1. **Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**
   Our project has no changes.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**
   Achieved:
   UI design, display detail information (movie, actor, director), display list (movie, actor, director), Login, Search, (add, edit, delete) review, edit user profile, recommendation system, pick out hot movie and good actor.
   Failed:
   User registration, filter movies according to some specific features such as genres

3. **Discuss if you changed the schema or source of the data for your application**
   We do not change the schema.
   Source of review: we use our own developed crawler to get the review from IMSB and Rotten Tomato.
   Source of movies' picture: we also use crawler to get the picture.
   Source of users: random generation, and then we randomly assigned reviews to users.

4. **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

   We add primary key to watch table(watchID) and act table(actID) because it is easy to operate data by ORM if each table have their own primary key instead of combining foreign keys as primary keys.

   We add "picture" attributes to movie table. It looks nice if we display picture for each movie.

5. **Discuss what functionalities you added or removed. Why?**

Add:

Application1: pick out the best movie in recent 3 decades.
Because people in different decades have different movie tastes, it's more likely for people to choose movies that they will like in their decades.

Application 2: pick the hot movie and good actor
Because some users do not clearly know which movie they want see, this functionality provide them a reference and tell them which are hot movies now. Generally, users always want to watch good movies. Therefore, this functionality provides users convenience.

Application3: when a user comments on a movie (after insert), according to the source, the movies' corresponding rating will be updated (update Table movie).

Because the functionality provides convenience for people to know the most recent rating of a movie whenever different resources ratings are changed.

Removed:

User registration:
We didn't have enough time to implement the functionality. We already have a creating review function, the way to create a user is same as creating a review.

Search actors:
This function is not often used by people. People always know movies at first, and then they can link to actors through movies. Therefore, Search Movies is same as Search actors.

6. Explain how you think your advanced database programs complement your application.

Trigger:
It can update movies' rating after users insert reviews, it can keep the movies' rating as the newest.

Procedure:
It picks out best movies for each decade. Because there are some users want to review old movie and others want to watch new movies. It is convenient for them to choose movie according to decade. Furthermore, it is a reference for users who has difficulties in choosing movie.

7. Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.

Jie Chang:
We didn't separate the front-end and back-end which causes problems. When the back-end needs to be added other functionalities and shown on the front-end, people who design the front-end need to wait until the back-end is finished which wastes a lot of time. Sometimes we need to merge the modified front-end and back-end in the same file which is troublesome. Thus, it's better to separate the front-end and back-end at the start of the project.

Jinghong Li:
Recommendation system often uses user's reviews to find those similar users, then it can recommend unseen movies for them. However, not all users have enough reviews especially when they just enroll, most recommendation algorithms find

difficult to solve this cold start problem. To better start our recommendation system, we need do some initial plans, such as recording user's preferred movie genres at first, collecting more movie's features for further content-based recommendation, etc.

Yeting Qi:

We use the ORM as relational database mapping, it is convenient for us to operate data as object. However, it is difficult to use ORM make complex sql query, so when we want to execute advanced query, we choose to connect directly to the database and use cursor to execute raw sql query. It is more convenient than ORM when we want to execute advanced query.

Yizhen Ma:

Start early and strictly follow online instructions if your project needs to use cookies to store user information. Standard cookie updating should involve sending a request to a new URL address. Do not skip this part, otherwise it will be very painful to get updated cookies from the user's browser, or you will probably encounter many unexpected bugs.

8. **Are there other things that changed comparing the final application with the original proposal?**

Our UI design is different from the original proposal.
We didn't implement user sign up functionality.
We can only search movies but not actors.
We filter out hot movies, good actors.
We pick out best movies and actors in each decade.

9. **Describe future work that you think, other than the interface, that the application can improve on**

1. Add user registration function.
2. Improve speed of recommendation algorithm.
3. Add trailers for all movies to let users know more about them.
4. Show upcoming movies.
5. Collect movie box office data.
6. Add Trigger to update movies' rating when users delete and edit their reviews.

10. **Describe the final division of labor and how well you managed teamwork.**

We cooperated very well, each teammate has tried their best in this project.
Jie Chang: front-end and back-end (list information, detail information)
Jinghong Li: data collection and back-end(recommendation algorithms, Procedure)
Yeting Qi： data processing, and back-end (detail information, CRUD, Tigger, Advance query)
Yizhen ma: data processing, and back-end (Login, CRUD, Procedure)