Gen AI Movie Recommender System - Chatbot:

1. Problem Statement / Objective:

The Objective of this Project is to build a Movie Recommender System which will provide a list of movies title to user based on the user requirements. We have a Movies Data set which contains around 5000 movies details. We will be building a Chatbot which will accept user inputs such as actor name, year of release, country, language, genre to provide user list of movies.

2. Understanding the Movies Dataset:

The movies data set contains around 5000 list of movies, and it has lot of columns, we will be using few sets of columns for our needs, the list of columns we will be using for the current project are:

```
#   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   director_name  4924 non-null   object
 1   genres         4924 non-null   object
 2   actor_name     4917 non-null   object
 3   title_year     4924 non-null   float64
 4   imdb_score     4924 non-null   float64
 5   movie_title    4924 non-null   object
 6   language       4924 non-null   object
 7   country        4923 non-null   object
```

We will be extracting details from the User input such as below to extract the movies name

- genre

- actor_name

- language

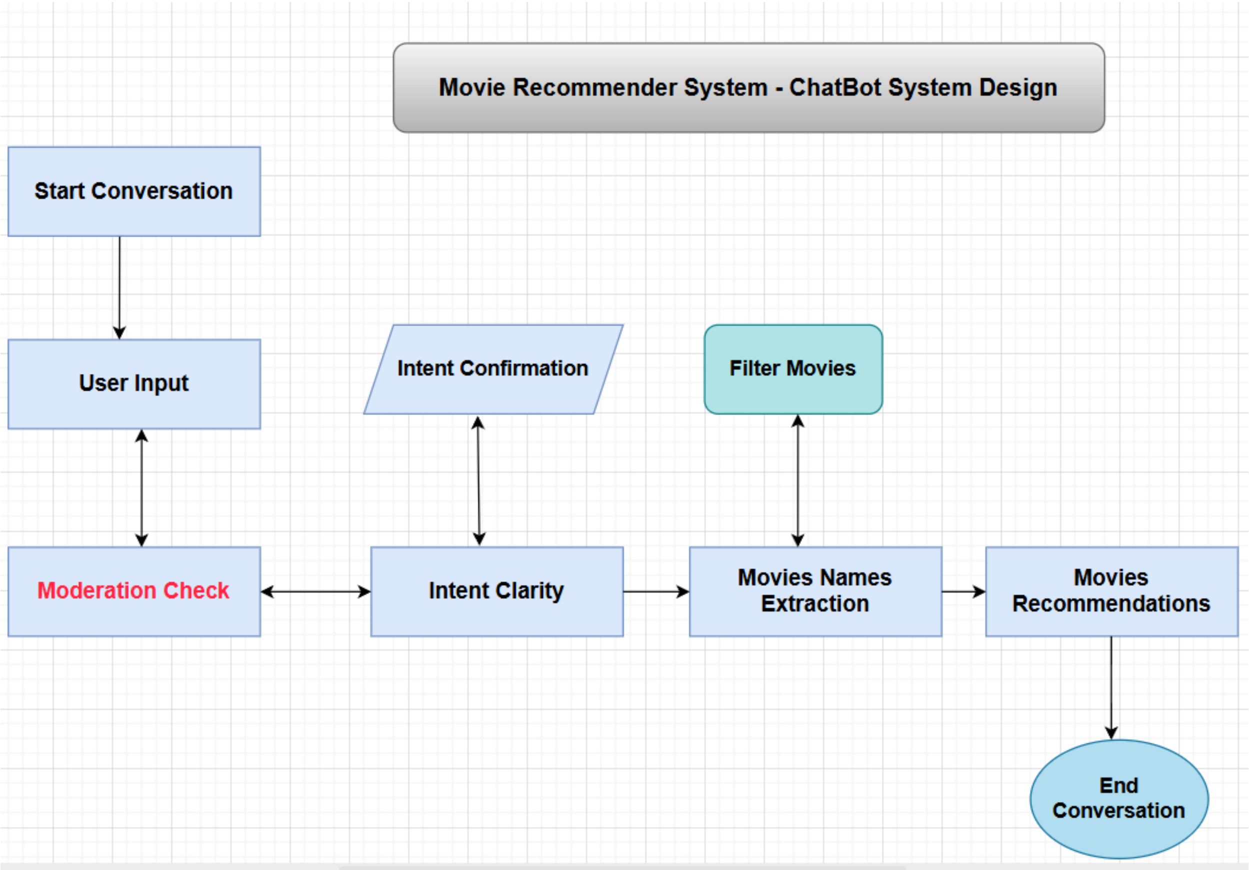
- country

- year of release

3. System Design:

For Building the Movie Recommender Chatbot , we are considering few inputs from the user to extract movie details those specific inputs will be extracted in the form of json object in the end, below are those inputs:

{'actor_Name':'values','movie_released_year':'number','genre:'values','language':'values','country':'values'}



4. Implementation:

We will be implementing our chatbot in three different stages

Stage 1

- Intent Clarity Layer
- Intent Confirmation Layer

Stage 2

- Filter Movies Layer
- Movies Extraction Layer

Stage 3

- Movies Recommendation Layer

4.1 : Stage 1

Let's start with the first part of the implementation - building the intent clarity and intent confirmation layers. As mentioned earlier, this layer helps in identifying the user requirements and passing it on to the product matching layer. Here are the functions that we would be using for building these layers:

- initialize_conversation(): This initializes the variable conversation with the system message.
- get_chat_completions(): This takes the ongoing conversation as the input and returns the response by the assistant
- moderation_check(): This checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, it ends the conversation.
- intent_confirmation_layer(): This function takes the assistant's response and evaluates if the chatbot has captured the user's profile clearly. Specifically, this checks if the following properties for the user has been captured or not such as actor_name , genre, language, country etc.
- dictionary_present(): This function checks if the final understanding of user's profile is returned by the chatbot as a python dictionary or not. If there is a dictionary, it extracts the information as a Python dictionary.

moderation_check():

This checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, you can add a break statement to end the conversation.

intent_confirmation_layer():

This function takes the assistant's response and evaluates if the chatbot has captured the user's profile clearly. Specifically, this checks if the following properties for the user has been captured or not

- actor_Name
- movie_released_year
- genre
- language
- country

Sample code for checking IntentConfirmation:

```
debug_response_assistant_2 = f"""
Great, thank you for providing all the details.
Based on your inputs, here is the final profile for the movies  you are looking for:
{{'actor_Name':'Tom Cruise',
 'movie_released_year':'2014',
 'genre':'Action',
 'language':'English',
 'country':'any'}}
"""

print(debug_response_assistant_2)
response = intent_confirmation_layer(debug_response_assistant_2)
response.get('result')
Yes
```

dictionary_present(): This function checks if the final understanding of user's profile is returned by the chatbot is a Python dictionary or not.

Sample Conversation of Stage 1:

```
debug_conversation = initialize_conversation()
debug_user_input = "Hi, I am John. I am looking for some movie
recommendations or suggestions, can you help me with that?"
debug_moderation = moderation_check(debug_user_input)
debug_conversation.append({"role": "user", "content": debug_user_input})
```

```
debug_response_assistant = get_chat_completions(debug_conversation)
debug_moderation = moderation_check(debug_response_assistant)
debug_conversation.append({"role": "assistant", "content":
debug_response_assistant})
debug_conversation.append({"role": "assistant", "content": "Yes, I am
looking for movies of Tom Cruise of Action genre of Year 2014 of english
lanaguage of country USA"})
debug_response_assistant = get_chat_completions(debug_conversation)
debug_response_assistant

##output

{'actor_Name': 'Tom Cruise', 'movie_released_year': 2014, 'genre':
'Action', 'language': 'English', 'country': 'USA'}
```

4.2 : Stage 2

In Stage 2 we will be building the Movies Extraction layer and filter layer , based on the final output we receive from Stage1

Implementing the Filter Movies and Movies Extraction Layers

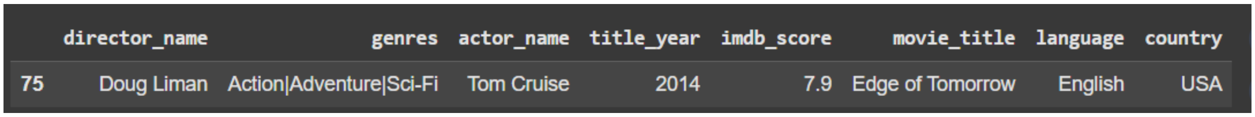
In this section, we take in the output of the previous layers, i.e. the user requirements, which is in the format of a Python dictionary.
Next we will extract the movies recommendations based on user's requirements.

filter_movies() :

This function will take the movies data frame and a movie_dict as input which is nothing but list of user requirements in the form of json dictionary , which will be used to extract movies which matches the movie_dict values and return a Dataframe as output.

```
# Example usage with response_dict_2:
filtered_movies = filter_movies(movies_details, response_dict_n)
filtered_movies
```

Output:



	director_name	genres	actor_name	title_year	imdb_score	movie_title	language	country
75	Doug Liman	Action|Adventure|Sci-Fi	Tom Cruise	2014	7.9	Edge of Tomorrow	English	USA

Movies Extraction List :

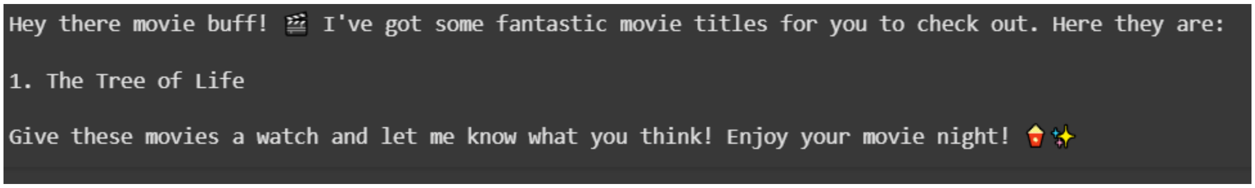
get_movie_titles : This function will return the list of movie titles it will accept a Dataframe as input.

4.3 : Stage 3

The final Stage which provide the List of movies name in chatty way,with the help of open api prompts

provide_movie_suggestions() : This function accepts list of movies and returns a well-define text to the user which contains the list of recommended movies along with some some useful custom message in chatty way.

```
# Example usage with the filtered movie titles:
chatty_response = provide_movie_suggestions(movie_titles)
print(chatty_response)
```



```
Hey there movie buff! 🎬 I've got some fantastic movie titles for you to check out. Here they are:

1. The Tree of Life

Give these movies a watch and let me know what you think! Enjoy your movie night! 🍿✨
```

5. Running All Stages Together Stage1+Stage2+Stage 3

initiate_brad_pitt_chat : This function will have all the logic to run continuous prompt with different set of checks like moderation and provide more info to get the final output

Sample Chat :

```
user: Hi, I am John looking for some movie recommendations , can you help
me with that?.
assistant: Great! As a Movie Recommender, I can provide you with great
movie recommendations. To better assist you, could you please let me know
if you have any specific genre or actor preferences for the movies you are
looking for?
user: Hi, I am looking for movies of actor Brad Pitt from the year 2001 of
Crime 'genre' of English language of 'any' country.Please provide movies
with this info only nothing else
assistant: Hey there! I'm so excited to share some awesome movie
recommendations with you. Here are a couple of fantastic titles for your
movie night:

1. Spy Game
2. Ocean's Eleven
```

```
These movies are both packed with suspense, intrigue, and a touch of
humor. I think you're in for a real treat with these choices. Enjoy
watching! 🍿🎬
```

6. Learning & Challenges:

- Need to provide proper prompting so that the LLM Can understand our set of requirements, and extract the relevant info form the user messages.

- Bad prompting can result-in improper result

-Has to provide inputs like this to get good result:

```
 example_user_dict = {'actor_Name': "Value",
                         'movie_released_year':"Value",
                           'genre': "Value",
                           'language': "Value",
                           'country': "Value",
                           }
```

- Has to provide clear prompting such as chain of thoughts and roles to the LLM so that it can verify the details

```
 You are a senior evaluator who has an eye for detail.The input text will
contain a user requirement captured through 5 keys.
    You are provided an input. You need to evaluate if the input text has
the following keys:
    {{
    'actor_Name':'values','movie_released_year':'number','genre:'values','
language':'values','country':'values'
    }}
    The 'movie_released_year' key can take only a numerical value.
    The 'country' key can take any country value of string format , also
the value 'any' is accepted for this field
    Next you need to evaluate if the keys have the the values filled
correctly.
    Only output a one-word string in JSON format at the key 'result' -
Yes/No.
    Thought 1 - Output a string 'Yes' if the values are correctly filled
for all keys, otherwise output 'No'.
    Thought 2 - If the answer is No, mention the reason in the key
'reason'.
    THought 3 - Think carefully before the answering.
```