

To run this program, the pre-requisites are

1. Install Python
2. pip install requests

After running the program (Scenario- 1)

1. Sort by Newest Releases- This will show the movies sorted with new release date.
2. Min Year- Selected 2025
3. Max Year- Selected 2025
4. Language/Region- Hindi (Bollywood)
5. Genre- None selected (This will choose all “Genres”)

Click on “GET RECOMMENDATIONS”

The program will show the Results

The screenshot shows a Windows application window titled "Movie Recommender". The main title is "Discover Your Next Favorite Movie". The interface includes the following input fields and controls:

- Sort By:** A dropdown menu set to "Newest Releases".
- Min Year:** A slider set to 2025.
- Max Year:** A slider set to 2025.
- Language/Region:** A dropdown menu set to "Hindi (Bollywood)".
- Genres:** A list of genres with checkboxes:
 - Action
 - Adventure
 - Animation
 - Comedy
 - Crime
 - Documentary
 - Drama
 - Family
 - Fantasy
 - Horror
 - Thriller
 - Western
- GET RECOMMENDATIONS**: A large green button.

The results section displays the following text:

-- Page 1 Results --

- [HI] Tu Meri Main Tera Main Tera Tu Meri (2025) - ★ 0.0
- [HI] Durlabh Prasad Ki Dusri Shadi (2025) - ★ 0.0
- [HI] Raat Akeli Hai: The Bansal Murders (2025) - ★ 0.0
- [HI] Janki (2025) - ★ 0.0
- [HI] Apna Amitabh (2025) - ★ 0.0
- [HI] Sholay: The Final Cut (2025) - ★ 10.0
- [HI] The Great Shamsuddin Family (2025) - ★ 0.0
- [HI] Kis Kisko Pyaar Karoon 2 (2025) - ★ 0.0
- [HI] Saali Mohabbat (2025) - ★ 0.0
- [HI] Mamta Child Factory (2025) - ★ 0.0
- [HI] Dhurandhar (2025) - ★ 6.3
- [HI] Flames (2025) - ★ 0.0
- [HI] Parvati (2025) - ★ 10.0
- [HI] Susu (2025) - ★ 0.0
- [HI] Me No Pause Me Play (2025) - ★ 0.0
- [HI] Kaisi Ye Peheli (2025) - ★ 0.0
- [HI] Gustaakk Ishq (2025) - ★ 0.0
- [HI] Tere Ishq Mein (2025) - ★ 7.0
- [HI] Small Clouds (2025) - ★ 0.0
- [HI] Ghor (2025) - ★ 0.0

A blue button at the bottom right of the results area says "Get more results".

If you need more results, click on the blue button, it will show more results

The screenshot shows a Windows application window titled "Movie Recommender". The main title "Discover Your Next Favorite Movie" is centered at the top. Below it, there are several filter options:

- Sort By:** A dropdown menu set to "Newest Releases".
- Min Year:** A slider set to 2025.
- Max Year:** A slider set to 2025.
- Language/Region:** A dropdown menu set to "Hindi (Bollywood)".
- Genres:** A list of genres with checkboxes:
 - Action
 - Adventure
 - Animation
 - Comedy
 - Crime
 - Documentary
 - Drama
 - Family
 - Fantasy
 - Horror
 - Thriller
 - Western

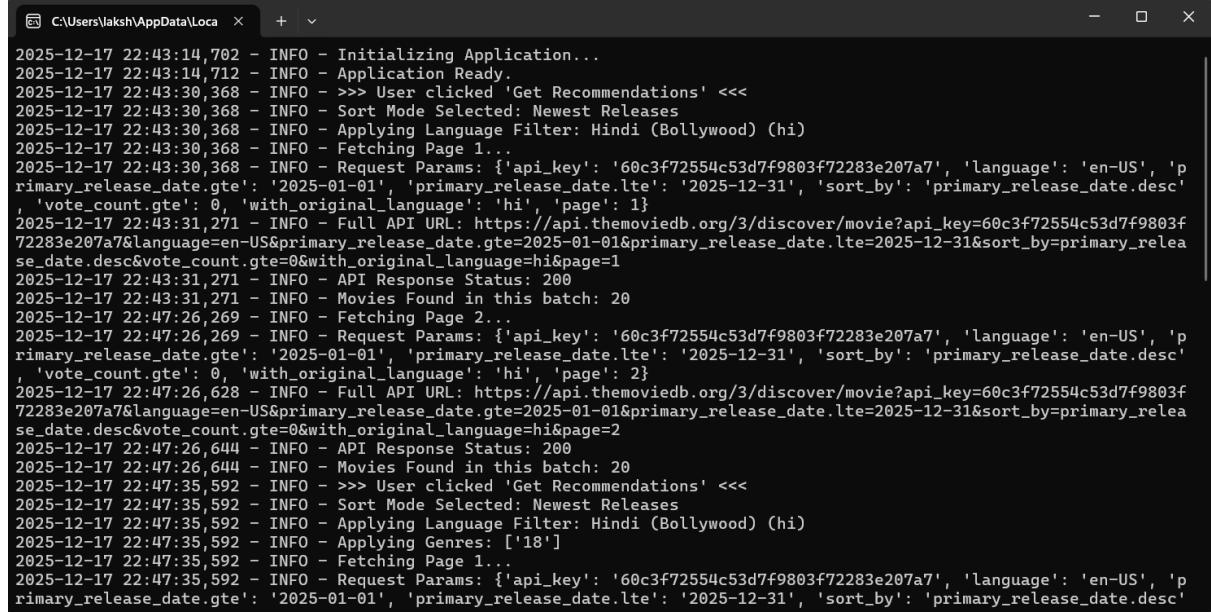
A large green button labeled "GET RECOMMENDATIONS" is centered below the filters. The main content area displays movie recommendations:

-- Page 2 Results --

- [HI] Hind Ki Chadar - Guru Ladho Re (2025) - ★ 0.0
- [HI] Mastiii 4 (2025) - ★ 6.0
- [HI] Dining with the Kapoors (2025) - ★ 1.0
- [HI] 120 Bahadur (2025) - ★ 5.0
- [HI] Hi Zindagi (2025) - ★ 0.0
- [HI] Nishaanchi 2 (2025) - ★ 8.0
- [HI] 2020 Delhi (2025) - ★ 5.0
- [HI] Yugantar (2025) - ★ 0.0
- [HI] Agra (2025) - ★ 0.0
- [HI] Kaal Trighori (2025) - ★ 0.0
- [HI] De De Pyaar De 2 (2025) - ★ 6.0
- [HI] Lalsa (2025) - ★ 0.0
- [HI] Jo's Tum (2025) - ★ 0.0
- [HI] Delhi is Choking - One Percent (2025) - ★ 0.0
- [HI] Songs of the Mist (2025) - ★ 0.0
- [HI] Sthiti (2025) - ★ 0.0
- [HI] Jassi Weds Jassi (2025) - ★ 0.0
- [HI] Whispers of a Qareen (2025) - ★ 0.0
- [HI] Baramulla (2025) - ★ 5.607
- [HI] My Amma's Tale (2025) - ★ 0.0

At the bottom of the content area is a blue button labeled "Get more results".

In the background, the cmd will show successful run



```
C:\Users\laksh\AppData\Loca + 
2025-12-17 22:43:14,702 - INFO - Initializing Application...
2025-12-17 22:43:14,712 - INFO - Application Ready.
2025-12-17 22:43:30,368 - INFO - >>> User clicked 'Get Recommendations' <<<
2025-12-17 22:43:30,368 - INFO - Sort Mode Selected: Newest Releases
2025-12-17 22:43:30,368 - INFO - Applying Language Filter: Hindi (Bollywood) (hi)
2025-12-17 22:43:30,368 - INFO - Fetching Page 1...
2025-12-17 22:43:30,368 - INFO - Request Params: {'api_key': '60c3f72554c53d7f9803f72283e207a7', 'language': 'en-US', 'primary_release_date.gte': '2025-01-01', 'primary_release_date.lte': '2025-12-31', 'sort_by': 'primary_release_date.desc', 'vote_count.gte': 0, 'with_original_language': 'hi', 'page': 1}
2025-12-17 22:43:30,368 - INFO - Full API URL: https://api.themoviedb.org/3/discover/movie?api_key=60c3f72554c53d7f9803f72283e207a7&language=en-US&primary_release_date.gte=2025-01-01&primary_release_date.lte=2025-12-31&sort_by=primary_release_date.desc&vote_count.gte=0&with_original_language=hi&page=1
2025-12-17 22:43:31,271 - INFO - API Response Status: 200
2025-12-17 22:43:31,271 - INFO - Movies Found in this batch: 20
2025-12-17 22:47:26,269 - INFO - Fetching Page 2...
2025-12-17 22:47:26,269 - INFO - Request Params: {'api_key': '60c3f72554c53d7f9803f72283e207a7', 'language': 'en-US', 'primary_release_date.gte': '2025-01-01', 'primary_release_date.lte': '2025-12-31', 'sort_by': 'primary_release_date.desc', 'vote_count.gte': 0, 'with_original_language': 'hi', 'page': 2}
2025-12-17 22:47:26,628 - INFO - Full API URL: https://api.themoviedb.org/3/discover/movie?api_key=60c3f72554c53d7f9803f72283e207a7&language=en-US&primary_release_date.gte=2025-01-01&primary_release_date.lte=2025-12-31&sort_by=primary_release_date.desc&vote_count.gte=0&with_original_language=hi&page=2
2025-12-17 22:47:26,644 - INFO - API Response Status: 200
2025-12-17 22:47:26,644 - INFO - Movies Found in this batch: 20
2025-12-17 22:47:35,592 - INFO - >>> User clicked 'Get Recommendations' <<<
2025-12-17 22:47:35,592 - INFO - Sort Mode Selected: Newest Releases
2025-12-17 22:47:35,592 - INFO - Applying Language Filter: Hindi (Bollywood) (hi)
2025-12-17 22:47:35,592 - INFO - Applying Genres: ['18']
2025-12-17 22:47:35,592 - INFO - Fetching Page 1...
2025-12-17 22:47:35,592 - INFO - Request Params: {'api_key': '60c3f72554c53d7f9803f72283e207a7', 'language': 'en-US', 'primary_release_date.gte': '2025-01-01', 'primary_release_date.lte': '2025-12-31', 'sort_by': 'primary_release_date.desc'}
```

Script Logic

Mode	What it sorts by	The "Secret Sauce" (Vote Count)
Top Rated	Quality (0-10)	Must have many votes (300+) to prove it's real.
Most Popular	Hype (Buzz)	Ignores votes, focuses on activity.
Newest	Date (Time)	Allows 0 votes (because it's brand new).
Hidden Gems	Quality (0-10)	Must have few votes (<1000) to keep it niche.

CODE OF THE SCRIPT

```
import tkinter as tk

from tkinter import scrolledtext, messagebox, ttk

import requests

from requests.adapters import HTTPAdapter

from urllib3.util.retry import Retry

import logging # --- NEW: Import Logging Module

# --- 0. DEBUGGING SETUP ---

# This configures the "cmd" output.

# level=logging.INFO means it will show everything from "INFO" warnings and errors.

logging.basicConfig(

    level=logging.INFO,

    format='%(asctime)s - %(levelname)s - %(message)s'

)

# --- 1. CONFIGURATION AND CONSTANTS ---

TMDB_API_KEY = "60c3f72554c53d7f9803f72283e207a7"

BASE_URL = "https://api.themoviedb.org/3"

# Font settings

HEADING_FONT = ("Arial", 16, "bold")

LABEL_FONT = ("Arial", 10, "bold")

UI_FONT = ("Arial", 10)

# Genre Map

GENRE_MAP = {

    "Action": 28, "Adventure": 12, "Animation": 16, "Comedy": 35,
    "Crime": 80, "Documentary": 99, "Drama": 18, "Family": 10751,
    "Fantasy": 14, "Horror": 27, "Thriller": 53, "Western": 37

}
```

```
# --- LANGUAGE GROUPS ---

LANGUAGE_GROUPS = {

    "Any Language": "",

    "English (Hollywood/Global)": "en",

    "Hindi (Bollywood)": "hi",

    "South Indian (Tamil/Telugu/Mal/Kan)": "ta|te|ml|kn",

    "European (Fr/Es/De/It)": "fr|es|de|it|sv|da",

    "East Asian (JP/KR/CN)": "ja|ko|zh|cn|th|id"
}
```

```
# --- SORT OPTIONS ---
```

```
SORT_MODES = [
    "Top Rated",
    "Most Popular",
    "Newest Releases",
    "Hidden Gems"
]
```

```
# --- 2. THE APPLICATION CLASS ---
```

```
class MovieRecommenderApp:

    def __init__(self, root):
        logging.info("Initializing Application...") # Log startup
        self.root = root
        self.root.title("Movie Recommender")
        self.root.geometry("850x850")
```

```
# -- State Variables --
```

```
    self.current_page = 1
    self.saved_search_params = {}
    self.genre_checkboxes = {}
```

```

self.lang_group_var = tk.StringVar()
self.sort_var = tk.StringVar()

# -- Setup --
self.session = self._create_network_session()
self._build_ui()
logging.info("Application Ready.")

def _create_network_session(self):
    retry_strategy = Retry(
        total=3, backoff_factor=1, status_forcelist=[429, 500, 502, 503, 504],
        allowed_methods=["HEAD", "GET", "OPTIONS"]
    )
    adapter = HTTPAdapter(max_retries=retry_strategy)
    session = requests.Session()
    session.mount("https://", adapter)
    return session

def _build_ui(self):
    # 1. Main Heading
    tk.Label(self.root, text="Discover Your Next Favorite Movie",
font=HEADING_FONT).pack(pady=15)

    # 2. Input Container
    self.input_frame = tk.Frame(self.root, padx=10, pady=5)
    self.input_frame.pack(fill="x")
    self.input_frame.columnconfigure(0, weight=0)
    self.input_frame.columnconfigure(1, weight=1)

    # 3. Add the controls
    self._add_sort_section()

```

```

    self._add_year_sliders()
    self._add_language_group_selector()
    self._add_genre_section()

# 4. Search Button (Green)

    btn = tk.Button(self.input_frame, text="GET RECOMMENDATIONS",
command=self.start_new_search,
                    bg="green", fg="white", font=LABEL_FONT)

    btn.grid(row=5, column=0, columnspan=2, pady=20, sticky="ew")

# 5. Results Area

    ttk.Separator(self.root, orient="horizontal").pack(fill="x", pady=5)

    self.results_text = scrolledtext.ScrolledText(self.root, height=15, font=UI_FONT)
    self.results_text.pack(fill="both", expand=True, padx=10, pady=10)

# 6. "Get More" Button (Blue)

    self.load_more_btn = tk.Button(self.root, text="Get more results",
command=self.fetch_next_page,
                    bg="#007bff", fg="white", font=LABEL_FONT)

# --- UI COMPONENT BUILDERS ---

def _add_sort_section(self):

    tk.Label(self.input_frame, text="Sort By:", font=LABEL_FONT).grid(row=0, column=0, sticky="e",
padx=10, pady=5)

    self.sort_combo = ttk.Combobox(self.input_frame, textvariable=self.sort_var,
values=SORT_MODES, state="readonly", font=UI_FONT)

    self.sort_combo.grid(row=0, column=1, sticky="w", padx=5)

    self.sort_combo.current(0)

def _add_year_sliders(self):

    tk.Label(self.input_frame, text="Min Year:", font=LABEL_FONT).grid(row=1, column=0,
sticky="e", padx=10, pady=5)

```

```

        self.min_year_slider = tk.Scale(self.input_frame, from_=1970, to=2025, orient="horizontal",
font=UI_FONT)

        self.min_year_slider.set(2000)

        self.min_year_slider.grid(row=1, column=1, sticky="ew")



        tk.Label(self.input_frame, text="Max Year:", font=LABEL_FONT).grid(row=2, column=0,
sticky="e", padx=10, pady=5)

        self.max_year_slider = tk.Scale(self.input_frame, from_=1970, to=2025, orient="horizontal",
font=UI_FONT)

        self.max_year_slider.set(2025)

        self.max_year_slider.grid(row=2, column=1, sticky="ew")


def _add_language_group_selector(self):

    tk.Label(self.input_frame, text="Language/Region:", font=LABEL_FONT).grid(row=3, column=0,
sticky="e", padx=10, pady=5)

    values = list(LANGUAGE_GROUPS.keys())

    self.lang_group_combo = ttk.Combobox(self.input_frame, textvariable=self.lang_group_var,
values=values, state="readonly", font=UI_FONT)

    self.lang_group_combo.grid(row=3, column=1, sticky="w", padx=5)

    self.lang_group_combo.current(0)


def _add_genre_section(self):

    tk.Label(self.input_frame, text="Genres:", font=LABEL_FONT).grid(row=4, column=0,
sticky="ne", padx=10, pady=5)

    genre_frame = tk.Frame(self.input_frame)

    genre_frame.grid(row=4, column=1, sticky="w")

    columns = 6

    for i, (genre_name, genre_id) in enumerate(sorted(GENRE_MAP.items())):

        var = tk.IntVar()

        self.genre_checkboxes[genre_name] = var

        row_pos = i // columns

        col_pos = i % columns

```

```
    tk.Checkbutton(genre_frame, text=genre_name, variable=var,  
font=UI_FONT).grid(row=row_pos, column=col_pos, sticky="w", padx=5)
```

```
# --- 3. LOGIC FUNCTIONS ---
```

```
def start_new_search(self):  
  
    logging.info("">>>> User clicked 'Get Recommendations' <<<")  
  
    self.current_page = 1  
  
    self.results_text.delete(1.0, tk.END)  
  
    self.load_more_btn.pack_forget()
```

```
# 1. Base Parameters
```

```
params = {  
  
    "api_key": TMDB_API_KEY,  
  
    "language": "en-US",  
  
    "primary_release_date.gte": f"{self.min_year_slider.get()}-01-01",  
  
    "primary_release_date.lte": f"{self.max_year_slider.get()}-12-31"  
  
}
```

```
# 2. Sort Logic
```

```
mode = self.sort_var.get()  
  
logging.info(f"Sort Mode Selected: {mode}")
```

```
if mode == "Top Rated":  
  
    params["sort_by"] = "vote_average.desc"  
  
    params["vote_count.gte"] = 300  
  
elif mode == "Most Popular":  
  
    params["sort_by"] = "popularity.desc"  
  
    params["vote_count.gte"] = 5  
  
elif mode == "Newest Releases":  
  
    params["sort_by"] = "primary_release_date.desc"
```

```

    params["vote_count.gte"] = 0

elif mode == "Hidden Gems":
    params["sort_by"] = "vote_average.desc"
    params["vote_count.gte"] = 10
    params["vote_count.lte"] = 1000

# 3. Language Group
selected_name = self.lang_group_var.get()
lang_codes = LANGUAGE_GROUPS.get(selected_name, "")
if lang_codes:
    params["with_original_language"] = lang_codes
    logging.info(f"Applying Language Filter: {selected_name} ({lang_codes})")

# 4. Genres
included_ids = [str(GENRE_MAP[name]) for name, var in self.genre_checkboxes.items() if
var.get() == 1]
if included_ids:
    params["with_genres"] = "|".join(included_ids)
    logging.info(f"Applying Genres: {included_ids}")

# 5. Save and Fetch
self.saved_search_params = params
self.fetch_next_page()

def fetch_next_page(self):
    self.saved_search_params["page"] = self.current_page

# --- DEBUG LOGGING ---
logging.info(f"Fetching Page {self.current_page}...")
logging.info(f"Request Params: {self.saved_search_params}")

```

```
try:
    url = f"{BASE_URL}/discover/movie"

    response = self.session.get(url, params=self.saved_search_params, timeout=10)

    # Log the actual URL being triggered
    logging.info(f"Full API URL: {response.url}")

    response.raise_for_status()

    data = response.json().get('results', [])
    logging.info(f"API Response Status: {response.status_code}")
    logging.info(f"Movies Found in this batch: {len(data)}")

if not data:
    if self.current_page == 1:
        logging.warning("No results found for these filters.")
        self.results_text.insert(tk.END, "No movies found. Try relaxing your filters.\n")
    else:
        logging.info("End of results reached.")
        self.results_text.insert(tk.END, "\n--- No more results available ---\n")
        self.load_more_btn.pack_forget()
else:
    self.results_text.insert(tk.END, f"\n--- Page {self.current_page} Results ---\n")

for movie in data:
    title = movie.get('title', 'Unknown')
    date = movie.get('release_date', '????')[::-1]
    rating = movie.get('vote_average', 'N/A')
    lang = movie.get('original_language', '??').upper()

    self.results_text.insert(tk.END, f"• [{lang}] {title} ({date}) - ★ {rating}\n")
```

```
        self.results_text.see(tk.END)

        self.current_page += 1

        self.load_more_btn.pack(pady=10)

except Exception as e:

    # This prints the FULL error trace to your cmd

    logging.error("CRITICAL ERROR occurred during fetch:", exc_info=True)

    messagebox.showerror("Error", f"Failed to fetch data:\n{e}")

# --- 4. MAIN EXECUTION ---

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