Evaluation Report: Book Recommendation System

1. Data and Experimental Setup
    - **Data source:** MongoDB "books" collection (includes book_id, max_genre, and similar_books fields) and wishlists collection.
    - **Evaluation harness:** A Node.js script (evaluate.js) connecting to MongoDB and the /api/recommendations endpoint.
    - **Synthetic users:** 40 total
        - 20 single-genre users (5 wishlist books drawn from one genre)
        - 20 mixed-genre users (5 wishlist books drawn from three genres)

2. Recommendation Logic
    - **Wishlist reset:** For each user, insert exactly their 5 synthetic wishlist entries into the wishlists collection.
    - **Candidate generation:** The API fetches all books in the user's wishlist and builds a candidateWeights map by parsing each book's similar_books array:
        - Weight +2 per similar-book link from wishlist items.
        - Weight +1 per similar-book link from explicit interactions (none in this synthetic test).
    - **Filtering & ranking:** Remove any candidate that matches a wishlist book, then sort candidates by descending weight.
    - **Top-K output:** Return the top 10 book documents via an aggregation pipeline.

3. Evaluation Metrics

    Let K = 10 and Rel = size of the relevant set.
    - HitRate@K: proportion of users for whom at least one of the top-K recommendations appears in the relevant set.

      Why it matters: measures whether the system can place any relevant item in the visible portion of the list.
    - Precision@K: (number of recommendations in the relevant set among the top K) ÷ K, averaged over all users.

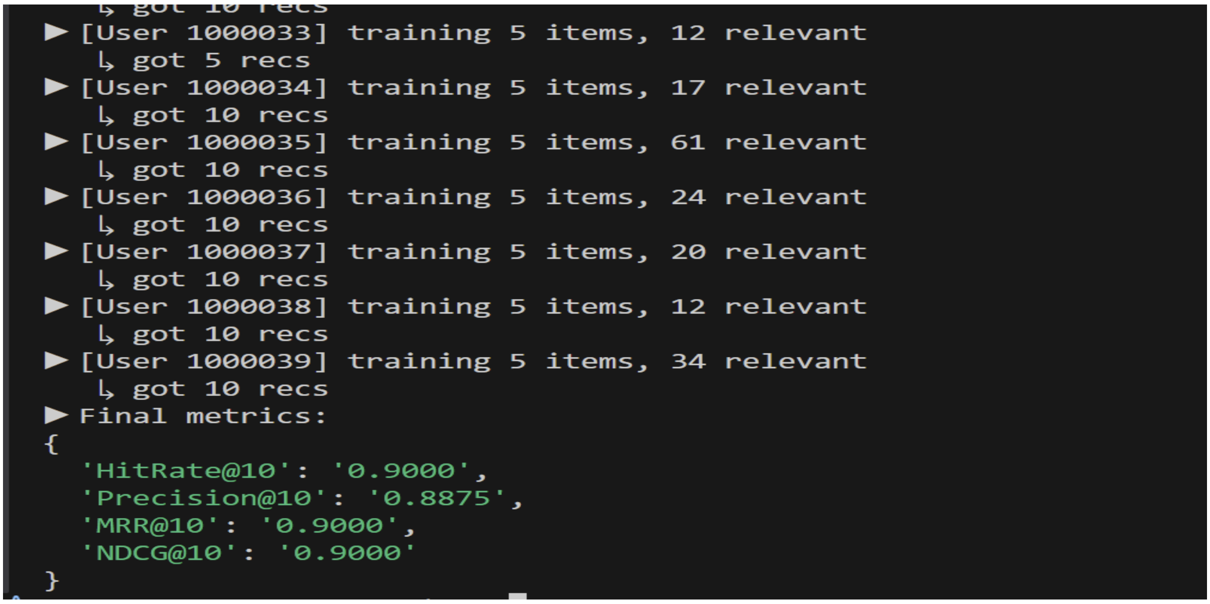
      Why it matters: indicates the purity of the top-K—it penalizes irrelevant items in the returned list.

- MRR@K (Mean Reciprocal Rank): average of 1 ÷ (rank position of the first relevant recommendation) over all users.
  Why it matters: rewards systems that place the first relevant item as early as possible in the ranking.

- NDCG@K (Normalized Discounted Cumulative Gain): a position-weighted measure of ranking quality. Compute DCG@K = Σ (gain_i ÷ log2(i+1)) for each position i in top-K (gain_i = 1 if item is relevant, 0 otherwise), then normalize by the ideal DCG (all relevant items at the top).
  Why it matters: reflects both relevance and position, giving diminishing returns for relevant items appearing deeper in the list.

4. Experimental Result:

Metric	Value
HitRate	0.90
Precision	0.8875
MRR	0.90
NDCG	0.90



```
  ↳ got 10 recs
▶ [User 1000033] training 5 items, 12 relevant
  ↳ got 5 recs
▶ [User 1000034] training 5 items, 17 relevant
  ↳ got 10 recs
▶ [User 1000035] training 5 items, 61 relevant
  ↳ got 10 recs
▶ [User 1000036] training 5 items, 24 relevant
  ↳ got 10 recs
▶ [User 1000037] training 5 items, 20 relevant
  ↳ got 10 recs
▶ [User 1000038] training 5 items, 12 relevant
  ↳ got 10 recs
▶ [User 1000039] training 5 items, 34 relevant
  ↳ got 10 recs
▶ Final metrics:
{
  'HitRate@10': '0.9000',
  'Precision@10': '0.8875',
  'MRR@10': '0.9000',
  'NDCG@10': '0.9000'
}
```

5. Interpretation
    - **High HitRate (90%):** 36 out of 40 users received at least one relevant (similar) book in the top-10—strong success at surfacing at least one good recommendation.
    - **High Precision (~89%):** On average 8.9 of the top-10 recommendations were relevant—very few irrelevant items shown.
    - **Strong MRR & NDCG (0.90):** Relevant items are not only present but appear in early positions, ensuring quick discovery.