Just Another Movie: Project Update 2

# Group Members:

**Emmett Storey, Robert Jones, Hayden Richards, Kaitlyn Hardin**

# Purpose

Just Another Movie is an application that uses AI to accurately recommend movies based on the user’s search criteria and user ratings.

# Updates

### UI Design/Frontend Development - robert

There have been a few changes from the last update. I have been successful with adding a color scheme. It could easily change because the color may be too dark and may not be clear enough. Just to reiterate from the previous update I have successfully created and implemented buttons, such as Next, Previous and Exit. They all play a unique role in this application. The exit button allows the user to exit from any screen of the application if they wish to close the page. The previous button allows the user to return to the last page that they have made progress on without losing the data that has been recorded. The next button allows the user to progress forward, within multiple screens. As the user proceeds to the search screen the he or she is allowed to enter their criteria which will then be populated on the Results page when the user clicks the Get List button located at the bottom of the page. In the remaining weeks of designing the User Interface a function will be created to store the user information on window 2 to be passed to the result page or window 3.

I have also come across challenges in the past, but Dr. Nicholson has been very helpful with helping me to recover my files that had errors. I have learned to be patient and to work at smaller sections on an application to avoid errors and decrease stress. I also had an instant with a corrupted file which led me to restart the application. As a team we have decided to implement the user to text their criteria and hold off on the drop-down box for now and to come back to it if time permits.

In the final weeks of development my goal is to make final corrections, such as discuss as a team if we need to change the color scheme. As discussed earlier it may be difficult to read a label called Title on the search screen. Secondly, we have determined a cold start problem and will need to create a solution. Finally, as a team we will create connections to our parts of the application called Just Another Movie for final demonstration.

Backend - EmmetT  
All of the classes have been implemented along with most of their functions being completely implemented, if not partially. However, the backend functions and code has changed significantly. The query class has been made to send queries to the database replacing the AI class. The Query class has TitleQuery(), DirectorQuery(), ActorQuery(), GenreQuery(), YearQuery(), GetGenreIds() have been added to its private methods. The AI class has been scrapped. One of the Error Handling classes functions allInputEmpty() has been scrapped. An addMovie() method has been added to the MovieList class and remMovFrmList() has been changed to removeMovie().

Furthermore, I implemented new adjustments allowing for the handling of multiple genre inputs up to 3. The unit tests have been mostly defined but not implemented. Most of these tests would involve the Query class and making sure it can access the data on the server and search through it appropriately.

Most of what needs to be implemented are The List<int> classes that holds the results of the query need to be changed to the QueryResults class that holds the movie id and recommendation level of the results.

The program that interacts with the frontend. One of the primary challenges we face is that due to Emmett’s lack of experience with databases and SQL and he is having to teach himself while focusing on implementing the Query class and other interactions within the server. Another problem will be debugging the searchQuery function. Emmett’s lack of experience with SQL will no doubt impede on his ability to understand defects and bugs in the code which will no doubt result in a longer debugging process than normal. For example, due to the database needing to be remodeled it has made testing a bit difficult and pushed it back. I have had some problems running the test I can at the moment, and it is somewhat hard to pinpoint the problem. All I can understand is that it is an invalid read-write access error.

### AI - hayden

The AI has been developed and implemented into the project, but we have encountered a problem with implementing the trained model. We unknowingly chose a model that can’t make a prediction on a new user without adding in their movie ratings into the training data and retraining the model. Because of this, we are going to have to ask the user which movies they like depending on the genre they choose and save what they choose to the csv file that stores the training data. Then we must re-train the model after the movies is generated. We might have to delete the user recommendations. Therefore, if the user wants to choose a new movie genre, we must display a new list of movies for the user to choose from and then retrain off of that new list of movies.

The problem I was having with using the function to split the data into testing and training data has been resolved. I found another tutorial online and what they did was different from what Microsoft showed how they wanted it to be done and it works. However, the function can take in an integer that works as a seed value so that it will split the data the same way each time it is called. As a result, I will need to play around with different values to see which seed value will give the best results. It is also necessary because if the user chooses one genre and wants recommendations for a different genre, and they re-enter their choices. This way the training and test data are the same between the different choices they make. This will provide a level of consistency in the recommendation engine.

The cold start user problem has been solved; we are going to display a list of movies based off what genre the user has entered. Next, we are going to save their choices to the csv file stores our data and then train the model. We might have to delete their choices after each time they choose a new genre, but the model has a parameter in it so that it won’t use duplicate values. We might have to make a function that will go through and put the movie ids in numeric order because the ids in the csv file we got from the dataset we are using is in numeric order. However, it might not be a problem. I just haven't tested the data with the movie ids not in numeric order. The machine learning algorithm maps all of the movie ids and user ids to key values so it shouldn’t be a problem. With this method of fixing the cold start problem we should be able to provide accurate recommendations to the user.

After fixing the problems and using the larger testing set, the AI can generate a list of movie ids and how likely the user is to like the movie. The r-squared value hasn’t changed and is still yielding us a value of only .4 and we are still aiming for a value of .5 or above. Although, the backend is going to be able to filter out some movies based on what the user enters. This should increase the likelihood that the user will enjoy what movies are displayed to the user. Saving the AI to a .zip file is no longer necessary because we are having to retrain the model for every recommendation. If possible, I will try and figure out a way for the model to be able to make predictions on new users, but from the research I have done, it seems like I would have to build a model for the model to be able to make predictions on new users. The two things left for the AI is just getting it to work with the backend and trying to increase its accuracy.

### Database - Kaitlyn

The database is hosted on MS SQL Server provided by APSU CIST Department. It was successfully set up and is managed by Kaitlyn Hardin. There are no current issues or any expected issues with hosting the database.

The design of the database has changed significantly since the latest update. The relationship between the tables in the database is different. Instead of two tables in the database, there are six tables. The field data types in each table are updated to adequately reflect the expected value for each column. These changes are documented in the updated ERD diagram. Moreover, the database will not connect to a web server, but it will directly connect to the application. After many trials with hosting a server, a significant amount of time has passed, and no solution has been found. Therefore, it is unlikely to complete the project following the previous system design. As a result, a direct connection to the database needs to be established.

All of the tables for the database have been created. Currently, there are seven tables in the database. One major issue with adding the rest of the tables is the extraction of data from the source dataset. The way the owner of the source dataset stored the information is unstructured and the format of the data is unique. This makes extraction difficult to automate. Luckily, we have found a technique to extract the data in our necessary format. Now, the technique needs to be implemented and applied to the various csv files in the source dataset to create the rest of the database tables.

The connection to the database has been implemented. Currently, we have no issues with the database connection. Successful queries have been executed using the connection to the database and all CRUD functionality is available to be executed. This was a critical objective. There were no major issues with established in this connection to the database.

### Diagram Description automatically generatedUpdateD Backend Class Diagram

### Updated AI Design

Diagram

Description automatically generated

### updated erd diagram

Graphical user interface, diagram

Description automatically generated

### System Design Diagram

System Diagram

Description automatically generated