Project 2: Media For You

Program Description:

For this project your group will be creating a system to allow a user to search through a collection of movies, tv shows, and books and find information regarding media they may wish to obtain. This system will both perform an analysis of the specified data, generate statistics, and provide a graphical user interface for the user to interact with the system.

Program Requirements

- This is a team-based project, as such, only one member of your group should create a private repository on GitHub to store the code and add the other group members to it
    - Each group member should make at least 3 substantial commits to the group repository during development
    - Consider making a commit to the repository after completing one of the superclasses or subclasses or after completing a major function
- The book files data will be in tab delimited format with the first line of the file containing the data headers, all other lines will contain data conforming to the headers, and the headers will always be in the order presented
    - The number of entries in a file will vary from file to file
- The shows files data will be in tab delimited format with the first line of the file containing the data headers, all other lines will contain data conforming to the headers, and the headers will always be in the order presented
    - The number of entries in a file will vary from file to file
    - This file contains data on both movies and tv shows, as indicated by the second column of data in the file
    - Note there may not be data for every column of data for each entry
- All member variables should be protected or private depending on if they are in an inheritance relationship or not
- Your program must have a `Media` class, stored in a **Media.py** file, that must contain at least the following:
    - Member variables to store an ID, a title, and an average rating
    - An appropriate constructor that takes in an ID, a title, and an average rating as parameters and assigns those values to the appropriate member variables
    - Appropriate accessor/mutator functions
- Your program must have a `Book` class, that is a subclass of `Media`, stored in a **Book.py** file, that must contain at least the following:
    - Member variables to store authors, an isbn number, an isbn13 number, a language code, the number of pages in the book, the number of ratings given to the book, the publication date, and the publisher

    - An appropriate constructor that takes in an ID, a title, an average rating, authors, an isbn number, an isbn13 number, a language code, the number of pages in the book, the number of ratings given to the book, the publication date, and the publisher as parameters and assigns those values to the appropriate member variables
    - Appropriate accessor/mutator functions
- Your program must have a `Show` class, that is a subclass of `Media`, stored in a **Show.py** file, that must contain at least the following:
    - Member variables to store the type of show, directors, actors, a country code, the date the show was added, the year the show was released, the rating, the duration, genres, and a description
    - An appropriate constructor that takes in an ID, a title, an average rating, the type of show, directors, actors, a country code, the date the show was added, the year the show was released, the rating, the duration, genres, and a description as parameters and assigns those values to the appropriate member variables
    - Appropriate accessor/mutator functions
- Your program must have a `Recommender` class, stored in a **Recommender.py** file, that must contain at least the following:
    - An appropriate constructor that instantiates three dictionaries
        - one to store `Book` objects, where the book's id is the key and the value is the object
        - one to store `Show` objects, where the show's id is the key and the value is the object
        - one to store dictionaries keeping track of associations, where a show or book id is the key and the value is a dictionary
            - For the inner dictionary, the key should be a show or book id and the value is the number of times the outer id and inner id are associated
    - A function named `loadBooks()`, that takes in no additional parameters, returns nothing, loads all of the data from a book file, and which should at least:
        - Prompts the user for the name of the file using an appropriate `filedialog`, and if the user does not choose an existing file, it should repeatedly prompt the user for a file in the same way
        - Opens and reads the file one entry at a time
        - Stores the entry for each book in a `Book` object
        - Stores each `Book` object in the appropriate dictionary using the book's ID as the key and the `Book` object as the value
        - Close the file once all of the data has been read in
    - A function named `loadShows()`, that takes in no additional parameters, returns nothing, loads all of the data from a show file, and which should at least:
        - Prompts the user for the name of the file using an appropriate `filedialog`, and if the user does not choose an existing file, it should repeatedly prompt the user for a file in the same way
        - Opens and reads the file one entry at a time
        - Stores the entry for each show in a `Show` object

        - Stores each `Show` object in the appropriate dictionary using the show's ID as the key and the `Show` object as the value
        - Close the file once all of the data has been read in
    - A function named `loadAssociations()`, that takes in no additional parameters, returns nothing, loads all of the data from an association file, and which should at least::
        - Prompts the user for the name of the file using an appropriate `filedialog`, and if the user does not choose an existing file, it should repeatedly prompt the user for a file in the same way
        - Opens and reads the file one entry at a time
        - Using the first id as a key, determine if there is a dictionary associated with it
            - It not, create a new dictionary and then add the second id to the new dictionary so that the second id is associated with the value 1
            - Otherwise, determine if the second id is a key in the second dictionary
                - If it is, increment the count associated with it by 1
                - Otherwise, set the count associated with it to 1
        - Perform the same steps again, but this time use the second id for the outer dictionary and the first id for the inner dictionary
        - Close the file once all of the data has been read in
    - A function named `getMovieList()`, that takes in no additional parameters, and returns the Title and Runtime for all of the stored movies, such that:
        - The data has the header Title and Movie
        - All of the data is in neat, even columns, whose width is determined based on the length of the entries in the data
    - A function named `getTVList()`, that takes in no additional parameters, and returns the Title and Number of Seasons for all of the stored tv shows, such that:
        - The data has the header Title and Seasions
        - All of the data is in neat, even columns, whose width is determined based on the length of the entries in the data
    - A function named `getBookList()`, that takes in no additional parameters, and returns the Title and Author(s) for all of the stored movies, such that:
        - The data has the header Title and Author(s)
        - All of the data is in neat, even columns, whose width is determined based on the length of the entries in the data
    - A function named `getMovieStats()`, that takes in no additional parameters, and returns the statistics regarding movies, such as:
        - Rating for movies (G, PG, R, etc…) and the number of times a particular rating appears as a percentage of all of the ratings for movies, with two decimals of precision
        - Average movie duration in minutes, with two decimals of precision
        - The director who has directed the most movies
        - The actor who has acted in the most movies
        - The most frequent movie genre

- A function named `getTVStats()`, that takes in no additional parameters, and returns the statistics regarding tv shows, such as:
  - Rating for tv shows (G, PG, R, etc…) and the number of times a particular rating appears as a percentage of all of the ratings for tv shows, with two decimals of precision
  - Average number of seasons for tv shows, with two decimals of precision
    The actor who has acted in the most tv shows
  - The most frequent tv show genre
- A function named `getBookStats()`, that takes in no additional parameters, and returns the statistics regarding books, such as:
  - The average page count, with two decimals of precision
  - The author who has written the most books
  - The publisher who has published the most books
- A function named `searchTVMovies()`, that takes in strings representing a movie or tv show, a title, a director, an actor, and a genre, and returns information regarding Movies or TV Shows such that:
  - If the string representing the movie or tv show is neither Movie nor TV Show, spawn a `showerror messagebox` and inform the user the need to select Movie or TV Show from Type first, and return the string No Results
  - If the strings representing title, director, actor, and genre are all empty, spawn a `showerror messagebox` and inform the user the need to enter information for the Title, Directory, Actor and/or Genre first, and return the string No Results
  - Otherwise, search through the dictionary of shows and select all objects that adhere to the user's data
  - Return a string containing the Title, Director, Actors, and Genre (with those titles at the top) in neat, even columns, whose width is determined based on the length of the entries in the data
- A function named `searchBooks()`, that takes in strings representing a title, an author, and a publisher, and returns information regarding books such that:
  - If the strings representing title, author, and publisher are all empty, spawn a `showerror messagebox` and inform the user the need to enter information for the Title, Author, and/or Publisher first, and return the string No Results
  - Otherwise, search through the dictionary of books and select all objects that adhere to the user's data
  - Return a string containing the Title, Author, and Publisher (with those titles at the top) in neat, even columns, whose width is determined based on the length of the entries in the data
- A function named `getRecommendations()`, that takes in strings representing a type and a title, and returns a string containing recommendations regarding Movies, TV Shows, or Books such that:
  - If the type is Movie or TV Show, search through the shows dictionary and determine the id associated with that title

        - If the title is not in the dictionary, spawn a `showwarning messagebox` informing the user that there are no recommendations for that title, and return No results
        - Otherwise, using that movie or tv show id, determine all of the books associated with that id in the association dictionary, and return a string containing all of the information for each book with appropriate titles for each piece of information
    - If the type is Book, search through the books dictionary and determine the id associated with that title
        - If the title is not in the dictionary, spawn a `showwarning messagebox` informing the user that there are no recommendations for that title, and return No results
        - Otherwise, using that book id, determine all of the movies and tv shows associated with that id in the association dictionary, and return a string containing all of the information for each movie or tv show with appropriate titles for each piece of information
- Your program must have a `RecommenderGUI` class, stored in a **RecommenderGUI.py** file, that must contain at least the following:
    - A constructor function that takes in no parameters and:
        - Creates an instance of a `Recommender` object and stores it in a variable
        - Creates a `Toplevel` main window, with an appropriate title, and dimensions of 1200 pixels wide by 800 pixels tall
        - Contains a `notebook` tab to display all of the movie titles and runtimes, as well as the movie statistics
            - This tab should be populated using the appropriate functions from the `Recommender` object
            - The user should be able to scroll through the title and runtimes
            - If no data has been loaded yet, both text areas should display default text informing the user that no data has been loaded yet
            - The user should not be able to alter the data in the text areas
        - Contains a `notebook` tab to display all of the tv show titles and seasons, as well as the tv show statistics
            - This tab should be populated using the appropriate functions from the `Recommender` object
            - The user should be able to scroll through the title and seasons
            - If no data has been loaded yet, both text areas should display default text informing the user that no data has been loaded yet
            - The user should not be able to alter the data in the text areas

- Contains a `notebook` tab to display all of the book titles and authors, as well as the book statistics
    - This tab should be populated using the appropriate functions from the `Recommender` object
    - The user should be able to scroll through the title and authors
    - If no data has been loaded yet, both text areas should display default text informing the user that no data has been loaded yet
    - The user should not be able to alter the data in the text areas
- Contains a `notebook` tab to allow the user to search through the movies and tv shows and should contain
    - A `Combobox` with the options Movie and TV Show
    - Appropriate `Label` and `Entry` widgets to collection information regarding the title, director, actor, and/or genre
    - A `Button` to trigger the search, which calls the appropriate function from the `Recommender` object and stores the results in the text area
    - If no searches have been performed yet, the text areas should display some default text to inform the user they need to enter data to perform a search
    - The user should be able to scroll through the results
    - The user should not be able to alter the data in the text area
- Contains a `notebook` tab to allow the user to search through the books and should contain
    - Appropriate `Label` and `Entry` widgets to collection information regarding the title, author, and/or publisher
    - A `Button` to trigger the search, which calls the appropriate function from the `Recommender` object and stores the results in the text area
    - If no searches have been performed yet, the text areas should display some default text to inform the user they need to enter data to perform a search
    - The user should be able to scroll through the results
    - The user should not be able to alter the data in the text area
- Contains a `notebook` tab to allow the user to obtain media recommendations and should contain
    - A `Combobox` with the options Movie, TV Show, Book
    - Appropriate `Label` and `Entry` widgets to collection information regarding the title
    - A `Button` to trigger the recommendation search, which calls the appropriate function from the `Recommender` object and stores the results in the text area
    - If no searches have been performed yet, the text areas should display some default text to inform the user they need to enter a title to receive a recommendation
    - The user should be able to scroll through the results
    - The user should not be able to alter the data in the text area
- `Buttons` below the notebook that will have appropriate names and will:
    - Load the show data using the `loadShows()` function
    - Load the book data using the `loadBooks()` function

        - Load the association data using the `loadAssociations()`
        - Spawn a dialog containing the information regarding your team using the `creditInfoBox()`
        - Quit the program
- A `loadShows()` function that takes in no parameters, returns nothing, and:
    - Calls the appropriate function from the `Recommender` object to read in all of the data for the shows
    - Calls the appropriate function from the `Recommender` object to obtain the string representing the list of movies and the string representing the movie statistics and displays them in the appropriate text area
    - Calls the appropriate function from the `Recommender` object to obtain the string representing the list of tv shows and the string representing the tv show statistics and displays them in the appropriate text area
- A `loadBooks()` function that takes in no parameters, returns nothing, and:
    - Calls the appropriate function from the `Recommender` object to read in all of the data for the books
    - Calls the appropriate function from the `Recommender` object to obtain the string representing the list of books and the string representing the book statistics and displays them in the appropriate text area
- A `loadAssociations()` function that takes in no parameters, returns nothing, and:
    - Calls the appropriate function from the `Recommender` object to read in all of the data for the associations
- A `creditInfoBox()` function that takes in no parameters, returns nothing, and:
    - Spawns a `showinfo messagebox` containing the names of each of your group members and what day the project was completed on
- A `searchShows()` function that takes in no parameters, returns nothing, and:
    - Extracts all of the data from the appropriate `Combobox` and `Entry` widgets to search for a movie or tv show
    - Calls the appropriate function from the `Recommender` object to search for a movie or tv show, passing in the information from the `Combobox` and `Entry` widgets, and then displaying the returned string in the appropriate text area
- A `searchBooks()` function that takes in no parameters, returns nothing, and:
    - Extracts all of the data from the appropriate `Entry` widgets to search for a book
    - Calls the appropriate function from the `Recommender` object to search for a book, passing in the information from the `Entry` widgets, and then displaying the returned string in the appropriate text area
- A `getRecommendations()` function that takes in no parameters, returns nothing, and:
    - Extracts all of the data from the appropriate `Combobox` and `Entry` widgets,
    - Calls the appropriate function from the `Recommender` object to obtain recommendations, passing in the information from the `Combobox` and `Entry` widgets, and then displaying the returned string in the appropriate text area

    - Outside of the class, the **RecommenderGUI.py** should also contain a main function that instantiates an instance of a `RecommenderGUI` and then calls the tkinter `mainloop()` function
- No global scoped variables, other than those specified are allowed for this project
- Your code should be well documented in terms of comments. For example, good comments in general consist of a header (with your name, date, and brief description), documentation comments for function, comments for each variable, and commented blocks of code

Submission

- Your program will be graded largely upon whether it works correctly
- Your program will also be graded based upon your program style. This means that you should use comments (as directed) and meaningful variable names
- You must submit the **Media.py, Book.py, Show.py, Recommender.py,** and **RecommenderGUI.py** files
- You must submit the URL link to your repository and set the repository from private to public on May 6th
- You must work in teams of 2 or 3 students for this project. You are not allowed to work with individuals outside of your team, other than the instructor and TA. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Additionally, you are not allowed to download code off of the internet or use generative AI for this project. Any discovered instances of this will be considered cheating and appropriate actions will the taken according to the course syllabus
- Be sure that you have tested the version of the program you wish to submit to make sure it works correctly. You will not be allowed to resubmit work after the deadline
- All students are expected to contribute relatively equally with respect to the coding for this project. If it is determined that one or more members of the team provided little to no substantive effort with respect to the coding, those member's project grades will be significantly penalized. If you are having issues with a teammate, please contact your instructor as soon as possible

Rubric

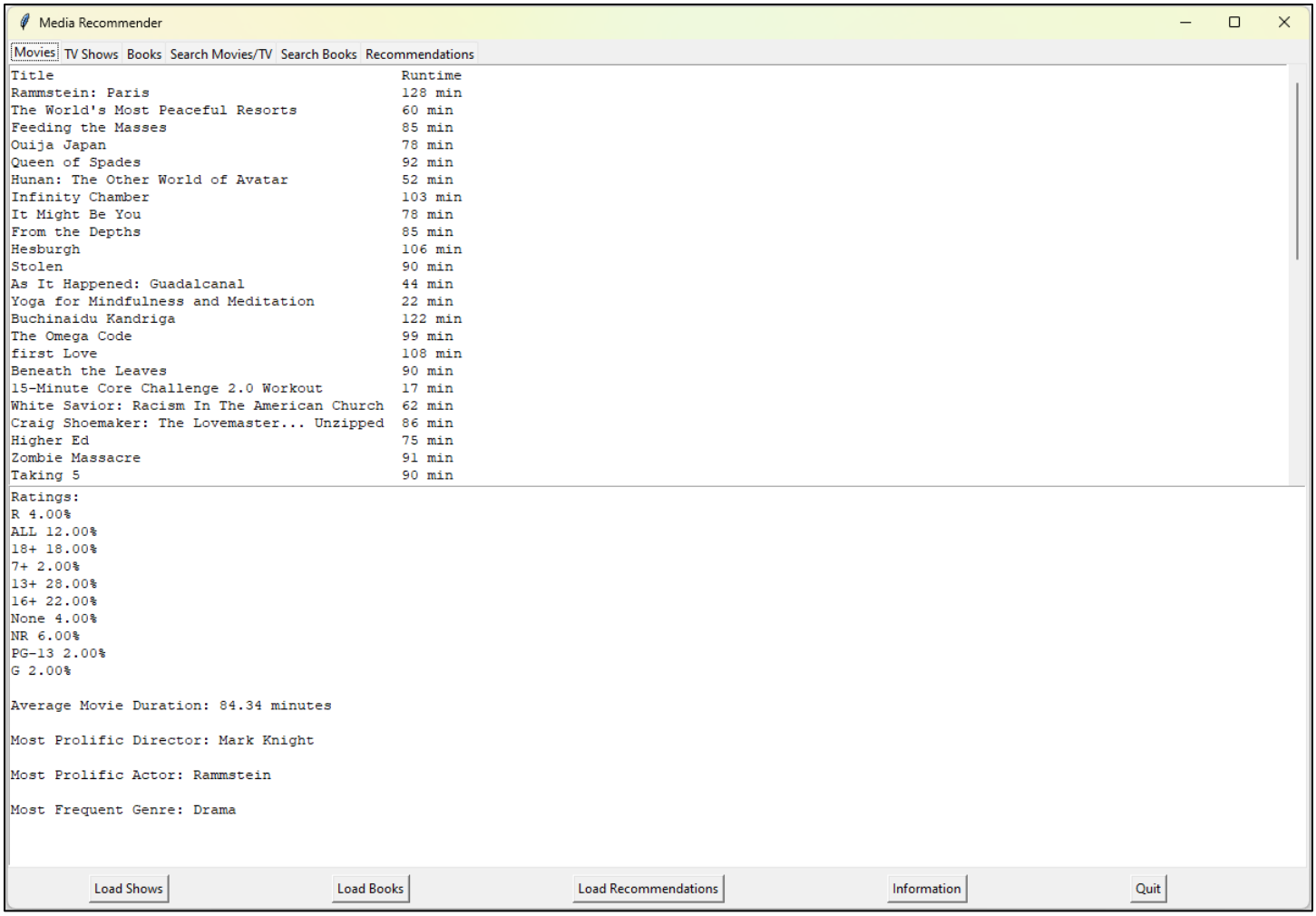
The entire assignment is worth 100 points and partial credit is possible. No credit will be given for portions of the program that cannot be tested due to the program crashing or the code being unreachable due to logic errors.

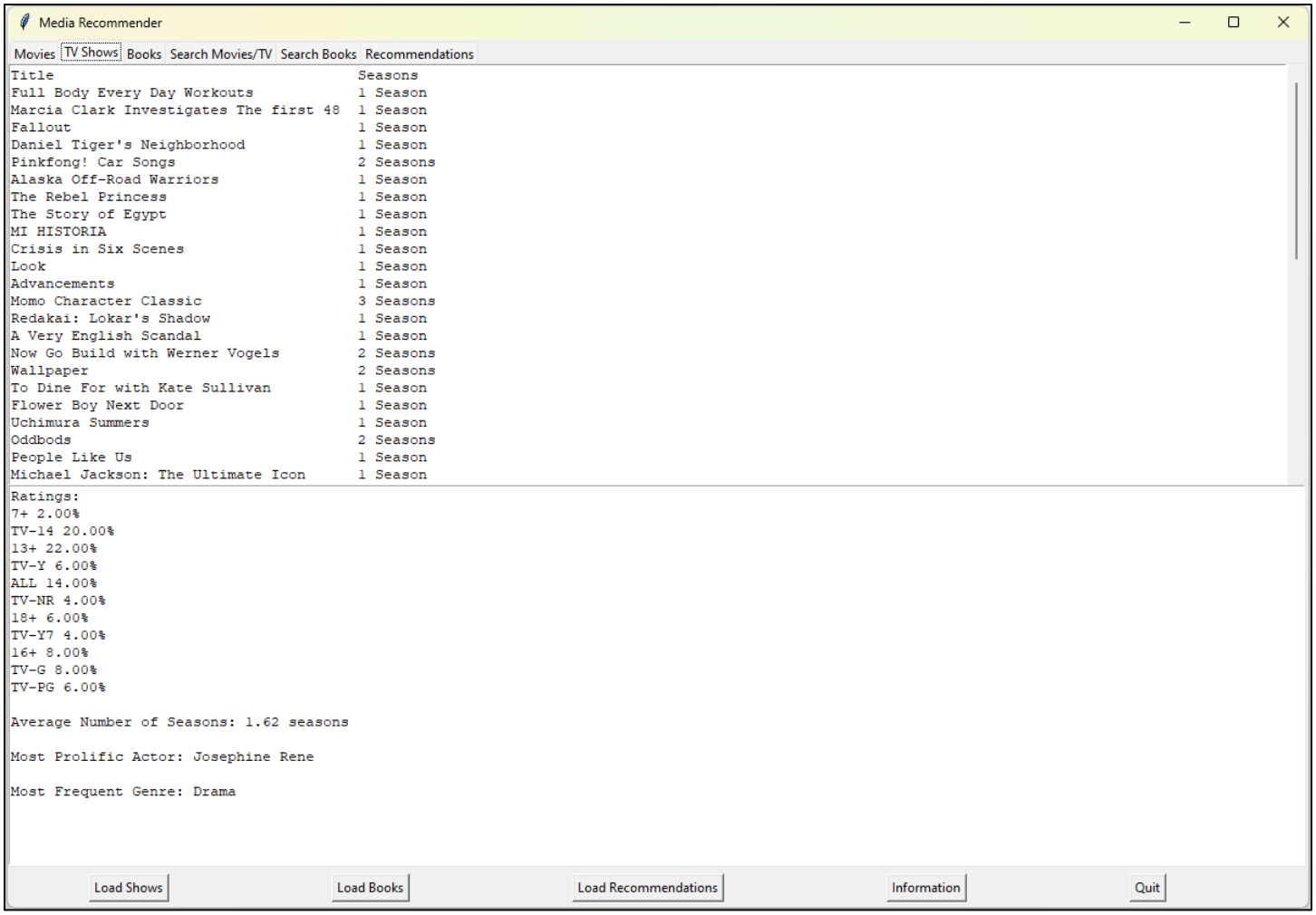
- **Program Executes Successfully**
  - If your program crashes due to syntax errors, you will lose 10 points, but we will attempt to fix minor issues (incorrect indentations, stray character, missing import) so that we can execute and test the program. We will not fix major issues that would require functionality to be further implemented, or a reorganization of logic in your code.
- **Data Storage (10 points)**
  - Each of the data storage classes `Media`, `Book`, and `Show` are setup as specified and use inheritance as appropriate
- **Movies Tab (10 points)**
  - The correct movie data is loaded into the appropriate text area, using the appropriate functions, when the Load Shows button is clicked (5 points)
  - The correct movie statistics data is calculated and loaded into the appropriate text area, using the appropriate functions, when the Load Shows button is clicked (5 points)
- **TV Shows Tab (10 points)**
  - The correct tv show data is loaded into the appropriate text area, using the appropriate functions, when the Load Shows button is clicked (5 points)
  - The correct tv show statistics data is calculated and loaded into the appropriate text area, using the appropriate functions, when the Load Shows button is clicked (5 points)
- **Books Tab (10 points)**
  - The correct book data is loaded into the appropriate text area, using the appropriate functions, when the Load Books button is clicked (5 points)
  - The correct book statistics data is calculated and loaded into the appropriate text area, using the appropriate functions, when the Load Books button is clicked (5 points)
- **Search Movies/TV Tab (10 points)**
  - Appropriate widgets are displayed and used to collect user information (5 points)
  - The correct movie or tv show data is loaded into the appropriate text area, using the appropriate functions, when the Search button is clicked (5 points)
- **Search Books Tab (10 points)**
  - Appropriate widgets are displayed and used to collect user information (5 points)
  - The correct book data is loaded into the appropriate text area, using the appropriate functions, when the Search button is clicked (5 points)
- **Recommendations Tab (15 points)**
  - Appropriate widgets are displayed and used to collect user information (5 points)
  - The correct movie and tv recommendations are loaded into the appropriate text area, using the appropriate functions, when the Get Recommendation button is clicked (5 points)
  - The correct book recommendations are loaded into the appropriate text area, using the appropriate functions, when the Get Recommendation button is clicked (5 points)
- **Buttons (10 points)**
  - Load Shows, Load Books, Load Recommendations, Information and Quit, buttons call the appropriate functions (10 points)
- **Each group member made at least three substantial commits to GitHub (5 points)**
- **Program contains sufficient comments (5 points)**
- **Group completed and submitted the Project Expectations Document (5 points)**

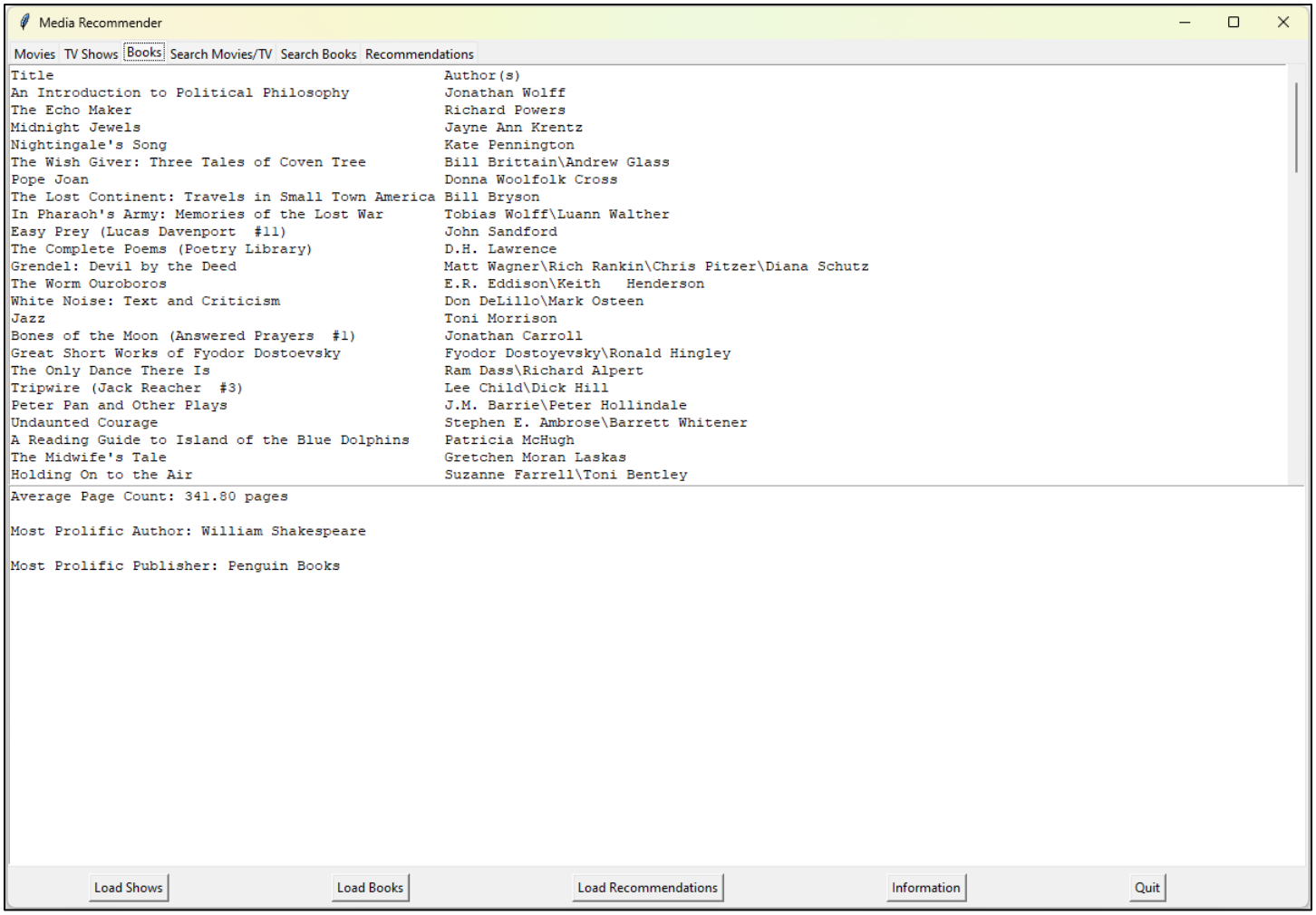
Bonus

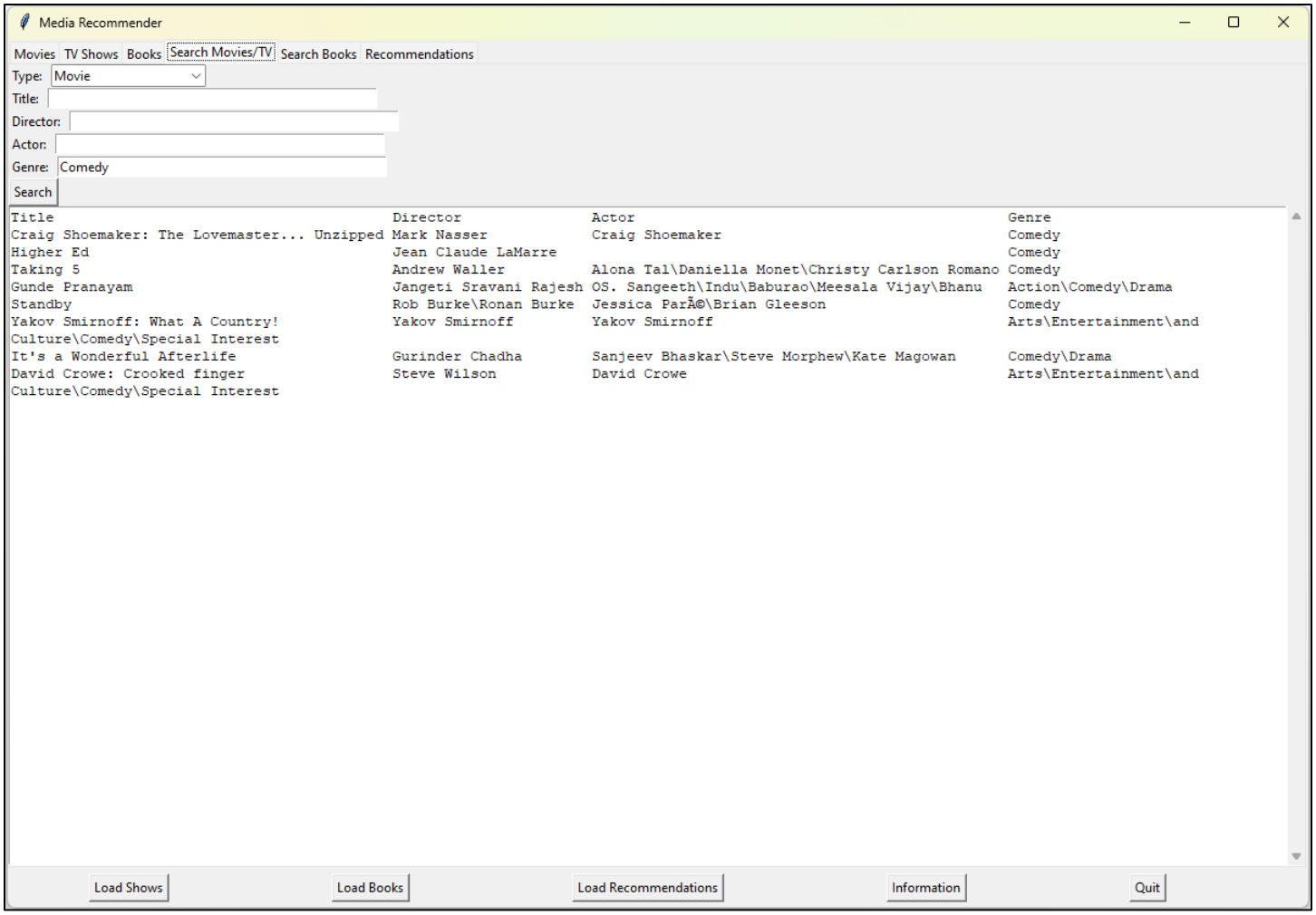
For 10 bonus points, create a new `Notebook` tab called Ratings that will store two matplotlib pie charts. One pie chart will show each percentage of ratings (G, PG, R, etc…) for movies, and the other pie charts will show the percentage of ratings for tv shows. Be sure to include the label (G, PG, R, etc) and percentage value (with two decimals of precision) for each slice of the pie chart. Note you will need to use a tkinter `Canvas` widget to display the pie charts.

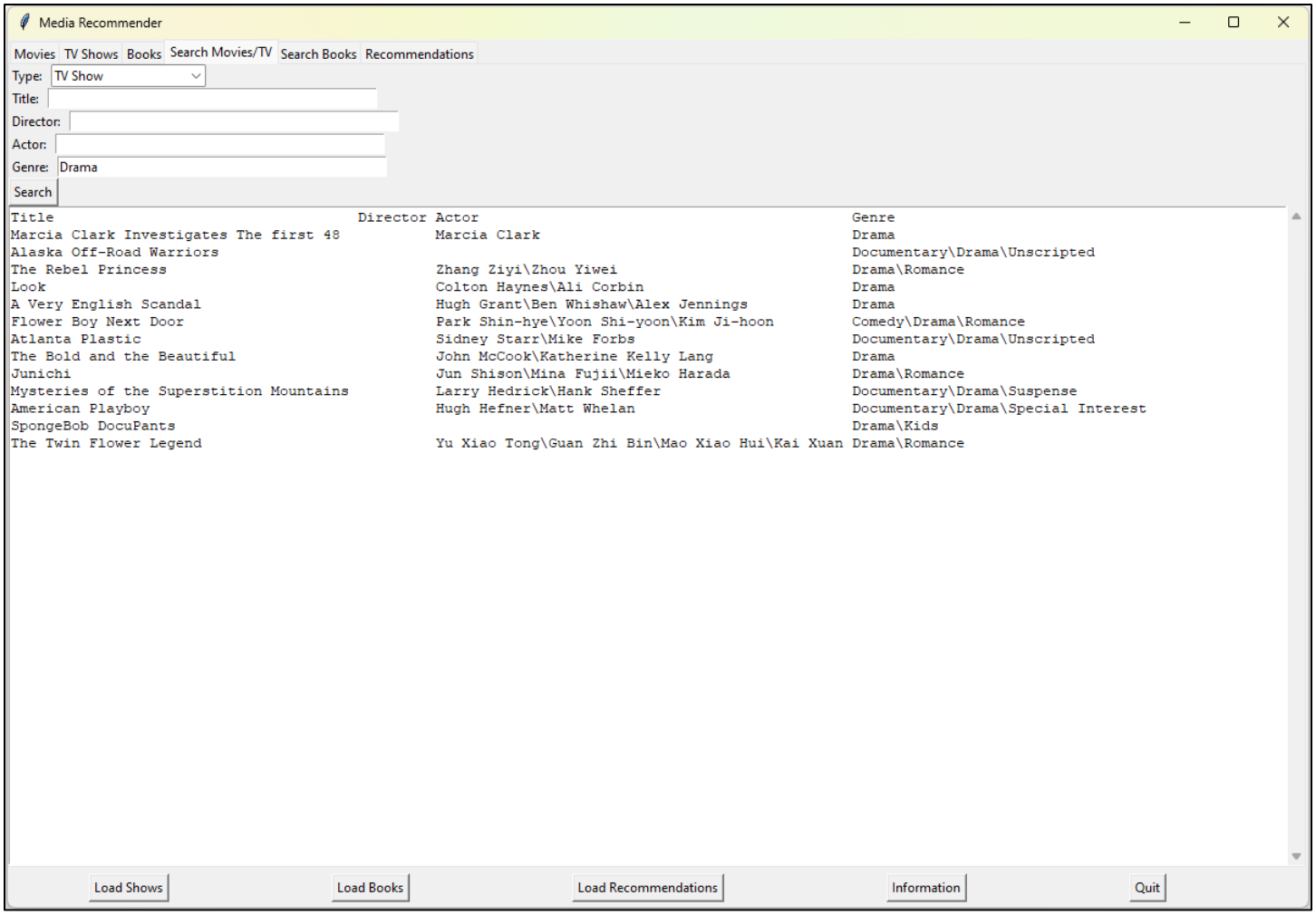
Sample Output

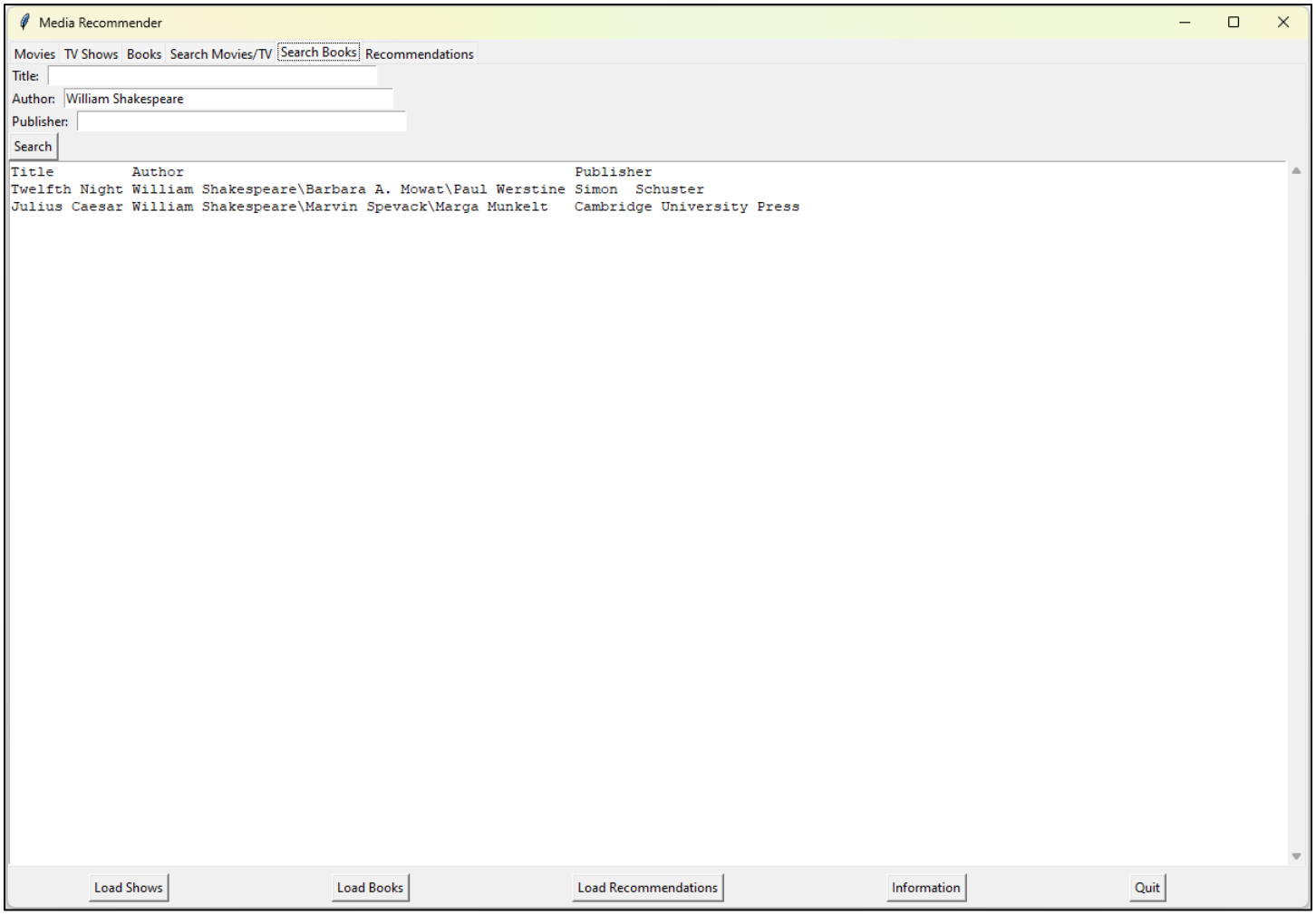


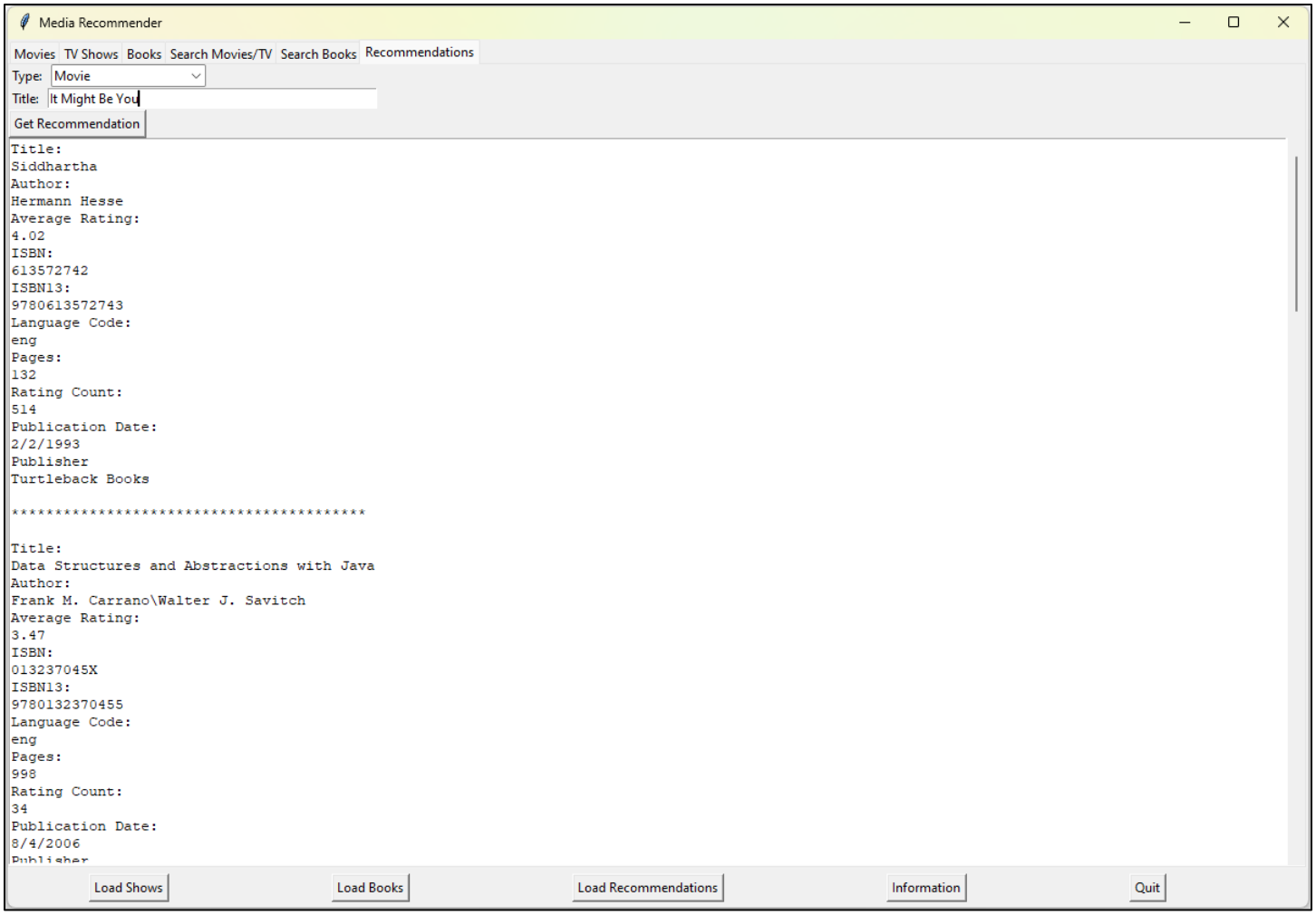

Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Title Seasons
Full Body Every Day Workouts 1 Season
Marcia Clark Investigates The first 48 1 Season
Fallout 1 Season
Daniel Tiger's Neighborhood 1 Season
Pinkfong! Car Songs 2 Seasons
Alaska Off-Road Warriors 1 Season
The Rebel Princess 1 Season
The Story of Egypt 1 Season
MI HISTORIA 1 Season
Crisis in Six Scenes 1 Season
Look 1 Season
Advancements 1 Season
Momo Character Classic 3 Seasons
Redakai: Lokar's Shadow 1 Season
A Very English Scandal 1 Season
Now Go Build with Werner Vogels 2 Seasons
Wallpaper 2 Seasons
To Dine For with Kate Sullivan 1 Season
Flower Boy Next Door 1 Season
Uchimura Summers 1 Season
Oddbods 2 Seasons
People Like Us 1 Season
Michael Jackson: The Ultimate Icon 1 Season
Ratings:
7+ 2.00%
TV-14 20.00%
13+ 22.00%
TV-Y 6.00%
ALL 14.00%
TV-NR 4.00%
18+ 6.00%
TV-Y7 4.00%
16+ 8.00%
TV-G 8.00%
TV-PG 6.00%
Average Number of Seasons: 1.62 seasons
Most Prolific Actor: Josephine Rene
Most Frequent Genre: Drama
Load Shows
Load Books
Load Recommendations
Information
Quit


Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Title Author(s)
An Introduction to Political Philosophy Jonathan Wolff
The Echo Maker Richard Powers
Midnight Jewels Jayne Ann Krentz
Nightingale's Song Kate Pennington
The Wish Giver: Three Tales of Coven Tree Bill Brittain\Andrew Glass
Pope Joan Donna Woolfolk Cross
The Lost Continent: Travels in Small Town America Bill Bryson
In Pharaoh's Army: Memories of the Lost War Tobias Wolff\Luann Walther
Easy Prey (Lucas Davenport #11) John Sandford
The Complete Poems (Poetry Library) D.H. Lawrence
Grendel: Devil by the Deed Matt Wagner\Rich Rankin\Chris Pitzer\Diana Schutz
The Worm Ouroboros E.R. Eddison\Keith Henderson
White Noise: Text and Criticism Don DeLillo\Mark Osteen
Jazz Toni Morrison
Bones of the Moon (Answered Prayers #1) Jonathan Carroll
Great Short Works of Fyodor Dostoevsky Fyodor Dostoyevsky\Ronald Hingley
The Only Dance There Is Ram Dass\Richard Alpert
Tripwire (Jack Reacher #3) Lee Child\Dick Hill
Peter Pan and Other Plays J.M. Barrie\Peter Hollindale
Undaunted Courage Stephen E. Ambrose\Barrett Whitener
A Reading Guide to Island of the Blue Dolphins Patricia McHugh
The Midwife's Tale Gretchen Moran Laskas
Holding On to the Air Suzanne Farrell\Toni Bentley
Average Page Count: 341.80 pages
Most Prolific Author: William Shakespeare
Most Prolific Publisher: Penguin Books
Load Shows
Load Books
Load Recommendations
Information
Quit


Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Type:
Movie
Title:
Director:
Actor:
Genre:
Comedy
Search
Title Director Actor Genre
Craig Shoemaker: The Lovemaster... Unzipped Mark Nasser Craig Shoemaker Comedy
Higher Ed Jean Claude LaMarre Comedy
Taking 5 Andrew Waller Alona Tal\Daniella Monet\Christy Carlson Romano Comedy
Gunde Pranayam Jangeti Sravani Rajesh OS. Sangeeth\Indu\Baburao\Meesala Vijay\Bhanu Action\Comedy\Drama
Standby Rob Burke\Ronan Burke Jessica ParÃ©\Brian Gleeson Comedy
Yakov Smirnoff: What A Country! Yakov Smirnoff Yakov Smirnoff Arts\Entertainment\and
Culture\Comedy\Special Interest
It's a Wonderful Afterlife Gurinder Chadha Sanjeev Bhaskar\Steve Morphew\Kate Magowan Comedy\Drama
David Crowe: Crooked finger Steve Wilson David Crowe Arts\Entertainment\and
Culture\Comedy\Special Interest
Load Shows
Load Books
Load Recommendations
Information
Quit


Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Type:
TV Show
Title:
Director:
Actor:
Genre:
Drama
Search
Title Director Actor Genre
Marcia Clark Investigates The first 48 Marcia Clark Drama
Alaska Off-Road Warriors Documentary\Drama\Unscripted
The Rebel Princess Zhang Ziyi\Zhou Yiwei Drama\Romance
Look Colton Haynes\Ali Corbin Drama
A Very English Scandal Hugh Grant\Ben Whishaw\Alex Jennings Drama
Flower Boy Next Door Park Shin-hye\Yoon Shi-yoon\Kim Ji-hoon Comedy\Drama\Romance
Atlanta Plastic Sidney Starr\Mike Forbs Documentary\Drama\Unscripted
The Bold and the Beautiful John McCook\Katherine Kelly Lang Drama
Junichi Jun Shison\Mina Fujii\Mieko Harada Drama\Romance
Mysteries of the Superstition Mountains Larry Hedrick\Hank Sheffer Documentary\Drama\Suspense
American Playboy Hugh Hefner\Matt Whelan Documentary\Drama\Special Interest
SpongeBob DocuPants Drama\Kids
The Twin Flower Legend Yu Xiao Tong\Guan Zhi Bin\Mao Xiao Hui\Kai Xuan Drama\Romance
Load Shows
Load Books
Load Recommendations
Information
Quit


Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Title:
Author:
William Shakespeare
Publisher:
Search
Title Author Publisher
Twelfth Night William Shakespeare\Barbara A. Mowat\Paul Werstine Simon Schuster
Julius Caesar William Shakespeare\Marvin Spevack\Marga Munkelt Cambridge University Press
Load Shows
Load Books
Load Recommendations
Information
Quit


Media Recommender
Movies
TV Shows
Books
Search Movies/TV
Search Books
Recommendations
Type:
Movie
Title:
It Might Be You
Get Recommendation
Title:
Siddhartha
Author:
Hermann Hesse
Average Rating:
4.02
ISBN:
613572742
ISBN13:
9780613572743
Language Code:
eng
Pages:
132
Rating Count:
514
Publication Date:
2/2/1993
Publisher
Turtleback Books
****
Title:
Data Structures and Abstractions with Java
Author:
Frank M. Carrano\Walter J. Savitch
Average Rating:
3.47
ISBN:
013237045X
ISBN13:
9780132370455
Language Code:
eng
Pages:
998
Rating Count:
34
Publication Date:
8/4/2006
Load Shows
Load Books
Load Recommendations
Information
Quit