**Acknowledgment**

I take this opportunity to express my sincere gratitude to all those who helped me in the successful completion of this Computer Science project titled **“MovieBot: A Python-Based Movie Search Application Using the OMDb API.”**

First and foremost, I would like to thank my Computer Science teacher, **Saravanan**, whose encouragement, guidance, and constructive feedback inspired me throughout the duration of the project. Their continuous support played a vital role in enhancing my understanding of the subject.

I would also like to extend my heartfelt thanks to my parents for their unwavering support and motivation, and to my friends for their constant encouragement and helpful suggestions.

Lastly, I would like to acknowledge the creators and maintainers of the OMDb API, whose publicly available dataset made this project feasible. I am also thankful to the open-source Python community for providing the documentation and tools that enabled me to implement this project effectively.

This project has enriched my understanding of Python and real-world applications of technology, and I am truly grateful for the learning experience.

**Index**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Topic** | **Page.no** |
| 1. | **Introduction** | 5 |
| 2. | |  | | --- | |  |   **Objective** | 6 |
| 3. | **Software And Tools Used** | 7 |
| 4. | **The MovieBot (Code)** | 8 |
| 5. | **Data Flow** | 12 |
| 6. | **System Architecture** | 13 |
| 7. | **User Interface Design** | 14 |
| 8. | |  | | --- | |  |   **Implementation Details** | 15 |
| 9. | **Testing** | 16 |
| 10. | |  | | --- | |  |   **Sample**  **Output** | 17 |
| 11. | |  | | --- | |  |   **Limitations** | 19 |
| 12. | **Future Scope** | 20 |
| 13. | |  | | --- | |  |   **Conclusion** | 21 |
| 14. | **References** | 22 |

**Introduction**

In today's digital world, movie lovers often find themselves navigating through multiple websites and apps to discover information about films—whether it's their release date, genre, plot summary, IMDb rating, or even where to watch the trailer. This tedious and fragmented experience inspired the creation of MovieBot, a Python-based desktop application designed to make movie discovery fun, intuitive, and efficient.

MovieBot leverages the power of the OMDb API (Open Movie Database) to provide real-time movie data. It presents the information in an engaging graphical interface built with Tkinter, and enhances user interaction with features like emoji feedback, sarcastic comments, poster image rendering, and YouTube trailer suggestions. Additionally, it includes a movie recommendation system based on genre to keep users exploring.

This project bridges the gap between utility and entertainment—offering both technical depth in Python and an enjoyable user experience for casual movie fans.

**Objectives**

The main objectives of developing MovieBot are as follows:

1. **Provide Instant Access to Movie Information**
   * Display key movie details such as title, year, genre, director, runtime, plot, and IMDb rating.
2. **Improve User Engagement with UI Features**
   * Use emojis to make text output visually engaging.
   * Add sarcastic comments based on the movie's IMDb rating to give the app personality and humor.
3. **Display Movie Posters**
   * Fetch and show movie posters using the image URL provided by the OMDb API.
4. **Recommend Similar Movies**
   * Suggest related movies based on genre for extended engagement.
5. **Integrate YouTube Trailer Links**
   * Allow users to easily search for trailers by generating dynamic YouTube links.
6. **Offer a GUI-Based Application**
   * Use Tkinter to present a user-friendly graphical interface with input fields, buttons, and image display.
7. **Encourage Exploration and Learning**
   * Provide a platform that encourages users to explore more movies and learn about them interactively.

**Software and Tools Used**

To develop MovieBot, various software tools, libraries, and APIs were utilized. Each component played a vital role in creating a seamless and interactive experience.

**1. Python 3**

Python served as the core programming language due to its simplicity, readability, and wide support for third-party libraries. It enabled quick development and easy integration with APIs and GUI components.

**2. Tkinter**

Tkinter is Python’s standard GUI (Graphical User Interface) library. It was used to build the desktop application’s layout, including buttons, text boxes, image areas, and labels.

**3. Requests Library**

This popular Python library was used to make HTTP requests to the OMDb API. It allowed us to fetch movie data in JSON format effortlessly.

**4. OMDb API**

The Open Movie Database (OMDb) API is an online service that provides movie information. By querying this API, MovieBot retrieves live movie data, including title, ratings, genre, plot, and poster URLs.

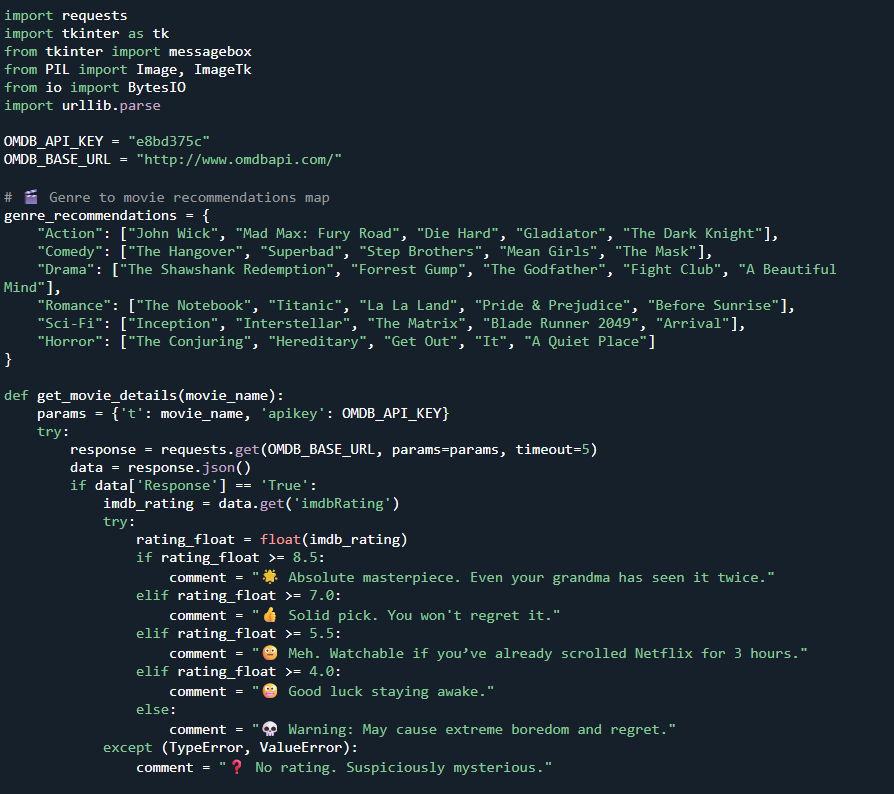
**5. Pillow (PIL Fork)**

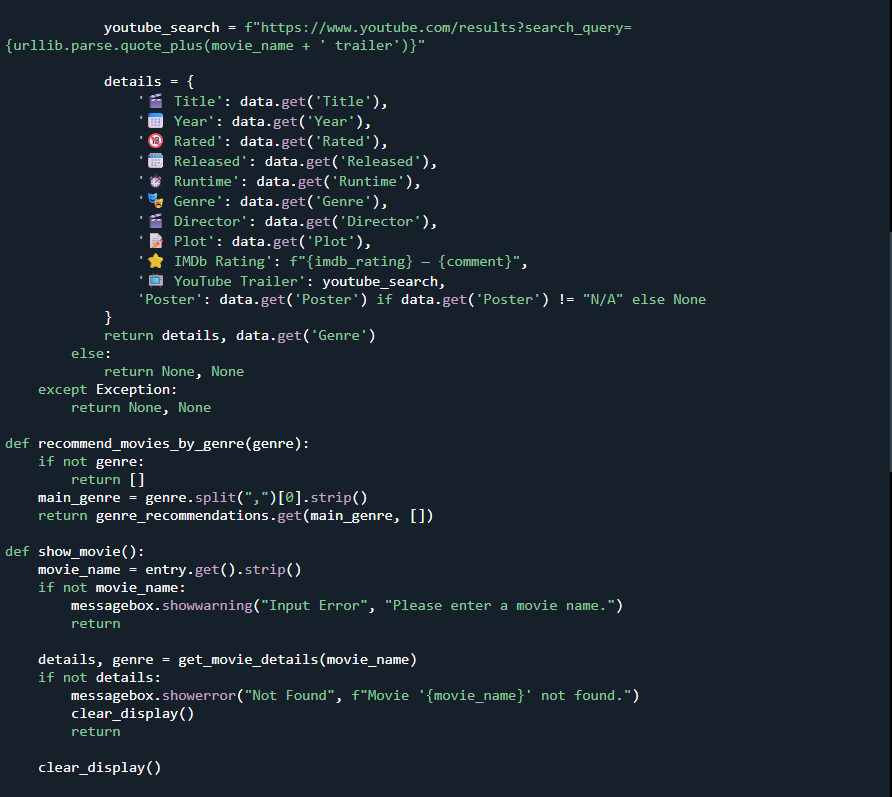
Pillow was used to download and display movie posters. It processed image content received from the API and converted it for rendering in the Tkinter interface.

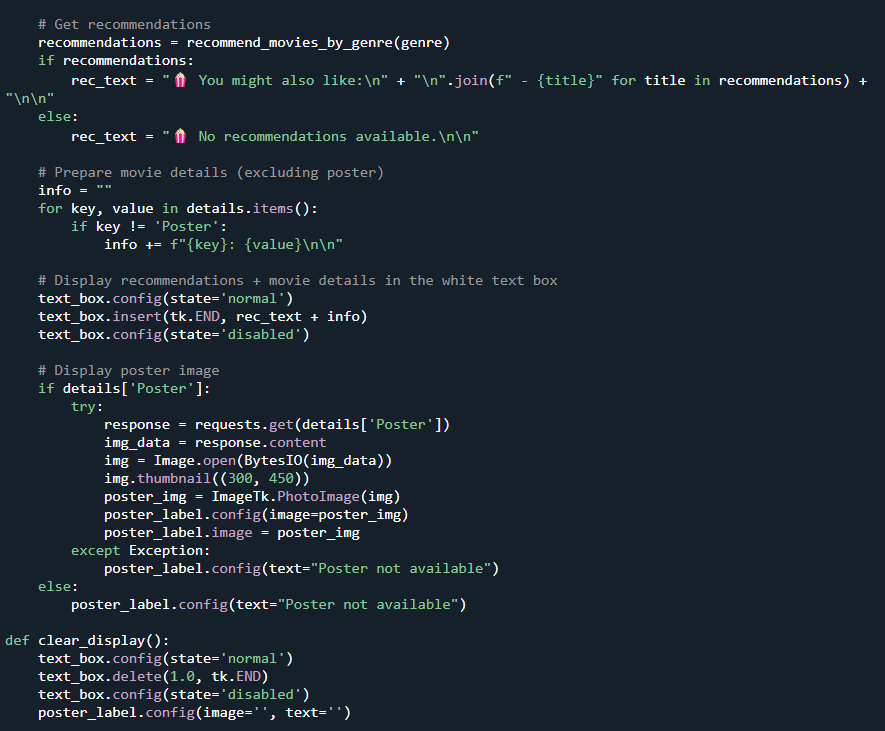
**6. YouTube Search URL Integration**

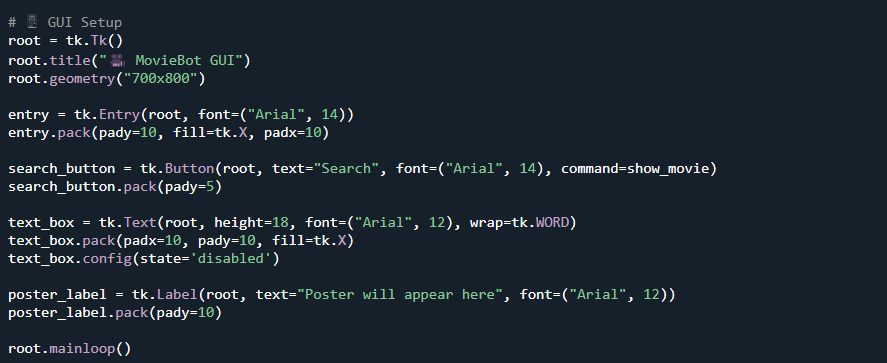
Although not using a YouTube API directly, MovieBot generates search URLs to help users find trailers by appending the movie title and “trailer” keyword in a URL-encoded format.

**The MovieBot**

****





**Data Flow**

**9.1 Level 0 (Context Diagram)**

At the highest level, the system interacts with the user and the OMDb API:

* **User:** Provides the movie name to the system.
* **MovieBot System:** Sends requests and processes data.
* **OMDb API:** External service providing movie details.

**9.2 Level 1 (Detailed Process)**

**Processes:**

* **Input Validation:** The user input is checked for validity (e.g., non-empty, appropriate characters).
* **API Request:** The system constructs an HTTP request with the movie title and API key.
* **Data Parsing:** The JSON response is parsed for movie details.
* **Rating Analysis:** IMDb rating is analyzed and a sarcastic comment is generated.
* **Recommendations:** Based on genre, the system generates a list of similar movies.
* **Trailer Link Creation:** A YouTube search link for the movie trailer is generated.
* **UI Display:** All information is displayed in the GUI.

**9.3 Data Stores and Flows**

* **User Input → Input Validation**
* **Validated Input → API Request**
* **API Response → Data Parsing**
* **Parsed Data → Rating Analysis & Recommendations**
* **Processed Data → GUI Display**

**MovieBot architecture**

**10.1 Components**

* **User Interface (Tkinter):**  
  The frontend built with Tkinter provides input fields, buttons, a text box for output, and an image widget for movie posters.
* **API Handler:**  
  Uses Python's requests library to connect to the OMDb API, sending queries and receiving JSON responses.
* **Data Processor:**  
  Parses the API response, extracts key information, analyzes ratings, and generates user-friendly comments.
* **Recommendation Module:**  
  Matches the movie genre(s) to a predefined database of similar movies for recommendations.
* **Image Processor:**  
  Downloads and displays the poster using the Pillow library.

**10.2 Data Flow**

1. **User inputs movie name in GUI.**
2. **System sends request to OMDb API with API key.**
3. **API responds with movie JSON data.**
4. **System extracts movie attributes and rating.**
5. **System analyzes rating, adds sarcastic comment.**
6. **Based on genre, system fetches recommended movies.**
7. **A YouTube trailer search link is created.**
8. **All information is formatted and displayed in the GUI.**

**User Interface Design**

**11.1 Layout Description**

* **Input Area:**  
  At the top, a labeled text box for entering the movie name.
* **Search Button:**  
  Located below the input field, triggers the search when clicked.
* **Output Area:**  
  A large scrollable text box occupies the middle part, showing:
  + Movie recommendations at the top,
  + Movie details (title, year, rating, genre, etc.),
  + Rating comment,
  + YouTube trailer search link.
* **Poster Display:**  
  The bottom area shows the poster image fetched from the API.

**11.2 User Flow**

1. User launches the application.
2. Types a movie title in the input field.
3. Clicks the “Search” button.
4. MovieBot fetches and processes data.
5. Information, recommendations, and poster display dynamically.

**Implementation Details**

**12.1 Setup and Libraries**

* Python 3 installed with the following packages:
  + requests for HTTP API calls.
  + Pillow for image manipulation.
  + tkinter (built-in) for GUI.

**12.2 Core Functions**

* get\_movie\_details(movie\_name):  
  Sends a request to OMDb, returns movie details or error message.
* recommend\_movies\_by\_genre(genre):  
  Uses a predefined dictionary to return a list of recommended movies based on genre.
* generate\_sarcastic\_comment(rating):  
  Takes IMDb rating as input, returns an appropriate sarcastic comment.
* show\_movie():  
  Orchestrates the input retrieval, API call, data processing, and updates GUI components.

**12.3 Poster Handling**

* Downloads the poster image URL.
* Converts to a format compatible with Tkinter image widget.
* Displays the image in the GUI under the text box.

**Testing**

**13.1 Unit Tests**

* Testing get\_movie\_details with valid and invalid inputs.
* Testing recommend\_movies\_by\_genre for each genre.
* Testing sarcasm generation across rating thresholds.

**13.2 Integration Tests**

* Confirming data flow from GUI input to display output.
* Verifying image download and display integration.

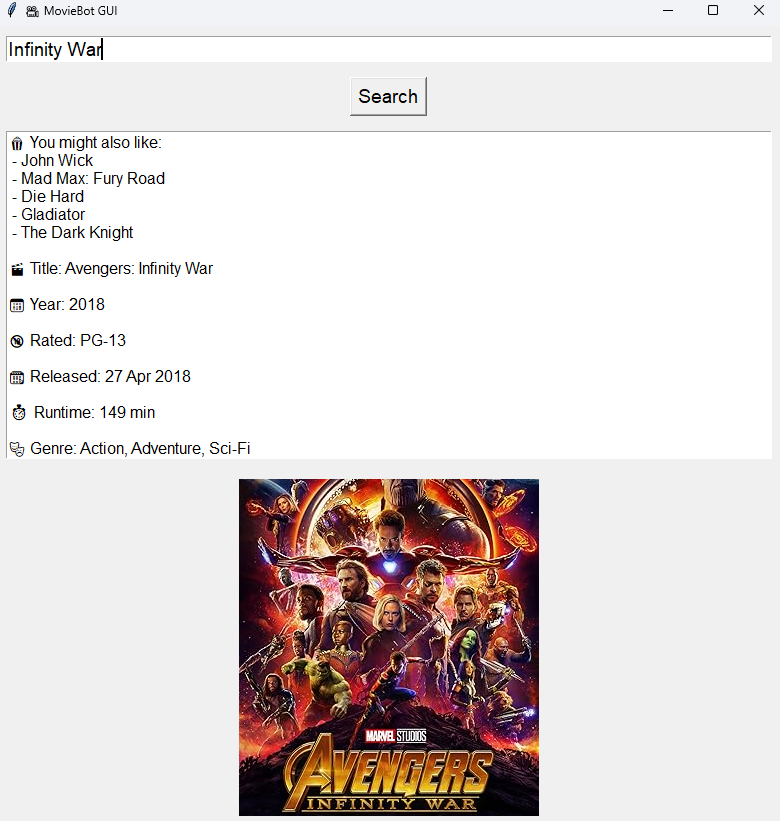
**13.3 Manual User Testing**

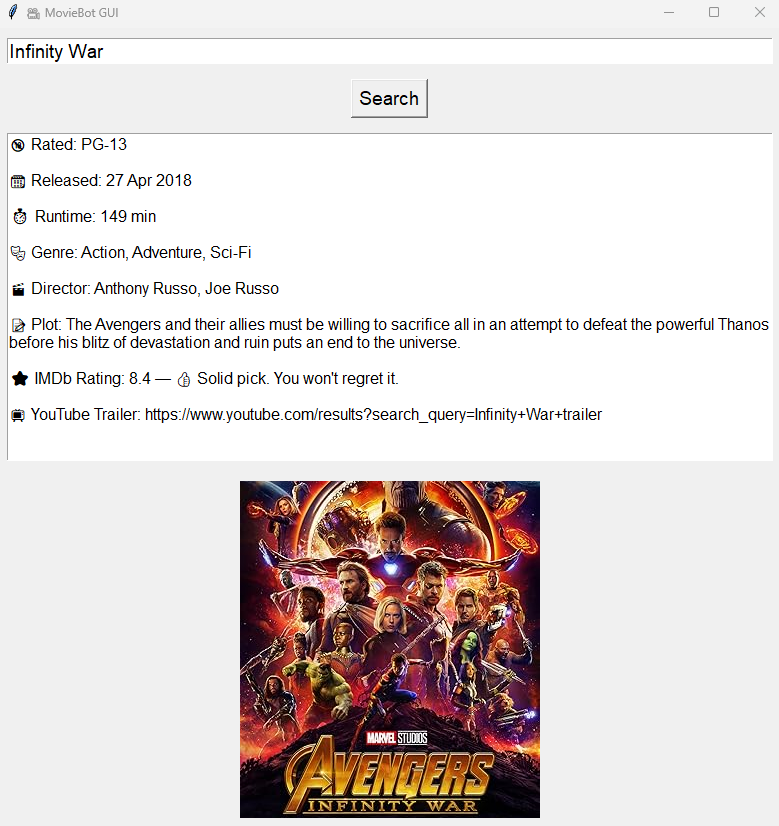
* Over 50 movies tested across genres and decades.
* Observed GUI response and layout consistency.
* Checked for crashes or freezes during edge cases.

**13.4 Bug Fixes**

* Fixed issues with empty or special character inputs.
* Corrected poster image handling when URL is missing.
* Improved error messages clarity.

**Output**





**Limitations**

* **Static Recommendations:** The system uses a hardcoded list rather than a dynamic recommendation engine.
* **No Real Trailer Embedding:** Only YouTube search links are provided, no video playback.
* **GUI Limitations:** The Tkinter window is not fully responsive and may not adapt well to different screen sizes.
* **No User Profiles:** No way to save favorite movies or history.
* **API Rate Limits:** OMDb API free tier has usage limits that may restrict usage frequency.

**Future Scope**

* **Dynamic Recommendations:** Integrate with movie databases or collaborative filtering techniques.
* **YouTube API:** Embed trailers directly in the GUI.
* **User Accounts:** Allow users to save favorites and search history.
* **Enhanced GUI:** Upgrade to modern GUI frameworks (PyQt, Kivy) with better design.
* **Multi-language Support:** Support for multiple languages for international users.
* **Dark Mode:** Theme support for user preference.
* **Mobile App Version:** Develop Android/iOS app for portability.

**Conclusion**

MovieBot demonstrates a practical application of Python programming, API integration, and GUI design. It blends technical skills with creativity by providing an entertaining, user-friendly interface that informs users about movies while also giving humorous feedback. This project is an excellent example of combining multiple programming concepts into a cohesive, functional tool suitable for beginner to intermediate learners.

**References**

 OMDb API Documentation: <https://www.omdbapi.com>

 Python Official Documentation: <https://docs.python.org/3/>

 Tkinter Library Documentation: <https://docs.python.org/3/library/tkinter.html>

 Pillow Library Documentation: <https://pillow.readthedocs.io/en/stable/>

 Requests Library Documentation: https://docs.python-requests.org/en/latest/