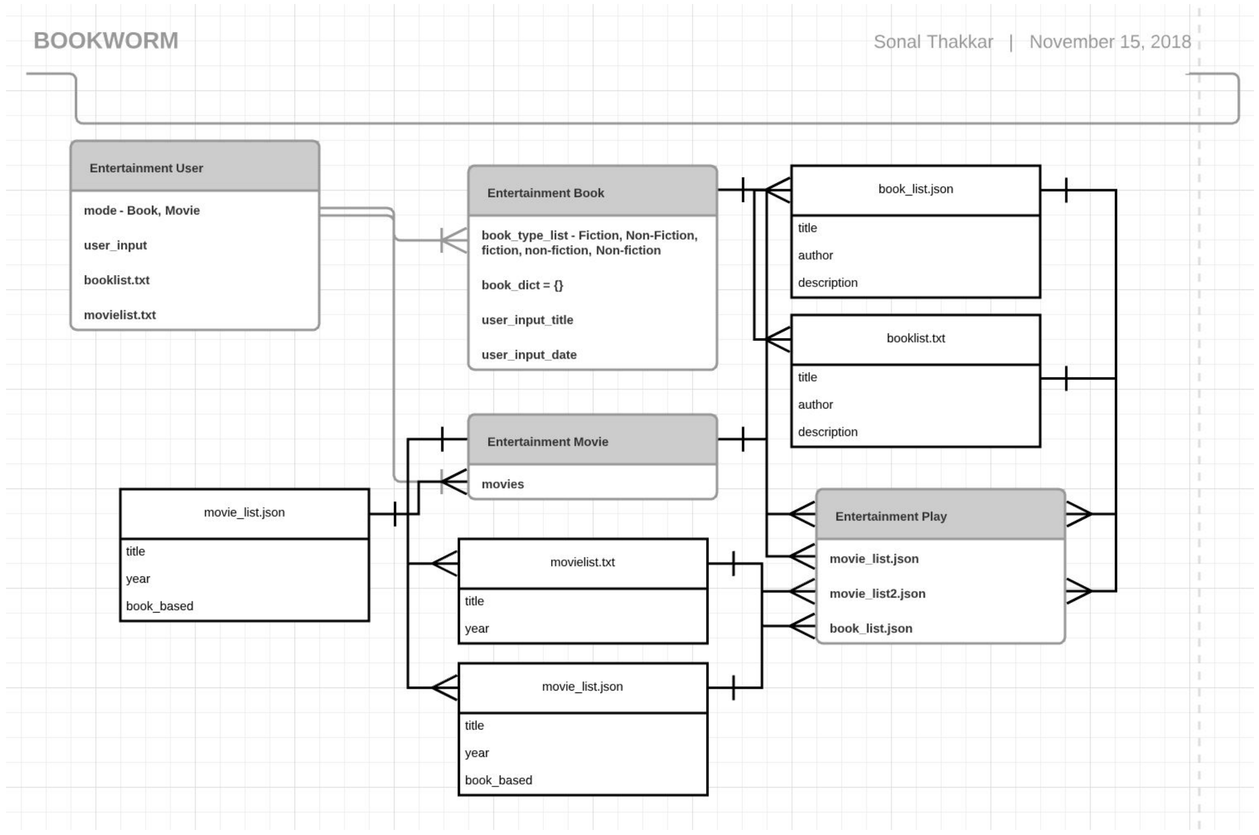
Introduction:
The bookworm project was conceived out of a love for books and the fascination with watching movies based on selected books and how they would match up to the story-line within the book. The fact that one of the async material videos used a Google API to find books and their publications lead to the idea behind this project.

Planning:
I was initially trying to use google APIs to search for movie details similar to the Google Books API. Towards this I researched the customsearch api and youtube api from url:
https://developers.google.com/api-client-library/python/apis/
But was unsuccessful in using either for this purpose.

I then switched to storing this data in a json file for the movie list. And using the API just for getting the book information. Another option could have been storing this information on a database and calling that within the code. However i went with the first approach since i have already used python with database connection through airflow-spark at work.

General Design and Flow of the Program:



Algorithm:
The program starts by calling the Entertainment class - which is the main class, through the main.py function which simply initializes the Entertainment class. The init function has the mode attribute: mode = ('Book', 'Movie')
The Entertainment class calls its main function from within the init function which requests for user to select from a book or movie.

The user choice leads to the initialization of the Entertainment_Book class or the Entertainment_Movie class after ensuring the user input exists within the class attribute mode. This code is wrapped within the Entertainment.entertainment_main(self, mode_val):
call.

The Entertainment_Book class has an attribute tuple: book_type_list = ('Fiction', 'Non-Fiction','fiction','non-fiction', 'Non-fiction')
The Entertainment_Book class lets the user specify a title (user_input_title) and a published year (user_input_dt) if desired. The user_input_title value is collapsed using: ''.join((user_input_title.split(' ')))
Since the google API cannot process NULLs - so if the user enters “How to train your dragon” - this value is processed as “Howtotrainyourdragon”
The user_input_dt value is checked against the numbers
 Package to ensure a valid number is entered if at all. The function then calls the Google API

```
urlis =
"https://www.googleapis.com/books/v1/volumes?q=categories="+user_input_book+"*"+user_input_title+"*"+us
er_input_dt+"'&maxResults=10"
response = urlopen(urlis)
rawData = response.read()
book_data = json.loads(rawData)
```

A list of 10 books is picked and printed to the console. The user can select one of these books which then gets plugged to the booklist.txt file (to be displayed later) and the value entered to a json file to enable the program to select movies based on the book selection.
The user can alternatively choose to go on browsing books which will be displayed in lists of 10.

Once the user makes a selection of a book the EntertainmentPlay class gets called that will load all the json files for books and movies. And depending on existence of movies based on book selected will write the movie details to the booklist.txt file. And then display the contents of the file from the entertainment_play_print function.

Similarly when the user selects movie the Entertainment class initializes the Entertainment_Movie class which displays a list of movies from the movie_list.json file.
I did use a random package to shuffle the movie names before presenting to the user. Once again the user is asked to select a movie or continue browsing. Once the user selects a movie, the details get written to a movielist.txt file to be displayed at the end of the process from the EntertainmentPlay class. The user selected movie also gets fed to another json file which is

used to extract the movie name which is fed to the books Google API to find any related books which then get added to the movielist.txt file.

Conclusion:
This was a fun project to work on! I would love a chance to improve upon the code and if possible to use a google API to search for movies similar to the book class just to make the entire code-base consistent.