```python
import customtkinter as ctk
from PIL import Image, ImageTk
import os
import itertools

ctk.set_appearance_mode("dark")
ctk.set_default_color_theme("dark-blue")

APP_WIDTH = 950
APP_HEIGHT = 560

# --- In-memory user ratings (movie_clean_title -> stars 1..5) ---
user_ratings = {}

# --- Movie data (50+ movies) ---
movies = {
    "Action": {
        "Avengers ⭐8.4": ["Iron Man", "Thor", "Captain America"],
        "KGF ⭐8.1": ["KGF 2", "Pushpa", "Vikram"],
        "Leo ⭐7.2": ["Kaithi", "Vikram", "Master"],
        "John Wick ⭐8.0": ["John Wick 2", "Nobody", "Equalizer"],
        "Mad Max Fury Road ⭐8.1": ["Dune", "Blade Runner 2049"],
    },
    "Romance": {
        "Titanic ⭐8.2": ["The Notebook", "La La Land"],
        "Sita Ramam ⭐8.6": ["96", "Hridayam"],
        "La La Land ⭐8.0": ["Whiplash", "A Star is Born"],
        "Notebook ⭐7.9": ["Dear John"],
        "Me Before You ⭐7.4": ["Fault in Our Stars"],
    },
    "Horror": {
        "Conjuring ⭐7.5": ["Annabelle", "Insidious"],
        "IT ⭐7.3": ["Conjuring", "Nun"],
        "Nun ⭐6.1": ["Annabelle", "Lights Out"],
        "Insidious ⭐6.9": ["Sinister"],
        "Annabelle ⭐6.4": ["Conjuring 2"],
    },
    "Sci-Fi": {
        "Interstellar ⭐8.7": ["Inception", "Martian"],
        "Inception ⭐8.8": ["Tenet", "Shutter Island"],
        "Avatar ⭐7.8": ["Avatar 2"],
    "Matrix ⭐8.7": ["Matrix Reloaded"],
    "Jurassic Park ⭐8.1": ["Jurassic World"],
},
"Comedy": {
    "Hangover ⭐7.7": ["21 Jump Street"],
    "Mr Bean ⭐8.5": ["Johnny English"],
    "Deadpool ⭐8.0": ["Deadpool 2"],
    "Jumanji ⭐7.0": ["Jumanji 2"],
    "Minions ⭐6.4": ["Despicable Me"],
},
"Tamil": {
    "Vikram ⭐8.7": ["Kaithi", "Rolex", "Leo"],
    "Master ⭐7.8": ["Maanagaram"],
    "Kaithi ⭐8.5": ["Vikram"],
    "Theri ⭐7.3": ["Mersal"],
    "Goat ⭐?": ["Leo"],
},
"Telugu": {
    "RRR ⭐8.0": ["Baahubali"],
    "Baahubali ⭐8.2": ["Baahubali 2"],
    "Pushpa ⭐7.6": ["KGF"],
    "Adhurs ⭐7.2": ["Kick"],
    "Eega ⭐7.7": ["Magadheera"],
},
"Anime": {
    "Your Name ⭐8.4": ["Weathering With You"],
    "AOT ⭐9.1": ["AOT S2"],
    "Naruto ⭐8.4": ["Naruto Shippuden"],
    "Demon Slayer ⭐8.7": ["Mugen Train"],
    "JJK ⭐8.6": ["JJK 0"],
},
"Kids": {
    "Frozen ⭐7.4": ["Frozen 2"],
    "Moana ⭐7.6": ["Encanto"],
    "Cars ⭐7.1": ["Cars 2"],
    "Toy Story ⭐8.3": ["Toy Story 2"],
    "Lion King ⭐8.5": ["Lion King remake"],
},
"Thriller": {
    "Shutter Island ⭐8.2": ["Inception"],
    "Gone Girl ⭐8.1": ["Girl on Train"],
        "Oldboy ⭐8.4": ["I Saw the Devil"],
        "Joker ⭐8.4": ["Batman"],
        "Fight Club ⭐8.8": ["Seven"],
    }
}

# --- Helpers to extract clean title and intrinsic rating from keys like "Avengers ⭐8.4" ---
def clean_title(key):
    if "⭐" in key:
        return key.split(" ⭐")[0].strip()
    return key

def intrinsic_rating(key):
    try:
        if "⭐" in key:
            return float(key.split("⭐")[1])
    except:
        pass
    return 0.0

# --- Build a flattened catalog for quick lookups: title -> (category, intrinsic_rating, full_key) ---
catalog = {}
for cat, group in movies.items():
    for key in group.keys():
        t = clean_title(key)
        catalog[t] = {"category": cat, "intrinsic": intrinsic_rating(key), "full_key": key}

# --- Poster loader (optional) ---
def load_poster(movie_name):
    try:
        cleaned = movie_name.split(" ⭐")[0]
        path = f"posters/{cleaned.lower().replace(' ', '_')}.jpg"
        if os.path.exists(path):
            img = Image×open(path)
            img = img.resize((130, 175))
            return ImageTk.PhotoImage(img)
    except:
        pass
    return None

# --- Recommendation engine (Option A: user-based preferences) ---
```

```
def get_user_preferences():
    """
    Returns dict: {category: average_user_rating_in_that_category}
    """
    sums = {}
    counts = {}
    for rated_title, stars in user_ratings.items():
        meta = catalog×get(rated_title)
        if not meta:
            continue
        cat = meta["category"]
        sums[cat] = sums.get(cat, 0.0) + stars
        counts[cat] = counts.get(cat, 0) + 1
    prefs = {}
    for cat in sums:
        prefs[cat] = sums[cat] / counts[cat]
    return prefs

def get_personal_recs(limit=10):
    """
    Generates personal recommendations:
    - Prioritize categories the user rated highly
    - Within a category, sort by intrinsic rating (descending)
    - Exclude movies the user already rated
    """
    if not user_ratings:
        # no ratings yet — fallback: top intrinsic movies across catalog
        all_movies = []
        for t,meta in catalog.items():
            all_movies.append((meta["intrinsic"], t, meta["category"]))
        all_movies×sort(reverse=True)
        return [t for _,t,_ in all_movies][:limit]

    prefs = get_user_preferences()  # category -> avg stars by user
    # Score every candidate movie:
    candidates = []
    for t, meta in catalog.items():
        if t in user_ratings:  # skip already rated by user
            continue
        base = meta["intrinsic"]  # use intrinsic rating as base
        # add preference boost if category in prefs
        boost = prefs×get(meta["category"], 0)
        # final score: weighted (higher prefs and intrinsic better)
        score = base × 0.7 + boost × 0.6
        candidates.append((score, t, meta["category"], meta["intrinsic"]))
    candidates×sort(reverse=True, key=lambda x: x[0])
    return [t for _,t,_,_ in candidates][:limit]

# --- Movie detail popup (no trailer) ---
def show_movie_details(movie_full_key):
    top = ctk×CTkToplevel(app)
    top.title(movie_full_key)
    top.geometry("450x520")

    cleaned = clean_title(movie_full_key)
    poster = load_poster(movie_full_key)

    if poster:
        img_label = ctk×CTkLabel(top, image=poster, text="")
        img_label×image = poster
        img_label×pack(pady=10)

    ctk.CTkLabel(top, text=movie_full_key, font=("Poppins", 23, "bold"))×pack(pady=5)

    # --- USER RATING UI ---
    ctk.CTkLabel(top, text="Your Rating:", font=("Poppins", 16))×pack(pady=8)
    rating_frame = ctk.CTkFrame(top, fg_color="transparent")
    rating_frame.pack()

    def give_rating(stars):
        user_ratings[cleaned] = stars
        update_stars()
        # optional: update quick suggestions panel
        refresh_quick_recs()

    def update_stars():
        for i, b in enumerate(star_buttons):
            if i < user_ratings.get(cleaned, 0):
                b.configure(text="⭐")
            else:
                b.configure(text="☆")
    star_buttons = []
    for i in range(5):
        btn = ctk×CTkButton(
            rating_frame, text="☆",
            width=40, fg_color="transparent",
            hover_color="#003f4f",
            command=lambda s=i+1: give_rating(s)
        )
        btn×grid(row=0, column=i, padx=5)
        star_buttons.append(btn)

    update_stars()

    # --- Similar movies (category-based) ---
    # find recommendations stored in movies dict (by full key)
    recs = []
    for cat, group in movies.items():
        if movie_full_key in group:
            recs = group[movie_full_key]
            break

    ctk.CTkLabel(top, text="Similar Movies:", font=("Poppins", 16))×pack(pady=12)
    ctk.CTkLabel(top, text="\n".join(recs) if recs else "No similar movies found",
                 font=("Poppins", 14)).pack()

# --- UI: load movies for a category into content frame ---
def load_category(category):
    for widget in content_frame.winfo_children():
        widget.destroy()

    row = col = 0
    for movie_key in movies[category]:
        poster = load_poster(movie_key)

        frame = ctk×CTkFrame(content_frame, corner_radius=12)
        frame×grid(row=row, column=col, padx=12, pady=12)

        if poster:
            img_label = ctk×CTkLabel(frame, image=poster, text="")
            img_label×image = poster
            img_label×pack(pady=6)
        else:
            # simple placeholder box
            ph = ctk×CTkLabel(frame, text=clean_title(movie_key), width=18, height=10,
fg_color="#222")
            ph×pack(pady=6)

        ctk.CTkButton(
            frame, text=movie_key, font=("Poppins", 12),
            command=lambda k=movie_key: show_movie_details(k)
        )×pack(pady=6)

        col += 1
        if col == 4:
            col = 0
            row += 1

# --- Search ---
def search_movie():
    q = search_entry.get().strip().lower()
    if not q:
        ctk.CTkMessagebox(title="Empty", message="Type a movie name or part of it.")
        return
    # find first match by startswith or contains
    for cat in movies:
        for key in movies[cat]:
            t = clean_title(key).lower()
            if t.startswith(q) or q in t:
                show_movie_details(key)
                return
    ctk.CTkMessagebox(title="Not Found", message="Movie not available.")

# --- Gradient animation (fast) ---
def animate_bg():
    colors = itertools×cycle(["#002244", "#004c66", "#008c99", "#00b8c6", "#00e6ff"])
    def shift():
        app.configure(fg_color=next(colors))
        app.after(150, shift)
    shift()

# --- Quick recommendations panel refresh ---
def refresh_quick_recs():
    # update the right panel quick recs (small list)
    recs = get_personal_recs(limit=6)
    quick_box.configure(state="normal")
    quick_box.delete("0.0", "end")
    if recs:
        quick_box.insert("end", "Recommended For You:\n\n")
        for r in recs:
            # show where it comes from (category)
            cat = catalog.get(r, {}).get("category", "Unknown")
            quick_box.insert("end", f"• {r}  ({cat})\n")
    else:
        quick_box.insert("end", "No recommendations yet. Rate movies to get suggestions.")
    quick_box.configure(state="disabled")

# --- Full "Recommended For You" screen ---
def show_recommended_screen():
    recs = get_personal_recs(limit=20)
    for w in content_frame.winfo_children():
        w.destroy()
    header = ctk×CTkLabel(content_frame, text="Recommended For You",
font=ctk.CTkFont(size=16, weight="bold"))
    header×grid(row=0, column=0, columnspan=4, sticky="w", padx=8, pady=(6,8))
    row = 1; col = 0
    for r in recs:
        # find full key from catalog
        fk = catalog[r]["full_key"]
        poster = load_poster(fk)
        frame = ctk×CTkFrame(content_frame, corner_radius=12)
        frame×grid(row=row, column=col, padx=12, pady=12)
        if poster:
            lbl = ctk×CTkLabel(frame, image=poster, text="")
            lbl×image = poster
            lbl×pack(pady=6)
        ctk.CTkButton(frame, text=fk, width=200, command=lambda k=fk:
show_movie_details(k))×pack(pady=6)
        col += 1
        if col == 4:
            col = 0; row += 1

# --- Build flat catalog for lookup (already built earlier) ---
catalog = {}
```

```
for cat, group in movies.items():
    for key in group.keys():
        t = clean_title(key)
        catalog[t] = {"category": cat, "intrinsic": intrinsic_rating(key), "full_key": key}

# --- UI Setup ---
app = ctk×CTk()
app.title("MovieVerse Premium")
app.geometry(f"{APP_WIDTH}x{APP_HEIGHT}")

animate_bg()

# Sidebar
sidebar = ctk×CTkFrame(app, width=170, corner_radius=12)
sidebar×pack(side="left", fill="y", padx=10, pady=10)

ctk.CTkLabel(sidebar, text="🎬 MovieVerse", font=("Poppins", 25, "bold"))×pack(pady=12)

# Category buttons
for category in movies.keys():
    ctk.CTkButton(
        sidebar, text=category, width=150, font=("Poppins", 13),
        command=lambda c=category: load_category(c)
    )×pack(pady=6)

# Recommended for you button (AI-style suggestions)
ctk.CTkButton(sidebar, text="Recommended For You", width=150, fg_color="#1db954",
              command=show_recommended_screen)×pack(pady=(12,8))

# Search entry
search_entry = ctk×CTkEntry(sidebar, placeholder_text="Search (type part)...", width=150)
search_entry×pack(pady=(8,6))
ctk.CTkButton(sidebar, text="Search", width=150, command=search_movie)×pack(pady=(0,12))

# Center content (scrollable)
content_frame = ctk×CTkScrollableFrame(app, fg_color="#0f0f0f", corner_radius=12)
content_frame×pack(side="right", fill="both", expand=True, padx=10, pady=10)

# Right quick panel
right = ctk×CTkFrame(app, width=220, corner_radius=12, fg_color="#0f0f0f")
right×place(relx=0.72, rely=0.03, relwidth=0.25, relheight=0.34)
```

```
ctk.CTkLabel(right, text="Quick Picks", font=ctk.CTkFont(size=14,
weight="bold"))×pack(pady=(8,6))
quick_box = ctk×CTkTextbox(right, width=200, height=260)
quick_box×pack(padx=8)
quick_box×configure(state="disabled")

# initial load
load_category("Action")
refresh_quick_recs()

app.mainloop()
```