Project Title: Video Recommendation Engine

Overview:
This project implements a personalized video recommendation system combining user interaction graphs and deep learning-based semantic matching. The goal is to serve a feed of videos relevant to a user based on past interactions and inferred content preferences, including cold-start handling using metadata.

Technology Stack:

- **Backend:** FastAPI
- **Database:** MongoDB
- **Recommendation Techniques:**
    - Graph-based collaborative filtering using NetworkX
    - Deep learning with `SentenceTransformer` (BAAI/bge-base-en-v1.5) for semantic similarity
- **Deployment/Test Tools:** Postman for API testing
- **Others:** NumPy, PyTorch, Pickle, re (regex)

Components:

1. **User-Post Graph Construction**
    - Parses interactions (likes, views, inspiration, ratings) from MongoDB.
    - Assigns weights to interactions and constructs a user-user graph based on shared post interactions.
    - Serializes the graph to a `.pkl` file for reuse.
2. **Graph-based Recommendations**
    - Given a user ID, finds top-k similar users from the graph based on edge weights.
    - Collects unseen posts liked/viewed by similar users and ranks them by frequency.
3. **Semantic Ranking with Deep Learning**
    - Uses `SentenceTransformer` to encode a textual project query and flatten metadata from posts.
    - Computes cosine similarity between the query and post embeddings.
    - Ranks posts semantically relevant to the user's context or mood.
4. **API Endpoint (`/feed`)**
    - Accepts `username` and optional `project_code` as query parameters.
    - Resolves username to `user_id` from MongoDB.

    - Returns a JSON list of recommended posts, optionally filtered by semantic relevance.

5. **Metadata and Post Fetching**
    - Retrieves and returns selected metadata fields like `id`, `title`, `video_link`, `username`, and `upvote_count`.
    - This list can be extended or customized as needed.

Design Philosophy:

- Each module independently connects to MongoDB to allow standalone use and simplify testing.
- The system blends collaborative filtering (for personalization) with deep models (for cold-start/content awareness).
- Modular and pluggable: developers can change metadata fields, similarity models, or graph parameters without major rewrites.

Usage Example:

- Send a GET request to `/feed?user_name=alice&project_code=cinematicstorytelling`
- The system returns a ranked list of post metadata based on Alice's user graph and the semantic meaning of "cinematic storytelling".

Conclusion:
This hybrid recommendation engine leverages both behavioral patterns and content semantics to improve the relevance of video suggestions. It is scalable, explainable, and adaptable for various creative or media-heavy platforms.