<https://amille73-su.github.io/ist363/Projects/Project2/project2.html>

Note: the API used for the movie ratings and director information has a limit of 10 uses per day – after the ten uses are up, the text should change to say unknown or unable to find information but the first API with the plot and release year should still work fine

For my project, I utilized two API’s that pulled information about movies. The first API is OMDb, an API that provides general information about films including the title, year of release, summary of the plot, and rating (ex. PG-13 or R). The second API is from RapidAPI, called Movies Ratings, that holds more specialized information including IMDb scores, Rotten Tomatoes ratings, and director names. These two APIs worked together by utilizing the IMDb ID (a unique key like *tt109830*) as a shared identifier. When a user selects a movie, my code first fetches basic details from OMDb, then using the same IMDb ID, the code pulls the ratings and director information from the Movie Ratings API. By using a shared identifier, I could combine both sets of data to give a complete overview of each film.

Since APIs don’t respond instantly, I managed delays to ensure that the data loaded in the right order. To do this, I used asynch/await which helped to keep the process smooth and prevent any errors (ex. after selecting an ID, the code waits for OMDb to pull data before moving on to the second API). This was done to ensure that the second API call always had the correct ID. I also included error handling with try-catch blocks so if an API failed or was missing information, the page could still display a message instead of simply breaking down. This approach ensures that the page stays responsive even if some data takes longer to load or is unavailable.

To structure the data on my page, I organized the page into clear sections using bootstrap to style and create a clean, responsive layout. At the top, I included a dropdown menu that allows users to select a movie. Once users select a movie and click “Get Info”, the page populates with details about the selected movie. There are three main areas in which the information populates: the basic info section (title, plot, MPAA rating…), detailed ratings (IMDb and Rotten Tomatoes scores), and additional facts (director, release year, and generalized star rating). Below these interactive sections, I added some general statistics about the top movies and top directors in the world. For extra visual appeal I also added a carousel of movie posters. Bootstrap’s grid system (row and col) kept everything aligned on both desktop and mobile, and I used cards with dark backgrounds to make the text stand out. The goal was to make the information easy to scan without overwhelming the user.

Throughout the project, I encountered a few challenges. One challenge was connecting the two APIs to one another using the IMDb ID. Originally I was pulling the title then converting to the ID but after making a few small changes to the movie selection value, I was able to connect the two APIs by sticking to the IMDb ID as my main unique key. Another challenge was converting the IMDb score into a star rating. Since the numerical value can return numbers such as 8.5, I wrote a function to map a 2 point range that displays stars instead of digits (ex. 8–10 = ★★★★★). Outside of these two errors, I also ran into some missing data specifically with Rotten Tomatoes scores. To handle this, I added a fallback message so users would not receive a blank space. With these limitations I added error messages to ensure that users understood that there was an issue loading the data rather than seeing blank spaces or having outside errors.