In [12]:
```
class Movie:
    def __init__(self, title, genre, rating, actors, language):
        self.title = title
        self.genre = genre
        self.rating = rating
        self.actors = actors
        self.language = language

    def __str__(self):
        return f"Title: {self.title}, Genre: {self.genre}, Rating: {self.rating}, Langu
```

In [13]:
```
class MovieDatabase:
    def __init__(self):
        self.movies = []
        self.genre_index = {}
        self.title_index = {}
        self.famous_actors = {"Leonardo DiCaprio", "Christian Bale", "Hugh Jackman", "S

    def add_movie(self, movie):
        self.movies.append(movie)
        if movie.genre not in self.genre_index:
            self.genre_index[movie.genre] = []
        self.genre_index[movie.genre].append(movie)

        self.title_index[movie.title.lower()] = movie
        self.adjust_ratings(movie)

    def delete_movie(self, title):
        movie = self.title_index.pop(title.lower(), None)
        if movie:
            self.movies.remove(movie)
            self.genre_index[movie.genre].remove(movie)

    def search_by_title(self, title):
        return self.title_index.get(title.lower(), None)

    def search_by_genre(self, genre):
        return self.genre_index.get(genre, [])

    def recommend_movies(self, top_n=5):
        sorted_movies = sorted(self.movies, key=lambda x: x.rating, reverse=True)
        return sorted_movies[:top_n]

    def adjust_ratings(self, movie, famous_actor_increment=0.5, language_increment=0.3,
        if any(actor in self.famous_actors for actor in movie.actors):
            movie.rating += famous_actor_increment
        if movie.language.lower() == "english":
            movie.rating += language_increment
        if not any(actor in self.famous_actors for actor in movie.actors):
            movie.rating -= non_famous_actor_decrement
```

In [*]:
```
import tkinter as tk
from tkinter import messagebox

class MovieRecommendationSystem:
    def __init__(self, root):
        self.db = MovieDatabase()

        self.db.add_movie(Movie("Inception", "Sci-Fi", 8.8, ["Leonardo DiCaprio"], "Eng
        self.db.add_movie(Movie("The Dark Knight", "Action", 9.0, ["Christian Bale"], "
        self.db.add_movie(Movie("Interstellar", "Sci-Fi", 8.6, ["Matthew McConaughey"],
        self.db.add_movie(Movie("The Prestige", "Drama", 8.5, ["Hugh Jackman"], "Englis
        self.db.add_movie(Movie("Martian", "Sci-Fi", 8.0, ["Mat Damon"], "English"))
        self.db.add_movie(Movie("Batman", "Action", 9.0, ["Robert Pattinson"], "English
        self.db.add_movie(Movie("DDLJ", "Drama", 8.6, ["ShahRukh Khan"], "Hindi"))
        self.db.add_movie(Movie("Bad Boys", "Action", 7.1, ["Will Smith"], "English"))
        self.db.add_movie(Movie("The Godfather", "Action", 9.1, ["Al Pacino"], "English
        self.db.add_movie(Movie("The Batman", "Action", 7.9, ["Ben Afflick"], "English"
        self.db.add_movie(Movie("The Revenant", "Adventure", 8.6, ["Leonardo DiCaprio"]
        self.db.add_movie(Movie("The Wolverine", "Action", 8.5, ["Hugh Jackman"], "Engl

        self.root = root
        self.root.title("CineMatch")

        self.title_label = tk.Label(root, text="Title")
        self.title_label.pack()
        self.title_entry = tk.Entry(root)
        self.title_entry.pack()

        self.genre_label = tk.Label(root, text="Genre")
        self.genre_label.pack()
        self.genre_entry = tk.Entry(root)
        self.genre_entry.pack()

        self.rating_label = tk.Label(root, text="Rating")
        self.rating_label.pack()
        self.rating_entry = tk.Entry(root)
        self.rating_entry.pack()

        self.actors_label = tk.Label(root, text="Actors (comma separated)")
        self.actors_label.pack()
        self.actors_entry = tk.Entry(root)
        self.actors_entry.pack()

        self.language_label = tk.Label(root, text="Language")
        self.language_label.pack()
        self.language_entry = tk.Entry(root)
        self.language_entry.pack()

        self.add_button = tk.Button(root, text="Add Movie", command=self.add_movie)
        self.add_button.pack()

        self.search_label = tk.Label(root, text="Search by Title")
        self.search_label.pack()
        self.search_entry = tk.Entry(root)
        self.search_entry.pack()

        self.search_button = tk.Button(root, text="Search", command=self.search_movie)
        self.search_button.pack()

        self.genre_search_label = tk.Label(root, text="Search by Genre")
        self.genre_search_label.pack()
        self.genre_search_entry = tk.Entry(root)
        self.genre_search_entry.pack()

        self.genre_search_button = tk.Button(root, text="Search", command=self.search_g
        self.genre_search_button.pack()
```

```
        self.delete_label = tk.Label(root, text="Delete by Title")
        self.delete_label.pack()
        self.delete_entry = tk.Entry(root)
        self.delete_entry.pack()

        self.delete_button = tk.Button(root, text="Delete", command=self.delete_movie)
        self.delete_button.pack()

        self.recommend_button = tk.Button(root, text="Recommend Top 5 Movies", command=
        self.recommend_button.pack()

        self.result_text = tk.Text(root, height=10, width=50)
        self.result_text.pack()

    def add_movie(self):
        title = self.title_entry.get()
        genre = self.genre_entry.get()
        rating = float(self.rating_entry.get())
        actors = self.actors_entry.get().split(',')
        language = self.language_entry.get()
        movie = Movie(title, genre, rating, actors, language)
        self.db.add_movie(movie)
        messagebox.showinfo("Success", "Movie added successfully!")

    def search_movie(self):
        title = self.search_entry.get()
        movie = self.db.search_by_title(title)
        self.result_text.delete('1.0', tk.END)
        if movie:
            self.result_text.insert(tk.END, str(movie))
        else:
            self.result_text.insert(tk.END, "Movie not found!")

    def search_genre(self):
        genre = self.genre_search_entry.get()
        movies = self.db.search_by_genre(genre)
        self.result_text.delete('1.0', tk.END)
        if movies:
            for movie in movies:
                self.result_text.insert(tk.END, str(movie) + '\n')
        else:
            self.result_text.insert(tk.END, "No movies found in this genre!")

    def delete_movie(self):
        title = self.delete_entry.get()
        self.db.delete_movie(title)
        messagebox.showinfo("Success", "Movie deleted successfully!")

    def recommend_movies(self):
        top_movies = self.db.recommend_movies()
        self.result_text.delete('1.0', tk.END)
        for movie in top_movies:
            self.result_text.insert(tk.END, str(movie) + '\n')

if __name__ == "__main__":
    root = tk.Tk()
    app = MovieRecommendationSystem(root)
    root.mainloop()
```

In [ ]: