality Constraint: For the labeling and recommendation part using K-nn, we should discard labels that are further than a threshold in order to control the quality of these found neighbors.

vii). Some problems and thoughts:

I think for the further consideration; the model's input can add a date tag. For example, when input the watching history of a user, we can relate the movies a user watched on Christmas Day to the tag family, love, happy and relate the movies watched on Halloween to trick, ghost, and so on.

Some parts of the movie data may be small. For example, some movies are in special class and there are not enough movies in that class, so the model may be overfitting. So, there is a chance that our bot model may be overfitting or underfitting. We should find some way to deal with it.

And I think it is a best idea the bot can better interact with users. For example, we can add some API like speech recognition and speaker such that the bot can even *talk* to users.