Department of Computer Science and Engineering,

Saveetha School of Engineering, SIMATS Thandalam, Chennai

March – 2024

“MOVIE INSIGHTS: Search, Summarize, Discover Cast Details using Fuzzywuzzy. “

## Submitted by

**Harikrishnan K (192124217)**

**Mohan Krishnan S(192224170)**

## Guided by

V.Saranya

Junior Research Fellow

Department of Neural Networks

# PROBLEM STATEMENT

This project is an interactive Python application that leverages the **TMDb API**, a popular and comprehensive database for movies and TV shows, to provide detailed information about movies. The TMDb API was chosen for its extensive collection of data, regular updates, and support for various metadata, making it ideal for retrieving accurate and up-to-date movie information. Users can input a movie name, and the application retrieves its summary, cast details, and other relevant information directly from the TMDb database. To handle spelling errors or approximate inputs, the application incorporates fuzzy matching using **FuzzyWuzzy**, a library that calculates the similarity between two strings and assigns a score. This ensures that even if users input a misspelled or incomplete movie name, the application can identify the closest match, providing a seamless and accurate user experience. The tool is designed to make movie exploration easy and accessible for everyone.

**DATASET ANALYSIS**

The application interacts with the TMDb API to retrieve data dynamically from its database. When a user inputs a movie name, the application sends a search request to the API, specifying the movie title as a query parameter. The API processes this request and returns a list of matching movie records, including details like movie IDs, titles, and release dates. The application then uses the most relevant match (determined using fuzzy matching) to extract the specific movie ID. This ID is used to make a second API call to fetch detailed information, such as the movie's summary, cast, and other metadata. The data is retrieved in JSON format, parsed by the application, and displayed to the user in a readable format. This API-driven approach ensures that the data is up-to-date and directly sourced from the TMDb database.

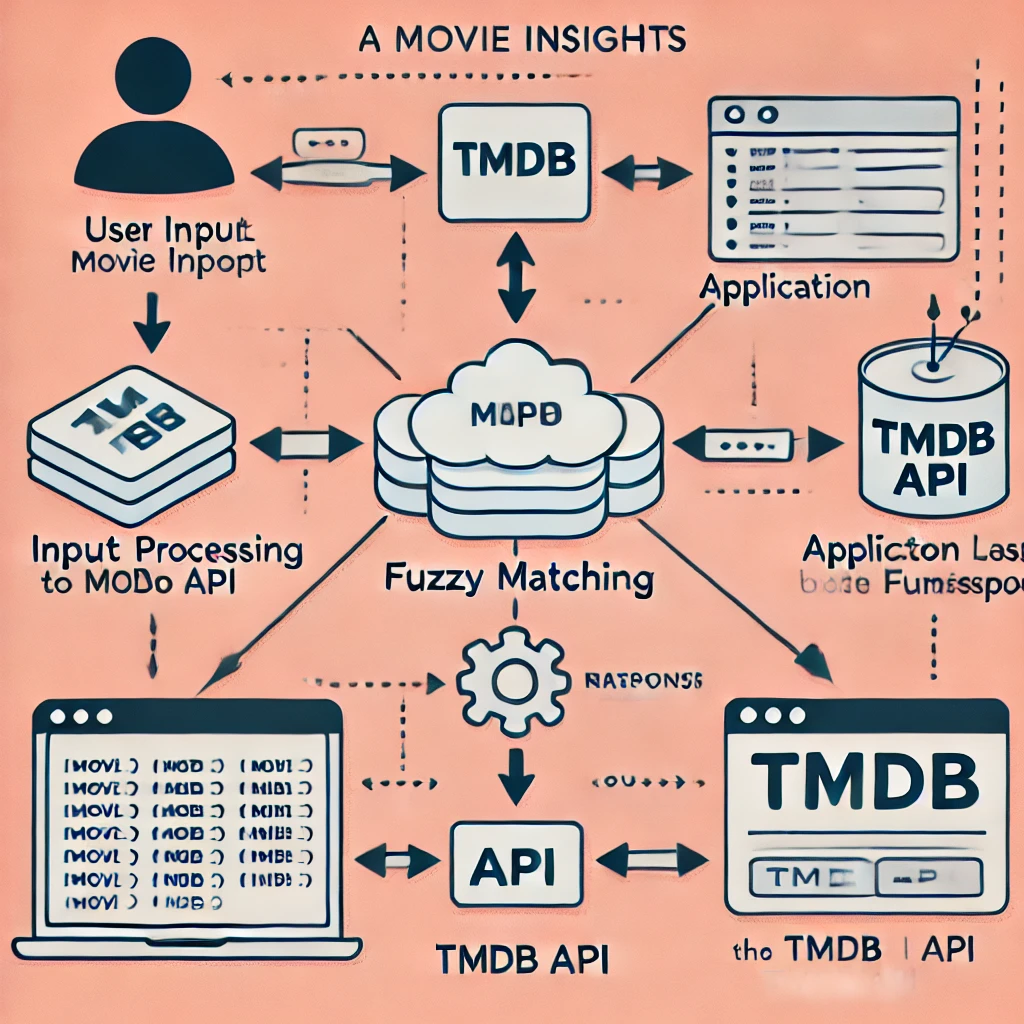
# ENVIRONMENTAL SETUP

1. **User Input:**
   * The user enters a movie name into the application through a text-based input field.
   * This input can be approximate or contain spelling errors, thanks to built-in fuzzy matching.
2. **Search Query:**
   * The application processes the input and sends a search request to the **TMDb API**.
   * TMDb responds with a list of potential matches based on the search term provided.
3. **Fuzzy Matching:**
   * The application uses **FuzzyWuzzy** to compare the user’s input with the results from the TMDb API.
   * The closest match is identified by evaluating the similarity score, ensuring that even slightly misspelled movie names are handled accurately.
4. **Fetch Movie Details:**
   * Once the best match is determined, the application retrieves detailed information about the movie using its unique **movie ID** from TMDb.
   * This includes the movie’s title, plot summary, cast list, and other relevant metadata.
5. **Display Results:**
   * The application organizes the retrieved data and presents it to the user in a structured format.
   * Key details such as the movie’s title, plot summary, and the names of the top five cast members are displayed for easy readability.
   * Additional metadata, such as release year and genre, can also be included to enhance the user experience.

**Areas of Improvement**

1. **Advanced Fuzzy Matching:**
   * Replace **FuzzyWuzzy** with a faster library like **RapidFuzz** to improve performance for larger datasets.
2. **Language Support:**
   * Add multi-language support using TMDb’s language parameter to fetch movie details in the user’s preferred language.
3. **Recommendation System:**
   * Suggest similar movies based on genres or user preferences.
4. **Rich Media Integration:**
   * Include movie posters, trailers, and metadata (e.g., release dates, genres, and ratings) for a visually enhanced experience.

**DATA FLOW DIAGRAM (OR) ARCHITECTURE DIAGRAM (OR) UML DIAGRAMS**



**RESULT ANALYSIS**

# The result analysis of the movie insights application highlights its strengths and areas for improvement. The use of fuzzy matching ensures accurate identification of movies even with misspelled or approximate input, enhancing user experience. The integration of the TMDb API allows the application to fetch real-time and reliable movie data, ensuring the results are both dynamic and up-to-date. The application is user-friendly, requiring minimal input and delivering concise outputs such as movie summaries and cast details. However, its reliance on the TMDb API makes it vulnerable to service outages, as it cannot function without API availability. The response time is generally fast but can be affected by network latency or API delays. While the application effectively retrieves text-based data, it lacks visual elements like movie posters or trailers, which could enhance user engagement. Additionally, the fuzzy matching mechanism, while useful, may occasionally struggle with edge cases if the similarity threshold is set incorrectly. To address these limitations, the application could integrate additional APIs for enriched information, incorporate visual enhancements such as posters and trailers, and provide advanced search filters for a more tailored user experience. Caching popular movie details locally could reduce dependency on live API calls, and improving scalability would ensure better performance for multiple users. Overall, the application is effective in its current state but has significant potential for enhancement.

**CODE SNIPPETS**

***import requests***

***from fuzzywuzzy import process***

***TMDB\_API\_KEY = 'a76d2f9cc87b26f475a83b8f8253356d'***

***def get\_movie\_details\_tmdb(movie\_name):***

***"""***

***Get movie details (summary, cast) using TMDb API.***

***"""***

***search\_url = f"https://api.themoviedb.org/3/search/movie?query={movie\_name}&api\_key={TMDB\_API\_KEY}"***

***search\_response = requests.get(search\_url)***

***if search\_response.status\_code == 200:***

***search\_data = search\_response.json()***

***if search\_data['results']:***

***best\_match = search\_data['results'][0]***

***movie\_id = best\_match['id']***

***title = best\_match.get('title', 'Unknown Title')***

***overview = best\_match.get('overview', 'No summary available.')***

***cast\_url = f"https://api.themoviedb.org/3/movie/{movie\_id}/credits?api\_key={TMDB\_API\_KEY}"***

***cast\_response = requests.get(cast\_url)***

***if cast\_response.status\_code == 200:***

***cast\_data = cast\_response.json()***

***cast = [member['name'] for member in cast\_data.get('cast', [])[:5]] # Top 5 cast members***

***cast\_list = ', '.join(cast) if cast else 'No cast information available.'***

***else:***

***cast\_list = 'Error fetching cast information.'***

***return f"Title: {title}\nSummary: {overview}\nCast: {cast\_list}"***

***else:***

***return "Movie not found in TMDb API."***

***else:***

***return "Error connecting to TMDb API."***

***def get\_movie\_summary(movie\_name):***

***"""***

***Use fuzzy matching to find the best movie match and then fetch details from TMDb API.***

***"""***

***search\_url = f"https://api.themoviedb.org/3/search/movie?query={movie\_name}&api\_key={TMDB\_API\_KEY}"***

***search\_response = requests.get(search\_url)***

***if search\_response.status\_code == 200:***

***search\_data = search\_response.json()***

***if search\_data['results']:***

***movie\_names = [movie['title'].strip().lower() for movie in search\_data['results']]***

***best\_match, score = process.extractOne(movie\_name.lower().strip(), movie\_names)***

***if score >= 70:***

***best\_match\_movie = next(movie for movie in search\_data['results'] if movie['title'].strip().lower() == best\_match)***

***return get\_movie\_details\_tmdb(best\_match\_movie['title'])***

***else:***

***return f"Movie not found with high confidence. Did you mean one of these?\n{', '.join(movie\_names[:5])}"***

***else:***

***return "Movie not found in TMDb API."***

***else:***

***return "Error connecting to TMDb API."***

***if \_\_name\_\_ == "\_\_main\_\_":***

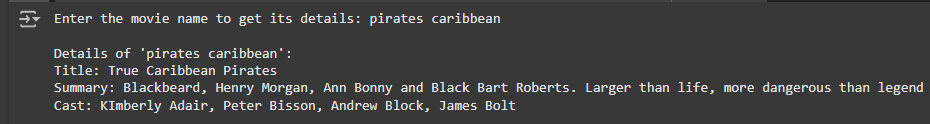
***movie\_name = input("Enter the movie name to get its details: ")***

***details = get\_movie\_summary(movie\_name)***

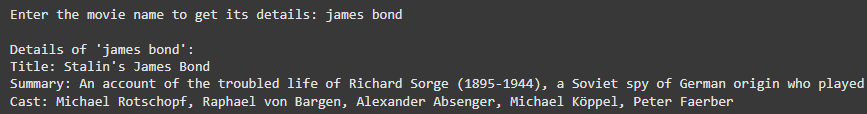
***print(f"\nDetails of '{movie\_name}':\n{details}")***

**OUTPUT SAMPLES**

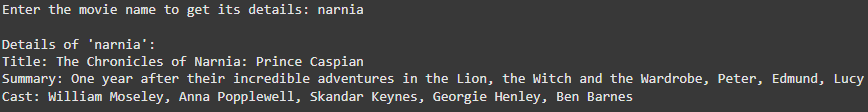
**Example 1:**

****

**Example 2:**

****

**Example 3:**

****