### ****Revised Scenarios with Limited Data****

#### ****1. New User (No History)****

* **Problem**: No user-item interaction data is available for the user.
* **Solution**:
  + Recommend **globally popular movies** (highest average ratings).
  + Use **Item-Item Collaborative Filtering**: If the user watches one movie, recommend movies similar to it based on shared user ratings (e.g., cosine similarity or Pearson correlation).

#### ****2. Returning User (History Exists)****

##### **a) Home Page Recommendations**

For a user with historic ratings, recommendations can combine:

1. **Similar Movies to Rated Movies (Item-Item Collaborative Filtering)**:
   * Fetch movies similar to the user's top-rated movies based on cosine similarity of ratings.
   * Fetch **N similar movies** for each of the user's top 5 movies (or highest-rated movies).
2. **Similar Users' Favorites (User-User Collaborative Filtering)**:
   * Identify the top 5 users most similar to the current user (based on cosine similarity of rating vectors).
   * Recommend movies that these users rated highly but the current user hasn’t seen.
3. **Predicted Ratings (Matrix Factorization using SVD)**:
   * Predict ratings for all unseen movies using SVD.
   * Recommend the top N movies with the highest predicted ratings.

**Blending**:

* + Combine these recommendations using a weighted strategy, e.g.:
    - 40% similar movies (item-item).
    - 30% similar users' favorites (user-user).
    - 30% SVD predictions.

##### **b) User Watches a New Movie**

* **Recommendations**:
  + Fetch movies similar to the newly watched movie using **Item-Item Collaborative Filtering**. ( 0.7 )
  + Use the **SVD matrix** to fetch also some movies are highly rated (predicted ratings > 4).(0.2)

### ****Steps for Implementation****

#### ****1. Precompute Similarity Matrices****

* **Item-Item Similarity Matrix**:
  + Compute cosine similarity between columns of the user\_movie\_matrix.
  + Store the most similar items for quick retrieval.
* **User-User Similarity Matrix**:
  + Compute cosine similarity between rows of the user\_movie\_matrix.
  + Store the most similar users for quick retrieval.

#### ****2. Precompute SVD Components****

* Decompose the user-item matrix into latent features:

R=UΣVTR = U \Sigma V^TR=UΣVT

* + UUU: User latent features.
  + Σ\SigmaΣ: Singular values (diagonal matrix).
  + VTV^TVT: Item latent features.
* Use this to predict ratings for unseen movies:

R^=UΣVT\hat{R} = U \Sigma V^TR^=UΣVT

#### ****3. Functions for Recommendations****

* **Global Popular Movies**: Identify movies with the highest average ratings.
* **Item-Item Recommendations**: Fetch similar movies using the item similarity matrix.
* **User-User Recommendations**: Fetch movies liked by similar users.
* **SVD Predictions**: Predict ratings for unseen movies and rank them.

### ****Challenges and Opportunities****

* **Cold Start Problem**:
  + New users and items cannot be handled without external metadata (e.g., movie genres).
  + Consider integrating auxiliary data in the future for better cold-start handling.
* **Diversity**:
  + Ensure recommendations are not overly similar by including randomness or exploring diverse genres (even if not explicitly present in the data).
* **Dynamic Updates**:
  + Ratings data grows, so precomputed matrices should be periodically updated.