IMDB Scraper and Web Application

1. ## Scraper Application - **IMDB_Scrape.py**

   The Scraper Application goes through the IMDB website and scrapes the list of movies along with the posters of each movies and stores the information in the local file system
   The Scraper Application has two functions

   scraper()
   - Scrapes the IMDB website for the movies name and link to the movies page and poster and stores it in the file IMDB_Movie_Index.csv

   poster_scraper()
   - Uses the scraper output to download the actual posters of each movie and stores it in the 'posters' directory

   Apart from these two functions the list also has **decorator** function time_calc(), Main() and writelisttocsv()

   time_calc()

   Takes in three boolean arguments 'enable' , 'logToFile', 'logTime'. The log of this is stored in the directory 'files' under 'venv'

   Main()

   The main function is used to define the argument parser for this program. The three arguments defined for this program are -t, -p, -tp.

   -t - run only the scraper()
   -p run only the poster_scraper()
   -tp run both the scraper() and poster_scraper()

   writelisttocsv()

   Takes in two arguments file path and data from scraper() to store it as a csv file

2. ## Web Application - **WebApp.py**

   The Web Application uses the stored information to display the contents to the user.
   Functionalities
   1. List movies
   2. Search movie from the list
   3. Delete movie from the list
   4. Modify a movie name from the list
   5. Dump data in CSV as well as JSON format
   6. Log Decorator to log the time of each function. The logs can be accessed from 'files' directory

*The application only displays the top 50 movies. The whole list can be accessed from the IMDB_Movies_Index.csv file located in the 'venv' directory.

**Two implementations has been included one with SQLAlchemy and another with DataFrames. The implementation with SQLAlchemy is indicated in the filename itself. This is just for viewing purpose alone, not for running.Data Frame implementation has been much more elegant and without error projections. SQLAlchemy projected a lot of troubles while execution and a lot of time consumption, hence I have chosen the DataFrame implementation for the main program.